UCF Senior Design I

Swarm Operations & Autonomous Rescue (SOAR)

Department of Electrical Engineering and Computer Science
Department of Mechanical and Aerospace Engineering
University of Central Florida
Dr. Chung Yong Chan

Group 15

Adrian Castillo	Computer Engineer
Cristian Gomez	Computer Engineer
Robert Hagerty	Aerospace Engineer
Michael Palin	Computer Engineer
Belle Perry	Aerospace Engineer
Kristopher Tellez	Computer Engineer

REVIEWERS:
Dr. Justin Phelps
Dr. Mark Maddox
Dr. Mike Borowczak
Lucas Ponte

ADVISORS:
DR. Chan (Senior Design Advisor/Supervisor)
DR. Amoruso (Faculty Advisor/UCF liaison to Sponsor)

SPONSOR:
Lockheed Martin
Joseph Rivera

Table of Contents

Table of Contents **1**
List of Tables **5**
List of Figures **8**
Chapter 1 - Executive Summary **8**
Chapter 2 - Project Description **9**
2.1 Project Background and Motivation 9
2.2 Current Commercial Technologies and Existing Projects 10
2.2.1 Drone Technologies in Search and Rescue 10
2.2.2 Swarm and Multi-Agent Systems 11
2.2.3 Sensor and Communication Technologies 11
2.2.4 Limitation of Current Technologies 11
2.2.5 AI and Autonomy 12
2.2.6 Summary 12
2.3 Goals and Objectives 13
2.3.1 Goals 13
2.3.2 Objectives 14
2.3.3 Hardware 16
2.3.3.1 Drone Platforms & Autonomy – Basic Goals: 16
2.3.3.2 Heterogeneous Sensing Payloads – Basic Goals: 16
2.3.3.3 Drone Functionality – Basic Goals 17
2.3.3.4 Swarm Communication Network – Advanced Goals: 17
2.3.3.5 Onboard Computation – Advanced Goals: 17
2.3.3.6 Ground Control Station (GCS) – Stretch Goal: 17
2.3.4 Software 18
2.3.4.1 Core Swarm Intelligence – Basic Goals: 18
2.3.4.2 Adaptive Machine Learning & Mapping – Stretch Goals: 18
2.3.4.3 Advanced Ground Control Station (GCS) Interface – Stretch Goal: 19
2.4 Expected Specifications 19
2.5 House of quality 27
Chapter 3 - Research and Investigation **29**
3.1 Mechanical Hardware
3.1.1 Types of Drone Frames 29
3.1.1.1 Key Performance Parameters 29
3.1.2 Quadcopter Types 33
3.1.3 Applications of Quadcopters 34
3.1.4 Frame Selection for SOAR 35
3.1.5 Summary of Drone Frames 35

3.1.6 Propulsion..... 36
3.1.6.1 Propeller Key Performance Parameters..... 36
3.1.6.2 Blade Number..... 37
3.1.6.3 Blade Shape..... 38
3.1.6.4 Propeller Configuration..... 40
3.1.6.5 Summary of Propeller Design Components..... 42
3.1.6.6 Propeller Products..... 43
3.1.8 Manufacturing Techniques..... 45
3.1.8.1 Additive manufacturing (3D printing)..... 45
3.1.8.2 CNC Machining..... 48
3.1.8.3 Injection Moulding..... 50
3.1.8.4 Summary of Manufacturing Techniques and Selection..... 51
3.1.9 Material Selection..... 52
3.1.9.1 Summary and Material Selection..... 54
3.1.10 Landing Gear..... 55
3.1.10.1 Key Performance Parameters..... 55
3.1.10.2 Landing Gear Types..... 56
3.1.10.2 Landing Gear Summary & Selection..... 58
3.1.11 Vibration and Shock Mitigation..... 58
3.1.11.1 Vibration in Drones..... 59
3.1.11.2 Shock in Drones..... 61
3.1.11.3 Vibration and Shock Mitigation Summary..... 63
3.2 Electronic Hardware..... 63
3.2.1 Flight Controller Technologies..... 63
3.2.1.1 Flight Controller Firmware..... 66
3.2.1.2 Flight Controller Selection..... 67
3.2.1.3 Carrier Board Selection..... 70
3.2.1.4 Flight Controller MCU Part Selection..... 73
3.2.1.5 AI Companion Computer..... 75
3.2.1.6 Environmental Sensors and GPS Module..... 78
3.2.2 Motor Technologies..... 84
3.2.2.1 Type of Drone Motors..... 86
3.2.2.2 Applications of Drone Motors..... 89
3.2.2.4 Maintenance and Care for Drone Motors..... 90
3.2.2.5 Drone Motor Selection and Suitability for SOAR..... 90
3.2.2.6 Summary of Drone Motors..... 93
3.2.2.7 Electronic Speed Controllers (ESCs)..... 94
3.2.2.8 Electronic Speed Controller (ESC) Selection and Configuration for SOAR.. 97

3.2.3 Power System and Energy Source..... 101
3.2.3.1 Types of Energy Source..... 102
3.2.3.2 Applications of Power System..... 104
3.2.3.3 Voltage Regulation and Power Management..... 105
3.2.3.4 Battery Maintenance and life cycle..... 106
3.2.3.5 Battery Selection and Suitability for SOAR..... 106
3.2.3.6 Summary of Drone Power Supply..... 108
3.2.3.6 Cell Requirement for Power Supply..... 108
3.2.3.7 Summary of Drone Power Supply..... 109
3.3 Communication Technology..... 110
3.3.1 Communication Parameter..... 110
3.3.2 Types of Wireless Networking Protocol..... 112
3.3.3 Telemetry Protocol Selection..... 113
3.3.2.4 Telemetry Radio Selection..... 114
3.3.4 Video Streaming Protocol..... 116
3.3.5 Video Stream Transmitter Selection..... 118
3.3.6 Drone-to-Drone Communication..... 120
3.3.7 Drone-to-Base Communication..... 120
3.4 Software of Drone..... 120
3.4.1 Computer Vision Frameworks..... 121
3.4.2 Computer Vision Algorithms..... 122
3.4.3 Evolution of Computer Vision Frameworks/Algorithms..... 123
3.4.4 Computer Vision Framework/Algorithm Comparison..... 128
3.4.5 Summary of Computer Vision Frameworks/Algorithms..... 130
3.4.6 Swarm Coordination Ecosystem..... 131
3.4.7 Comparing Swarm Coordination Ecosystems..... 133
3.4.8 Summary of Swarm Coordination Ecosystem Components..... 135
3.4.9 Human Detection Objective..... 136
3.4.10 Ground Control and User Interface..... 143
Chapter 4 - Standards and Design Constraints..... 144
4.1 Standards..... 144
4.1.1 Drone Flight Standards..... 144
4.1.2 Battery and Power Safety Standards..... 145
4.1.3 Communication Standards..... 147
4.1.4 Autonomous System Safety Standards..... 149
4.1.5 Mechanical Standards and Regulations..... 150
4.1.6 Flying over Humans and Vehicles..... 151
4.1.7 Mechanical Design Regulations..... 151

4.1.8 Material Standards..... 152
4.2 Design Constraints..... 152
4.2.1 Weight and Structural Constraints..... 153
4.2.2 Budget and Resource Constraints..... 153
4.2.3 Power Efficiency and Thermal Constraints..... 154
4.2.4 Environmental and Testing Constraints..... 155
4.2.5 Manufacturability and Assembly Constraints..... 155
4.2.6 Ethical and Safety Constraints..... 156
4.2.7 Vibrational and Shock Constraints..... 157
4.2.8 Time and Project Management Constraints..... 158
4.2.9 Summary of Design Constraints..... 159
Chapter 5 - Case Study..... 159
5.1 Case Study One - Belle Perry..... 159
5.2 Case Study Two - Adrian Castillo..... 160
5.3 Case Study Three - Robert Hagerty..... 161
5.4 Case Study Four - Kristopher Tellez..... 162
5.5 Case Study Five - Michael Palin..... 163
5.6 Case Study Six - Cristian Gomez..... 164
Chapter 6 - Mechanical Modeling..... 165
6.1 Overview and Initial Modeling..... 165
6.1.1 Updated Design..... 167
6.1.2 Flight Performance Hand Calculations..... 169
6.2 FEA Analysis..... 170
6.2.1 Overview..... 170
6.2.2 Deformation..... 171
6.2.3 Stress and Strain..... 172
6.3 Flight Plan..... 173
6.3.1 Overview..... 173
6.3.2 Assembly..... 173
6.3.3 Calibration..... 174
6.4 Structural Fatigue Analysis..... 174
6.4.1 Fatigue Mechanisms and Behavior..... 174
6.4.3 Mitigation Strategies for Fatigue..... 174
Chapter 7 - Electrical Hardware Design..... 176
7.1 Overview of the Electronic System..... 176
7.2 Flight Control System..... 177
7.3 Jetson Xavier Companion Computer..... 179
7.4 Electronic Speed Controller..... 180

7.5 Sensors and Peripheral Modules....181
7.6 Power Supply and Regulation....182
7.7 Communication Architecture....184
7.8 Integration and Safety Considerations....185
7.9 Summary....186
Chapter 8 - Software Design....187
8.1 Object Detection and Obstacle Avoidance....187
8.2 Communication and Ground Control....194
8.3 Swarm Operations....195
8.4 Landing Sequence and Failsafe Behavior....197
8.5 Summary....197
Chapter 9 - Prototyping and Fabrication....197
9.1 Drone Body....197
9.2 Drone Construction....199
Chapter 10 - System Testing and Evaluation....199
10.1 NVIDIA Jetson TX Implementing Human Detection....199
10.1.1 Human Detection Model Evaluation....199
10.2 Mechanical Integration Testing....205
10.2.1 Flight Testing....206
Chapter 11 - Administrative Content....207
11.1 Budget....207
11.2 Distribution of Work....208
11.3 Milestones....210
Appendix A – References....217
Appendix B – copyright permission....224
Appendix C – etc....224
Appendix D - Software Code....224
Appendix E - AI Prompts and Outcomes....236

List of Tables

Table 2.4.1 Overall System Specifications of Drone....18
Table 2.4.2 Specifications of Drone....21
Table 3.1.1.1 Performance Parameters Comparison Between Fixed Wing and Rotating Wing....28
Table 3.1.1.2 Performance Parameters Comparison Between Rotating Wing Types....29
Table 3.1.6.2.1 Blade Number Comparison....34
Table 3.1.6.3.1 Blade Shape Comparison....36

Table 3.1.6.4.1 Propeller Blade Pitch Comparison....38
Table 3.1.6.4.1 Propeller Blade Pitch Comparison....40
Table 3.1.8.1 Performance Parameters Comparison Between Additive Manufacturing Techniques 44
Table 3.1.8.4 Manufacturing Technique Comparison....48
Table 3.1.9.1.1 Material Comparison....51
Table 3.1.10.2.1 Landing Gear Comparison....54
Table 3.2.1.1 Flight Controller Firmware Comparison....62
Table 3.2.1.2 Flight Controller Pros and Cons....65
Table 3.2.1.3 Flight Controller Carrier Board Pros and Cons....68
Table 3.2.1.4 Flight Controller FMU Pros and Cons....71
Table 3.2.1.5 AI Companion Computer Constraints Comparison....73
Table 3.2.1.6.1 GPS Module Comparisons Table Selection....75
Table 3.2.1.6.2 LiDAR Specifications....77
Table 3.2.1.6.3 RGB Camera Specifications....78
Table 3.2.1.6.4 EO Camera Specifications....78
Table 3.2.1.6.5 Thermal Camera Specifications....79
Table 3.2.1.6.6 Stereo Depth Camera Specifications....80
Table 3.2.2.5.1 Motor Technology Selection for SOAR....87
Table 3.2.2.5.2 Motor Product Selection for SOAR....89
Table 3.2.2.7 Motor Encoding Protocol Selection for SOAR....92
Table 3.2.2.8.1 ESC Type Selection for SOAR System....93
Table 3.2.2.8.2 ESC Configuration Selection for SOAR System....94
Table 3.2.2.8.3 ESC Product Selection for SOAR....96
Table 3.2.3.5 Battery Selection Comparison Chart....103
Table 3.2.3.6 Component Voltage & Current Requirements chart....105
Table 3.3.3 Telemetry Communication Protocol Selection Chart....110
Table 3.3.2.4 Telemetry Radio Selection Comparison Chart....112
Table 3.3.4 Video Streaming Communication Protocol Selection Chart....114
Table 3.3.5 Video Streaming Transmitter Selection Chart....115
Table 3.4.3.1 Evolution with Pros and Cons on OpenCV Versions....119
Table 3.4.3.2 Evolution with Pros and Cons on TensorFlow / TensorFlow Lite Versions....120
Table 3.4.3.3 Evolution with Pros and Cons on PyTorch Versions....121
Table 3.4.3.4 Evolution with Pros and Cons on YOLO Versions....122
Table 3.4.3.5 Evolution with Pros and Cons on SSD Versions....122
Table 3.4.3.6 Evolution with Pros and Cons on R-CNN Versions....123
Table 3.4.3.7 Evolution with Pros and Cons on NanoOWL Versions....124
Table 3.4.4.1 Comparison of Computer Vision Frameworks with YOLO for SOAR AI System....

125
Table 3.4.4.2 Comparison of YOLO Versions for SOAR AI System.....125
Table 3.4.7.1 Comparison of Swarm Coordination Ecosystem components.....130
Table 3.4.7.2 Swarm Coordination Ecosystem components Pros and Cons..... 131
Table 3.4.9.1 Data Preprocessing Pipeline Steps.....136
Table 3.4.9.2 Data Augmentation..... 137
Table 11.1 Estimated Budget..... 193
Table 11.2 Distribution of Work..... 195
Table 11.3.1 Project Initialization: Senior Design 1 Milestones.....197
Table 11.3.2 Project Initialization: Senior Design 2 Milestones.....198
Table 11.3.3 Project Fabrication Milestones.....198
Table 11.3.4 Bill Of Materials..... 202

List of Figures

Figure 2.4.1 System Hardware Diagram....24
Figure 2.4.2 Software Flowchart.... 25
Figure 2.5 House of Quality....27
Figure 3.4.9.1 RGBTDronePerson Human Detection.... 136
Figure 3.4.9.2 RGBTDronePerson Human Detection Additional Sets.... 137
Figure 3.4.9.3 AU-AIR Dataset (for RGB model) Display.... 138
Figure 3.4.9.4 Organization of Datasets.... 140
Figure 3.4.9.5 Splitting of Datasets.... 141
Figure 6.1 Initial CAD Model....165
Figure 6.2 Initial CAD Drawing.... 166
Figure 6.3 X4-500 Quadcopter.... 167
Figure 6.4 Initial Design of Modified Central Drone Body....168
Figure 7.1 SOAR System-Level Electronic Architecture....169
Figure 7.2 Cube Orange Flight Controller Connection Diagram.... 170
Figure 7.3 Jetson Xavier Connectivity Diagram....172
Figure 7.4 T-Motor F66A 4-in-1 ESC Wiring and Signal Flow.... 174
Figure 7.6 Power Supply Wiring.... 175
Figure 7.7 Communications Wiring and Signal Flow.... 177
Figure 8.1.1 ArduCopter Sensor Coverage Diagram....181
Figure 8.1.2 Obstacle Avoidance via LiDAR.... 183
Figure 8.1.3 Human Detection via onboard camera processed through Jetson AGX Xavier..... 186
Figure 8.1.4 Initialization Process for Jetson Xavier AGX.... 187
Figure 8.2 Communication Process for Raspberry Pi 4....187
Figure 8.3 Swarm Algorithm Software Diagram....190
Figure 9.1.1 Initial Drone Kit.... 191
Figure 10.1.1 Model 3 Datasets Performance....193
Figure 11.1 Gantt Chart.... 202

Chapter 1 - Executive Summary

The aftermath of a disaster can be devastating—whether caused by natural events such as hurricanes, earthquakes, and floods, or by incidents like building collapses and transportation accidents. In these situations, people may become trapped or go missing. Local authorities,

emergency management agencies, and specialized organizations deploy first responders to search for survivors; however, traditional search-and-rescue efforts can be slow, resource-intensive, and prone to overlooking critical areas.

To address these limitations, the SOAR system—Swarm Operation and Autonomous Rescue—was developed. Sponsored by Lockheed Martin and designed by engineering students at the University of Central Florida, SOAR is a heterogeneous drone swarm equipped with advanced communication and detection capabilities.

The SOAR drones autonomously navigate disaster zones, scanning the area using both RGB and thermal cameras. Through an integrated inter-drone mesh network, each drone shares video and sensor data with other swarm members and with the base control station. When a significant heat signature is detected and a potential target is identified, the system transmits an alert and GPS coordinates to the base station, enabling rescuers to pinpoint the exact location of an individual in need.

Deploying multiple drones allows the swarm to cover large areas efficiently, significantly reducing search time while increasing the likelihood of finding survivors. The mesh network supports long-range communication and ensures that drones remain connected even when dispersed across wide regions. Each drone is also designed for autonomous flight, with onboard obstacle detection and avoidance systems to safely navigate complex environments. Power is supplied by rechargeable batteries capable of sustaining medium-duration missions.

The drones operate in coordination with a base controller station—also sponsored by Lockheed Martin and developed by another team of UCF engineering students—which receives sensor data, processes video streams, and applies computer vision algorithms to detect elevated heat signatures.

This document provides a comprehensive overview of the design and development of the SOAR system. It details component selection, technologies used, underlying theories, assembly and implementation processes, and project limitations. The discussion begins with background information and project context, followed by the system's goals, objectives, and specifications. Subsequent sections outline the applicable constraints and standards, along with schematics and diagrams illustrating how the drone's components integrate into a complete system. The documentation continues with the full system design, final conclusions, and appendices containing references and copyright information.

Chapter 2 - Project Description

2.1 Project Background and Motivation

When disaster strikes, the clock starts ticking. Search and rescue teams face tremendous challenges, including limited resources and hazardous conditions that put responders at risk. Our project, SOAR (Swarm Operations & Autonomous Rescue), was conceived to help overcome these obstacles. With sponsorship from Lockheed Martin, the team is developing a coordinated fleet of specialized drones capable of working together autonomously to locate people in need.

These drones will utilize a range of sensor technologies to rapidly and safely search hazardous environments.

As part of our senior design capstone at the University of Central Florida, SOAR represents a comprehensive application of our engineering education. It challenges us to integrate principles from aerospace, electrical, computer, and mechanical engineering into one cohesive system. This project mirrors the complexity of real-world engineering work, requiring our team to collaborate effectively, manage time and resources, and innovate with purpose. A central focus of our work involves developing the artificial intelligence that will allow the drones to communicate, make decisions, and operate as a coordinated swarm.

The potential of this project extends far beyond the classroom. If successful, SOAR could serve as a transformative tool for emergency response operations, greatly reducing search times and increasing the chances of locating survivors. Furthermore, the swarm intelligence and sensor fusion techniques we are developing can be adapted to other important domains, such as environmental monitoring, disaster assessment, and security applications. This highlights the broader real-world value of autonomous collaborative systems and reinforces the importance of our work under the guidance and support of Lockheed Martin.

2.2 Current Commercial Technologies and Existing Projects

In the development of SOAR, it is critical to examine the current state of commercial drone technology, sensor systems, communication methods, and artificial intelligence capabilities. Existing platforms and research projects reveal valuable design strategies and highlight significant limitations, which our project intends to address. Additionally, the 2023–2024 SOAR project laid the groundwork for integrating swarm functionality, sensor payloads, and autonomous decision-making. This section builds upon their efforts by surveying the technologies they explored, identifying areas of improvement, and outlining the direction our current team will take to enhance the system's capabilities.

2.2.1 Drone Technologies in Search and Rescue

Modern drone platforms have already proven valuable in emergency and disaster response. Companies such as DJI and Parrot have developed drones with robust sensor payloads and dependable flight control, which are now used by agencies around the world. Platforms like the DJI Matrice 300 RTK and Parrot Anafi USA demonstrate how UAVs can provide vital aerial views in scenarios where access is limited or dangerous. These drones are often equipped with thermal imaging cameras, LiDAR, and optical sensors, enabling operators to locate individuals in collapsed buildings, dense forests, or other hazardous zones.

Despite their strengths, these systems remain constrained by their design for single-unit, operator-controlled missions. Their effectiveness is often limited by the reliance on skilled human operators, making coordinated multi-drone operations difficult. The 2023–2024 SOAR team used these commercial platforms as a reference point but began exploring autonomy and swarm communication, areas where off-the-shelf systems currently fall short.

2.2.2 Swarm and Multi-Agent Systems

While single-drone systems provide immediate utility, multi-drone swarms offer revolutionary potential by scaling operations and reducing human workload. Research institutions and government agencies are at the forefront of developing swarm coordination technologies. Programs such as DARPA's OFFSET (Offensive Swarm-Enabled Tactics) and ETH Zurich's Flying Machine Arena have demonstrated that fleets of autonomous drones can coordinate in real-time, achieving complex tasks through distributed control. Swarm technologies offer clear advantages in scalability, redundancy, and adaptability, making them well-suited for unpredictable environments.

However, most current swarm implementations are limited to laboratory demonstrations or restricted to military applications, which reduces their accessibility and relevance to humanitarian rescue. The previous SOAR team began experimenting with swarm coordination by developing a basic mesh network and role-based task assignment among drones. Our project will expand upon this by pursuing more advanced task distribution methods, improving network efficiency, and enabling dynamic role switching within the swarm.

2.2.3 Sensor and Communication Technologies

Sensors form the backbone of environmental awareness for drones, and when paired with communication systems, they enable UAVs to navigate, detect objects, and transmit data efficiently. Thermal imaging, for example, allows drones to detect body heat in low-visibility conditions such as smoke or darkness, making it one of the most widely deployed tools in search and rescue. Multispectral and hyperspectral cameras extend this capability by capturing data across multiple wavelengths of the electromagnetic spectrum, which can highlight differences between vegetation, soil, and man-made materials. This makes it possible to identify disturbed terrain or locate survivors whose clothing blends into the background. LiDAR adds another dimension, producing detailed 3D maps in debris-filled or smoke-obstructed environments where optical sensors struggle.

Communication technologies are equally critical. Many current UAVs still rely on direct operator links, such as DJI's OcuSync or Parrot's enhanced Wi-Fi systems, which provide real-time video and control but remain limited by line-of-sight and interference. Emerging solutions, such as mesh networks and 5G-enabled communication, allow multiple UAVs to act as relays, maintaining data sharing and situational awareness even when conventional infrastructure has collapsed. The prior SOAR team deployed a thermal camera and experimented with Wi-Fi mesh networking. Building on this, the current project will test long-range sensors, improve upon communication modules, and RF-based mesh protocols with potential error handling and encryption support to support both ground-station links and inter-drone communication within the swarm.

2.2.4 Limitation of Current Technologies

Despite the advances in UAV hardware and software, several constraints continue to limit their application in fully autonomous rescue missions. Short battery life, often less than 30 minutes

per flight, restricts mission duration and requires frequent redeployment. Operator-linked communication restricts Beyond Visual Line of Sight (BVLOS) missions due to both technical limitations and regulatory restrictions, reducing their operational range in disaster zones. Additionally, the absence of true swarm intelligence prevents UAVs from coordinating at scale in dynamic, complex environments.

The 2023–2024 SOAR team addressed some of these concerns by experimenting with modular payloads and early swarm networking, but their systems still faced limitations in coordination reliability, autonomous navigation, and power efficiency. Our project will directly tackle these issues using improved flight algorithms, modular sensor integration, and more energy-efficient designs.

2.2.5 AI and Autonomy

Artificial intelligence continues to redefine drone capabilities, pushing UAVs beyond operator dependence and toward autonomous decision-making. Vision-based object detection models can now recognize humans, vehicles, and movement from aerial imagery with high accuracy, while SLAM (Simultaneous Localization and Mapping) enables UAVs to navigate without GPS—a critical function in collapsed buildings or underground environments. Companies such as Skydio and Google's Wing project have integrated AI into their platforms, offering intelligent obstacle avoidance and autonomous route planning, though these systems are typically closed-source and optimized for delivery or inspection rather than unpredictable disaster response.

The 2023–2024 SOAR team advanced in areas such as thermal image recognition and early computer vision but did not achieve full obstacle avoidance or robust GPS-denied navigation. Building on this foundation, our team will pursue lightweight machine learning models optimized for real-time inference on UAV hardware, enabling drones to identify hazards, locate survivors, and make navigation decisions.

2.2.6 Summary

The technologies examined above provide a robust foundation for developing autonomous rescue drones, but they also highlight persistent gaps in endurance, autonomy, communication, and coordination. Commercial UAVs offer proven platforms with advanced sensors, swarm research validates the potential of multi-agent systems, and AI has begun to demonstrate real-world autonomy. However, these elements remain fragmented, preventing widespread adoption in life-saving applications.

SOAR 2023–2024 demonstrated feasibility and outlined several core mechanisms. Building on their work, this year's SOAR project aims to bridge these limitations by combining multi-sensor integration, swarm coordination, and onboard AI in a scalable, field-ready rescue system. With continued sponsorship and guidance from Lockheed Martin, the project serves not only as a proof of concept but also as a step toward real-world deployment in humanitarian aid and disaster recovery.

2.3 Goals and Objectives

The SOAR project sets forth a structured approach toward designing and deploying a heterogeneous drone swarm. Our goals begin with establishing foundational control and communication systems, then extend to intelligent coordination and autonomous navigation. Ultimately, we aim to create a modular and expandable system capable of responding dynamically to real-world disaster scenarios.

2.3.1 Goals

Basic goals:

The basic goals of this project are to develop a heterogeneous swarm of three drones capable of autonomous flight and inter-drone communication. Each drone will be equipped with distinct sensing modalities, such as infrared and optical cameras, to enable complementary data collection. A foundational communication network will be established to share telemetry and status updates, and the system will be demonstrated through a proof-of-concept search pattern within a defined area. In addition, the drones will be structurally sound and be able to withstand extensive search and rescue missions.

- Develop a functional heterogeneous swarm of three drones capable of basic autonomous flight and inter-drone communication.
- Integrate distinct sensing modalities (e.g., Infrared, Optical Camera) onto individual drones to enable complementary data collection.
- Establish a foundational communication network for sharing basic telemetry and status updates between drones.
- Demonstrate a proof-of-concept search pattern within a defined area.

Advanced goal:

Our advanced goals are to implement distributed AI algorithms that enable dynamic task allocation, allowing drones to autonomously divide and conquer a search area. By achieving real-time sensor fusion, data from multiple drones will be combined to generate a unified and confident target identification. The project will also develop and demonstrate efficient route optimization algorithms to minimize search time while ensuring full area coverage. Ultimately, the system will be validated through its ability to robustly locate and identify mock targets, such as heat signatures or objects, within a strict time limit. We have put the advanced goals into bullet points below for easier viewing:

- Implement distributed AI algorithms for dynamic task allocation, allowing drones to autonomously divide and conquer a search area.
- Achieve real-time sensor fusion, combining data from multiple drones to create a unified, confident identification of a target.
- Develop and demonstrate efficient route optimization algorithms that minimize total search time while ensuring complete area coverage.

- Create a robust system that can successfully locate and identify mock targets (e.g., heat signatures, objects) within a strict time limit.
- Create aerodynamic drone design to maximize stability and lower drone weight

Stretch Goal:

The stretch goal of this project is to enhance the swarm's capabilities by enabling it to autonomously adapt its search strategy in response to real-time findings from individual drones. A ground control station will be incorporated to provide live visualization of swarm movement, sensor data, and target identification. Additionally, machine learning-based target recognition will be implemented to improve classification accuracy, and the swarm's functionality will be expanded to include basic mapping of the search area.

- Enable the swarm to autonomously adapt its search strategy based on real-time findings from a subset of drones.
- Incorporate a ground control station to visualize swarm movement, sensor data, and target identification in real-time.
- Implement machine learning-based target recognition to improve classification accuracy of objects of interest.
- Expand the swarm's operational capabilities to include basic mapping of the search area.
- Enable each drone to chain with each other for extended communication range.

2.3.2 Objectives

Basic Objective:

The basic objective of this project is to construct and configure three flight-ready drones, each equipped with a flight controller for autonomous operation. The swarm will integrate at least three different sensor types, these will be infrared cameras, LIDAR, and EO cameras, with each sensor fully functional and streaming data. A communication network will be established to enable successful transmission of GPS coordinates and status messages between all drones. Finally, the system will be programmed to execute a basic pre-defined search flight path, allowing the drones to simultaneously cover a defined area of approximately 100 meters.

- Construct and configure three flight-ready drones, each equipped with a flight controller for autonomous operation.
- Integrate at least three different sensor types, such as infrared camera, LIDAR, EO camera, across the swarm, with each sensor functional and streaming data.
- Establish a network communication, demonstrating successful transmission of GPS coordinates and status messages between all drones.
- Program a basic pre-defined search flight path that all drones can execute simultaneously to cover an area, such as five hundred meters.

- Create three drones that can mechanically withstand the swarm and environmental elements.
- Design a drone swarm capable of holding required electronic payload and perform flight and landing operations.

Advanced Objective:

The advanced objective of this project is to develop a distributed algorithm that allows drones to broadcast discovered sub-areas, enabling the swarm to avoid redundant searches and achieve over 95% area coverage. The system will fuse sensor data to accurately identify and classify at least three distinct target types, such as a person, vehicle, and backpack, with 90% accuracy in a controlled setting. Additionally, a route optimization algorithm will be implemented to reduce total search time by at least 10% compared to a basic parallel lawnmower pattern. Ultimately, the full swarm must successfully locate and identify all mock targets within a 20-minute time limit in a designated outdoor search area.

- Develop a distributed algorithm where drones broadcast discovered sub-areas, enabling the swarm to collectively avoid redundant search efforts and achieve >95% area coverage.
- Fuse the different sensor responses to correctly identify and classify a minimum of three distinct target types, such as person, vehicle and backpack, with 90% accuracy in a controlled environment.
- Implement a route optimization algorithm that reduces total search time by at least 10% compared to a simple parallel lawnmower pattern.
- The full swarm must successfully locate and identify all mock targets within a 20-minute time constraint in a designated outdoor search area.
- Using software analysis to design and optimize the drone structure to decrease the frame rate.
- Using FEA analysis, optimize drone weight and thrust variables to obtain the desired flight parameters that enable stable and efficient flight.

Stretch Objective:

The stretch objective of this project is to design a decision-making algorithm that enables a drone detecting a potential target to autonomously summon the nearest drone with a complementary sensor for verification. A ground control software interface will be developed to display live drone positions, sensor status, and target confirmation alerts. The system will also be designed to support the addition of more drones with varied sensor types, while incorporating a configuration that allows one drone to relay information to another, thereby extending the swarm's operational range

- Design a decision-making algorithm where a drone that identifies a potential target can summon the nearest drone with a complementary sensor for verification, without human intervention.
- Develop a ground control software interface to display live drone positions, sensor status, and target confirmation alerts.
- Allow more drones to be added to the swarm with possibly different types of sensors.
- Design a drone configuration, where one drone will be able to relay the information to another, thus increasing the range of the drones.
- Design the drone structure to be manufactured using carbon fibers.

2.3.3 Hardware

2.3.3.1 Drone Platforms & Autonomy – Basic Goals:

The SOAR system's core consists of three quadcopter drones, forming a scalable and heterogeneous swarm. Each drone platform must be capable of stable, autonomous flight and serve as a robust base for specialized sensor payloads. The primary autonomy for each unit will be managed by a Pixhawk 4 (or similar) flight controller, running the PX4 open-source autopilot software. This enables pre-programmed mission execution, failsafe behaviors, and low-level flight stability.

The critical hardware differentiator is the assignment of a unique sensing modality to each drone, creating a complementary data-gathering system. The physical integration of these sensors requires careful consideration of weight distribution, power draw, and vibration damping to ensure both flight performance and sensor data integrity. Each drone will be powered by a high-capacity 4S Lithium-Polymer battery to provide sufficient flight time for search missions, with all electronic speed controllers (ESCs), motors, and propellers selected to provide the necessary thrust-to-weight ratio for the added payload.

2.3.3.2 Heterogeneous Sensing Payloads – Basic Goals:

The swarm's effectiveness hinges on its heterogeneous sensing units, with each drone equipped for a specific detection role: Infrared Drone: This unit will be equipped with an infrared camera. Its primary function is to detect heat signatures, crucial for locating living persons or recent mechanical activity through visual obstructions like foliage or in low-light/night conditions. The camera will be connected to an onboard companion computer (e.g., NVIDIA Jetson Nano) for initial image processing.

EO camera Drone: This drone will carry a high-resolution RGB camera capable of streaming live video and capturing still images. Its role is to provide detailed visual confirmation and identification of targets located by other drones, and to gather general situational awareness of the search area, where algorithms will run on it to perform real-time object detection.

LIDAR Drone: This unit will be fitted with a LIDAR sensor. Its objective is to generate real-time elevation maps. This data is critical for identifying potential hazards, understanding the topography for route optimization, and locating objects based on structural shape rather than thermal or color properties.

Ultrasonic Drone: This drone will be equipped with an array of ultrasonic sensors capable of short-range distance measurement and obstacle detection, where it will complement the other sensing units by providing precise, close-range environmental awareness, especially in cluttered or GPS-denied environments such as forests, caves, or partially collapsed structures.

2.3.3.3 Drone Functionality – Basic Goals

The drone swarm starts with the mechanical base. Each drone is designed to have different operating capabilities, however the drone frame and structure will be designed in parallel. Each drone will be altered to fit each detection role, and be able to fly during the 30 minute operating time. The material and sizing of the drone shall be determined as the hardware for each drone is selected.

2.3.3.4 Swarm Communication Network – Advanced Goals:

The swarm will implement a star communication topology in which all drones communicate both with a central hub and directly with each other when needed, enabling coordinated operations and data sharing in real time. Each drone will be equipped with a radio module capable of transmitting and receiving sensor data, positional information, and alerts within the swarm, ensuring low-latency, high-reliability links. This network allows drones to share processed sensor outputs while maintaining situational awareness and dynamic path planning. Radio frequency management will be optimized to minimize interference and allow all three drones to operate cohesively, leveraging their specialized sensing capabilities while maintaining seamless connectivity with the central computer.

2.3.3.5 Onboard Computation – Advanced Goals:

Each drone's sensing payload will be managed by an onboard companion computer, such as an NVIDIA Jetson Nano or Raspberry Pi 4. These devices are responsible for the initial processing of raw sensor data (e.g., cropping thermal images, running inference on object detection models, parsing LIDAR point clouds) before transmitting relevant findings to the swarm. This edge computing approach minimizes the volume of data that needs to be transmitted wirelessly, reducing latency and bandwidth requirements. The companion computers will also host the distributed AI algorithms that allow for decentralized decision-making, such as negotiating task division based on proximity and sensor capability.

2.3.3.6 Ground Control Station (GCS) – Stretch Goal:

A central Ground Control Station will serve as the human-machine interface for the swarm. The hardware for the GCS will be a ruggedized laptop running mission planning software like

QGroundControl or a custom-developed application. This station will be equipped with a long-range radio telemetry link to maintain a constant connection with the swarm, allowing an operator to monitor the overall mission progress, view live sensor feeds from individual drones, update mission parameters on the fly, and receive automated alerts upon target identification. The GCS will also log all telemetry and sensor data for post-mission analysis.

2.3.4 Software

2.3.4.1 Core Swarm Intelligence – Basic Goals:

The software architecture for SOAR is the central nervous system that transforms individual drones into a cohesive, intelligent swarm. The core software goals are divided into four pillars: autonomy, communication, coordination, and perception.

The foundation is Autonomous Flight & Navigation, managed by the PX4 flight stack on each drone's Pixhawk controller. This software handles low-level stabilization, waypoint navigation, and failsafe procedures, providing a stable platform for higher-level algorithms.

The Communication Layer will be implemented using the MAVLink protocol, a lightweight messaging standard for communicating with drones. To create a resilient swarm, a mesh networking protocol will be built on top of MAVLink, allowing any drone to act as a router, relaying commands and data to extend the network's range and prevent single points of failure.

The most critical component is the Swarm Coordination Algorithm. This distributed AI will be hosted on each drone's companion computer and is responsible for decentralized decision-making. Its primary functions include:

Dynamic Task Allocation: Using a market-based or consensus algorithm, drones will autonomously negotiate and assign search sectors based on their proximity, battery life, and sensor capability.

Collaborative Path Planning: Drones will not only plan their own routes but also share their intended paths to optimize overall area coverage and avoid intra-swarm collisions, ensuring the entire search area is scanned efficiently.

Distributed Sensor Fusion: Software modules will be developed to fuse data from different drones. For example, the thermal drone's detection of a heat signature will trigger a request for the optical drone to provide visual confirmation, creating a unified, multi-modal target identification.

Finally, the Perception & Detection software will include machine learning models for real-time analysis. This includes a convolutional neural network (CNN) for the optical drone to identify objects of interest and computer vision algorithms for the thermal drone to isolate human-sized heat signatures from background clutter.

2.3.4.2 Adaptive Machine Learning & Mapping – Stretch Goals:

A key stretch goal is the development of Adaptive Learning Algorithms. This software would allow the swarm to learn from its search patterns and target discoveries in real-time. If a specific area or type of terrain consistently yields false positives, the swarm could collectively adjust its

detection parameters or search strategy for subsequent missions, continuously improving its performance without human intervention.

Another advanced objective is the creation of a Real-Time Collaborative Map. Software on the ground control station (GCS) would ingest and synthesize LIDAR data, target coordinates, and geotagged images from all drones to construct a live, shared 2D/3D map of the search area. This map would be broadcast back to the swarm, providing each unit with an updated common operational picture to inform their future navigation and search decisions.

2.3.4.3 Advanced Ground Control Station (GCS) Interface – Stretch Goal:

The stretch goal for human-swarm interaction is a custom application that would be built using a framework similar to ROS. Its features would include:

- Live Swarm Visualization: A real-time map display showing the position, status, and live video feed of every drone in the swarm.
- Mission Orchestration: A user-friendly interface for defining search areas, setting priorities, and initiating the mission with a single command, leaving the complex task division to the swarm.
- Target Management Console: An alert system that aggregates and displays all potential targets identified by the swarm, complete with confidence levels and fused sensor data (e.g., a thermal image alongside an optical confirmation shot).
- Data Logging & Post-Mission Analysis: Comprehensive logging of all telemetry, communication messages, and sensor data to enable detailed performance review and algorithm refinement after a mission.

2.4 Expected Specifications

Our overall system specifications define how the full drone swarm should perform during a mission. These targets help guide our design and give us clear goals we can test in the real world.

Requirement	Target Value & Units	Description
Flight Time	≥ 10 minutes	Minimum required to execute a full search-and-rescue sweep during a single sortie.
Number of Drones in Swarm	≤ 3 count	Defined by test and safety constraints. Allows for robust swarm communication testing while minimizing system complexity. Matches

		available Jetson and telemetry units.
Number of Drone Sensors	≥ 3 count	The system includes one RGB camera, one infrared (IR) sensor, and optionally a LiDAR. Enables multi-modal sensing for object recognition, terrain awareness, and thermal detection.
Drone-to-Ground Communication Range	≥ 100 meters	Ensures consistent telemetry transmission and command reception from the GCS (Ground Control Station), even in semi-obstructed environments..
Drone-to-Drone Communication Range	≥ 50 meters	Critical for decentralized swarm coordination and real-time position sharing. Enables autonomous behavior and obstacle avoidance between units..
GPS Accuracy	≤ 2 meters	Required for precise localization during autonomous operations and waypoint navigation. Important for swarm spacing and map-based mission planning.
Obstacle Detection Range	≥ 5 meters	Allows early object recognition and safe maneuvering at nominal speeds. Supports fail-safe behaviors and autonomous rerouting.
Ground Station	Laptop	Central node for mission planning, live monitoring, and telemetry logging. Interfaces with

		MAVLink-based control software.
Flight Controller	Cube Orange	Provided by sponsor; supports GPS, MAVLink, companion compute integration.
Companion Computer	Jetson AGX Xavier	Handles real-time AI processing, including object detection and classification using onboard camera feeds
Video Streaming Quality	720-1080 pixels	Supports clear video transmission for real-time video stream and post-mission analysis.
Environmental Operating Range	0 – 40 °C	Based on outdoor testing conditions in Florida.
Frame Rate (FPS)	≥ 20 FPS	Minimum frame rate for real-time detection
Inference Latency	≤ 100 ms	Time between frame capture and detection output
Detection Accuracy	≥ 80% mAP	Mean Average Precision across target classes
False Positive Rate	< 10%	Fraction of incorrect detections
Model Size / Memory Footprint	< 100 MB	Fraction of incorrect detections
Power Consumption	< 15W	Processing power draw from Jetson module

Table 2.4.1 Overall System Specifications of Drone

By Authors

Taking constraints and our goals into consideration. These are our expected component specifications for our SOAR project:

Electrical Specifications		
Requirement	**Estimated Value**	**Description**
GPS Update Rate	10-20 Hz	Based on Cube Orange GNSS capabilities for rapid position updates.
Video Transmission Frequency	Up to 5.8 GHz	Industry-standard video frequency for real-time low-latency feed.
Communication Range	Up to 500 meters	Conservative range estimate for 915 MHz telemetry radio in open field with line-of-sight.
Maximum Current Draw	~70 A	Estimated based on 4 × high-thrust motors and ESCs under full load.
Voltage Input	12 - 16 volts	Standard voltage range for a Lithium-Polymer battery.
ESC Type	4-in-1 BLHeli_32	Supports DShot protocol, fast response, and telemetry feedback to flight controllers. Compatible with Cube Orange.
Compute Power	32 TOPS (INT8)	Jetson AGX Xavier offers high-performance AI acceleration, sufficient for CNNs and real-time decision-making.
Camera & Sensor Specifications		
RGB Camera Resolution	~2.5 MP (e.g., 2688×1520)	Approximate resolution for visible light cameras used for mapping and video stream.

RGB Camera FOV	~100° horizontal	Wide enough for area coverage; balanced for detection accuracy.
RGB Camera FPS	≥ 20 FPS	Ensures smooth live feed and real-time AI inference.
Thermal Camera Resolution	~32×24 pixels	Entry-level thermal sensor for heat signature detection.
Thermal Detection Range	-40°C to 300°C	Sufficient for identifying people, vehicles, and power sources.
Thermal Camera Weight	≤ 5 g	Must remain ultra-light to avoid overloading frames.
Obstacle Sensor FOV	360° horizontal	Full peripheral coverage via multiple ultrasonic/IR sensors or rotating LiDAR
Mechanical Specifications		
Motor Speed	~400 KV	Low-KV motors are preferred for stable flight and efficient hover with larger props.
Thrust per Motor	~2.5 kg	Total thrust across 4 motors exceeds 2× estimated AUW for safe maneuverability.
Motor Size	~47.5 mm × 37.5 mm	Common size for high-thrust quadcopters in this class.
Motor Weight	~165 g	Mid-range weight for motors with strong torque output.
Frame Size	495 mm x 363 mm	Common size for quadcopters in this weight class.

Frame Weight	~300 g	Weight of the frame based on the volume of the expected carbon fiber make up.
Propeller Sizing	~ 4400mm^2	Size of propeller based on predicted weight requirements and drone size
Drone Weight	~ 5 lbs	Predicted weight considering all components and hardware.
Lift	~ 10 lbs	Common to achieve a lift to weight ratio of 2:1

Table 2.4.2 Specifications of Drone

By Authors

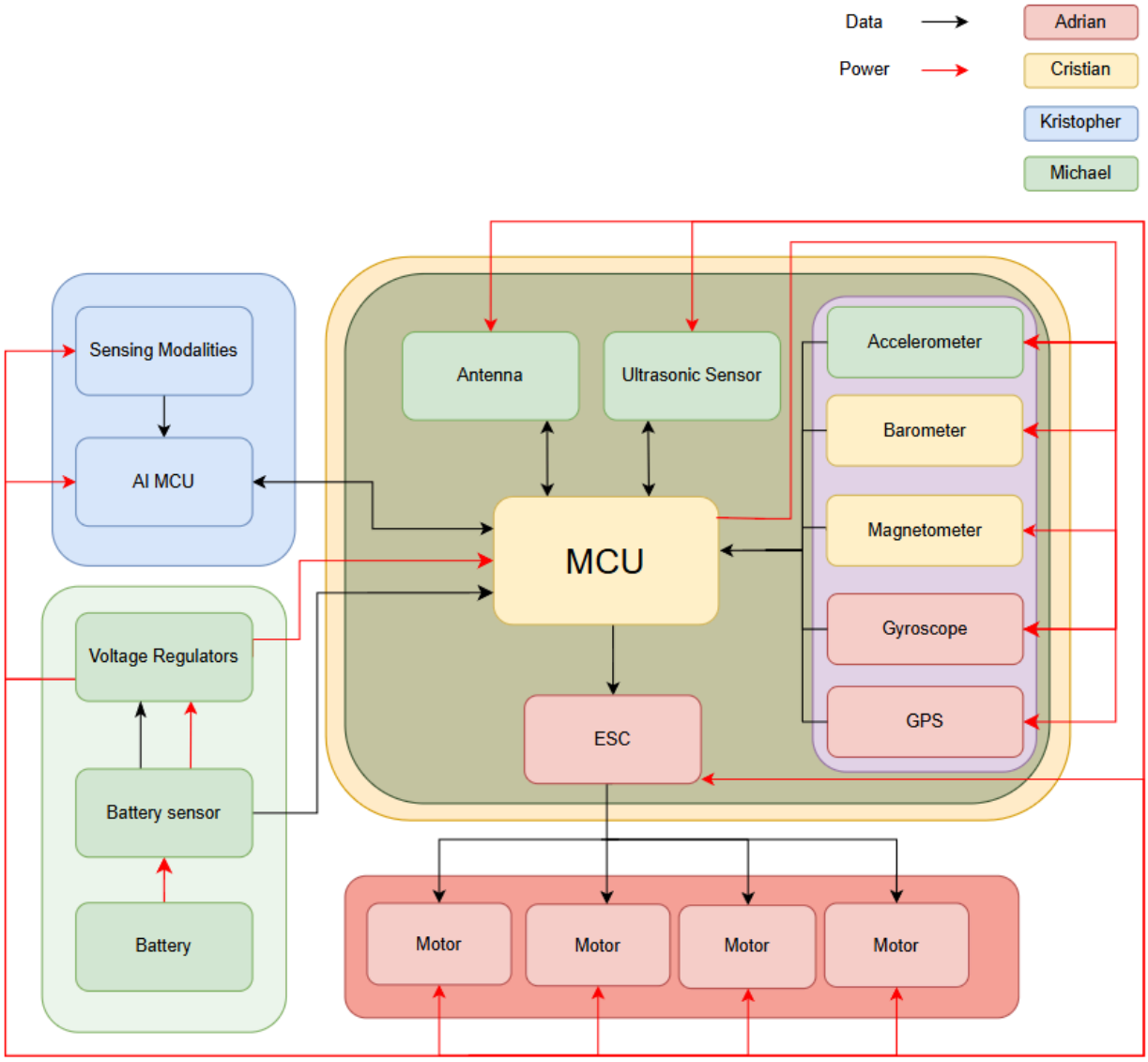


Figure 2.4.1 System Hardware Diagram

By Authors

SOAR hardware system diagram is shown above. The diagram demonstrates the core hardware components needed for SOAR to perform flight and detection. Separated by color, are the different part systems.

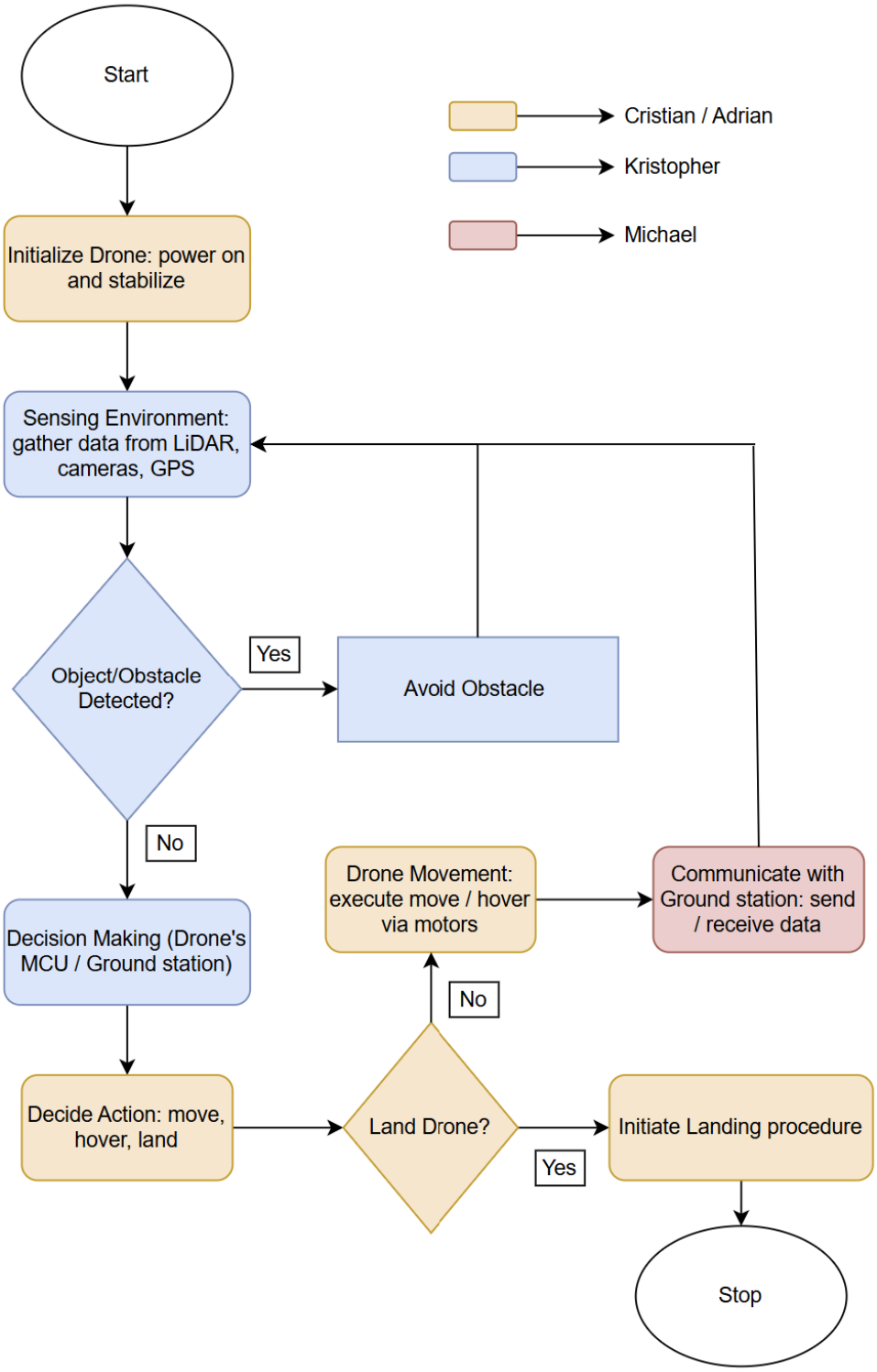


Figure 2.4.2 Software Flowchart

By Authors

The Software Flowchart outlines the process of the SOAR following directions and detections.

2.5 House of quality

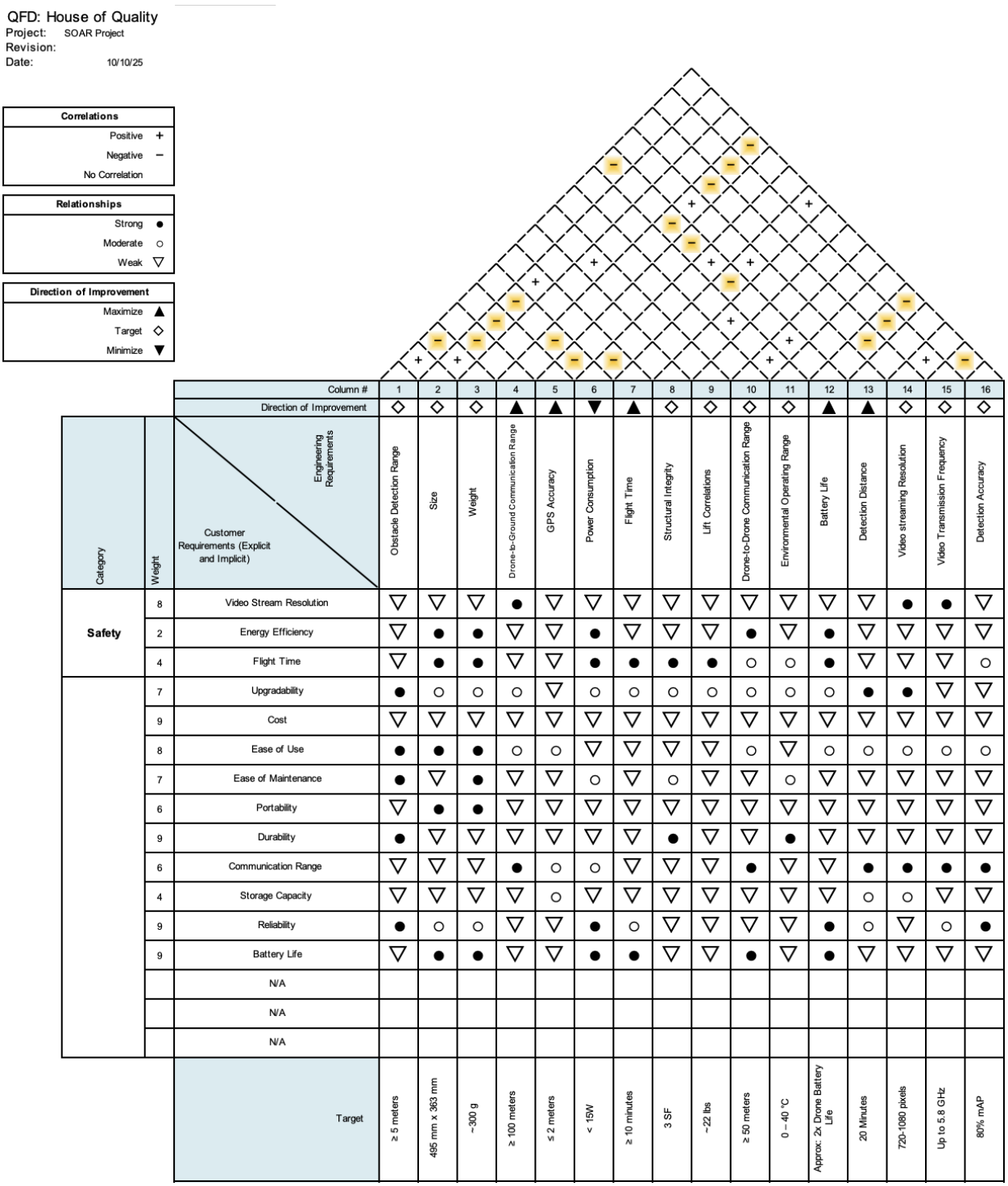


Figure 2.5 House of Quality

By Authors

The picture above is the house of quality for our SOAR which provides a visual aid of engineering characteristic tradeoffs. The left-handed columns depict features that are deemed

important to the end user. The columns located at the top of the table demonstrate the features that could potentially be traded off during engineering parameter considerations. (Note: the spaces in the columns at the top represent that there is no relationship between the two features)

Chapter 3 - Research and Investigation

3.1 Mechanical Hardware

3.1.1 Types of Drone Frames

Drones have existed for more than one hundred years. The applications of the drones vary from search and rescue to package delivery. Recreation drones are common for students and novice drone flyers due to their weight and small electronic housing. On company/military use, drones have been used for investigations, Disaster Response, Traffic Collision, Crowd Monitoring and Surveillance. For relevance to the SOAR Project goals and mission objectives, this research focuses on common industry drone types.

There are two main types of drones: fixed wing and rotating wing. A fixed wing, like the name, is a plane looking structure whereas a rotating wing has moving propellers and looks more like the typical drone that is commonly recognized. Rotating wing drones is a general name and has many types of specific drone versions. This report will discuss Quadcopter, Quad Air, Hexa Copter, AR-Drone Quadcopter and Octocopters. Unmanned Aerial Vehicles (UAVs) have classifications by weight, range, altitude and wing loading. The SOAR team will be able to identify the classification of the drone by Senior Design 2. However, to support the teams' decision-making process, key performance parameters have been outlined for the drone frame.

3.1.1.1 Key Performance Parameters

Weight

Drones, specifically drones housing a complex hardware system are heavy. The more material used for the drone frame, the more thrust will be needed to create lift. When designing the frame, the weight of the design must be a priority to give each drone in the swarm the best ability during its operation.

Size

Based on the application of drones, their size may differ. Based on the knowledge and research of other drones used for search and rescue or adjacent operations, these drones will be approximately 460 mm from wing tip to wing tip. These drones are larger in size and therefore will be more susceptible to drag.

Structural Durability

The frame design must be able to withstand the drag as well as any other elements that may be faced during flight time. The drone swarm should be able to perform multiple operations once tested, and the frame should be designed to withstand multiple missions and possible crashes.

Hover Capability

Stability is a common issue for drones. When a drone is hovering over the target, it should be steady so the electronics and cameras have good quality control. In order for the drone swarm to perform well, the drone's frame should be structurally sound to promote stability during flight.

Environmental Durability

Drones most commonly operate outdoors. These drones will need to operate in different outdoor conditions. Wind and gust are the primary weather conditions faced, but mist, rain, and heat are other factors that could affect the drone. Ensuring that testing is done to mitigate the risk with any environmental condition is essential to good mission data.

Speed

The search and rescue operations are deployed when a person is missing. These missions must be vigilant in finding the person in order to perform rescue operations. Although the swarms are not designed for racing, they will need to be quick and easily maneuverable to search within the search radius.

Stability

Reliability is a primary factor for drone operations. This reliability is increased when stability is high. Stability is a crucial factor for the SOAR project, since the drone swarm will be responsible for carrying out search and rescue missions. Stability plays a role in imaging and data, where lack of stability interrupts imaging quality and data collection.

Motor Efficiency

Mission duration can be heavily influenced by motor efficiency. By designing a frame and choosing propellers that complement the motors size and weight, the motors will be able to work more efficiently and give the drone swarm the highest fly time.

Fixed Wing Drones

Fixed Wing Drones act like a small aircraft. They often mimic the look of an aircraft and fly like a plane. Fixed Wing UAVs are used because they are reliable and have good flight speed. They can fly flights for long durations due to their size. The drone types, however, need a larger takeoff and landing area and have greater wind resistance due to the shape of the drone [25]. They are often selected for long-term surveillance operations.

The Lockheed Martin Desert Hawk is a fixed wing drone used to support small unit intelligence teams. The drone can run surveillance operations and utilize the terrain surrounding it for take off and landing procedures. This drone is ideal for tactical missions due to the weight of the

drone, 8.2 pounds. While SOARs operations may be similar, the team aims to have short term search and rescue operations, ranging 10-20 minutes.

The Lockheed Martin RQ-3 DarkStar is another fixed wind UAV that was used for tactical operation. This UAV is larger, weighing 3900 kg and could fly for up to 12 hours. This drone was designed for long distance mission operations, with the drones' range being 575 miles [26]. While this drone has been useful for many operations, it is out of the scope of ability for the SOAR Senior Design Team.

Fixed Wing Drones have been extremely successful in mission operations for Lockheed Martin and other companies in the past. They are consistently able to perform surveillance operations and long duration missions. While long duration missions do not align with the current goals and objectives of SOAR, the fixed wing drone is still an option.

Rotating Wing

Rotating Wing drones or UAVs utilizes propellers, making the UAV act more as a helicopter rather than a plane. These types of drones create more versatility in flight. It also enables the users to have rapid movement, hovering, and quicker operations. Many recreational drones are also designed with Rotating Wings, so the user can maneuver the aircraft with ease. Two challenges are often faced when designing and building rotating wing drones. Stability is the primary challenge, since increasing the propellers often causes issues for stability and navigation. The second issue often seen is weight. The maximum weight a rotating wing drone can be is significantly lower than the weight of a fixed wing. The thrust of the propellers must be greater than the force of the drone due to the weight.

Rotating Wing drones are optimal for tactical operations, such as search and rescue, Disaster Response, Surveillance, package delivery and Military applications. These types of drones have the ability to maneuver quickly, go to specific locations, and communicate with other drones if used in a swarm application. The Kargu Rotary Wing Attack Drone is an example of an application rotating wing drone [27]. Developed by the Turkish Defense Company, the rotary drone weighs 7060 grams and performs attack operations in a defensive act. The drones lock onto a location and can maneuver to that specific coordinate to perform its operation.

Performance Parameters	**Fixed Wing**	**Rotating Wing**
Weight	20-100 kg	~3 kg
Size	3-12 ft	1-4 ft
Durability (time in flight)	Size dependent, ranging from 30 min to multiple hours	Short term flight, ranging 10-30 minutes depending on battery life, weight, etc.
Hover Capability	Not typically possible	20-40 minutes

Environmental Capability	Can withstand harsh environmental conditions including rain, snow and ice	Sensitive to wind gust, temperature and precipitation
Speed	60-100 km/h	40-80 km/h
Stability	High	Moderate-High
Motor Efficiency	Moderate	Moderate-High

Table 3.1.1.1 Performance Parameters Comparison Between Fixed Wing and Rotating Wing

By Authors

Rotating wing drones is the ideal choice for the SOAR project. The search and rescue operations outlined in the project objective require quick takeoff and landing, maneuverability and speed. These drones are easily on the market, and extensive research exists to support the project's mission objectives. There are four types of rotating wing drones being researched for the SOAR project. They are the most common rotating wing type and have been used in practical applications.

Hexacopter

Hexacopters, like quadcopters, use four propellers to create lift on the drone. Unlike quadcopters, these propellers are housed close to the center of the drone, creating a rectangular shape. These drones are optimal for recreational flying, video recording and small level projects.

Quadcopter

Quadcopters are one of the most common types of rotating wing drones. Designed with four propellers, most quadcopters have central housing for electronics, and four arms that extend out with propellers. Quadcopters are often preferred due to structural durability and crash resistance. These drones are designed to distribute the force of a crash between at least 2 arms if landing on the side, or across the entire drone if landing at the bottom or top of the drone.

AR-Drone Quadcopter

AR Drone Quadcopters are designed to look like a computer fan. The propellers have a built-in 2-1 feature: creating lift and keeping the electronics cool. These drones are most used for recording purposes, as the drones mostly stay still in the air, only moving in the Z direction. They are less maneuverable and can only move in one direction well at a time.

Octocopter

Octocopters are lifted using 8 propellers. Like other drones, the octocopters house electronics in the center of the drone, and the 8 propellers extend off the main housing. While octocopters are not as commonly seen, the drone can recover if one propeller loses functionality during a flight.

Drone Type	Hexacopter	Quadcopter	AR-Drone Quadcopter	Octocopter
Weight	Medium Heavy (2-5 kg) depending on payload	Light Medium (0.5-8 kg) adjusted for payload	Light (0.4-1 kg)	Heavy (5-12 kg) Professional use
Size	Larger than Quadcopters	Small - medium	Compact	Very Large
Durability (time in flight)	20-30 minutes average	20-30 minutes average	10-15 minutes	20 - 40 minutes
Hover Capability	Very stable	Stable, widely used for hovering	Good, designed for easy use	Extremely stable, good for heavy payloads
Environmental Capability	Can handle moderate wind	Limited in strong wings	Limited, better in indoor use	Strong wind resistance
Speed	Moderate-high (40 - 65 km/h)	Moderate (30-50 km/h)	Low - Moderate (15-30 km/h)	High (50-80 km/h)
Stability	High	Moderate - High	Moderate	High
Motor Efficiency	Moderate-High	Moderate-High	Moderate	Moderate - High

Table 3.1.1.2 Performance Parameters Comparison Between Rotating Wing Types

By Authors

SOAR Drone Frame Selection

Based on research and mission objectives of SOAR, the team will be utilizing rotating wing drones. Based on the previous project and research, the team has decided to utilize the quadcopter frame design. The functionality of the quadcopter aligns with the search and rescue goals of the SOAR project. The next section looks at the different types of quadcopters, and what each functionality is.

3.1.2 Quadcopter Types

With quadcopters being the team's selection, there are three types of common quadcopters that were researched. These three drone fixture designs each have their own advantages and disadvantages, discussed in the following sections. It is critical that the frame of each drone is reliable, stable and able to house all sensors and electronics necessary for the drone operations.

H Frame

The H Frame quadcopter is an older design and was most seen in early drone designs. These drone frames are favorable due to their crash resistance. These drones are predicted to survive crashes from higher altitude but have significantly lower speed and range capabilities due to their weight. H frames have the heaviest “dry weight”, meaning that the frame itself weighs more than any other drone structure without electronics installed.

X Frame

The X Frame is designed in a symmetrical pattern, with each arm extending the same distance out from the electronic housing in the center. Like the letter X, the drone arms go out the corners of the housing in a diagonal pattern allowing for larger propellers. With motors installed at each arm tip, the propellers each have a different role in maneuverability during flight. These drones are easy to conduct roll, pitch and yaw maneuverers, and are often seen for search and surveillance purposes. The X Frame design's advantage includes distributed weight in a crash, and weight distributions that allow the drone to fall or crash in an upward direction.

Cross Frame

Cross Frame Quadcopters are extremely like X Frame quadcopters. They are designed with the propeller arms turned, therefore the arms coming from the flat side of the copter. In this design, the arms are not symmetrical, since motors 1 and 3 (front and back) are designed to be slightly longer. These frames cannot withstand crashes as well, since most crash scenarios end with one arm taking the entire force of the crash.

3.1.3 Applications of Quadcopters

Quadcopters are a common drone frame type in the current industry. The frame of these copters is essential for mission operations and are the foundation of drones. Understanding the functionality of Quadcopters is helpful for determining what elements and mission profiles may be seen during operations.

One of the major uses, and project goal, is search and rescue operations. Quadcopters are an ideal choice for swarm and rescue operations due to functionality and speed. Quadcopters can take off quickly, operate and communicate in flight, and navigate in tighter spaces. Unlike fixed wing drones, the quadcopter has significantly shorter battery life and cannot perform long duration operations.

Quadcopters are also used for surveillance operations. Like search and rescue, the drones can go to locations that the average human would not be able to enter. The drone is able to utilize on board video to take surveillance and check for safety in a location. This allows the operator to ensure that the location being entered is safe.

Delivery drones became popular when delivery companies began testing remote delivery using drones. Quadcopters are often a great choice, since they can carry a package and hold a

significant amount of weight. These quadcopters are often larger, to house a larger battery to ensure operational safety.

These drone types are useful for the SOAR team to determine the best frame for the drone swarm. Researching the frame's abilities to create lift, carry weight and perform duration long mission profiles is essential in the research for the drone swarm.

3.1.4 Frame Selection for SOAR

The foundational aspect of the SOAR Drone swarm is the frame. Without the structural ability to fly, the drone swarm will not be operational. In addition, with most drone crashes and issues, the mechanical system (Frame and propellers) are the first aspects to break or get damaged. Each drone in the swarm will carry slightly different electronics and sensors, but all must be able to fly as a swarm. For this to be successful, the drone frame must be designed to carry the weight of the motors, AI modules, video streaming, sensors, flight controllers and any other essential hardware.

The X style quadcopter is the ideal choice for SOAR. With specific design considerations and project functionality, the X style gives the drone swarm the greatest ability to maneuver, perform operation, and return safely. With electronic safety in mind, the SOAR team is most confident in the X Frame quadcopters ability to distribute force in a crash and protect the electronics housed. The quadcopter frame will be approximately 460 mm x 360 mm, which gives enough space for both the housing and the propellers.

As mentioned, one of the requirements for each drone is for the thrust-to-weight ratio to be at least 2:1 to ensure that the drone is easily maneuverable and withstand the flight duration. For this, the frame must be practical and lightweight, to ensure that there is enough space/weight available for the electronics, motor, and propeller.

3.1.5 Summary of Drone Frames

The frame of the drone is the foundational aspect to flight. It houses all electronics, motors and propellers and brings all components together. It acts as the drones line of defense in case of a crash, and will be a symmetrical piece of equipment that will generate lift and allow the swarm to fly. As mentioned before, the two main types of drones are fixed wing and rotating wing. After initial research and project description, the team has decided that the rotating wing is the best choice for the swarm.

Quadcopters are widely used and easily maneuverable. They are the best choice for the SOAR drone swarm, as they will be able to be quick, fast and work together to perform the search and rescue operations. The quadcopter frame should be lightweight and promote stability, as the design will follow the X Frame pattern. The X Frame will be symmetrical, supporting stability and durability. The arms will be attached to the main housing, which will be screwed in and reinforced once studied further.

Maintenance and reusability is another important factor to this project. The drone swarm is to be used for more than one mission, and the frame is most likely to have damage from use to use. To

mitigate this, the frame will be designed with reusability in mind, and add supports to help distribute the landing forces.

Overall, the frame will be the building block of the drone. Each frame will be designed specifically to house the different components and sensors designated in the swarm, and will work to all be around the same weight and distribution. Future research will discuss material options for manufacturing the drone swarm, and basic FEA analysis for shear and bending.

3.1.6 Propulsion

For all aerial vehicles, the main objective is to generate a force to oppose the weight of the vehicle and keep the vessel in the air; this is known as lift. There have been many methods developed over time to create such forces. For rotating wing drones, the primary method of lift generation is propellers. A propeller is a rotating multi-winged component that is driven by the motor to aid in lift generation. In this case, a drone propeller is driven by a motor to push air down in order to cause a pressure difference which generates lift. The following contains research on the various types and parameters of drone propellers including their relevance to the objectives pertaining to the SOAR Project.

Rotating wing drones have various propeller types implemented based on the specifications and requirements the drone must uphold. In order to select a propeller type for the SOAR Project, it is crucial to lay out the essential performance parameters that a propeller design may affect.

3.1.6.1 Propeller Key Performance Parameters

Speed

Flight speed depends on the application of the drone. Drones for Search and Rescue operations need to be able to locate targets and act quickly to ensure the mission can be executed before the situation escalates. This will require a propeller that ensures quick traversal.

Efficiency

With any battery powered device, they lose charge while in use. The drone must be able to stay airborne and complete its objectives throughout the mission duration. An efficient drone will lead to less load on the battery systems resulting in a longer battery life. To ensure the drone remains in flight, a propeller that promotes efficiency will contribute to lasting and more predictable flight times

Thrust

In order to navigate the field, drones must be able to generate vertical thrust and directional thrust. Propeller selection should support directional flight as well as vertical flight to aid in mission traversal.

Stability

When searching for or encountering a target, it is critical that the drone is stable in order to correctly identify the target and administer aid in order to complete the search and rescue operation. To guarantee an effective product, propeller design should have a focus on stable flight and operations.

Complexity

Propellers must be easy to acquire and replace in situations resulting in failure due to crashes or impacts. With this in mind, propellers must lack design and implementation complexities to be obtainable and replaceable if needed.

(Parameters are subject to change during additional research or other considerations involving propellers).

3.1.6.2 Blade Number

Propellers work by rotating a select number of specifically shaped blades around an axis to generate thrust and propel a vehicle in a desired direction. The number of blades that rotate about this axis is very important to several characteristics that should be considered for drone flight. The following breaks down the impact of the blade number for a propeller.

Two-Blade Propellers

Two-Blade Propellers have only two blades that rotate about an axis to generate the lift necessary for flight. These propellers are traditionally used for their higher efficiency and their low drag when compared to other multi-blade configurations [29]. However, these propellers are found to generate less thrust and are less stable in comparison to propellers with more blades [29]. These propellers are used in various hobby or personal drone structures that rely on speed and efficiency rather than stability and control. The efficient nature of the propeller contributes to longer flight times. An example of a drone that uses a two-blade propeller is the DJI Mini 2; a small, fast, lightweight drone made for video capture.

Three-Blade Propellers

Three-Blade Propellers offer an additional rotating blade on the axis that aids in generating more thrust compared to propellers with less blades. The additional blade introduces more drag and is less efficient but is complimented by added stability [29]. The Three-Blade Propeller is a moderate option providing a compromise between function and efficiency. These propellers find use in situations where moderate specifications of weight, stability, and flight time are called for.

Four-Blade+ Propellers

Propellers offering blade configurations offering four or more propeller blades are great for vehicles that have large weight requirements or that must be as stable as possible. As blades are added to a propeller design, it is found that drag, thrust, and stability are increased, while efficiency decreases [29]. The inefficiencies of four-blade or more propellers cause flight times to suffer due to the greater power draw.

Design Aspect	Two-Blade	Three-Blade	Four-Blade+
Speed	High	Moderate	Low
Efficiency	High	Moderate	Low
Thrust	Low	Moderate	High
Stability	Low	Moderate	High
Complexity	N/A	N/A	N/A

Table 3.1.6.2.1 Blade Number Comparison

By Authors

SOAR Blade Number Selection

Propeller blade number is an essential characteristic when selecting a design for the propellers used on a drone. The blade number directly correlates to several key performance parameters for the SOAR Projects' mission objectives, those being speed, thrust, efficiency, and stability. Based on these parameters and surrounding research, the team has elected to utilize a three-blade propeller. As stated, these propellers offer a good compromise between functional characteristics and efficiency whereas propellers with more or less blades offered too high of a trade off.

3.1.6.3 Blade Shape

The shape of the blades on a propeller also have significant influence on the flight parameters of a drone. These shapes help to influence different factors like efficiency and thrust capability. Blade shape can be broken down into two subsystems, that being planform and tip shape.

Tip shape is the description of the furthest point from the pivot point of the propeller. These points have great influences on the performance parameters of a propeller due to their interaction with vortices. Planform is the chord distribution along the blade. This planform shape would describe the variations in chord length of the blade as it approaches the tip. These structures can vary the thrust production of the propeller along with the efficiency of the rotating blades.

Straight Planform and Tip

A straight planform provides a constant chord length along the blade. This provides an intermediate design function that doesn't provide any superior benefits in the listed key performance parameters. The design is easy to manufacture due to its simplistic shape but this simplicity harms the efficiency of the propeller.

Tapered Planform and Tip

For a tapered planform and tip, the blade gets narrower as one approaches the tip of the blade. For certain configurations of a tapered blade, it is shown that this shape has a larger aerodynamic efficiency compared to other blade tip shapes [30]. This shape contributes to less drag and thus results in an increase in efficiency of the propeller. This increase in efficiency can lead to longer flight times due to less load being required of the battery.

Bullnose Planform and Tip

The bullnose planform and tip design contains a widening towards the blade tip. The blade tip is a wider and thicker section compared to the base of the blade section. This widening contributes to more drag but in doing so helps to contribute to more thrust as well as noise. With the greater drag and noise, efficiency is inhibited and thus battery life may suffer.

High Efficiency Shaped Planforms (Elliptical, Prandtl-Bell, etc)

For some applications, it may be desired to achieve the lowest induced drag and use the most efficient propeller possible. This can be obtained by using an elliptical or Prandtl-Bell shaped propeller due to their ability to reduce tip vortices and possess an elliptical lift distribution which both contribute to reducing drag [31]. These blade shapes would be great in an application requiring long flight times or tight power constraints.

Swept Tip

The swept tip is a tip design that angles the tip back. This can be done at varying degree angles to move the vortices to the back side of the blade tip. The result from this blade shaping was a large decrease in the noise of the propeller. This low noise is important for any sound sensors or processing but currently does not seem to fit the scope of the SOAR Project.

Design Aspect	Straight	Tapered	Bullnose	Efficiency Shaped
Speed	Moderate	Moderate	Moderate	Moderate
Efficiency	Moderate	High	Low	Maximized
Thrust	Moderate	Low	High	Moderate
Stability	Moderate	Low	High	Moderate
Complexity	Low	Moderate	Moderate	High

Table 3.1.6.3.1 Blade Shape Comparison

By Authors

SOAR Blade Shape Selection

The blade shape of a propeller holds great weight on the flight performance of an aerial drone. These shapes can lead to variations in thrust output, efficiency, and noise. For the SOAR Project's mission objectives and goals, it is in the best interest of the team to use a blade shape that is efficient and could contribute to better thrust outputs if needed. The design must also be simple and easy to obtain. With these parameters in mind, the team has decided to either use a tapered or bullnose shaped blade. These blades offer excellent and thrust performance respectively. Depending on the power constraints or the weight of the vehicle will contribute to the final blade shape selection.

3.1.6.4 Propeller Configuration

Outside of the design of the propellers being used, the manner in which they are configured is also important to the flight of an aerial drone. The configurations of these propellers entail their angle (pitch), the directions in which they spin, and their housings. These design aspects directly

affect the key performance parameters previously mentioned, the most influential being thrust, stability, and efficiency.

Low Pitch

A low pitch propeller design implements a slight angle causing a lesser distance that a propeller moves in a revolution of its blades. This results in a lower angle of attack during flight and directly impacts the thrust output of the vehicle. With a lower angle of attack, the thrust is reduced but efficiency and stability at lower speeds is increased. A diminishing return in thrust will produce a slower drone that is more equipped to handle longer flight times and more stable operations.

High Pitch

Higher pitch propellers are the opposite of propellers with lower pitch. These propellers have a harsher angle and thus leading to a higher angle of attack during flight. This larger angle of attack leads to an increase in thrust output and allows for faster flight. This corresponds with higher speeds and for greater accelerations. However, higher pitching results in more noise and less efficiency overall.

Variable Pitch

Variable pitch propellers have the freedom to change their pitch depending on the flight conditions. The pitch can decrease in order to have a more stable flight at lower speeds for deployment or data capturing and the pitch can increase to generate more thrust and have a quicker and more responsive flight when needed [32]. A potential fault of this pitch method is that variable pitching adds complexity to the propeller design and could introduce implementation issues.

Design Aspect	Low Pitch	High Pitch	Variable Pitch
Speed	Moderate	High	Moderate
Efficiency	High	Low	Moderate
Thrust	Low	High	Variable
Stability	High	Low	Variable

Complexity	Low	Low	Extreme

Table 3.1.6.4.1 Propeller Blade Pitch Comparison

By Authors

Coaxial

Coaxial propeller configurations involve the stacking of propellers that rotate in opposite directions. This opposite rotation helps to alleviate torque and rotational forces that would typically need to be counteracted. Due to the drone frame selected being a quadcopter, the need for a coaxial system is diminished due to the quadcopter design naturally diminishing these rotational forces [33]. The coaxial systems also add complexity and are largely more inefficient due to the need of two motors; one motor for each propeller. This overall results in more thrust and less spin, but requires more power and it does not seem to correct a rotational issue for our frame.

Ducted

A ducted propeller system is one that implements a cage or chassis around the propeller. These ducts help to produce more thrust by eliminating effects that may inhibit thrust production like downwash [34]. Since there is more thrust produced with the same propeller when a duct is introduced, the addition of a duct can also be seen as more efficient [34]. The drone may be more heavy and bulky due to the addition of ducts around the propellers, but the trade off in increased thrust should be considered depending on application

SOAR Propeller Configuration Selection

The implementation in which the propellers are configured for a drone have substantial influence on the flight parameters of the vehicle. For the SOAR Project, the aim is to have a vehicle that is capable of reactive steady flight, while containing payloads for use in emergency response. This objective requires substantial thrust demands, stable flight for deployment and data capture, and efficient operating for 10-30 minute long missions. With these parameters in mind, the SOAR Project team has come to the conclusion to utilize a drone that implements a low pitch propeller that is uni-axial and contains a duct if needed. The low pitch propeller design offers stable low speed flight which would aid in data capture and communications and doesn't possess the complexities of a variable pitch propeller. The coaxial design was deemed to be too inefficient and complex for the scope of this project and was speculated to hinder the operation of the drone. Ducts could be implemented if more thrust is required for greater responsiveness and weight capabilities.

3.1.6.5 Summary of Propeller Design Components

Propulsion is required for all vehicles. All vehicles require a means to move in their desired direction. For the case of the SOAR Project, propellers are essential for the propulsion system of a rotating wing drone. Propellers convert mechanical torque into thrust in the form of lift in order to allow a drone to become airborne. With propellers there are several design parameters that

influence the flight of the vehicle. For the SOAR Project, a propeller that allows for the stable and efficient operation of the drone is what is most sought after. This propeller must also lack extraneous complexities that add to the difficulty of implementation. With the above parameters in mind, the team selects to consider propellers that have a three blade design with tip shape that is most effective for the mission. The propeller must also implement a fixed pitch design that also increases mission effectiveness. These propeller parameters have been shown to provide less complexity than other implementations like coaxial and variable-pitch propellers and provide the most stable flight possible while also considering thrust, drag, and efficiency.

3.1.6.6 Propeller Products

As the above research shows, the best fit for the drone appears to be a propeller with great efficiency, power, and stability. Durability is another factor that should be considered with the final selection. Price should also be considered for initial purchase and any replacements that must occur. For ease and simplicity, the following are pre-made propellers that are under consideration for use on the drones for the SOAR project.

HQProp 13x9x3 15-Inch X Class

The HQProp 13-Inch propeller is a 3-blade propeller (X class) that is 13 inches in diameter and has a pitch of 9 inches. This combination of specifications will prove to provide plenty of lift, and get the drone above the desired 2:1 lift to weight ratio. The propellers come in the configuration of two counter clockwise and 2 clockwise which is necessary for the stable flight of a quadcopter. The 3-blade design will provide more lift than a 2-blade design while still possessing mild efficiency. The HQProp is also constructed using fiberglass reinforced nylon which is more elastic and durable than carbon fiber. If the drone was to encounter a shock, the nylon would be less susceptible to failure. The price for a set of 4 of these propellers is $42.09.

SpeedyFPV 14x5.5 Carbon Fiber

The SpeedFPV 14x15.5 propeller is a 2-blade propeller that is 14 inches in diameter and has a pitch of 5.5 inches. This propeller configuration also provides the amount of lift for a 2:1 lift to weight ratio, which is essential for the proper flight of the drone. Like the previous propeller, both clockwise and counterclockwise propellers are provided. The 2-blade design offers less drag and therefore a reactful drone, however the stability and drag may suffer in comparison. It should also be known that the propeller is made of carbon fiber, which is not as durable as other discussed materials, but still will prove to be effective; especially with vibration mitigation. The price is the largest appeal for this propeller as a set of 4 of the SpeedFPV 14x5.5 is $19.99.

CB2 Pro 16x5.4 Carbon Fiber

Additionally, the CB2 Pro 16x5.4 is a 2-blade propeller that is very similar to the SpeedyFPV propeller discussed prior. The FLUXER has slightly less pitch (5.4 inches) but is slightly larger

in diameter (16 inches). This difference will show a subtle increase in stability and lift, especially in lower RPM flight. This propeller possesses a matt surface finish which will lead to less wind noise which helps improve efficiency and vibration mitigation. The material used is 12k carbon fiber which is a very strong and durable material, essential for shock mitigation. This propeller will provide less drag than a 3-blade propeller but due to the increased diameter, may prove to provide the lift necessary for flight. The price for a set of 2 of the CB2 Pro propellers is $49.00.

EOLO 15x5.5 Carbon Fiber Reinforced Nylon

Lastly, the EOLO 15x5.5 is a 2-blade propeller that is similar to the other 2-blade propellers mentioned but implements a carbon fiber and reinforced nylon design. The propellers claim to have a very efficient design, with reduced noise compared to other propellers. The propeller offers this with weight and durability in mind as well. This propeller is also very similar to the T-Motor propeller that was matched to the motor we selected. Finally, this propeller was used by the previous SOAR team and extra left over propellers were provided to be used. This propeller provides all the benefits of the 2-blade propeller design as previously mentioned but with additional familiarity. The price for a set of 2 of the EOLO 15x5.5 is $17.99.

Design Aspect	HQProp 13x9x3	SpeedyFPV 14x5.5	CB2 Pro 16x5.4	EOLO 15x5.5
Weight (each)	63.9	20g	18g	21g
Efficiency	Moderate	High	High	High
Lift	High	Moderate	Moderate	High
Drag	High	Low	Moderate	Low
Material	Nylon	Carbon Fiber	Carbon Fiber	Carbon Fiber & Nylon
Stability	Moderate	Moderate	Moderate	Moderate
Price	$42.09	$19.99	$49.00 x 2	$17.99 x 2

Table 3.1.6.4.1 Propeller Blade Pitch Comparison

By Authors

Propeller Product Selection

Of the propellers analyzed above, the EOLO appears to be the best fit for the SOAR Project. This propeller offers roughly 5kg of thrust per propeller which is plenty of thrust to offer a lift to weight ratio of 2:1. The propeller is a 2-blade propeller that offers high efficiency (low drag) and offers decent stability. The EOLO is made up of carbon fiber and nylon which are reliable materials that should withstand the vibrations and shocks that may occur. If excessive shock is to occur, the propeller's low price makes it less costly to replace.. Lastly, the EOLO was used in a previous iteration of the SOAR Project, which provides familiarity and comfort with the component in addition to already possessing extra EOLO propellers from the previous project team.

3.1.8 Manufacturing Techniques

Manufacturing techniques are essential for the SOAR project. The drone frames will need to be manufactured and built to house all the components required for SOAR. It is important to understand different manufacturing technologies and techniques commonly used for autonomous drone building. This section aims to discuss different types of manufacturing techniques that exist for drone manufacturing.

The SOAR drone swarm will need to be reliable and reusable. The manufacturing technique used should be able to print/design/build the drone to be sturdy, reusable and limit any non essential waste.

3.1.8.1 Additive manufacturing (3D printing)

Additive manufacturing is a modern fabrication method that produces components by building them layer-by-layer, rather than removing material as in traditional subtractive processes. This approach allows for the direct creation of complex geometries that would be difficult or impossible to achieve using conventional machining techniques. Among the most recognizable forms of additive manufacturing is 3D printing, which has rapidly grown in popularity in recent years for professional development.

One of the major advantages of additive manufacturing is its efficiency in material usage. Since parts are built directly from digital models with precise deposition of material, the process generates minimal waste compared to alternative methods. This makes additive manufacturing an environmentally conscious choice, limiting bi-product. Another key benefit is the ability to produce lightweight and strong structures, which is valuable in fields where weight reduction leads to significant performance gains, such as aerospace engineering and biomedical uses [36].

There are many different techniques under the umbrella of additive manufacturing, each with its own strengths, limitations, and applications. This research will focus on some of the most

common and widely used methods, including Material Extrusion, Material Jetting, Binder Jetting, Powder Bed Fusion, and Vat Photopolymerization [37]. By comparing these processes, the goal is to highlight their respective capabilities and limitations to provide a clearer understanding of how each technique can support the objectives of SOAR.

Material Extrusion (FDM - Fused Deposition Modeling)

Fused Deposition Modeling as an advanced method of additive manufacturing. It uses a Computer Aided Design file and performs a melt-extrusion of plastic filament, completing a layer-by-layer operation. This method is done by filament being delivered from a coil and melts as it passes through the extruder, where melting point is reached[38]. This allows for precise layer by layer manufacturing. Some of the common filament types for Material Extrusion include PLA, PVA, ABS and TPE. FDM has limitations for material types, since this form of manufacturing can only be done with polymer based materials.

Material Jetting

Material Jetting is another form of additive manufacturing that selectively cures liquid photopolymers to build parts and components. It is a common additive manufacturing tool for polymer printing that can create both matte and glossy finishes. The printing process solidifies when hit by UV light. This process can be related to inkjet printing due to the similarities with regular paper printing [39]. Material Jetting, in comparison to other techniques, is not able to withstand as much force when a component is completed. However, honeycomb and varying stiffness levels can be achieved using this form of additive manufacturing.

Binder Jetting

This form of additive manufacturing works by dispensing a liquid binding agent on powder to form a 2D pattern onto a layer. These layers are stacked to create the 3D model. Binder Jetting was developed in 1993 from MIT, which promoted working with multiple powdered feedstock. By using the jet with the binder, better detail can be achieved when creating the layer by layer. With the use of the liquid agent, a wide variety of materials can be used, including metals, polymers, sands, ceramics, and more [40]. Binder Jetting is most popular for designs of multiple feet, but are able to print on a small scale.

Powder Bed Fusion

Powder Bed Fusion uses lasers or electron beams to fuse powder bed areas. This type of additive manufacturing is extremely popular in biomedical applications as it is known for its precision and accuracy in comparison to other additive manufacturing techniques. Using Powder Bed Fusion allows for strength and durability for products. There are two main types of techniques for Powder Bed Fusion; Selective Laser Sintering (SLS) and Electron Beam Melting (EBM).

Both can use metals, polymers, ceramics and composites to create these products. PBF can create complex geometric shapes with high precision, but often has long printing time and scalability issues [41].

Vat Photopolymerizations

This method of additive manufacturing is a popular technology for low production volume projects. It is a type of 3D manufacturing technique that uses light to cure photocurable resin. The part is "grown" on a platform in this process. This method of manufacturing has many variables that are considered based on print time and material [42]. The practice calls for very intricate manipulation of how and when things will cure or harden.

Based on research and scope of the SOAR project, material extrusion is the best additive manufacturing technique if selected for designing the frames of the drone. Its general endurance and reasonable cost gives the team the highest certainty that the drone will be fully functional for the project goals. If this project were to continue on a larger scale, binder and material jetting would allow for higher quality drone frames. These methods would increase budget and material needs, and is outside the scope of the senior design project. The table below shows the overall comparison of the different additive manufacturing techniques [43].

Comparison	Material Extrusion	Material Jetting	Binder Jetting	Powder Bed Fusion	Vat Photo.
Typical Materials	Thermoplastics (ABS, PLA, Carbon-Fiber)	Photopolymers, composites	Metals and Ceramics	Polymers, metals	Photopolymer resins
Strengths	Low Cost, Accessible, Easy to Scale	High Precision, multi-material prints	Large and complex parts, lower cost	High strength and durability	High Accuracy, smooth surface finish
Limitations	Finish often rough, slow print time	Materials often brittle, high cost	Requires post-processing, parts can be weaker	Expensive, support removal and finishing	Resins are often brittle
Drone Applications	Lightweight housings	Detailed housings, covers	Metal brackets, lightweight	Propeller components	Small detailed components

			non critical components		and sensor housing

Table 3.1.8.1 Performance Parameters Comparison Between Additive Manufacturing Techniques

By Authors

Based on the research done, measures can be put in place to increase the success rate of material extrusion, by adding supports, vibration damping and more to have a successful drone swarm.

3.1.8.2 CNC Machining

CNC Machining is a different manufacturing method known for its high precision, repeatability and automation. These are essential for designing and making engineering parts with tight tolerances and complex geometries. In engineering, CNC machining is used for prototyping, component fabrication, finishing, and sometimes for small batches or custom components [43].

There are many advantages and disadvantages for CNC machining. CNC machining is ideal due to its ability to work with a wide variety of materials, such as metals, plastics and composites. It has the ability to prototype and integrate with CAD systems for seamless design-to-manufacture pipeline. However, tool wear, heat generation and material removal limits make machining complex components with CNC a challenge. This research discusses the different types of common CNC machining practices [44].

2.5 Axis Machining

2.5 Axis Machining is ideal for creating prismatic parts and sheet components. The system moves in the X and Y direction, with limited Z-axis. As a result, 2.5 axis machining is effective for operations involving slots, holes and flat profiles, but is not capable of cutting at extreme angles. 2.5 axis machining can be used for electronic enclosures, automotive brackets, medical device houses and simple aerospace tooling applications. The product is low cost and fast operations for suitable geometries, but is unable to smooth contoured surfaces or perform more complex machining as mentioned. However, the limitations should be noted, 2.5 axis machining cannot easily produce smooth 3D contours or perform sophisticated machining required for complex geometries [44]. For projects demanding higher precision on curved surfaces or intricate shapes, 3-axis or 5-axis machining techniques are typically preferred.

3 Axis Machining

This type of CNC machining allows the cutting tools to move in the X, Y and Z directions at once, enabling more complex geometries and contoured surfaces than in the 2.5 axis machining. This enables the machine to create more intricate parts and curved surfaces. This machining technique requires more complex programming skills and is more expensive than 2.5 Axis or additive manufacturing techniques. Feed rates are generally slow to simpler machining methods,

which can extend production time for products [45]. Despite additional printing time, 3 Axis machining is a commonly used manufacturing technique that acts as a solution for industries needing high precision and the ability to produce complex geometric shapes.

4 Axis Machining

4 Axis Machining builds off of 3 Axis machining techniques, with an additional rotating axis. This additional axis is typically in the A-axis, which enables an additional degree of motion to increase the amount of complexity in each component. This makes creating cut-outs, holes and features on the sides of parts more accessible and able to be machined without moving the structure. Both approaches improve flexibility and efficiency, with simultaneous machining offering greater precision for complex geometries [45]. By reducing the need for multiple setups, 4 axis machining saves time, improves accuracy, and expands the range of components that can be produced compared to standard 3 axis systems.

5 Axis Machining

5 Axis CNC Machining is the most advanced and versatile form of CNC milling, offering simultaneous movement along the X, Y, and Z axes while incorporating two additional rotational axes. These extra degrees of freedom allow the cutting tool to approach the workpiece from virtually any direction, enabling the machining of highly complex geometries in a single setup. This capability makes 5 axis machining ideal for industries such as aerospace, automotive, and medical, where intricate shapes, undercuts, and tight tolerances are required. While the equipment and programming are significantly more costly and complex compared to lower-axis systems, the process reduces the need for multiple fixtures, minimizes errors from repositioning, and shortens overall production time. As a result, 5 axis machining is the preferred solution for producing precision components with demanding geometries and surface quality requirements [45].

CNC Routers

CNC Routers are designed for cutting, carving and shaping materials with exceptional precision. These machines can handle intricate designs and shapes. CNC routers are known for their precision and accuracy, specifically for aerospace materials. High tolerance levels are required and can achieve tolerances within thousandths of inch. In addition, most machines have integrated quality assurance that monitors production accuracy [46]. These machines are able to adapt to high volumes while maintaining precision and enabling facilities to streamline operations.

CNC Lathes

This manufacturing technique is ideal for high precision cuts on cylindrical components. Lathes are often used for landing gear components, engine parts and any small aerospace components. This type of machining is not applicable for SOAR and will not be considered for design [47].

CNC Laser Cutters

Laser Cutters can be used on more than metal. Laser cutters can generate any desired shape in a singular setup. Laser cutters are ideal due to simplicity in programming. Laser cutting eliminates costs for sharpening and replacing punches and maintaining the edge of a shear [48]. This process is very expensive compared to other machining techniques or additive manufacturing.

CNC Plasma Cutters

Plasma cutters were invited to cut through very thick pieces of metal. These applications of this advanced technology are outside the scope of the SOAR project. If this project were to increase in scale and be needed for very large quantities, plasma cutting would be a great alternative to other machining methods [49]. Axis methods are more practical and easier for projects of this size.

3.1.8.3 Injection Moulding

Injection molding is a popular method of manufacturing that involves injecting melted material into a pre-built frame. The following research discusses different types of injection methods common for developing aerospace materials.

Standard Injection Molding

Standard injection molding is a high volume manufacturing process known for developing structures and products at a high rate. It works by using a mold to inject melted material into it and cooling it. The most common injection material is polymer pellets. The high pressure used to force the material into the mold allows it to create identical parts each time. Injection molding is preferred due to high volume production, precision, and consistency. For SOAR, this method would be great to create four uniform drone frames or propellers if opting to design and build propellers instead of buying them.

Gas-Assisted Injection Molding

This injection method limits common injection molding issues with distorted parts caused during the normal cooling process of the plastic. Nitrogen, the common gas to be used in this process, is introduced into the gas channels, producing smoother surfaces and limiting deformations. The gaps being natural with the gas injection, the structure strength is maximized, and not affected in a negative way [50]. This method would promote overall smoothness to the SOAR swarm drones.

Over Molding

Over molding allows for an injection method to be integrated with two different material types. As the first material is cooling, the second mold is added to create the strongest bonds possible. This minimizes secondary operations often needed during injection molding. This second component is often rubber, to add to the integrity of the part. A common example of this is

electronic device grips. The over molding makes it more usable for the consumer, but also adds the additional level of structural support.

Multi-Material Injection Molding

Multi-material injection molding can be described as two separate materials being mixed together prior to injection molding. These materials are often opposite in qualities, and are combined to create very complex parts with specific requirements. Mixing rigid and flexible material, a common occurrence in this type of injection molding, allows for the enhancement of the finished project, having qualities from both types of materials. For drones, this method would be useful for creating a strong frame structure [50].

Micro Injection Molding

Similar to standard material injection molding, micro injection molding works by injecting a heated material into a premade mold. In this case, the material is typically 0.1 to 1 gram of plastic part, being heated into the mold. This method can create the smallest of parts with extreme accuracy. This method does not fit the scope of SOAR, since the drone swarm will be on a larger scale.

3.1.8.4 Summary of Manufacturing Techniques and Selection

Additive manufacturing, CNC machining and Injection molding are all common manufacturing techniques used in the professional industry. The vast usability and diversity of production enhances opportunities for engineering solutions. SOAR has requirements to be met for the success of the project. In the mechanical aspect of the drones, the drone frames need to be durable, cost effective and be able to withstand the force of flight for operational testing. The table looks at the overall effectiveness of each of the manufacturing techniques with drone design in mind.

Design Aspect	Additive Manufacturing	CNC Machining	Injection Molding
Cost	Low	High	Moderate-High
Accessibility	High	Low	Low
Material Range	Moderate-High	High	Moderate
Precision	Moderate	High	High
Environmental Impact	Low-Moderate	High	Moderate

Surface Finish	Moderate	High	Moderate

Table 3.1.8.4 Manufacturing Technique Comparison

By Authors

After researching the different types of manufacturing techniques, there are clear advantages and challenges to each. Additive manufacturing is an accessible tool that can be easily obtained for the SOAR project. Additive manufacturing techniques are cheaper and still can't print at high efficiency and detail. The surface finish of these printed parts are not the smoothest, and additional operations would have to be done post print. CNC Machining technology is advanced and widely used in industry. For SOAR, CNC machining is expensive, and must be done with at least 3 axis machining. Although CNC machining can be done with extreme precision, there is still a significant amount of environmental impact, or unused material, that is wasted during the project. Due to these factors, CNC machining would not be ideal for this project. If this project were to continue at a larger scale, CNC machining techniques would be extremely useful. Similarly, injection molding has a significantly higher precision compared to additive manufacturing and CNC machining. The cost and lack of access to these machines for SOAR, has limited the team to choose additive manufacturing and the manufacturing technique for this project. The next section discusses common materials used for additive manufacturing and what is best for the SOAR team.

3.1.9 Material Selection

Selecting the best material is a critical step in the design and fabrication of search and rescue drones, as it directly impacts performance, reliability, and mission success [51]. These drones must be capable of carrying out flights lasting up to 20 minutes while maintaining stability to ensure sensors and cameras can collect accurate data in challenging environments. Material choice influences not only the structural integrity and weight of the drone but also its durability, fatigue resistance, and overall cost-effectiveness [51]. By carefully evaluating candidate materials such as aluminum alloys, carbon fiber composites, polymers, and advanced options like graphene, this research aims to identify solutions that balance strength, lightweight performance, and affordability. Optimizing material selection is essential for creating drones that are both efficient and dependable for the team's SOAR project scope.

Overview of Materials

Additive manufacturing offers a unique way to design and print drones. Due to the wide range of material options, 3D printing gives the user an expanded choice when determining what material is good to print. This versatility enables engineers to tailor material selection to meet specific performance needs, such as lightweight structures for longer flight times, high-strength alloys for load-bearing components, or flexible polymers for shock absorption. This research will focus on materials that could positively impact the success of SOAR.

Aluminum Alloys (6061 and 7075)

Aluminum alloys have become increasingly popular in additive manufacturing, offering a lightweight yet strong option for structural applications. While aluminum has been used in traditional machining, 3D printing introduces new advantages by enabling design flexibility and the creation of complex geometries that would be difficult to achieve with traditional machining. Features such as internal channels, hollow sections, and intricate internal structures can now be printed, enhancing both performance and efficiency. However, challenges remain, particularly in managing thermal conditions during printing. These challenges are mitigated with post process operations. Among the many aluminum alloys available, 6061 and 7075 are two of the most widely used due to their balance of strength, durability, and workability [51]. This study compares these alloys to determine which is most suitable for the SOAR project's drone applications.

Aluminum 6061 is one of the most widely used alloys due to its excellent balance of strength, weight, and manufacturability. As an aluminum–magnesium–silicon alloy, its classification reflects its composition and properties. 6061 is especially valued for its weldability, corrosion resistance, and cost-effectiveness, making it suitable for a wide range of structural applications. In the context of additive manufacturing, 6061 can be effectively processed with current 3D printing technologies, allowing for the production of lightweight, durable components with minimal post-processing [51].

Aluminum 7075 is a high-strength aluminum–zinc alloy commonly used in aerospace and high-performance UAV applications where maximum structural integrity is required. Known for its exceptional tensile strength and stiffness, 7075 provides superior load-bearing capabilities compared to most other aluminum alloys. Advances in 3D printing technology have made it possible to fabricate components from 7075 with careful thermal management and post-processing [51]. Its combination of strength and lightweight characteristics makes 7075 an attractive option for drones that require high-performance structural elements.

Carbon Fiber Composites

Carbon fiber composites are one of the most popular of the 3D printed filaments for aerospace engineering applications. It began growing in popularity in 1960 when NASA began exploring the use of carbon composites for aerospace applications. Carbon fiber composites offer a unique combination of high strength, stiffness and reduced density. Optimizing weight on any device with flight applications is ideal, and printing/utilizing carbon fiber composites allows that to happen. When carbon fibers are utilized, extensive testing is required to ensure its durability in flight operations. These tests are run in the form of FEA analysis, where factors such as frequency, stress distribution and failure modes are analyzed to predict the composite performance [52]. Lockheed Martin has utilized carbon fiber composites before, specifically in the F-35 Lighting II. The wings are designed with carbon fiber composites to decrease weight and maximize stealth. Carbon fibers are excellent for when metal is preferred, but not feasible.

Polymers (Nylon, ABS)

Polymer printing is common in many University labs. They are relatively inexpensive and can print a variety of structures, designs and drawings. Nylon and Acrylonitrile Butadiene Styrene (ABS) are the most popular polymers for injection molding. This project will not utilize injection molding technology, but if SOAR were to be produced on a larger scale, injection molding with polymers would be beneficial for creating the housing and guards for the electronics.

Graphene/Graphene Composites

Graphene is low mass and has high thermal conductivity. Technology for 3D printing graphene is very new and experimental at this time. Graphene would be a great composite for drones due to its ultra-lightweight and high-strength components such as propellers. Graphene has significantly higher cost than the other materials discussed in this research, but should be studied further if applications were to arise larger than the scope of the SOAR project [53].

3.1.9.1 Summary and Material Selection

A variety of materials for 3D printing exist and have advantages and disadvantages. These materials can be used in a variety of applications, and would contribute to the success of the SOAR project. Based on the table and summary, it has been determined that the team will use carbon fiber composites to 3D print the drone frame. Other materials, such as wood or polymers may be used to stabilize the housing of the drone frames. Initial CAD drawings and analysis are discussed later in this paper.

Material	6061	7075	Carbon Fiber	Nylon	ABS	Graphene
Density	2.70 g/cm^3	2.81 g/cm^3	1.4-1.8 g/cm^3	1.1 g/cm^3	1.2 g/cm^3	0.77 g/cm^3
Tensile Strength	310 MPa (T6 Temper)	572 MPa (T6 Temper)	>350 MPa	60 MPa	34 MPa	Dependent
Elastic Modulus	69 GPa	71.7 GPa	Dependent on fiber fraction	1-4 GPa	2.1-2.27 GPa	1 TPa
Fatigue	500 mil cycles	500 mil cycles	Depends on quality	Poor	Poor	Unknown
Relative Cost	Moderate	High	Moderate-High	Moderate	Moderate	High

Table 3.1.9.1.1 Material Comparison

By Authors

3.1.10 Landing Gear

After the objective has been completed, the drone must return to a landing zone safely in order to maintain function and allow for repeat operation and use. Landing is a harsh task, as the drone is trying to halt all motion and come to a complete stop at a desired location while keeping the sensitive components on board intact. To land safely and intact, the drone must have an implementation of specialized landing procedures to retain its structural integrity from the impact of landing. The most common means to uphold the integrity of the vehicle is to include landing gear that helps the drone stop its flight in a safe and effective manner.

For the SOAR mission, the drone may encounter harsh or damaged environments and must be able to take-off and land from these rugged terrains. In order to do so, various landing gear technologies must be considered with a primary focus on a lightweight design and stability in order to take-off and land safely and effectively from various surfaces while not hindering the flight and efficiency of the vehicle. The following are key performance parameters for the landing gear of the drone.

3.1.10.1 Key Performance Parameters

Weight

Having increased weight for the vehicle requires more thrust in order to maintain maneuverability and flight operation. With this in mind, it is best to keep the weight of these components as low as possible to aid the drones with their operation by requiring less thrust to execute the mission.

Drag

Like weight, drag is another parameter that directly impacts the aerodynamic efficiency of the vehicle. With more drag, more thrust is required to maintain a desired speed of the drone which leads to demands in other components. With drag being a factor, it is best to keep this parameter minimized to require less from other components.

Stability

When considering landing gear, stability is essential to provide a safe and effective landing when the drone comes in contact with the ground. A more stable landing procedure keeps the drone

upright and unsusceptible to tipping. This contributes to the sensitive components of the vehicle remaining intact and ready for further operation.

Shock Absorption

The main objective of landing gear is to keep the sensitive and internal parts safe and protect them from any of the impact from coming in contact with the landing site. This is attained through shock absorption found in the landing gear and thus it is desired to implement a landing technology that is effective at absorbing this impact.

Complexity

Some landing gear may be more effective at the previous performance parameters but have a drastic increase in difficulty to implement. Due to the nature of the project and its time restraints, a simple design is sought after to ease manufacturing and testing.

3.1.10.2 Landing Gear Types

Skid-Type

Skid-type landing gear implements two skids that are parallel to one another that help distribute the load of the aircraft across the two skids. This type of landing gear is a fixed type where the skids do not move and are fixed to their position throughout the take-off, flight, and landing of the vehicle. These skids are attached to the underside of the drone frame. This attachment style and the parallel nature allows for an equal weight distribution for the drone body. This helps to contribute to the safe and effective landing on various surfaces and good shock absorption depending on the material. However, due to the size of the skids, it is often found that they are heavier and cause more drag on the vehicle. Also, skid-type landing gear is often attached by an arm between the skid and the drone frame. During landing, it can be seen that the weight is distributed along the skid but where the arm and skid attach is a point of high stress which may lead to failure [35]. An example of skid-type landing gear being employed can be seen in helicopters.

Leg-Type

Another fixed-type landing gear is leg-type. Leg-type landing gear has a rod or “leg” below each motor/propeller which helps to support the drone in take-off and landing functions. These legs implement a very simple design that eases the creation and potential repair of the rods involved. The slender nature of these legs contribute to a lightweight and drag efficient design that would minimally impact the flight of the drone. However, these legs distribute the weight of the drone solely through the end of the rod that encounters the landing site. This leads to a potentially

unstable landing in difficult terrain and a more impactful landing that may not be as effective in protecting drone components.

Flat Plate/Pad-Type

Attaching a flat plate to the bottom of the drone is another technology used for landing gear. The landing pad is traditionally attached to the underside of the drone and allows for a more efficient flight due to the lack of framing. This lack of framework also requires less material which contributes to a lighter and more compact design. These pads are great for flat even terrains but lack the stability and absorption to land on uneven or hazardous terrain.

Retractable

Landing gear that is capable of being deployed and undeployed during take-off and lift is known as retractable landing gear. This is often seen in planes how they are able to store their landing equipment when approaching flight and then redeploying the gear when beginning to land. This process allows for a minimalization of drag due to the landing equipment being stored away, where it can't increase the drag of the drone. Also, with the gear being stored away, this clears the area underneath the drone for any of the sensors necessary for the SOAR mission. In creating this type of landing gear, the weight of the mechanical components that move and enclose the landing gear is a factor along with the complexity of implementing a design that is capable of being deployed in and out of use.

Inflatable

Inflatable landing gear employs an airbag like structure to act as a cushion that protects the sensitive materials on the vehicle from landing. In this landing gear type, a cushion is either fixed to the bottom of the drone or deployed when landing is desired. These airbags would add minimal weight to the vehicle due to the materials being used. The inflatable design has great shock absorption due to the equal impact distribution throughout the airbag [35], but due to the fluid nature of the contents within the bag, it will lead to a lack of stability in landing which can contribute to tipping or bouncing. The inflatable will also lack the durability of a rigid design due to most inflatable devices being susceptible to tears or leaks which would impact effectiveness. This design, whether fixed or deployable, also adds complexity to the drone design which may make manufacturing, reliability, and repeatability more difficult.

Wheel Based

A landing gear design that uses wheels for take-off and landing is also in use. This technology requires the use of wheels that absorb the shock of landing and allow for a rolling take-off/landing. For the scope of this project, the drones being designed for SOAR are designed

with vertical take-off and landing in mind (VTOL) and thus landing gear implementing wheels is out of the scope for this project.

Key Performance Parameter	Skid-Type	Leg-Type	Flat Plate/Pad-Type	Retractable	Inflateable
Weight	Moderate	Low	Moderate-Low	High	Minimal
Drag	Moderate	Low	Low	Minimal	Various
Stability	Moderate	Moderate	Low	Moderate	Low
Shock Absorption	Moderate	Low	Low	Various	High
Complexity	Low	Low	Low	Maximal	High

Table 3.1.10.2.1 Landing Gear Comparison

By Authors

3.1.10.2 Landing Gear Summary & Selection

Many technologies exist for landing gear on a drone. These technologies have different implementations that attempt to keep the sensors and technologies inside the drone safe. Each implementation has different advantages and disadvantages in the performance parameters discussed throughout this section that contribute to the objectives of the SOAR Project. It is sought to have a landing system that is lightweight, stable, aerodynamically efficient, able to absorb impact, and simple to implement. With the previous specifications mentioned, the SOAR Project team has elected to use a modified leg-type landing gear. This landing gear provides decent stability and lacks complexity. It also is lightweight and has little effect on drag. The only downside of this landing gear is its shock absorption. This will be corrected using a modified end that is cushioned to provide less of an impact during landing. Initial CAD drawings and analysis of this design can be found later in this report.

3.1.11 Vibration and Shock Mitigation

Vibration and shock refer to the constant shaking or repeated oscillatory movement of an object (vibration) and the large sudden or unsuspected forces encountered (shock). In the case of

vibration, this shaking causes a wearing of the different components that encounter this effect. It can be caused by any rotating parts found in the system. Reducing vibration allows for a more reliable and sustainable system. In the case of shock, this unexpected impact can lead to a sudden failure in the components it may encounter. Shock is caused by any unpredicted or accounted for force that strikes the system. Eliminating shock can help protect the system from failure due to these infrequent impacts. For drones, especially those with sensitive hardware or equipment, these two effects must be mitigated as much as possible in order to ensure the proper use and performance of these sensitive components.

With the mission and objectives of the SOAR Project considered, it is in the best interest of the project to reduce and eliminate vibration and shock as much as possible in order to ensure the operation and longevity of the drones. If these effects are left unchecked, it could lead to issues in sensor reading which would impact the ability of the drones to record accurate information and their ability to relay this information to one another. It could also lead to the long term and sudden breakdown of the components on the drone. In order to minimize these effects, the following sections provide insight on both vibration and shock; including their impacts and methods that contribute to their elimination.

3.1.11.1 Vibration in Drones

Sources of Vibrations

In order for rotating wing drones to take flight, they require the spinning of propellers driven by motors to generate lift. Unfortunately, this rotating motion directly contributes to vibration across the entire drone. This can be caused from mechanical, material, and environmental factors. Any imbalances in the spinning of the propellers or motors can cause vibration to occur. Also propeller damage or imperfections affecting the spin can contribute to vibration. Different drone frame materials can resonate at the frequency in which the motors and propellers oscillate which cause a greater vibrational effect. Lastly, wind or other environmental factors can contribute to vibration. These various causes of vibration should be considered when attempting to mitigate these vibrational effects. The following sub-sections present different methods for minimizing vibration in rotating wing drones.

Propeller and Motor Balancing and Maintenance

As previously mentioned, vibrational movement can be caused by an imbalance in the propeller or the motor. A propeller is deemed imbalanced when there is a tilt in the propeller. Rather than spinning flat, there is a tilt on the propeller which causes it to spin at an off angle respective to the horizontal. When in motion, the tilt causes one side to be up while the other is down and in 180 degree phase, the opposite is true. This causes an oscillation and thus a vibration throughout the drone [54]. This tilt is often a result of a weight difference between propeller blades. This imbalance/tilt can be caused by improper mounting, manufacturing defects, or even damage

caused from crashes or other impacts. Propellers and motors can be manually adjusted to make sure they are balanced; this can include adding weight to a blade or fixing the mount of a propeller or motor. Before and after operation, the propellers and motors should be checked for any imbalances (damage or defects) to ensure as little vibrational effects as possible.

Frame Material and Design

In order to counter vibrational effects, a material may be selected for the frame of the drone that is more rigid thus resonating at a frequency differing from that of the frequency of the vibration caused by the propellers and motors. The stiffer the material the higher the resonance frequency and due to the low frequency of the vibrational energy caused by flight [55] the less vibrational effect on the drone frame. There is a return on the stiffness of the material as if it is too stiff, it will become more susceptible to shock and the stresses on the frame that it presents. The weight also is a factor as the weight directly affects the flight parameters of the drone. The solution is to find a material that is moderate in both stiffness and ductility along with being lightweight as well. These factors have led to the decision of using carbon fiber as the material of choice for the drone frame.

The structural design of the drone also impacts its influence to vibrational effects. Propeller arm thickness and length influence the bending stiffness of each arm which impacts their susceptibility to vibration. Due to this, arms that are short and thick should be considered if further vibrational mitigation is required. Also, symmetry should be considered to have equal weight distribution which will further help to reduce vibration across the entire drone body.

Vibration Damping Connections and Isolation

When installing sensors or other sensitive hardware, the method in which they are mounted to the drone frame can influence their susceptibility to vibration. In order to reduce the effect of vibration on these components as much as possible, material known to absorb these vibrational effects should be installed between the sensor and the frame. This method can be further employed through all connections of the drone which will lead to less interference from vibration across the entire drone. These connections can also implement plates or other means to distribute the connectional stress which would be less affected by vibration as well. A similar practice can be implemented with the mounting of the motors to the frame. Dampening materials can be introduced between the motor and the frame which would lead to further mitigation in vibrational effects across the drone frame, due to the difficulty of these vibrations to translate across the damped medium.

Filtering and RPM Ranges

Vibration mitigation can also occur on the electronics side. Filters can be employed to reduce the magnitude of the frequencies in which the propellers and motors spin at. These filters could help

clean sensing and transmission since the vibrational frequencies would be eliminated from the data. Additionally, harmful RPM ranges of the propellers and motors can be avoided to not encounter the resonance frequency of the drone frame.

Vibration Summary

Vibrational effects have various negative effects on drones. These effects cause a difficulty in sensor reading and communication and cause a degradation of structural components on the vehicle. In order to prevent these consequences, various technologies can be implemented that help counteract the vibrational energy produced by propeller and motor motion. Propeller balancing and damping connections and mounts between sensors and motors can help protect these on board components from being affected by these vibrational effects. Materials used in the drone body also have an impact on the potency of vibration; it is desired to use stiff and rigid materials. Filtering can also be used on the digital side to further reduce the electrical presence of these effects. These methods, working with one another, can lead to a more reliable system which improves flight stability and operation.

3.1.11.2 Shock in Drones

Sources of Shock

Shock is the sudden and non-repetitive forces a system may encounter during operation. These forces are high in magnitude and can contribute to immediate failures in the structure or components of the drone. These unexpected forces can be caused by impacts with surroundings, improper landing, or from environmental factors like wind gusts. When attempting to mitigate shock, there should be an attempt to neutralize these forces before they can cause a structural or electronic/sensing failure. The following sub-sections present different methods for minimizing shock in drones.

Shock Absorbing Material

Like for vibration, shock mitigation is also determined by the materials that make up the frame and other components. Materials that are able to elastically deform are materials that are capable of returning to their original shape after deformation has occurred. During shock, a large force is applied to the system and this may cause a deformation. For example, If the drone body is capable of elastic deformation, then after the shock has occurred, the drone body will return to its normal operating shape prior to the deformation. In order to counteract the previous mentioned vibrational effects, the drone body has been selected to be made out of carbon fiber. However, due to carbon fibers' brittle nature, it is susceptible to shock and failure from it. A solution for this has the landing gear and other drone parts that may hit the drone body being made up of materials that are able to absorb and dissipate the shock. This dissipation allows for the energy from the shock to be handled through the ductile material, rather than transferring to the rest of

the drone body and causing a failure. The landing gear can also utilize a combination of rigid materials for structure and stability and ductile materials for shock mitigation.

Shock Mitigating Connections and Mounts

To combat vibrational effects, soft, energy absorbing materials were called for. Similarly, these same materials are used to mitigate shock. These energy absorbing materials also dissipate the sudden large forces that may be encountered during drone operation. To help neutralize shock, these materials are put in between sensitive components and the drone. The materials can be placed between the different sensors or electrical components essential for proper drone operation and communication. These mediums allow for less effects from shock and provide a more reliable and stable drone system. Connections can also be wider and spread out rather than localized to distribute shock across a wider area. Stress has larger and more devastating effects on pinpoint areas where it can not spread the load out. Making the connection area larger, it allows for further distribution of these sudden loads thus resulting in further shock mitigation; especially when paired with energy absorbing mediums in between connections or mounts.

It should be noted that some of these energy absorbing materials are susceptible to the environment. Some are factored by temperature or humidity. In most cases, when it is cold, materials tend to stiffen and become brittle. Rubbers and other materials used as mediums are no exception. If they were to become hot from the vibrational/shock mitigation or external temperature, they would then bend or deform too much, also hindering their ability to mitigate these effects. Mediums should be made up of various materials that are capable of providing energy absorbing effects for a range of operating temperatures to ensure proper mitigation.

Active Methods

Some drones implement more advanced technology that is able to consciously mitigate shock. One such method employs the use of sensors to detect the drones distance from the ground and has a burst of thrust right before impact to slow the speed in which the drone lands. Another method uses servos to adjust the stiffness of landing gear depending on the terrain to mitigate the impact of landing. These methods are very complex and are difficult to implement effectively.

Shock Summary

Shock can cause an immediate and critical failure in the structure or components of a drone. When encountering shock, the drone must dissipate this sudden load effectively in order to not break or cause damage to any of the on board electronics or structural components. To dissipate or mitigate this load, shock absorbing material and connections can be used along with more advanced methods to help reduce the load and stress caused by shock. This will result in a drone with higher reliability that is more capable of operation where sudden and unexpected forces may be encountered.

3.1.11.3 Vibration and Shock Mitigation Summary

To ensure the proper flight of a drone, vibration and shock mitigation must be considered. Left unchecked, sensing and structure may critically suffer thus leading to failures in both communication and mechanics. To avoid this, balanced propellers and motors along with material selection and damping connections can help eliminate these vibrational effects. Further material selection in landing components and damping connections can also eliminate shock effects. Filters and active methods can both be employed if further vibrational and shock mitigation is required. By implementing these technologies, drones are able to better sense their surroundings and are less susceptible to damage when unexpected interactions arise. This leads to a more reliable drone, with the ability to encounter difficulties when in use in the field.

3.2 Electronic Hardware

The electronic hardware of the drones integrates all critical systems needed for autonomous flight and data collection. Each drone includes a Pixhawk 4 flight controller for stability and navigation, powered by a high-capacity Li-Po battery. An onboard companion computer, such as a Jetson Nano or Raspberry Pi 4, processes sensor data from cameras, LiDAR, and ultrasonic modules in real time. These components communicate through MAVLink and mesh networking modules to enable coordination within the swarm. Together, the electronic hardware ensures reliable control, efficient power management, and seamless data exchange for autonomous search-and-rescue operations.

3.2.1 Flight Controller Technologies

The flight controller is the central brain of a drone, functioning much like a computer's CPU by processing inputs from sensors and translating them into precise motor commands. Without it, the drone would have no means of flight or maneuverability. It manages all three dimensions of motion (i.e roll, pitch, and yaw) as well as altitude, by constantly adjusting rotor thrust to stabilize and direct movement along the x, y, and z axes. Modern flight controllers use ARM-based chips (such as F745, F765, or H7 series) and rely on sensor fusion, combining data from the inertial measurement unit (accelerometer and gyroscope), magnetometer, barometer, GPS, and sometimes ultrasonic sensors to create a full picture of the drone's position and orientation. This integration allows the drone not only to hover steadily and respond to pilot inputs but also to perform autonomous tasks like waypoint navigation, altitude hold, and precision landing. In essence, the flight controller transforms raw sensor data into coordinated motion, making it an indispensable component for stable and intelligent drone operations. Below are descriptions of each component in a flight controller:

Inertial Measurement Unit (IMU)

An Inertial Measurement Unit is a critical sensor module within a flight controller that allows a drone to interpret and maintain its orientation, stability, and motion in three-dimensional space. Using sensor fusion, an IMU combines the telemetry from accelerometers, gyroscopes, and sometimes magnetometers to measure linear acceleration, angular velocity, and heading relative to the Earth's magnetic field. The flight controller continuously processes IMU data to calculate the drone's altitude (more specifically its roll, pitch and yaw) and to issue precise motor adjustments that counteract disturbances such as wind or sudden movement. The closed-loop feedback allows the drone to steadily hover, execute flight maneuvers, and maintain control in dynamic environments. Without an IMU, the flight controller would have no manner of sensing or correcting changes in orientation, resulting in loss of control and stability. Advanced flight controllers come with two or more IMUs so that they can be secure if one fails, this is known as IMU redundancy.

Accelerometer

An accelerometer is a sensor that measures linear acceleration (the rate of change of velocity) along the x, y, and z axes, allowing a drone to detect motion and orientation in three-dimensional space. In drones, it is part of the Inertial Measurement Unit (IMU) and plays a critical role in flight stability and control. Most accelerometers are MEMS (Micro-Electro-Mechanical Systems) devices that use tiny capacitive plates suspended by microscopic springs; when the drone accelerates or tilts, these plates shift due to inertia, changing their capacitance. This change is converted into an electrical signal proportional to the acceleration experienced. The flight controller uses this data to determine the drone's tilt, movement, and orientation relative to gravity, enabling it to stabilize flight and respond to changes in motion. When combined with data from the gyroscope, magnetometer, barometer, and GPS through sensor fusion, the accelerometer helps provide a complete understanding of the drone's position and attitude in 3D space.

Barometer

A barometer is a sensor that measures air pressure to estimate a drone's altitude above sea level and detect changes in vertical motion. In drones, it is a key component of the Inertial Measurement Unit (IMU) system, used primarily for maintaining stable altitude and supporting autonomous flight. Most barometers are MEMS-based pressure sensors that contain a small, flexible diaphragm which bends slightly as air pressure changes; this deflection alters the capacitance or resistance (depending on the barometer) within the sensor, producing a voltage signal proportional to the surrounding air pressure. The flight controller interprets these pressure readings to calculate altitude using the barometric formula, and by monitoring pressure changes over time, it can determine the drone's climb and descent rates. Barometers are especially useful for smooth hovering and altitude hold functions, but their accuracy can be affected by temperature, weather changes, and propeller wash, which is why they are often paired with GPS and IMU data through sensor fusion for more reliable and precise altitude estimation.

Magnetometer

A magnetometer is a sensor that measures the strength and direction of the Earth's magnetic field to determine a drone's heading or orientation relative to magnetic north. In drones, it serves as a digital compass and works alongside the gyroscope and accelerometer within the Inertial Measurement Unit (IMU) to provide accurate directional data for navigation and stabilization. The sensor operates based on the Hall Effect, where moving electric charges in a semiconductor are deflected by magnetic fields, generating a measurable voltage that corresponds to the field's intensity and direction. By combining readings from three orthogonal magnetometer sensors (aligned with the x, y, and z axes), the flight controller reconstructs the full three-dimensional vector of Earth's magnetic field. This allows the drone to maintain correct yaw orientation and prevent rotational drift during flight. However, magnetometers are sensitive to electromagnetic interference from motors and nearby electronics, requiring periodic calibration. When fused with gyroscope and GPS data, the magnetometer ensures long-term heading accuracy and reliable directional control, which are essential for autonomous navigation and waypoint flight.

Gyroscopic Sensor

A gyroscopic sensor, or gyroscope, measures a drone's angular velocity (rate at which it rotates around its roll, pitch, and yaw axes). It is one of the core components of the Inertial Measurement Unit (IMU) and is essential for maintaining stability and orientation during flight. Most modern drones use MEMS (Micro-Electro-Mechanical Systems) gyroscopes, which operate using a tiny vibrating element, often shaped like a tuning fork. When the drone rotates, the Coriolis effect causes the vibration pattern of the element to deflect slightly; this deflection is proportional to the angular rate of rotation and is converted into an electrical signal. The flight controller uses these real-time rotational measurements to quickly detect and correct any unintended tilts or spins, ensuring smooth and stable flight performance. However, gyroscopes are subject to drift (gradual accumulation of small measurement errors over time) which can cause inaccuracies if used alone. To counteract this, data from the gyroscope is combined with readings from the accelerometer and magnetometer through sensor fusion, allowing the drone to maintain precise orientation and accurate long-term heading control.

GPS (Global Positioning System)

A GPS (Global Positioning System) is a satellite-based navigation sensor that provides a drone with precise positioning, velocity, and timing data. It works by receiving time-stamped signals from at least four satellites and calculating the drone's distance from each one based on how long the signals take to arrive. Using this information, the GPS receiver determines the drone's latitude, longitude, and altitude, effectively pinpointing its 3D position on Earth. In drones, GPS is essential for autonomous flight, position hold, and waypoint navigation, allowing the aircraft to hover in place, follow pre-programmed routes, and return to its launch point automatically. GPS data is often combined with IMU sensors (accelerometer, gyroscope, and barometer) through sensor fusion to improve accuracy and compensate for short-term drift or lag. While GPS offers position accuracy within about 2–5 meters, it can be affected by signal blockage, interference, or atmospheric conditions, leading to slower update rates compared to onboard sensors. Despite these limitations, GPS remains crucial for ensuring reliable, large-scale navigation and situational awareness in drone operations.

UltraSonic Sensor

An ultrasonic sensor is a distance-measuring device that helps a drone determine its altitude relative to the ground by using sound waves. It operates by emitting a short burst of ultrasonic pulses toward the ground and measuring the time it takes for the echoes to return after reflecting off a surface. The sensor then calculates the distance using the formula: *distance = (speed of sound × time of flight) / 2*. In drones, ultrasonic sensors are especially useful for low-altitude flight, takeoff, and landing, where barometers and GPS signals may be unreliable or imprecise. They provide accurate short-range altitude data, allowing the flight controller to maintain stable hovering and avoid ground collisions. However, their effectiveness is limited by range (typically 5–10 meters) and surface type, as soft or uneven surfaces can scatter sound waves instead of reflecting them. Environmental conditions such as temperature, humidity, and wind can also slightly affect the speed of sound and, consequently, the accuracy of readings. Despite these limitations, ultrasonic sensors play a vital role in providing precise ground-distance data for indoor or near-surface drone operations.

3.2.1.1 Flight Controller Firmware

Open-source Architecture vs Proprietary Architecture

To begin, it is important to define the two system architectures. Open-source architecture is a layered hardware–software framework for drones in which flight stabilization, sensor integration, and mission management are controlled by publicly available and modifiable firmware such as ArduPilot or PX4. This firmware can run on a wide range of compatible flight controllers, allowing significant flexibility and customization. In contrast, proprietary architecture provides the same core functions but relies on manufacturer-specific hardware and closed firmware that cannot be modified, limiting adaptability while offering tighter integration and vendor support. Open-source systems thus excel in versatility but may introduce configuration challenges that require additional troubleshooting, whereas proprietary systems are designed to function reliably out of the box but sacrifice flexibility and long-term adaptability.

Type of firmware	Flexibility/Adaptibility	Reliability OOB
Open-source	High	Low
Proprietary	Low	High

Table 3.2.1.1 Flight Controller Firmware Comparison

By Authors

Choice of Architecture: Open-source

For this project, open-source architecture is the optimal choice because it offers the flexibility needed to adapt the system to the specific requirements of the SOAR project. Unlike proprietary systems, which are limited to pre-packaged functionalities, open-source firmware such as ArduPilot or PX4 can be modified to integrate custom swarm algorithms, our specific sensors, and the mission-specific failsafes required by Lockheed Martin. This adaptability is crucial for developing features like coordinated area coverage, victim detection and verification, and dynamic task reassignment among drones. Although open-source systems may require additional troubleshooting and fine-tuning during development, the ability to customize both the software and hardware layers ensures that the swarm can evolve alongside mission demands. In contrast, a proprietary system may provide stability out of the box, but its rigid design would restrict innovation and prevent the implementation of advanced swarm behaviors. For these reasons, open-source architecture provides the most practical and scalable foundation for a drone swarm intended to function as a reliable search and rescue tool.

3.2.1.2 Flight Controller Selection

Selecting the right flight controller is one of the most critical decisions for this project, as it directly determines the swarm's reliability, adaptability, and performance in real-world search and rescue missions. A suitable controller must support the necessary suite of sensors including an IMU, magnetometer, barometer, and GPS while also offering expansion options for additional components such as cameras, ultrasonic sensors, or LIDAR. Equally important is the processing capability of the main chip, since higher-level tasks like object detection and swarm coordination require both robust onboard computation and efficient communication with a companion computer. Finally, the controller must integrate seamlessly with open-source firmware, ensuring the flexibility to implement custom algorithms and swarm behaviors tailored to mission requirements. By carefully balancing sensor support, processing power, and open-source compatibility, the chosen flight controller becomes more than just a stabilization tool; it serves as the foundation that enables advanced autonomy and coordinated swarm operations.

GEPRC TAKER F745 BT

The GEPRC TAKER F745 BT is an excellent choice for the SOAR project due to its robust features and compatibility with open-source firmware. The flight controller is powered by a high-performance STM32F745 chip and is fully compatible with ArduPilot, providing the flexible, advanced flight control necessary for the complex swarm coordination that will be required. Its dual IMU (MPU6000 and ICM42688-P) configuration enhances reliability and precision by providing sensor redundancy and sensor fusion, while the large 512MB onboard storage allows for detailed flight data logging and post-mission analysis. The flight controller's seven UART ports and single I2C port are critical for integrating additional hardware, such as GPS modules for precise navigation, communications systems for the swarm, and a companion computer for handling the intensive processing required for computer vision payloads. This combination of a capable flight controller with a robust expansion architecture makes the GEPRC TAKER F745 BT an ideal controller for our goals in the SOAR project.

Matek F765-WSE

The Mateksys F765-WSE is a versatile and powerful flight controller that makes it an excellent candidate for use in SOAR operations as a flight controller for our worker drones with more robust sensing modalities. At its core, it runs on an STM32F765VIH6 ARM Cortex-M7 processor clocked at 216 MHz, supported by 2 MB of flash memory and 512 KB of RAM, providing ample processing capacity for open-source firmware such as ArduPilot, PX4, and iNav. The board is equipped with modern sensors including an ICM-42688-P IMU for precise attitude estimation and a DPS310 barometer for altitude measurement, with expansion options for external GPS and compass modules. Its robust communications suite (featuring 6.5 UARTs, dual I2C buses, a CAN interface, USB connectivity, and 12 PWM outputs) makes it highly adaptable to a wide range of peripherals that will be integral to the drone's swarm search and rescue operations. Among these peripherals are companion computers, payload sensors, and extra navigation tools. The inclusion of an SD card slot enables detailed blackbox logging for post-flight analysis, while integrated power management ensures reliable operation in demanding environments. Compact yet capable, the F765-WSE strikes an enticing balance between affordability and performance, making it especially well-suited for support drones in swarm operations where flexibility, sensor integration, and open-source compatibility are essential.

Pixhawk 6X

The Holybro Pixhawk 6X is an ideal choice for the primary "leader board" in the SOAR Project because it combines high processing power, advanced sensor redundancy, and unmatched connectivity, making it the most capable controller for coordinating swarm operations. Powered by an STM32H753 processor running at 480 MHz with ample memory capacity (2MB Flash, 1MB RAM), the Pixhawk 6X can handle complex flight control tasks while managing communications with multiple companion computers and support drones. Its extensive communication suite including numerous UARTs, CAN buses, I2C ports, and even Ethernet ensures seamless integration with the swarm network and external payloads(i.e. companion computer for computer vision and payload sensors). On the sensing side, its triple redundant IMUs, dual barometers, and onboard magnetometer provide robust sensor fusion that enhances reliability, stability, and long-term heading accuracy. This level of redundancy allows the leader drone to safely navigate through treacherous terrain, adapt to unpredictable weather conditions, and maintain precise situational awareness—critical factors in high-stakes search and rescue missions. By serving as the central hub for coordination, the Pixhawk 6X not only guarantees flight stability but also enables the swarm to operate cohesively, making it the most strategic choice for the leader role.

The Cube Black

The Cube Black is built as a high-reliability autopilot platform, using triple-redundant IMUs mounted on an isolated, vibration-damped internal structure and supported by dual barometers

and redundant power paths. This hardware design provides consistent state estimation even under severe vibration, temperature shifts, or partial sensor failure, which is a requirement for multi-aircraft coordination and for missions conducted in unstable or cluttered environments. Its modular carrier-board system exposes a large set of interfaces—multiple UARTs, dual CAN buses for DroneCAN peripherals, I2C, SPI, USB, and dedicated power and safety circuits—allowing clean integration of RTK-GPS receivers, long-range telemetry, companion computers, and SAR-specific payloads without overloading any single bus. Because it runs ArduPilot and supports advanced failsafe logic, real-time sensor voting, and automatic reversion when a subsystem degrades, the Cube Black maintains continuous navigation accuracy and link stability across all swarm units. This makes it effective for swarming, where synchronized trajectories, collision-avoidance logic, and distributed decision-making depend on stable state data and deterministic timing. In search-and-rescue operations, the same hardware redundancy and interface capacity support thermal cameras, mapping sensors, and high-bandwidth communication modules needed to search large areas, operate in uncertain conditions, and maintain uninterrupted control even when individual sensors or links experience faults.

Flight Controllers	**Pixhawk 6X**	**GEPRC TAKER F745 BT**	**Matek F765-WSE**	**Cube Black PX4**
Memory Capacity	High	Low	Moderate	High
Sensor Support	High	Moderate	Moderate	High
Companion Computer Integration	High	Moderate	Low	High
Weight	Moderate	Moderate	Moderate	Moderate
Cost	High	Low	Low	High

Table 3.2.1.2 Flight Controller Pros and Cons

By Authors

Choice of Flight Controller: The Cube Black

While our team determined that the Holybro Pixhawk 6X would be the most optimal flight controller for the SOAR Project based on our scope and performance requirements, we are utilizing the Cube Black because it is the hardware provided by our sponsor, Lockheed Martin. The Cube Black is a highly capable controller, featuring an onboard 32-bit ARM Cortex M4 with FPU processor that delivers the computational power needed for real-time swarm coordination, autonomous path planning, and sensor data processing. Its triple redundant, temperature-controlled IMUs and dual barometers offer strong reliability and fault tolerance, ensuring that flight stability and mission performance are not compromised by single-sensor failures. Although the Pixhawk 6X offers slightly broader communications capabilities and superior integration for swarm networking, the Cube Black provides a robust and flexible platform that fully supports our open-source architecture and allows us to meet project objectives using the hardware currently available to us.

3.2.1.3 Carrier Board Selection

Carrier boards play a vital role in the overall function of a drone's flight control system. They act as the central interface that allows the flight controller (the drone's "brain" or "control center") to communicate with and manage all other onboard components. Through integrated communication buses (such as UART, CAN, I2C, and SPI) and a wide array of connection ports, the carrier board distributes power, data, and control signals between the flight controller, sensors, telemetry modules, GPS units, cameras, and electronic speed controllers (ESCs). In essence, the carrier board serves as the bridge between hardware and control logic, ensuring that every subsystem operates in coordination. Without one, even the most advanced flight controller would lack the means to communicate with the drone's sensors and actuators, making reliable flight and system integration impossible.

ADS-B Carrier Board

The ADS-B is an ideal carrier board for our swarm as it provides a vital connectivity hub for our SOAR Project architecture. The ADS-B expands the capabilities of the Cube Orange+ by linking it to a wide array of sensors, peripherals, and communications systems. This carrier board provides an extensive set of interfaces and ports such as: multiple UARTs, CAN buses, I2C and SPI connections, PWM output, USB, and power distribution channels. This allows seamless integration of telemetry modules (such as our RGB camera, Lidar, and thermal camera), GPS units, and payload sensors that are essential to SOAR missions. Among its many features, the most valuable is the integrated 1090 MHz ADS-B IN receiver. This receiver enables the drone to detect and track nearby aircrafts broadcasting ADS-B signals, providing real-time air traffic awareness for the Cube Orange+ to process. This allows the system to autonomously execute collision-avoidance maneuvers and maintain safe airspace separation. Additionally, the carrier board's built-in voltage regulation, redundant power inputs, and vibration-dampened mounting design ensure stable operation even in the harshest of flight conditions. Combining the robust connectivity with intelligent sensing, the ADS-B carrier board transforms the Cube Orange+ into a highly adaptable command center, capable of managing complex sensor networks, maintaining

constant communication across the swarm, and safely coordinating multiple drones during SOAR operations.

Kore Carrier Board

The Kore Carrier Board is an ideal carrier board for our swarm as it is designed to maximize the functionality and connectivity of the Cube Orange+. Its feature-rich, integrated platform is engineered to simplify wiring and improve reliability by consolidating key systems (such as power distribution, voltage and current sensing, and redundant power inputs) directly onto the board. This reduces the need for external modules and minimizes failure points when out in the field. The Kore carrier board offers a wide range of communication interfaces (including multiple UARTs, two CAN buses, I2C and SPI connections, USB ports, and PWM outputs). This myriad of communications methods enables seamless integration of telemetry radios, GPS modules, companion computers and essential payload sensors (such as our Jeston AGX Orin, RGB, thermal, and LiDAR systems). The Kore also provides dedicated ports for RC input, a safety switch, buzzer, and LEDs along with redundant power circuitry and a built-in 5V and 3.3V regulator to ensure stable operation even under the most demanding mission conditions. When paired with the Cube Orange+, the Kore's robust connectivity and ingenious engineering allow for high-bandwidth communication between swarm members, reliable sensor fusion, and precise command coordination (all these qualities being critical to maintaining safety, efficiency, and situational awareness during complex SOAR operations).

Mini Carrier Board

The Mini Carrier Board is a compact yet highly capable platform designed to provide the essential connectivity and functionality for the Cube Orange+ in a smaller, lightweight form factor. This compactness makes it ideal for SOAR operations where space, weight, and efficiency are imperative criteria. Despite its reduced size, the Mini carrier board retains robust features, these include dual power inputs with automatic redundancy, power distribution for ESCs, and integrated voltage and current sensing for accurate power monitoring. Among communication methods, the Mini carrier board supports multiple UARTs, CAN buses, I2C and SPI connections, and PWM outputs, allowing seamless integration with GPS modules, telemetry radios, companion computers, and payload sensors (like our Thermal, RGB, and LiDAR systems). Though it is a smaller form factor, the board provides dedicated ports for RC input, a safety switch, a buzzer, and I/O expansion, ensuring that it has operational flexibility without adding unnecessary bulk. When paired with the Cube Orange+, the Mini carrier board delivers reliable data and power communications across all drone subsystems while simultaneously maintaining a compact footprint, making it ideal for agile search coverage and rapid deployment. Its balance of connectivity, redundancy, and lightweight design makes it a practical and efficient choice for our drones in our SOAR project.

EDU450 Carrier Board

The EDU450 Carrier Board is an educational and development-focused carrier designed to provide a versatile and accessible platform for integrating the Cube Orange+ into multirotor systems. This makes it a strong candidate for our SOAR project. Intended to be a part of the EDU450 Quadcopter Frame Kit, the board consolidates vital drone electronics into a compact and organized layout. It features power distribution, voltage and current sensing, and support for redundant power inputs to ensure flight stability during grueling missions. In terms of communications methods, it boasts multiple UARTs, CAN buses, I2C and SPI connections, and PWM outputs. This myriad of communications methods enables seamless connection between the Cube Orange+ and imperative peripherals like GPS units, telemetry modules, companion computers, and sensor payloads (including Thermal cameras, RGB cameras, and LiDAR systems). The board also features dedicated ports for RC input, safety switch, buzzer, and LEDs. This simplifies setup while maintaining compatibility with the full Cube Orange+ ecosystem. When paired with the Cube Orange+, the EDU450 offers reliable connectivity, stable power management, and straightforward integration. This makes it well suited for coordination, modularity, and resilience, all parameters encompassed by our SOAR Project.

Carrier Boards	**Kore**	**Mini**	**ADS-B**	**EDU450**
Connectivity	High	High	Moderate-High	Moderate
Power Management	High	Moderate	Low	Moderate
Flight Controller Compatibility	High	High	High	High
Integration Complexity	Low	Low	Moderate	Low
Redundancy Support	High	Moderate	Moderate	Low

Table 3.2.1.3 Flight Controller Carrier Board Pros and Cons

By Authors

Final Choice: ADS-B Carrier Board

After evaluating all available carrier boards, our team selected the Kore Carrier Board as the most suitable option for the SOAR Project due to its optimal balance of connectivity, reliability, and system integration efficiency. The Kore offers a comprehensive feature set that supports robust field operations, combining advanced power management capabilities (including integrated voltage and current sensing) with redundant power inputs to enhance flight safety and operational resilience. Its streamlined wiring layout reduces complexity during assembly and maintenance, while the vibration-dampened design contributes to improved sensor stability and overall flight performance. Additionally, its superb connectivity brings out the full potential of the Cube Orange+, allowing it to connect to most peripherals without issue. However, since our project sponsor has provided the ADS-B Carrier Board, our team will be proceeding with this board. While the ADS-B board lacks some of the Kore's power management refinements, it provides valuable additional functionality, such as an integrated 1090 MHz ADS-B IN receiver for real-time air traffic awareness. This will be very useful for maintaining safety in shared airspace during swarm operations. While it is not our first choice, the ADS-B board is still a very capable and top of the line board. Its robust set of features will be imperative for our swarm operations.

3.2.1.4 Flight Controller MCU Part Selection

STM32H757

The STM32H757 series represents the most advanced line of ARM Cortex-M7 microcontrollers currently used in flight controllers, offering the highest level of performance and reliability for demanding drone applications such as our SOAR project. With clock speeds up to 480 MHz and exorbitant memory capacity, these chips deliver significantly greater processing power than the F4 and F7 counterparts, enabling them to handle complex control algorithms, high-frequency sensor fusion, and extensive communication tasks simultaneously. Flight controllers, such as the Holybro Pixhawk 6X, leverage the STM32H757 to provide robust redundancy features, such as triple IMUs, dual barometers, and advanced failsafe mechanisms. The chip's enhanced communications capabilities (like multiple UARTs, CAN buses, SPI and I2C interfaces, and Ethernet support) make it ideal as the flight controller in our swarm architecture, where seamless networking and reliable coordination are paramount for swarm success. For our SOAR Project, the STM32H757-based flight controller provides the processing headroom and sensor reliability required to safely navigate the complex and unpredictable search and rescue environments, manage the multiple data streams from the other swarm members, and ensure stable operation even in treacherous conditions. Its combination of speed, memory, and connectivity makes it the most strategic choice as the flight controller for our swarm.

STM32F765

The STM32F765 is a high-performance member of the ARM Cortex-M7 family that strikes an effective balance between processing capability, memory, and cost, making it well-suited for worker drones in our SOAR Project. Though it is not as robust as the STM32H7, running at 216 MHz with 2 MB of flash memory and 512 KB of RAM, the F765 provides ample resources for open-source flight stacks such as ArduPilot, PX4, and iNav, while maintaining efficiency and stability in real-time flight control. Flight controllers like the Matek F765-WSE take advantage of this processor's capabilities by integrating reliable sensor suites (such as IMUs and barometers) and offering extensive connectivity through multiple UARTs, I2C buses, CAN support, and PWM outputs. This makes the F765 versatile enough to manage a variety of mission-critical peripherals such as GPS, telemetry rallying, companion computers, and payload sensors. For SOAR applications, the STM32F765 provides the computational headroom we need to process sensor fusion, execute autonomous flight routines, and coordinate with external systems (such as the "leader" drone with the STM32H7 and the ground station), all while remaining lightweight and cost-effective. Its balance of power, expandability, and open-source compatibility makes it an excellent choice for the support drones in our swarm, where reliability and adaptability are crucial to our mission success.

STM32F745

The STM32F745 is a capable ARM Cortex-M7 microcontroller that provides solid performance for smaller or lighter drone platforms within our swarm. Operating at 180 MHz with 1 MB of flash memory and 320 KB of RAM, it delivers less processing headroom than the STM32F765 or H7 series but remains more than sufficient for stabilization, sensor fusion, and standard open-source flight control tasks. It supports popular firmware such as ArduPilot, PX4, and iNav, ensuring full integration within the open-source architecture. Flight controllers like the GEPRC TAKER F745 BT leverage this processor to provide essential functionality, including IMU and barometer support, multiple UARTs, and I2C interfaces for GPS, telemetry, and companion computer links. Its compact design and lower cost make it ideal for our swarm "support units" where weight and affordability are critical, but advanced onboard processing is not as heavily demanded (i.e. our drones with the lighter sensors or less computationally demanding sensors). In the context of SOAR missions, the STM32F745 allows these drones to carry out reliable area coverage and communication roles while leaving more complex tasks such as swarm coordination and heavy sensor payloads to F765- and H7-based drones.

Flight Controller FMU	STM32H757	STM32F765	STM32F745
Memory Capacity	High	Low	Moderate
Sensor	High	Moderate	Moderate

Support			
Dimensions	High	Moderate	Low
Weight	Moderate	Moderate	Moderate
Cost	High	Low	Low

Table 3.2.1.4 Flight Controller FMU Pros and Cons

By Authors

Flight Controller MCU Selection: STM32H757

Based on the project scope and parameters, the STM32H757 micro-controller was the best choice because its performance is essential for the complexity of the mission. As a faster H7 System on Chip (SOC), the chip provides the necessary high-speed processing and 1MB of integrated RAM to execute complex real-time algorithms simultaneously across the drone swarm. This computational power is critical for enabling dynamic swarm coordination and communication, running sensor fusion algorithms to integrate data from the redundant IMUs and search payloads, and maintaining the precise flight algorithms required for reliable autonomous navigation during critical rescue missions. The speed and multitasking capability of the STM32H757 is crucial for the responsiveness and stability required for effective, coordinated SOAR efforts.

3.2.1.5 AI Companion Computer

Each SOAR drone will carry a dedicated AI companion computer. This companion functions in tandem with the flight controller: it ingests raw sensor data (from cameras, LiDAR, etc.) and executes compute-intensive workloads such as object detection, classification, and tracking. Offloading those tasks to a separate processor ensures that the flight controller can remain fully focused on essential real-time tasks (stabilization, control loops, navigation, communication) without being burdened by heavy inference workloads. In effect, the companion computer enables a true “edge AI” capability on the drone, processing data locally and immediately — a necessity when latencies, bandwidth constraints, or communications dropouts prevent relying on a ground or cloud server.

Below, we compare three candidate companion computers against SOAR’s objectives and constraints to determine which may be the best fit.

NVIDIA Jetson AGX Xavier

The NVIDIA Jetson AGX Xavier is a powerful AI companion that strikes a balance between computational performance and energy efficiency, making it a strong candidate for the SOAR project. With up to 32 TOPS of AI performance, it provides high inference throughput for demanding real-time applications such as object detection, classification, and tracking in complex environments. Its 512-core Volta GPU with 64 Tensor Cores, combined with an 8-core

Carmel ARM CPU and up to 32 GB of LPDDR4x memory, ensures fast data processing and smooth multitasking between AI workloads and communication tasks. The Xavier operates on 9–20V input and typically draws 30W or less, which is energy efficient given its performance class—an important consideration for drone endurance. Its low latency processing allows rapid decision-making for autonomous navigation and obstacle avoidance. Additionally, NVIDIA's JetPack SDK offers robust software support, streamlining AI model deployment and sensor integration. Lighter than the Orin, the Xavier offers excellent scalability and reliability for airborne systems. Overall, the Xavier fits SOAR's objectives by providing high computational performance without overly compromising weight or power consumption.

NVIDIA AGX Orin

The NVIDIA Jetson AGX Orin represents the next generation of embedded AI computing, offering exceptional performance and efficiency for autonomous systems like SOAR. Delivering up to 275 TOPS, it dramatically surpasses the Xavier in inference throughput and supports more complex neural networks, making it ideal for advanced perception tasks such as multi-object tracking and environmental mapping in dynamic conditions. It features a 2048-core Ampere GPU with 64 Tensor Cores and a 12-core ARM Cortex-A78AE CPU, backed by up to 64 GB of LPDDR5 memory. Despite this immense power, the Orin remains energy efficient with configurable power modes between 15W and 60W, adjustable to match drone endurance needs. Its lower latency and higher scalability enable it to handle simultaneous sensor streams (e.g., LiDAR, EO/IR, GPS) and perform real-time AI-driven decision-making. With strong support from NVIDIA's ecosystem (JetPack, DeepStream, TensorRT), the Orin integrates seamlessly into modern robotics frameworks. Although it is slightly heavier and consumes more power than the Xavier, its superior AI capacity and flexibility make it highly suitable for the SOAR project's long-term scalability and autonomy goals, particularly for advanced environmental intelligence and multi-sensor fusion.

Raspberry Pi 4

The Raspberry Pi 4 serves as a lightweight and low-cost companion computer option, but its limited computational resources make it less ideal for intensive AI workloads in the SOAR project. It features a 1.5 GHz quad-core ARM Cortex-A72 CPU, up to 8 GB of LPDDR4 RAM, and lacks a dedicated GPU optimized for AI inference, resulting in significantly lower throughput (measured in single-digit GOPS rather than TOPS). Its power consumption of around 5V at 2–3A (10–15W) is extremely efficient, making it suitable for simple control and data logging tasks, but not for complex real-time perception such as object detection or classification. Latency is higher due to the lack of parallel processing cores, and scalability is limited by its memory and bandwidth constraints. However, it offers excellent software flexibility and community support, making it an excellent platform for basic prototyping, telemetry processing, or as an auxiliary node for non-critical AI tasks. While not suitable as the primary AI companion for SOAR, the Raspberry Pi 4 can complement the system as a lightweight processor for communications or data management where high inference performance is not required.

	Jetson AGX Xavier	Jetson AGX Orin	Raspberry Pi 4
Input Voltage	9V - 20V	9V - 20V	5V
Power Consumption	10W - 30 W	15W - 60W	≤ 15W
CPU	8-core NVIDIA Carmel ARMv8.2	12-core ARM Cortex- A78AE	Quad-core ARM Cortex-A72
GPU	512-core Volta GPU + 64 Tensor Cores	2048-core Ampere GPU + 64 Tensor Cores	No dedicated AI GPU / tensor cores — uses VideoCore GPU
TOPS	≤ 32	≤ 275	≤ 26
RAM	16GB - 32 GB	32GB - 64 GB	1GB, 2GB, 4GB, 8GB
I/O Support	Rich: multiple PCIe lanes, camera interfaces, etc	Very Rich: PCIe, multiple camera interfaces, more bandwidth, next-gen features	Standard: USB, CSI (for camera), Ethernet, GPIO; limited high-throughput
Community	Strong: JetPack, CUDA, ROS support	Very Strong: supports latest frameworks, backward compatibility with Jetson ecosystem	Strong: Linux support, many libraries — but less optimized for heavy AI models
Weight (including ADX DevKit)	≈ 665g	≈ 872.5g	≈ 70g

Table 3.2.1.5 AI Companion Computer Constraints Comparison

By Authors

Summary

Based on the evaluation of the three AI companion computers — the NVIDIA Jetson AGX Xavier, NVIDIA Jetson AGX Orin, and Raspberry Pi 4 — the most suitable option for the SOAR project is the Jetson AGX Orin. The Orin stands out as the most capable companion computer due to its greater number of CPU and GPU cores, enabling faster and more parallelized processing of real-time sensory input. Its large memory capacity (up to 64 GB LPDDR5) allows for efficient handling of advanced perception tasks such as object detection, classification, and tracking without the risk of memory bottlenecks. Additionally, the Orin offers exceptional I/O

and peripheral support, making it highly compatible with SOAR's planned camera and LiDAR integration. This high bandwidth and interface flexibility allow the Orin to synchronize multiple data streams in real-time, which is critical for accurate sensor fusion and autonomous decision-making. While the Orin has the highest power consumption and weight among the three, its superior AI performance and ability to handle complex neural networks justify these trade-offs, especially for missions that demand rapid and intelligent environmental analysis.

Despite these advantages, budget constraints and the availability of a Jetson AGX Xavier unit from a previous SOAR team led to the decision to continue development using the Xavier. Although not as powerful as the Orin, the Xavier remains a very capable and balanced AI platform, offering up to 32 TOPS of AI performance with a 512-core Volta GPU and 8-core ARM CPU. It maintains strong I/O support for connecting cameras, LiDARs, and other sensors while consuming less power than the Orin, which benefits flight endurance. Its 32 GB of memory—though smaller than the Orin's—can still support complex AI inference workloads when paired with optimized algorithms, model compression, and efficient memory management. Therefore, while the Xavier may not provide the same ceiling of computational performance, it still meets the operational needs of the SOAR project by enabling reliable, real-time onboard processing for perception and navigation. This makes it a cost-effective and mission-capable solution given the project's current scope and resource limitations.

3.2.1.6 Environmental Sensors and GPS Module

Environmental Sensors

Each SOAR drone will be equipped with an AI companion computer, specifically the NVIDIA Jetson AGX Xavier, which handles onboard vision processing, object detection, and coordination with other drones in the swarm. The Jetson platform is optimized for GPU-accelerated AI workloads, allowing real-time inference for tasks such as human detection, target tracking, and collision avoidance.

To enable environmental awareness and situational perception, each drone will also integrate two categories of sensors:

1. **Obstacle avoidance sensors**, such as LiDAR or ultrasonic modules.

2. **Imaging sensors**, such as RGB, EO, or thermal cameras, depending on mission profile and environmental conditions.

These sensors collectively support the SOAR drone's autonomy by providing depth, environmental, and visual information essential for navigation and target identification.

GPS Module

To back our choice of the Cube Orange PX4 flight controller we have to choose a GPS module because there isn't one inside the Cube Orange PX4. Although it doesn't carry a GPS module the Cube Orange PX4 supports external GPS modules through dedicated UART ports. GPS modules connect through the GPS1 or GPS2 ports on the Cube Carrier Board (the breakout board that the Cube plugs into). These are UART + I²C interfaces that supply power (5V) and data (TX/RX + SDA/SCL) to GPS receivers. These following GPS modules fit our requirements:

Criteria	Here2 (CAN/UART)	Here3 / Here3+ (CAN RTK)	u-blox SAM-M8Q (UART/I²C)
Positioning Accuracy	1.5m	0.01m (RTK mode)	2m
Update Rate	10 Hz	10-20 Hz	10Hz
Interface Type	UART / I²C	CAN / UART	UART / I²C
RTK Support	No	Yes (RTCM3)	No
Built-in Magnetometer	Yes	Yes	No
Voltage Input	5V	5V	3.3V - 5V
Power Consumption	0.5W	0.7W	0.3W
Weight (g)	38g	50g	18g
Operating Temperature	–40 to +85(°C)	–40 to +85(°C)	–40 to +85(°C)
Data Protocols	NMEA, UBX	NMEA, UBX, CAN	NMEA, UBX
Compatibility with Cube Orange	Yes (GPS1 Port)	Yes (Preferred – CAN1 Port)	No (Requires custom wiring)
Jetson AGX Integration (via MAVLink)	Limited	Fully Supported (via ArduPilot)	Limited
Cost (USD)	$79	$139	$35
Overall Reliability	90%	98%	85%
Suitability for SOAR	88%	98%	70%

Table 3.2.1.6.1 GPS Module Comparisons Table Selection

By Authors

The Here3 GPS module was selected for the SOAR project because it provides the most reliable and precise positioning system compatible with the Cube Orange flight controller. Its CAN-based communication allows seamless integration with the Cube ecosystem, ensuring low-latency data transfer and minimal interference compared to UART-based alternatives. The Here3 also supports Real-Time Kinematic (RTK) correction, which significantly improves positional accuracy to the centimeter level, an essential feature for autonomous navigation and precise waypoint tracking during search and rescue operations. Its built-in magnetometer and dual IMU sensors enhance flight stability by improving heading and orientation data, ensuring consistent performance in dynamic environments. Additionally, the Here3 integrates smoothly with the Jetson AGX Xavier through ArduPilot and MAVLink, enabling synchronized data exchange for high-level processing tasks. Although it has a slightly higher cost and power requirement than other GPS modules, its precision, robustness, and full compatibility with SOAR's hardware architecture make it the most effective and future-proof choice for the system.

LiDAR

LiDAR, short for Light Detection and Ranging, is a sensing technology that measures distance by emitting rapid laser pulses and calculating the time it takes for each pulse to reflect off nearby surfaces and return to the sensor. By repeating this process thousands of times per second, LiDAR builds a precise 2D or 3D spatial map of the environment with centimeter-level accuracy. This capability makes it invaluable for obstacle avoidance, terrain mapping, and environmental awareness in autonomous drones.

For the SOAR platform, LiDAR enables the detection of trees, buildings, and other aerial obstacles in real time — even under challenging lighting conditions such as low light or direct sunlight. The technology offers excellent spatial accuracy and range resolution, performing reliably in both day and night operations. However, its limitations include higher power consumption compared to other proximity sensors and performance degradation in adverse weather such as heavy rain, fog, or dust, where laser scattering reduces effective range.

To support SOAR's obstacle avoidance and autonomous navigation capabilities, the selected LiDAR must provide a 360° field of view and output real-time 2D or 3D environmental scans. The following LiDAR systems meet these requirements:

	RPLIDAR C1	RPLIDAR A2M8
Frequency	≤ 8000 samples/sec	≤ 8000 samples/sec
Rotational Speed	10 Hz (typical 5.5 Hz)	5 - 15 Hz (typical 10 Hz)
Angular Resolution	~ 0.72 degrees	~ 0.45 degrees
Weight	110 g	190 g
Range	≤ 12 m	≤ 12 m

Supply Voltage	5 V	5 V
Current	230 mA	450-600 mA
Power Consumption	~1.15 W	2.25-3 W
Accuracy	≤ 1% up to 12 m	≤ 1% up to 12 m
Cost	≈ $70	≈ $150

Table 3.2.1.6.2 LiDAR Specifications

After evaluating low-cost 360° RPLIDAR A2 C1 series options, the RPLIDAR C1 was selected as the most suitable choice for SOAR's obstacle avoidance system. The C1 provides a full 360-degree field of view, up to 12 meters of range, and centimeter-level accuracy, which is sufficient for low-altitude flight and close-range obstacle detection. Its lightweight design and low power consumption (~1.15 W) make it ideal for integration on small UAVs where weight and power budgets are limited. The C1's 10 Hz scan rate provides frequent environmental updates, ensuring reliable avoidance of trees, buildings, and other drones in real time. Additionally, its affordable cost and USB/UART compatibility simplify both hardware integration and data interfacing with onboard computers such as the NVIDIA Jetson. Overall, the C1 offers the best balance between performance, efficiency, and cost for SOAR's real-time aerial navigation needs.

RGB Camera

An RGB camera captures standard color imagery using three channels — red, green, and blue — similar to human vision. Used for object detection, classification, and visual tracking under normal lighting conditions. It provides rich visual data for AI-based detection models. The advantages are high-resolution color imaging, compatibility with most vision AI frameworks, and relatively low cost and lightweight. The downsides are degrading performance in low light or fog and limited ability to distinguish depth. Below are two RGB cameras that fulfill the SOAR object detection requirements.

	EMEET SmartCam C970	Sony IMX477
Frequency	≤ 60 fps	≤ 60 fps
Resolution	1920 x 1080 (Full HD)	12.3 MP, 4056 x 3040 px

Supply Voltage	5V	2.8V
Spatial Precision	Not Published	±1 pixel
Cost	≈ $40	≈ $100

Table 3.2.1.6.3 RGB Camera Specifications

After evaluating both options, the EMEET SmartCam C970 best meets the SOAR drone's requirements. While both sensors provide comparable frame rates and resolution, the EMEET offers a more balanced cost-to-performance ratio. Its effective range and resolution are sufficient for mid-range object detection and environmental awareness, and its significantly lower cost makes it more practical for multi-drone deployment.

EO Camera

An EO camera captures imagery in the visible and sometimes near-infrared spectrum using optical sensors. It is often paired with gimbals for stabilized aerial imaging. Provides stabilized, high-zoom imaging for reconnaissance, tracking, and visual confirmation of targets in daylight. EO systems are often used in tandem with IR or thermal cameras for dual-sensor payloads. Some advantages are high image clarity and long-range optical zoom and compatibility with multi-spectral fusion when combined with an IR camera. The disadvantage is dependent on lighting and weather conditions and higher cost to fix and maintain. Below are two EO cameras that fulfill the SOAR object detection requirements.

	Zingto INTTO A12 Gimbal	**4MP UAV Drone Camera**
Frequency	1080p at 30 fps	1080p at 25 fps
Weight	460 g	255 g
Zoom	10x optical zoom	10x optical zoom
Supply Voltage	Not Published	12V
Accuracy	±2–3 % pixel-level error	±5–8 % pixel-level error
Cost	$800	$500

Table 3.2.1.6.4 EO Camera Specifications

A EO camera will be selected after knowing what cameras can be re-used from the last SOAR project.

Thermal Camera

A thermal camera detects infrared radiation emitted by objects and converts it into temperature-based images, allowing visibility in total darkness or through smoke/fog. Used for night operations, search and rescue, or human detection, where temperature contrast is more reliable than visual cues. Thermal cameras can work in total darkness or low-visibility conditions and complement EO/RGB data for sensor fusion. The disadvantages are lower image resolution compared to RGB, more expensive, and sensitive to ambient temperature calibration. Below are two thermal cameras that fulfill the SOAR object detection requirements.

	Teledyne FLIR BOSON 320 x 256	384 x 288 FPV Thermal Imaging Module
Frequency	60 fps	25-30 fps
Weight	~7.5g	~20g
Supply Voltage	5V	4-5.5V
Power Consumption	~500mW	Not Published
Accuracy	Not Published	±2 °C
Cost	~$2k-$3k	~$600

Table 3.2.1.6.5 Thermal Camera Specifications

After evaluating both options, the FPV Thermal Imaging Module meets the SOAR drone's requirements. While the FPV sensor provides a lower frame rate, it offers a more balanced cost-to-performance ratio. Even though the accuracy cannot be compared, the FPV is sufficient for mid-range object detection and environmental awareness, and its significantly lower cost makes it more practical for multi-drone deployment.

However, last year SOAR team bought and used the Teledyne FLIR BOSON thermal camera, so we plan to use this camera.

Stereo Depth Camera

A stereo depth camera estimates distance (depth) by analyzing the disparity between two horizontally spaced lenses—much like human vision. By comparing how objects appear in the left and right images, the camera computes depth and produces both RGB and depth maps, enabling a drone to "see" its environment in three dimensions without the need for LiDAR.

These sensors are highly effective for vision-based navigation, SLAM (Simultaneous Localization and Mapping), and obstacle avoidance, particularly in scenarios where weight and power efficiency are critical. Their advantages include lightweight construction, compact size, and lower power consumption compared to LiDAR. However, stereo depth cameras have limitations: depth accuracy decreases with distance, they are sensitive to lighting conditions, and they require sufficient visual texture (patterns or contrast) to compute reliable depth.

The following stereo depth cameras meet the SOAR project's requirements for environmental mapping and object detection:

Criteria	**Intel RealSense D455**	ZED 2i
Frequency	$\leq$ 90 fps	$\leq$ 100 fps
Weight	170 g	166 g
Range	0.6 – 6 m	$\leq$ 8 m
Supply Voltage	3.3 or 5 V	5 V
Power	2.25 W	1.9 W
Accuracy	$\leq$ 2% up to 4 m	$\leq$ 1% up to 3 m $\leq$ 5% up to 15 m
Cost	$\approx$ $350	$\approx$ $500

Table 3.2.1.6.6 Stereo Depth Camera Specifications

After evaluating both options, the Intel RealSense D455 best meets the SOAR drone's requirements. While both sensors provide comparable frame rates, accuracy, and power consumption, the D455 offers a more balanced cost-to-performance ratio. Its effective range and precision are sufficient for mid-range object detection and environmental awareness, and its significantly lower cost makes it more practical for multi-drone deployment.

Additionally, the D455's broad software support through Intel's RealSense SDK and native compatibility with ROS 2 simplify integration with the existing SOAR swarm control stack. This combination of reliability, affordability, and ease of integration makes the D455 the most suitable stereo depth camera for SOAR's onboard perception system.

3.2.2 Motor Technologies

Drone motors are essential components responsible for generating the thrust that enables unmanned aerial vehicles (drones) to lift off, maneuver, and remain stable during flight. In

modern quadcopters and multirotor drones, brushless DC (BLDC) motors are the preferred choice due to their high efficiency, low maintenance, and extended lifespan compared to brushed alternatives.

These motors operate in coordination with electronic speed controllers (ESCs), which regulate the power supplied to the motor based on control signals from the flight controller. Together with carefully selected propellers, the motor-ESC-propeller system enables precise control of thrust, yaw, pitch, and roll, making them crucial for stable flight and dynamic aerial maneuvers.

Key Motor Performance Factors:

Revolutions Per Minute (RPM)

RPM is the speed at which the motor shaft spins and is directly related to both the KV rating and the input voltage. The higher the RPM, the faster the drone's propellers will spin, which translates to quicker acceleration and more agile flight. However, extremely high RPMs can reduce efficiency and strain the motor and ESC. Monitoring RPMs during flight helps identify performance issues and can be used to fine-tune motor and propeller combinations for different mission profiles, like fast scouting or slow-hover search operations.

KV Rating (RPM per Volt)

The KV rating tells you how many revolutions per minute (RPM) a motor will spin when one volt is applied with no load. A high KV motor spins very fast, which is great for light drones or racing drones that need high speed and responsiveness. However, it sacrifices torque, which means it is not ideal for carrying heavy payloads. On the other hand, a low KV motor spins slower but produces more torque, making it better for larger drones that need to lift more weight such as those used in mapping, delivery, or search and rescue. Choosing the right KV rating helps balance between speed, torque, and power efficiency.

Power Output (Watts)

This is the measure of how much electrical power the motor can convert into useful mechanical thrust. Motors with higher power output can lift heavier loads or sustain longer flights at high speeds, but they also draw more current, which drains the battery faster and may generate more heat. Matching power output with the weight of our drone and the goals of our mission is critical. For example, a rescue drone might prioritize power to lift extra sensors or batteries, while a reconnaissance drone might value efficiency for longer flight duration.

Thrust-to-Weight Ratio

This ratio is essential in drone design. As a general rule, our drone’s motors should be able to generate at least twice the total weight of the drone in thrust. This provides enough lift for takeoff, hovering, and maneuvering while also allowing room for dynamic movement and payload recovery. If the thrust-to-weight ratio is too low, the drone may struggle to lift off or maintain stable flight, especially under windy conditions or in case of sudden load shifts during flight.

Efficiency (grams per Watt)

Efficiency describes how much thrust a motor can produce per watt of power consumed. This is crucial for maximizing battery life, especially in long missions or when weight must be minimized. A motor with high efficiency will help extend the drone's flight time without sacrificing performance. This is especially important in multi-drone swarm missions, where consistent flight duration across all drones is needed to maintain coordination.

Motor Size and Mounting

Drone motors come in various sizes, typically indicated by numbers such as 2207 or 1806. These refer to the diameter and height of the motor stator. Larger motors generally offer more torque and are better suited for heavy-lift drones, while smaller motors are lighter and better for compact, agile designs. Matching motor size with drone frame dimensions and weight goals is vital for stable flight. Improperly sized motors can lead to poor handling, excessive vibration, or mechanical stress on the frame.

Cooling and Durability

Brushless motors heat up under load, and excessive heat can degrade performance or cause failures. Well-ventilated motor designs or motors with built-in cooling features such as exposed bell housings can help dissipate heat effectively. In demanding environments or long missions, maintaining low motor temperatures helps extend the lifespan and ensures reliable performance. Motors with sealed housings may also help protect against dust or moisture when flying in outdoor or hazardous conditions.

ESC Compatibility

Electronic Speed Controllers (ESCs) regulate the power sent to each motor, and not all ESCs are compatible with all motors. It's important to match the ESC's current rating to the motor's maximum current draw to avoid overheating or system failure. Additionally, the ESC must support the correct signal protocol (such as PWM, OneShot, or DShot) that the flight controller is using. Mismatches in ESC and motor specs can lead to sluggish response, erratic behavior, or even component damage.

Propeller Pairing

Choosing the right propeller to go with a motor can dramatically affect performance. Larger propellers with low pitch are ideal for stable, slow-flying drones or those carrying heavy payloads. Smaller, high-pitch props are better for racing or high-speed agility. Propeller material, blade count, and balance also affect vibration and efficiency. A well-paired propeller and motor combination ensures smooth flight, optimal thrust, and reduced energy consumption.

3.2.2.1 Type of Drone Motors

Drone motors are at the heart of unmanned aerial vehicles, converting electrical energy into the thrust that allows the drone to take off, hover, and maneuver in the air. Selecting the right type of

motor depends on many factors, including the purpose of the drone, its weight, desired flight characteristics, and power source. Below is a breakdown of the main types of motors used in drones today, along with when and why each one is used.

Brushless Drone Motors

Brushless motors are by far the most commonly used motors in modern drones, and for good reason. These motors are known for being highly efficient, reliable, and long-lasting. They do not rely on brushes to transfer electricity, which reduces internal friction and results in less wear and tear over time. Because of this, brushless motors generate less heat and require less maintenance. You'll find them powering everything from lightweight racing drones to professional photography platforms and even heavy-lift industrial drones. They provide smooth and precise control, which is critical for applications that rely on stability and accuracy, like mapping or cinematography.

Brushless motors also tend to perform well under varying loads and voltages, making them versatile across different drone designs. Their wide availability and range of sizes make them ideal for custom-built drones like those in our project.

Brushed Drone Motors

Brushed motors are the predecessors of brushless motors and are still used in some simple or budget-friendly drones. These motors work by using physical brushes to transmit current to the motor windings. While they are cheaper to manufacture and easier to replace, their design results in lower efficiency and shorter lifespan. You'll often find brushed motors in small indoor drones, toy quadcopters, and micro aerial vehicles that do not require long flight durations or heavy lifting.

These motors may be suitable for prototyping or for low-cost testing environments, but they are generally not recommended for advanced drone designs due to their limited durability and performance.

Heavy-Lift Drone Motors

Heavy-lift motors are specially designed to produce significant torque and thrust, allowing drones to carry heavy payloads such as medical supplies, camera rigs, or agricultural sprayers. These motors are usually larger in size and work at lower KV ratings to ensure strong lifting power. Their primary function is to move large drones safely and efficiently, especially when operating in less-than-ideal conditions such as wind or high elevation. You'll find these motors in logistics drones, crop-dusting UAVs, and emergency response drones where carrying extra weight is a key part of the mission.

Using heavy-lift motors typically means the drone needs higher-capacity batteries and stronger frames, so their selection often affects many other design choices in the system.

Racing Drone Motors

Racing motors are built for speed, rapid acceleration, and sharp maneuverability. These motors have high KV ratings, meaning they spin extremely fast per volt of input. They are designed for drones that need to make quick movements and change directions in tight spaces, often while broadcasting live video in competitive environments. While they are not optimized for efficiency or carrying loads, they shine in FPV (first-person view) racing, stunt flying, and any application that values responsiveness over endurance.

They are usually compact, lightweight, and paired with aggressive propellers and specialized electronic speed controllers (ESCs) for maximum control.

Quadcopter Motors

Quadcopter-specific motors are optimized for drones with four rotors, offering balanced performance in thrust, size, and power consumption. These motors are often chosen for drones used in consumer photography, hobby flying, and general surveillance applications. They strike a good balance between efficiency and thrust, and are usually designed to pair well with standard drone ESCs and propellers.

Because quadcopters are among the most common drone types, motors in this category are widely available and often supported by extensive documentation and user communities.

Mini Drone Motors

Mini drone motors are small, low-power motors used in compact drones. These are typically used for indoor flying, classroom drones, or experimental swarm drones where size and weight need to be kept to a minimum. Though they don't provide much thrust, they are lightweight and ideal for situations where battery life and maneuverability are more important than payload capacity. They're often found in hobby kits or educational drones for learning environments.

These motors can be brushed or brushless, depending on cost and design goals, and may include built-in propellers in ultra-compact designs.

Permanent Magnet Synchronous Motors (PMSMs)

Permanent Magnet Synchronous Motors are high-performance brushless motors that use strong permanent magnets to maintain a consistent magnetic field. This leads to very smooth operation, low vibration, and high torque output per weight. PMSMs are frequently used in industrial drones, professional mapping UAVs, and long-endurance applications where precision and control are critical. Their high efficiency makes them attractive in scenarios where flight time is limited by battery life, such as in inspections of power lines, infrastructure, or large outdoor areas.

They are often used with more advanced control electronics and flight controllers and are a strong choice for future drone designs that prioritize stability, control, and long-term reliability.

3.2.2.2 Applications of Drone Motors

Drone motors serve a wide range of purposes depending on the mission and environment the drone is designed for. Different types of applications place unique demands on motor performance, from flight stability and torque to efficiency and responsiveness. Understanding these use cases helps justify why certain motors are selected for specific drone roles in the SOAR project.

One of the most common applications is in aerial photography and videography. Drones used for professional filming, surveying, or mapping need motors that offer exceptionally smooth and stable performance. This is especially important when capturing high-resolution images or video footage that must remain steady despite wind disturbances or rapid repositioning. Brushless motors are preferred in these scenarios because they produce less vibration, operate more quietly, and offer precise control over speed and acceleration.

Agricultural drones, on the other hand, often carry larger payloads such as fertilizer tanks, sprayers, or multispectral cameras for crop analysis. For these types of drones, motors must be capable of producing significant lift while remaining energy efficient. Heavy-lift motors with high torque are essential to maintain both flight stability and maneuverability when navigating across vast fields or hovering over crops for extended periods.

Delivery drones also require motors that can support substantial weight while offering long endurance and reliability. In these applications, it's not just about lifting the package; it's about sustaining flight over long distances while minimizing energy loss. The motors must be optimized for both performance and efficiency to ensure successful deliveries, especially in urban or remote environments where recharging infrastructure may be limited.

For FPV racing drones, the demands shift entirely toward speed, agility, and fast response time. High-performance brushless motors with high KV ratings are commonly used in these drones because they spin faster and allow the pilot to make rapid maneuvers. These motors are designed to deliver bursts of acceleration and withstand the stress of frequent throttle changes during competitive racing.

Surveillance and security drones require a blend of reliability and discretion. Motors in these drones must support silent or near-silent operation to avoid detection. At the same time, they need to provide steady thrust for stable hovering and quick directional changes when tracking a subject or surveying a perimeter. Quadcopter motors are often used here because of their balanced performance across power, size, and efficiency.

Each of these applications informs the selection of drone motors in the SOAR project. Whether prioritizing lift capacity, endurance, stealth, or agility, the motor system must be tailored to the mission profile. Understanding these diverse use cases helps ensure that the selected motors meet both the technical and operational requirements of our drone swarm.

3.2.2.4 Maintenance and Care for Drone Motors

Proper maintenance is essential to ensure that our drone motors continue to perform reliably throughout the duration of missions and over the full lifecycle of the system. Whether used for aerial photography, payload delivery, or swarm coordination, motors are constantly exposed to environmental stressors and mechanical wear that can affect performance if not managed carefully.

One of the most important maintenance routines is regular cleaning. After each flight, motors should be inspected and cleared of any accumulated dust, dirt, grass, or debris that could interfere with rotation or cause internal damage over time. This is especially critical when flying in rural, wooded, or sandy environments, where particles can easily find their way into the motor housings.

In addition to cleaning, routine visual inspections help catch early signs of wear or malfunction. For brushed motors, it's important to check for erosion of the brushes or discoloration, which are indicators of excessive friction or overheating. Even brushless motors, while more durable, should be examined for unusual sounds, shaft wobble, or bent motor mounts that might affect balance and stability.

To ensure consistent performance, drone motors should also be tested periodically for their rotational speed and thrust output. Simple bench tests using a thrust stand or tachometer can reveal whether the motor is losing efficiency or failing to meet its rated specifications. Early detection of performance issues allows us to replace or repair motors before a flight-critical failure occurs.

Cooling and proper airflow also play a big role in the longevity of motors. Brushless motors generate heat during extended operation, and without sufficient ventilation, they can overheat and degrade. Ensuring that the drone frame design promotes airflow, or integrating heat sinks where necessary, can significantly extend motor life and prevent thermal throttling.

For some of our drone configurations, particularly those involving enhanced agility or vertical takeoff and landing (VTOL) features, motors may be integrated with thrust vectoring mechanisms. These systems redirect the orientation of thrust dynamically, providing fine-grained control over pitch, roll, and yaw. While they improve maneuverability and responsiveness, they also add mechanical complexity, meaning that these assemblies require even more diligent maintenance and lubrication to prevent binding or mechanical wear.

By integrating regular care practices into our operational checklist, we can preserve the efficiency, stability, and safety of our drone fleet while extending the usable life of one of the most critical components in the system.

3.2.2.5 Drone Motor Selection and Suitability for SOAR

Motor Technology Selection

In designing the propulsion system for the SOAR project, selecting the right drone motors is a critical step that directly impacts flight stability, payload capacity, endurance, and overall performance. Our mission requires drones to support on-board AI modules, high-resolution video streaming, and multiple sensors, all of which contribute to a heavier payload and higher power demands. Therefore, our motor selection must balance thrust generation, efficiency, current draw, and heat management while staying within budget and weight constraints. This is our motor technology selection table for SOAR:

Criteria	Brushed DC	Brushless DC (BLDC)	Coreless	Inrunner BLDC	Outrunner BLDC	Switched Reluctance (SRM)
Weight	Moderate	High	Low	Low	High	Moderate
Power Consumption	High	Low	Moderate	Moderate	Moderate	Moderate
Torque Output	Moderate	High	Moderate	Low	High	Moderate
Durability	Low	High	Low	High	High	High
Efficiency	Low	High	Moderate	High	High	High
Speed (RPM)	Moderate	High	High	High	Low	Moderate
Agility	Moderate	High	High	High	Moderate	Moderate
Cooling Needs	Low	Moderate	Low	High	High	High
Payload Capacity	Low	Moderate	Low	Moderate	High	Moderate
ESC Compatibility	Simple	Requires ESC	Simple	Requires ESC	Requires ESC	Complex
Maintenance Needs	High	Low	Moderate	Low	Low	Low
Cost	Low	Moderate	Low	Moderate	High	High
Suitability for SOAR	Low	High	Low	Low	Moderate	Low

Table 3.2.2.5.1 Motor Technology Selection for SOAR

By Authors

Brushless DC motors (BLDC) are the industry standard for modern drones due to their high torque, low maintenance, and superior energy efficiency. For the SOAR drones, we are primarily considering mid- to heavy-lift BLDC motors that offer a low KV rating (between 400 and 800 KV). These motors operate at slower rotational speeds but provide high torque, which is essential for lifting the additional weight of AI processors such as the Jetson Orin Nano, thermal and optical sensors, and long-range communication hardware.

One of the key requirements is a thrust-to-weight ratio of at least 2:1 to ensure responsive maneuverability and flight stability under load. If the total estimated takeoff weight of each drone is approximately 3 kilograms, we aim for motors capable of generating at least 2.5 to 3 kilograms of thrust per motor, allowing for safe and efficient quadcopter configurations. Additionally, motors must be paired with properly sized electronic speed controllers (ESCs) and propellers to optimize performance and prevent overheating or power loss during extended missions.

Ultimately, our choice of motors will be guided by detailed thrust tests, datasheet comparisons, and simulation results to validate real-world performance. We are committed to choosing components that meet SOAR's demanding operational profile, ensuring that each drone can safely and reliably complete its autonomous search and rescue missions.

Motor Product Selection

Selecting the right motor for the SOAR drone is a critical step in achieving the performance, efficiency, and reliability needed for autonomous search and rescue missions. Motors must provide high thrust-to-weight ratios, operate efficiently on 4-6S LiPo power systems, and be compatible with advanced flight controllers and AI modules. The emphasis is on motors with a KV rating between 350-450 KV, which enables higher torque and lower RPM operation. Which is ideal for stable hovering, smooth video capture, and payload lifting. This section compares motors based on weight, thrust, current draw, voltage range, and manufacturer quality to determine the best fit for the mission profile.

T-Motor MN3510

This motor is designed for reliability and balance, offering around 2.5-3 kg of thrust at 360 KV with relatively low weight. It's well-regarded for efficiency and smooth performance, making it a popular choice for long-endurance quadcopters and aerial photography drones. T-Motor’s long-standing reputation for professional-grade UAV motors reinforces its appeal.

X4110S

This motor is a budget-friendly and proven alternative from a previous SOAR team. It provides adequate thrust (~2 kg) at 400 KV and has been tested with similar configurations. While slightly heavier, it offers solid build quality and thermal performance, though it's approaching the edge of thrust capacity for heavier payloads.

3542

This motor is a compact and low-cost high-KV motor typically used in racing or lightweight drones. Its high RPM and low torque profile make it ill-suited for carrying compute modules or sensor payloads.

Criteria	T-Motor MN3510	X4110S	3542
KV Rating	360 KV	400 KV	900-1000 KV
Motor Size & Weight	35×10 mm, 140g	47.5×37.5 mm, 165g	64×35 mm, 155g
Thrust Output	~2.5–3 kg	~2 kg	~2 kg
Voltage Range	4–6S	6S	3–4S
Current Draw	35–45A	45A	52A
Operating Temp.	~50 °C	53 °C	N/A
Company Reputability Index *(1–10 scale)*	9.5	9.0	6.5
ESC Compatibility	High	High	Low
Quality Rating *(MTBF, hours)*	30,000 h	25,000 h	12,000 h
Durability (Cycle Life)	95% integrity after 1000 h	93% after 1000 h	75% after 1000 h
Efficiency	88%	85%	78%
Cost (USD)	$79.99	$46.99	$31.99

Table 3.2.2.5.2 Motor Product Selection for SOAR

By Authors

After evaluating all options, the T-Motor MN3510 (360 KV) was selected as the most suitable motor for the SOAR platform. It offers an ideal balance of torque, efficiency, and weight, while delivering enough thrust (2.5–3 kg per motor) to support a total takeoff weight of around 3–4 kg. Its compatibility with 4–6S LiPo systems, low noise, and reputation for stability make it ideal for autonomous missions involving AI inference, long-range navigation, and sensor fusion. While more expensive than alternatives like the X4110S, its performance and build quality justify the cost for a mission-critical system like SOAR.

3.2.2.6 Summary of Drone Motors

Drone motors are at the heart of everything we hope to accomplish with the SOAR project. From basic flight to complex maneuvers, they are the components responsible for turning electrical

power into the physical thrust needed to get airborne and stay in control. As we explored in the previous sections, the type of motor we chose has a huge impact on how well each drone performs. Whether it's carrying a payload, scouting a wide area, hovering for surveillance, or working as part of a coordinated swarm.

Among all the options, brushless DC motors stand out as the most practical and reliable choice for our system. They offer better efficiency, less maintenance, and a longer lifespan than brushed motors. This makes them ideal for a mission-driven system like ours that needs to be dependable and high-performing across multiple flights. Pairing these motors with the right electronic speed controllers and propellers allows us to fine-tune each drone for its specific role. Whether that be quick maneuvering, lifting a Jetson module and sensors, or flying long distances.

Throughout this section, we've looked at the critical specs that determine motor performance: KV rating, torque, RPM, efficiency, thrust-to-weight ratio, and cooling needs. Each of these plays a role in how our drones will behave in the field. For instance, high-KV motors are fast, nimble and perfect for lightweight scouting drones. On the other hand, low-KV, high-torque motors are essential for drones that need to carry extra equipment or heavier batteries. For SOAR, we're leaning toward a hybrid motor strategy to balance different performance characteristics across the swarm so each drone excels in its own specialized task.

The type of motor we choose also depends on how the drone will be used. We explored examples from aerial photography to agriculture, delivery, surveillance, and FPV racing. Each application puts different demands on the motor, from quiet operation to high thrust or energy efficiency. Understanding those needs has helped us narrow down which motors will best support our objectives, especially in scenarios like search-and-rescue where reliability and endurance are critical.

Maintenance is another important consideration. Even the best motors can degrade over time if not cared for properly. Our plan includes regular cleaning, inspection, and performance testing to catch issues before they affect flight. We're also accounting for cooling and environmental exposure, especially since our drones may need to operate outdoors for long periods. In more advanced use cases, such as VTOL or thrust vectoring, we're factoring in the extra mechanical complexity and how that affects motor reliability.

Overall, selecting the right motors is about more than just specs. It's about understanding how they integrate into the full system: the airframe, the electronics, the mission profile, and the maintenance plan. For SOAR, we're building a diverse fleet of drones with varied capabilities, and our motor choices reflect that need for flexibility, reliability, and performance. By grounding our decisions in the applications, performance requirements, and long-term sustainability of the system, we're setting a solid foundation for the aerial success of the SOAR project.

3.2.2.7 Electronic Speed Controllers (ESCs)

The Electronic Speed Controller (ESC) plays a critical role in a drone's propulsion system. Acting as a bridge between the flight controller and the motors, the ESC interprets throttle signals and modulates the electrical power delivered to each motor. This modulation governs the motor's speed and responsiveness, which in turn affects everything from lift generation and

maneuverability to energy efficiency and system stability. For an advanced system like **SOAR**, where intelligent control, coordinated swarm behavior, and mission endurance are essential, selecting the right ESC becomes more than just a hardware choice, it also becomes a key performance determinant.

Function of an ESC

At its core, the ESC is a power electronics module that converts the direct current (DC) from the drone's battery into a three-phase alternating current (AC) required to drive Brushless DC (BLDC) motors. This process involves rapid switching of transistors using a technique called Pulse Width Modulation (PWM), which controls the average voltage and, in turn, the motor speed.

The ESC receives throttle signals from the flight controller, interprets them, and adjusts the power supplied to each motor accordingly. This allows the drone to ascend, descend, hover, rotate (yaw), or move forward/backward (pitch and roll). For multirotor UAVs, such as quadcopters or hexacopters, each motor requires its own ESC to allow for individual and independent speed control, which is crucial for coordinated flight dynamics.

ESC and Motor Compatibility

When selecting ESCs for a drone, it is essential to ensure compatibility with the motor specifications, especially in terms of:

- Current rating: The ESC must be capable of handling the maximum current draw of the motor without overheating or failing. Using an ESC with too low a current rating can result in permanent damage or fire.
- Voltage range: The ESC must support the battery voltage range used in the drone (e.g., 3S, 4S, 6S LiPo).
- Firmware support: Modern ESCs run on firmware like BLHeli or SimonK, which can offer features like active braking, regenerative braking, and RPM telemetry which is helpful for AI or advanced flight monitoring.

ESC Protocols and Flight Performance

Advancements in communication protocols between ESCs and flight controllers have led to faster and more reliable control. Some commonly used protocols include:

Criteria	D-shot	OneShot125	Multishot	PWM
Precision (µs)	0.25	1.0	0.3	4.0
Latency (ms)	0.05	0.12	0.07	1.00
Speed (kHz)	40	8	32	0.4

Signal Reliability (%)	98%	85%	90%	70%
Hardware Compatibility	Moderate	Moderate	Low	High
Hardware Compatibility (% of ESCs)	85%	90%	60%	100%
Power Efficiency	96%	90%	88%	80%
Error Detection (bit CRC)	16-bit CRC	None	None	None

Table 3.2.2.7 Motor Encoding Protocol Selection for SOAR

By Authors

For a high-performance drone like SOAR, which requires precision and possibly heavy-lift capabilities, using DShot ESCs can significantly improve flight response, especially when handling rapidly changing AI-based control commands from an onboard Jetson.

Thermal Management and Placement

ESCs can generate substantial heat under high loads, especially in drones designed for payload lifting, high-speed flight, or long endurance missions. As such, it's important to mount ESCs in ventilated locations, possibly with heatsinks or integrated cooling. Many drone frames now come with dedicated ESC arms to isolate heat sources and reduce EMI (Electromagnetic Interference).

Integration with Flight Controllers

In some drone designs, ESCs are integrated into the flight controller (AIO FC), simplifying wiring and saving weight. However, for high-powered drones like SOAR, using separate, robust ESCs is more practical to support larger motors and handle higher current. Additionally, some ESCs support telemetry feedback, which can be sent back to the AI module or flight controller to monitor motor performance and detect faults in real time.

System Compatibility: Cube Orange and Jetson AGX Xavier

Our drone architecture relies on the Cube Orange flight controller, a high-performance autopilot board known for its advanced sensor fusion, real-time operating system (RTOS) support, and compatibility with both PX4 and ArduPilot firmware. Complementing this is the Jetson AGX Xavier, which handles AI-based mission control, path planning, and perception algorithms such as image segmentation or obstacle detection using onboard cameras and LIDAR.

The ESC selected must integrate seamlessly with both of these core components. From the Cube Orange perspective, support for DShot protocols and telemetry passthrough is essential, as it ensures accurate motor control and real-time feedback. From the Jetson Xavier's standpoint, telemetry data forwarded via MAVLink allows high-level systems to reason about motor load, system health, and energy usage in real time.

Additionally, compact design, electrical simplicity, and software support are critical for reducing latency and wiring complexity which are two concerns especially relevant to swarm-capable UAVs operating under time-sensitive scenarios like autonomous search and rescue.

3.2.2.8 Electronic Speed Controller (ESC) Selection and Configuration for SOAR

To ensure the chosen ESC was optimal for our system, we analyzed and compared it against several other leading options. The table below presents a breakdown of technical specifications and system integration factors.

Criteria	BLHeli_S (8-bit)	BLHeli_32	APD F4 60F3
Architecture	8-bit MCU	32-bit MCU (STM32)	32-bit Proprietary FET driver
Protocol Support	PWM, Oneshot, DShot150	DShot600, DShot1200, PWM	DShot1200, CAN
Telemetry Support	Limited	Yes (ESC telemetry)	Yes (more robust)
Cube Orange Compatibility	Yes (via PWM/DShot passthrough)	Yes (fully supported in ArduPilot)	Yes (requires CAN setup)
Jetson Companion Integration	Manual parsing	Supported via MAVLink passthrough	Yes, through CAN/MAVLink
Current Rating (per motor)	~20–30A	~35–45A (typical for 4-in-1)	60A continuous
Weight	12g	18g	28g
Integration Simplicity	Simple	Simplified wiring, fewer connectors	Complex (per ESC wiring)
Cost (USD)	$25	$45	$120
Suitability for SOAR	75%	95%	90%

Table 3.2.2.8.1 ESC Type Selection for SOAR System

By Authors

The BLHeli_32 ESC was ultimately chosen not because it was the most powerful or most open solution, but because it strikes the best balance between system compatibility, feature richness, cost-efficiency, and development simplicity. It supports all the key functionalities required for reliable propulsion control in SOAR, including digital throttle signal support, bidirectional telemetry, and a compact form factor.

Its compatibility with ArduPilot and the Cube Orange ensures that our low-level control systems remain stable and well-supported throughout the lifecycle of the project. Its ease of calibration and support for high KV motors also make it suitable for future testing with varied motor types as the swarm design evolves. Additionally, its popularity in the FPV community means there are extensive resources available for debugging, repair, and tuning.

ESC Configuration for SOAR

When selecting ESCs, it's important not only to consider the firmware or current rating, but also the physical configuration of the ESCs. The layout chosen whether a single ESC per motor (1-for-1), dual-channel (2-in-1), or integrated four-channel (4-in-1) can affect wiring complexity, cooling, maintainability, failure modes, and overall space efficiency. Each configuration has its trade-offs depending on the intended application, repair accessibility, and system integration constraints. For SOAR, which prioritizes compactness, modular integration, and streamlined assembly, the configuration type directly influences build feasibility and long-term reliability.

Criteria	1-for-1	2-in-1	4-in-1
Integration Density (ESCs per unit)	1	2	4
Weight Efficiency (per motor)	20g	17g	12g
Connector Count	16	12	8
Ease of Replacement(min)	10 minutes	15 minutes	25 minutes
Cooling Surface Area (cm^2)	14	11	8
Risk of Total Failure	25%	50%	100%
Wiring Length (cm)	120 cm	80 cm	45 cm
Cost per Setup	$80	$95	$110

Space Savings on Frame	Low	Moderate	High
Build Time (min)	90 minutes	75 minutes	45 minutes

Table 3.2.2.8.2 ESC Configuration Selection for SOAR System

By Authors

Based on the analysis, the 4-in-1 ESC configuration was selected for the SOAR system. This choice aligns with the project's goals of reducing overall weight, saving internal frame space, and minimizing wiring complexity. Which are all crucial in a compact, performance-optimized drone. Although the 4-in-1 layout has a higher failure impact (since a failure may affect all four channels), the benefits in terms of simplified installation and space efficiency make it the most suitable option for a quadrotor research platform. The selected BLHeli_32 4-in-1 ESC leverages these advantages and ensures compatibility with both the Cube Orange flight controller and Jetson AGX Xavier companion computer, reinforcing its viability for SOAR's mission objectives.

ESC Product Selection

The electronic speed controller (ESC) is the critical interface between the flight controller and each motor, regulating speed, direction, and braking. For SOAR, the ESC must support modern digital protocols like DShot1200, offer robust telemetry, and integrate well with both the Cube Orange flight controller and the Jetson AGX Xavier. The team also prioritized ESCs with high continuous and burst current ratings, lightweight design, and integrated protection features. In this section, several top-performing ESCs were compared to identify the most reliable and compatible option for the drone's mission profile.

T-Motor F45A 4-in-1

This ESC is a professional-grade BLHeli_32 controller with full support for DShot1200, making it ideal for high-resolution throttle control and telemetry. It supports up to 75A burst current and includes protections for over-current, temperature, and voltage drops. It is lightweight and designed to pair well with T-Motor motors, offering a plug-and-play solution for serious UAV platforms.

Lumenier 60A

This ESC is another strong candidate with excellent current ratings and full BLHeli_32 support. It provides robust protection and supports ArduPilot integration. However, its larger size and lack of BEC output add complexity when integrating with auxiliary electronics or flight controllers that benefit from onboard voltage regulation.

SpeedyBee 60A

This ESC offers a cost-effective option with DShot600 and BLHeli_S firmware. While lighter and more affordable, it lacks BLHeli_32 features such as telemetry and higher resolution throttle signals. It's better suited for racing drones or hobbyist builds where cost and speed matter more than redundancy and data.

Criteria	T-Motor F66A 4-in-1	Lumenier 60A	SpeedyBee 60A
Continuous Current	66A	60A	60A
Burst Current	77A	100A	70A
Voltage Rating	3–6S LiPo	2–6S LiPo	2–6S LiPo
ESC Protocol	DShot1200 (BLHeli_32)	DShot1200 (BLHeli_32)	DShot600 (BLHeli_S)
Weight	18g	27g	10.5g
BEC Output	N/A	N/A	N/A
Protection	Over-Current, Over-Temp, Low-Volt, Signal Loss	Over-Current	Surge
ArduPilot Compatibility	Yes	Yes	Yes
DShot Compatible	Yes	Yes	Yes
Cost (USD)	$138.99	$84.99	$75.99

Table 3.2.2.8.3 ESC Product Selection for SOAR

By Authors

The T-Motor F66A 4-in-1 ESC was selected as the optimal match for the MN3510 motors and the overall SOAR system. Its tight integration with the DShot1200 protocol, compatibility with BLHeli_32 firmware, and built-in protections make it ideal for handling variable loads and extended mission durations. The included BEC output simplifies power routing to the Cube Orange and auxiliary systems, while its lightweight construction maintains the agility needed for precision maneuvers. This ESC provides the stability, monitoring, and performance essential for SOAR's autonomous operations under challenging conditions.

3.2.3 Power System and Energy Source

Selecting the right energy source determines whether your drone can fly long enough, carry the desired payload, and do so safely and reliably. A poorly chosen energy source results in short flight time, poor handling, overheating, a fire risk, or a drone that can't lift the equipment you planned. To make the correct choice, specific parameters need to be established.

Capacity (mAh)

Capacity represents the total amount of electric charge a battery can store, usually measured in milliamp-hours. A higher capacity means the drone can fly longer before needing to be recharged. However, capacity directly adds to the weight of the battery pack, so designers must balance endurance against payload and flight efficiency.

Required energy (Wh)

Required energy is the total energy needed for the drone to complete its intended mission, which is determined by estimating the current draw of all subsystems, such as motors, sensors, and communication modules, over the anticipated flight time. Ensuring the energy source meets or surpasses this requirement is required for safe and reliable operation.

Specific energy (Wh/kg)

Specific energy describes the amount of energy a battery stores per unit of weight, which is a key parameter for drones, as minimizing weight while maximizing endurance is essential. Higher specific energy batteries enable lighter designs and longer flight durations, but may involve trade-offs in terms of safety or cycle life.

Specific power (W/kg)

Specific power measures how quickly energy can be delivered relative to battery weight. Drones often require bursts of high power during maneuvers, takeoff, or carrying payloads. A battery with high specific power ensures the system can supply enough current for these demanding conditions without voltage sag or overheating.

Cycle life

Cycle life refers to the number of charge and discharge cycles a battery can undergo before its capacity significantly degrades, where longer cycle life reduces replacement costs and improves reliability in swarm or multi-mission operations. Although higher-performance chemistries often sacrifice cycle life for energy density.

Charge time

Charge time refers to the duration required to fully recharge a depleted battery, and shorter charge times enhance operational readiness, particularly in rescue or field missions where rapid redeployment is crucial. Fast charging requires compatible chargers and careful thermal management to prevent battery degradation or safety risks.

Safety and Thermal Stability

Safety and thermal stability refer to the battery's resistance to dangerous failures such as thermal runaway, fire, or explosion, especially under stress like physical damage, overcharging, or high current draw. This parameter is critical for operational safety, as a battery failure during flight could result in the total loss of the drone and pose a hazard on the ground. Factors such as the chemical composition, the quality of internal construction, and the presence of a robust Battery Management System (BMS) directly influence a battery's safety, where more stable chemistries may trade off for lower performance metrics.

Cost

Cost encompasses the initial financial outlay required to purchase the battery, which must align with the project's overall budget constraints. A lower cost allows for the acquisition of more units, facilitating parallel testing and providing essential spares, while a higher cost may limit the quantity or force compromises in other system areas. Designers must balance performance and safety against financial limitations, as the most capable batteries often come at a premium, making cost a decisive factor in the final selection.

Availability

Availability describes how easily and quickly a battery can be sourced from suppliers, which directly impacts the project's development timeline and logistical sustainability. A readily available battery can be replaced or supplemented without significant delays, ensuring that testing and iteration can proceed uninterrupted. Conversely, a battery with long lead times or limited stock poses a major risk to project schedules, making widespread availability a key practical consideration alongside technical specifications.

3.2.3.1 Types of Energy Source

The selection of an appropriate energy source is critical for the project, as it directly impacts flight endurance, payload capacity, and overall mission success. For the drone swarm's power needs, rechargeable battery technologies are the primary consideration.

Li-ion

Lithium-ion (Li-ion) batteries are widely used in electronics due to their high energy density and relatively low maintenance requirements, as they have a long cycle life and are generally safe and stable. However, Li-ion batteries typically have moderate discharge rates, making them less ideal for applications that require very high bursts of current, which could be best suited for endurance-focused missions or when weight and reliability are more critical than instantaneous power delivery.

LiPo

Lithium-polymer (LiPo) batteries are the most common choice for UAVs because they offer very high energy density, extremely lightweight construction, and the ability to deliver high discharge rates, making them well-suited for powering motors that demand large amounts of current during flight. The trade-offs are shorter cycle life compared to Li-ion and higher sensitivity to misuse, as LiPo cells can swell, overheat, or even ignite if improperly charged or discharged. Despite these risks, their performance benefits make LiPo packs the standard in drones and RC systems.

NiCd

Nickel-cadmium batteries are an older rechargeable technology that offers excellent robustness and tolerance to abuse. They can handle high discharge rates and have very long cycle lives. However, NiCd cells suffer from the "memory effect," requiring complete discharge cycles for optimal performance, and they have a lower energy density compared to other options, resulting in a heavier pack than Li-ion or LiPo, which because of their weight and inefficiency, NiCd batteries are rarely used in modern UAVs, though they remain viable in rugged applications where durability matters more than efficiency.

Rechargeable Batteries

Rechargeable alkaline batteries are a less common option for high-power systems but offer some advantages in terms of cost and accessibility. They are inexpensive, widely available in standard sizes, and more environmentally friendly than disposable alkaline batteries, as they can be reused multiple times. However, their energy density is relatively low compared to Li-ion or LiPo cells, and they cannot sustain high discharge rates, making them unsuitable for applications that require rapid or sustained power delivery, such as drone motors.

$LiFePO_4$

Lithium Iron Phosphate ($LiFePO_4$) batteries are a type of lithium-ion chemistry that prioritizes safety, thermal stability, and long cycle life. However, their energy density is lower than that of Li-ion or LiPo batteries; they are far more resistant to thermal runaway and degradation. They are increasingly used in robotics, electric vehicles, and heavy-duty UAVs where safety and long-term reliability are critical.

3.2.3.2 Applications of Power System

The power system is a foundational element of any drone, directly influencing its flight duration, payload capacity, and operational reliability. Different mission profiles place unique demands on the energy source, battery management, and power distribution subsystems. Understanding these applications helps guide the design of a power system that meets the specific needs of the SOAR project and ensures mission success across varied scenarios.

One of the most demanding applications is search and rescue (SAR) operations. Drones deployed in SAR must support extended flight times to cover large areas, often while carrying multiple sensors such as thermal cameras, LIDAR, and communication modules. The power system must provide high energy density to maximize endurance, while also supporting peak power demands during takeoff, ascent, or high-wind conditions. Batteries with high specific energy (Wh/kg) and robust discharge capabilities are essential, along with efficient power regulation to minimize losses across subsystems.

In aerial mapping and surveying, drones often follow pre-planned flight paths at consistent speeds and altitudes. These missions prioritize energy efficiency and stable voltage output to ensure sensors like RGB and multispectral cameras operate without interruption. Power systems in these applications benefit from high-cycle-life batteries and intelligent power management that adapts to varying computational loads from onboard processing units, such as the NVIDIA Jetson.

Delivery and logistics drones require a power system capable of supporting significant payload weight over moderate to long distances. This demands batteries with high specific power (W/kg) to handle the increased thrust requirements during takeoff and climb. Additionally, fast-charging capabilities are valuable in logistics operations to minimize downtime between missions. Safety features such as battery management systems (BMS) with overcurrent and thermal protection are critical to prevent failures during flight.

For surveillance and security drones, stealth and endurance are key. These systems often operate at low throttle for extended hovering or slow patrols. Power systems must be optimized for low-power consumption during loitering, while still providing reserve capacity for rapid acceleration when tracking a target. Low-noise power components and efficient DC-DC converters help reduce the acoustic and electromagnetic signature of the drone.

In FPV racing and agile maneuvering applications, the power system must deliver extremely high burst currents to support rapid throttle changes and high-speed flight. This requires batteries with very high discharge rates (C-rating) and low internal resistance. However, these systems often sacrifice energy density for power density, resulting in shorter flight times—a trade-off acceptable in racing but not suitable for endurance-focused missions like SAR.

Each of these applications informs the power system design for the SOAR project. By analyzing the power demands of different operational modes—hovering, cruising, sensing, and communicating—we can tailor the battery selection, voltage regulation, and power distribution to ensure reliable, efficient, and safe swarm operations under real-world conditions.

3.2.3.3 Voltage Regulation and Power Management

A reliable and efficient power management system is critical for the SOAR drones, which host a variety of components with different voltage and current requirements. The main battery pack (e.g., a 4S LiPo, ~14.8V) provides power for the high-current propulsion system, but the sensitive avionics, sensors, and companion computers require stable, lower voltages (e.g., 5V, 3.3V, 12V). Voltage regulators and DC-DC converters are essential for providing these clean, stable power rails from the fluctuating battery voltage.

Two primary types of regulators are considered: Low-Dropout Regulators (LDOs) and Switching Regulators (Buck/Boost Converters). LDOs are simple, provide very clean output with low noise, and are inexpensive. However, they are highly inefficient when the voltage difference between input and output is large, as they dissipate the excess power as heat. This makes them unsuitable for high-current applications or when power efficiency is a priority. In contrast, switching converters are more complex but offer high efficiency (85-95%) by rapidly switching the input voltage on and off and using inductors and capacitors to smooth it into a lower output voltage. This efficiency is paramount for maximizing flight time.

For the SOAR project, where power efficiency directly impacts mission endurance, switching buck converters are the preferred choice for the primary power conversion stages. LDOs may still be used for ultra-low-noise applications powering specific sensors, where their inefficiency is an acceptable trade-off for signal integrity.

The final power management design will utilize a multi-stage approach. A high-efficiency buck converter will step down the main battery voltage to a stable 12V rail for the companion computer and higher-power sensors. Subsequent buck converters will provide 5V and 3.3V rails for the flight controller, MCUs, and communication modules. This architecture ensures high overall efficiency, minimizes heat generation, and provides stable power to all critical subsystems, which is essential for reliable operation and data integrity.

3.2.3.4 Battery Maintenance and life cycle

For the SOAR project, disciplined battery maintenance is essential to ensure safety, maximize lifespan, and guarantee reliable performance during testing and missions. The lithium-polymer (LiPo) batteries selected for their high energy density are sensitive to misuse, and improper handling can lead to rapid degradation, swelling, or even fire hazards. A comprehensive understanding of the battery life cycle and a proactive maintenance routine are therefore critical operational pillars.

The lifespan of a LiPo battery is primarily influenced by several key factors. The Depth of Discharge (DoD) is a major contributor; regularly draining a battery to its minimum voltage severely shortens its useful life. To maximize cycle life, it is recommended to limit the DoD to 80%, preserving the bottom 20% of capacity. The charge and discharge rate (C-Rating) also plays a significant role; consistently operating near the battery's maximum current rating generates excessive heat and accelerates internal degradation. Furthermore, storage conditions are crucial. Batteries should never be stored long-term at full charge or full depletion, as both states cause chemical stress. The ideal storage voltage is approximately 3.8V per cell. Finally, exposure to extreme temperatures during operation or charging can cause permanent damage, necessitating that batteries cool to ambient temperature before any recharge cycle.

A proactive maintenance strategy will be adopted for SOAR to extend battery life and ensure safety. This strategy surpasses minimal care, which simply involves charging and using batteries until failure, and standard care, which includes basic storage charging. Our proactive approach involves rigorous visual inspections before and after each use to check for physical damage or swelling, regular voltage monitoring with a cell checker to ensure all cells within a pack remain balanced, and storing all batteries at their optimal storage voltage in a fireproof container. Additionally, we will maintain a cycle log for each battery pack to track usage and anticipate end-of-life replacement needs. This disciplined methodology is chosen to directly support our project's budget constraints and reliability requirements, ensuring our battery fleet remains a dependable asset throughout the development lifecycle.

3.2.3.5 Battery Selection and Suitability for SOAR

The selection of an appropriate battery is a critical system-level decision that directly impacts the SOAR project's core performance metrics, including flight endurance, payload capacity, and operational safety. The battery must supply sufficient power for the high-current demands of the propulsion system while also supporting the continuous energy needs of the avionics, sensors, and onboard AI computation. After evaluating the common rechargeable battery chemistries, Lithium-Polymer (LiPo) emerges as the most suitable candidate to meet the demanding profile of our search and rescue drone swarm.

The primary alternatives were evaluated against key performance parameters. Lithium-Ion (Li-ion) batteries offer excellent energy density and a longer cycle life, but their lower discharge

rates (C-rating) make them insufficient for the high burst currents required during takeoff and agile maneuvering. Lithium Iron Phosphate (LiFePO4) batteries provide superior safety and an exceptional cycle life but are hampered by a significantly lower energy density, which would result in heavier battery packs and reduced flight times for a given weight. Nickel-Cadmium (NiCd) batteries are robust but are obsolete for this application due to their very low energy density, weight, and pronounced memory effect.

Criteria	Li-ion	LiPo	NiCd	LiFePO4	Rechargeable Alkaline
Capacity (Ah)	2.5–3.5 Ah	3.0–4.0 Ah	0.6–1.0 Ah	1.5–3.2 Ah	0.6–0.9 Ah
Specific Energy (Wh/kg)	150–250	180–265	40–60	90–140	50–80
Specific Power (W/kg)	250–340	300–500	150–250	200–300	50–100
Required Energy (Achievability)	90–95% efficiency	95–98% efficiency	70–80% efficiency	85–90% efficiency	50–60% efficiency
Cycle Life (cycles)	500–1,000	300–500	1,000–1,500	2,000–5,000	50–100
Charge Time (hours)	1.5–3	1–2	5–7	2–3	8–12
Safety and Thermal Stability	60–80°C safe range	50–70°C safe range	60–80°C safe range	250°C thermal runaway threshold	80–100°C safe range
Cost ($/kWh)	$120–$200	$150–$220	$80–$120	$250–$350	$20–$50
Availability	Very High	High	Moderate	High	Moderate

Table 3.2.3.5 Battery Selection Comparison Chart

By Authors

The LiPo battery's superiority for the SOAR project is rooted in its very high specific energy and specific power. This combination allows for a lightweight power source that can simultaneously provide extended flight times and the high burst currents necessary for stable flight under a full sensor payload. While LiPo batteries have a moderate cycle life and require careful handling to mitigate safety risks, these drawbacks can be effectively managed through a disciplined maintenance protocol. This includes using a smart Battery Management System (BMS), adhering to strict storage voltage protocols, and utilizing fireproof containment bags. Therefore, for the SOAR project's need to balance performance, weight, and mission duration, LiPo batteries represent the optimal energy source, providing the necessary power and energy density to enable effective and prolonged autonomous search and rescue operations.

3.2.3.6 Summary of Drone Power Supply

The power supply system forms the foundational pillar of the SOAR drone swarm's operational capability, directly determining mission endurance, payload capacity, and overall reliability. Our investigation into power systems encompassed the energy source, voltage regulation, and power management, culminating in a selected design that prioritizes high performance while incorporating necessary safety measures. The core of this system is the Lithium-Polymer (LiPo) battery, chosen for its superior balance of very high specific energy and specific power. This chemistry provides the optimal compromise, enabling extended flight times through high energy density while delivering the high burst currents required for stable flight under the full weight of sensors, AI computation modules, and communication systems. While other chemistries like LiFePO4 offer advantages in safety and cycle life, their lower energy density would impose a significant weight penalty, directly conflicting with our endurance and agility objectives.

To ensure the LiPo batteries operate safely and reach their maximum lifespan, a proactive maintenance and monitoring strategy is essential. This includes the implementation of a robust Battery Management System (BMS) to prevent over-charge and over-discharge, adherence to strict storage protocols at 3.8V per cell, and meticulous cycle logging. This disciplined approach mitigates the inherent risks of LiPo technology and ensures consistent performance across the swarm. Furthermore, the varying voltage requirements of the onboard electronics are managed by an efficient, multi-stage power regulation architecture. High-efficiency switching buck converters are employed to step down the main battery voltage to stable 12V, 5V, and 3.3V rails, powering the flight controller, companion computer, and sensors with minimal energy loss as heat. This careful regulation is critical for the stability of sensitive avionics and the integrity of sensor data. In conclusion, the selected power system—centered on high-performance LiPo batteries, governed by rigorous safety protocols, and distributed via efficient voltage regulation, provides the necessary energy foundation for the SOAR drones to execute their complex, autonomous search and rescue missions effectively.

3.2.3.6 Cell Requirement for Power Supply

The power system must support a diverse set of components, each with specific voltage and current needs. The avionics define the operational power envelope; where under maximum load conditions, the total system demand reaches approximately 57 V and at the minimum required voltage around 50 V, which ensures all components remain within their specified operating ranges during lower-power states.

Component	Name	Voltage	Current
Flight Controller	Cube Orange	5 V	250 mA

Autonomous Computer	Jetson Agx Xavier	12 V (9 - 20)	250 mA
ESC	T-Motor F66A 4-in-1	25.2V (11.1 - 25.2)	55A
Transmitter Telemetry	3DR SiKTelemetry Radio	5 V	3 A
Transmitter Video	Raspberry Pi 4	5 V (4 - 12)	1.5 A
GPS Module	Here3 / Here3+	5 V	0.7 A
Total (Max)	Jetson = 19V	64.2 V	71.7 A
Total (Min)	Jetson = 12V	57.2 V	71.7 A

Table 3.2.3.6 Component Voltage & Current Requirements chart

By Authors

To meet these requirements, the selected power source is a 6S LiPo battery which will power the electronic speed control and a 10S to power the the flight controllers and radios; however those components will need converters to ensure the proper input is coherent with the component, therefore a voltage regulator will be needed to achieve optimal use of the equipment.

3.2.3.7 Summary of Drone Power Supply

The power supply system forms the foundational pillar of the SOAR drone swarm's operational capability, directly determining mission endurance, payload capacity, and overall reliability. Our investigation into power systems encompassed the energy source, voltage regulation, and power management, culminating in a selected design that prioritizes high performance while incorporating necessary safety measures. The core of this system is the Lithium-Polymer (LiPo) battery, chosen for its superior balance of very high specific energy and specific power. This chemistry provides the optimal compromise, enabling extended flight times through high energy density while delivering the high burst currents required for stable flight under the full weight of sensors, AI computation modules, and communication systems. While other chemistries like LiFePO4 offer advantages in safety and cycle life, their lower energy density would impose a significant weight penalty, directly conflicting with our endurance and agility objectives.

To ensure the LiPo batteries operate safely and reach their maximum lifespan, a proactive maintenance and monitoring strategy is essential. This includes the implementation of a robust Battery Management System (BMS) to prevent over-charge and over-discharge, adherence to strict storage protocols at 3.8V per cell, and meticulous cycle logging. This disciplined approach mitigates the inherent risks of LiPo technology and ensures consistent performance across the

swarm. Furthermore, the varying voltage requirements of the onboard electronics are managed by an efficient, multi-stage power regulation architecture. High-efficiency switching buck converters are employed to step down the main battery voltage to stable 12V, 5V, and 3.3V rails, powering the flight controller, companion computer, and sensors with minimal energy loss as heat. This careful regulation is critical for the stability of sensitive avionics and the integrity of sensor data. In conclusion, the selected power system—centered on high-performance LiPo batteries, governed by rigorous safety protocols, and distributed via efficient voltage regulation, provides the necessary energy foundation for the SOAR drones to execute their complex, autonomous search and rescue missions effectively.

3.3 Communication Technology

Communication technology forms the backbone of the SOAR system, enabling both individual drones and the swarm as a whole to function as a cohesive, intelligent unit. Two primary categories define communication within the project: on-board communication, which governs the flow of data between sensors, processors, and controllers inside each drone; and long-distance communication, which manages information exchange between drones and with the ground control station. On-board communication protocols such as UART, SPI, and I2C ensure that sensors deliver critical data to flight controllers with the speed and reliability needed for stable flight and perception. Meanwhile, long-distance communication methods, including RF, Wi-Fi, LoRa, and mesh networking, allow the swarm to coordinate across large search areas and relay mission-critical information back to human operators. Together, these technologies establish the foundation for real-time situational awareness, coordinated search patterns, and mission resilience in unpredictable environments.

3.3.1 Communication Parameter

Range

Range defines the maximum distance over which a drone can accurately send and receive data (i.e. telemetry, control commands, and video signals) from the ground station or between other drones in its swarm. Range is determined by several factors such as transmission power, antenna quality and orientation, frequency band, and environmental interference. For our purposes, maintaining a robust communication range is imperative to ensure continuous control, coordination, and data exchange during swarm and SAR operations. Longer and steadier ranges allow drones to cover larger search areas (needed for our target distance coverage), maintain formation in swarm operations, and transmit sensor telemetry without interruption.

Data Rate

Data rate, or bandwidth, refers to the amount of information that can be transmitted per second between a drone and its ground station or between drones in a swarm. It is typically measured in bits per second (bps). This metric determines how quickly telemetry, control signals, and sensor data (i.e. GPS coordinates, video feeds, and environmental readings) can be sent and received. For our purposes, maintaining an adequate bandwidth is essential for real-time responsiveness and coordination, specifically since each drone will transmit high-frequency data from multiple sensors such as LiDAR, RGB, and thermal cameras. Faster bandwidth allows for faster updates, smoother video transmission, and more accurate swarm synchronization. These are all critical aspects of our swarm SAR operations. The tradeoff for all the benefits mentioned just now is increased power consumption and reduced range. There is a balance that must be found so that we can have sufficient range while still having fast data rates. This balance ensures stable, efficient, and support for simultaneous data streams. It also enables precise coordination and situational awareness across all drones in the swarm.

Latency

Latency refers to the time delay between data being sent and when it is received by a transmitter and receiver. In the context of our SOAR Project, minimizing latency is crucial for maintaining real-time communication and control across the swarm. Low latency ensures that commands from the ground station (or between drones) are executed at a lightning pace, allowing for precise coordination, dynamic obstacle avoidance, and synchronized flight maneuvers. Responsiveness is imperative for our purposes, where delays of even a fraction of a second can affect swarm coordination, navigation accuracy, or mission effectiveness. High latency can cause sluggish reactions in the swarm. This can be seen in desynchronized swarm behavior, unstable flight patterns, or sluggish reading of our telemetry data. It is highly undesirable for our use case.

Power consumption

Power consumption refers to the amount of electrical energy required to operate a drone's communication systems, including transmitters, receivers, and signal processors. In the SOAR Project, minimizing power consumption is critical because every watt used by communication modules reduces the energy available for flight, sensors, and payload operations. Lower power usage helps extend flight endurance, allowing drones to remain airborne longer during search and rescue missions, where extended coverage and persistence are essential. Efficient communication systems not only conserve battery life but also help reduce heat generation and system load, improving overall reliability. Balancing communication performance with low power consumption ensures that the SOAR swarm can maintain continuous connectivity and coordination without compromising mission duration or operational efficiency.

Reliability

Reliability refers to the consistency and stability of a communication system's connection, even under challenging environmental conditions such as obstacles, interference, or signal loss during flight. In the context of the SOAR Project, reliability is vital to ensure that each drone maintains

a constant and dependable link with the ground station and other swarm members. A highly reliable communication system prevents data dropouts, control loss, and coordination failures, all of which could jeopardize mission safety and performance. During complex search and rescue operations, where conditions can change rapidly and signal paths may be obstructed, reliable communications guarantee continuous data exchange and control, allowing the swarm to operate cohesively and respond effectively to dynamic situations. Ultimately, strong reliability ensures that our fleet can perform critical tasks with precision, resilience, and operational confidence in real-world environments.

3.3.2 Types of Wireless Networking Protocol

The effectiveness of the search and rescue is utterly dependent on the communication, which serves as the backbone for real-time information delivery. Reliable transmission of essential information, such as GPS coordinates, live video streams, and telemetry, is indispensable for both navigation and the core mission of locating individuals. Without these continuous data links, the swarm's ability to assess its environment and execute coordinated search patterns would be compromised, where the capacity to dynamically issue flight path adjustments is critical for maintaining control and adapting to unpredictable obstructions.

RF

Radio Frequencies (RF) offer excellent long-range communication, making it highly reliable for transmitting critical telemetry and command data between drones and the ground station over distances of several kilometers, which is vital for Search and Rescue operations. However, its lower data rates make it unsuitable for high-bandwidth tasks, such as real-time video streaming, and careful frequency management is required to avoid interference with other systems.

Wi-Fi

Wi-Fi provides high-bandwidth data transfer, ideal for streaming live video feeds from optical or thermal cameras and for rapid sharing of large sensor datasets within the swarm; however its main drawbacks are relatively short range and high power consumption, and its performance can degrade significantly due to signal obstruction or interference from other networks, making it less reliable for maintaining essential control connections in cluttered or crowded RF environments.

Bluetooth

Bluetooth is advantageous for its low power consumption and cost, making it suitable for short-range, low-energy tasks such as initial drone configuration or firmware updates when in proximity. However, its significantly restricted range and vulnerability to interference render it

impractical for any in-flight communication between drones or with the ground station, excluding it from the core swarm networking strategy.

LoRa

Long Range (LoRa) technology is exceptional for its remarkably extended range capabilities and very low power consumption, enabling drones to transmit small packets of essential data, such as GPS coordinates or emergency alerts, over long distances. The trade-off is an extremely low data rate, which prevents it from being used for any real-time control or sensor data fusion, limiting its role to a secondary, long-range beacon or backup communication channel.

Mesh Network

Mesh Network protocols are ideally suited for swarm applications, as they create a resilient, self-healing network where each drone can act as a node, relaying data for others to extend the overall communication range and eliminate single points of failure, which ensures robust inter-drone coordination even if the link to the ground station is lost. The primary challenge is the added complexity in implementing and managing the network routing protocols to maintain low latency for time-sensitive swarm commands.

3.3.3 Telemetry Protocol Selection

For the SOAR project, selecting an appropriate telemetry protocol is critical for ensuring reliable, low-latency communication between the drones and the ground control station. The protocol must support real-time transmission of essential data—such as GPS coordinates, battery status, sensor readings, and system alerts—while operating efficiently within the constraints of a distributed, mobile swarm. Key considerations include bandwidth efficiency, interoperability with existing drone software stacks, support for multi-vehicle communication, and resilience to packet loss or interference in dynamic outdoor environments.

Zigbee

Zigbee is a low-power, low-data-rate wireless protocol known for its support of mesh networking, which allows devices to relay data for one another, extending network coverage without requiring a central hub. This makes Zigbee suitable for creating resilient local networks among drones operating in close proximity. However, its limited bandwidth makes it unsuitable for high-throughput applications like video streaming, and its relatively short range and susceptibility to interference in crowded 2.4 GHz bands may restrict its use in large-scale or noisy environments.

APRS

APRS (Automatic Packet Reporting System) is a ham radio-based protocol designed for real-time, simple communication of short data packets, such as position reports and short messages. It excels in long-range, low-power scenarios and can operate independently of cellular or internet infrastructure, making it useful for backup or emergency communication in remote areas. On the downside, APRS offers very low data rates, lacks support for complex or bidirectional data exchange, and requires amateur radio licensing in most regions, limiting its practicality for primary telemetry use.

MAVlink

MAVLink (Micro Air Vehicle Communication Protocol) is a lightweight, open-source protocol specifically designed for UAV systems. It is highly efficient, supports a wide range of common drone telemetry and command messages, and is natively integrated with popular flight stacks like PX4 and ArduPilot. MAVLink's simplicity, low overhead, and broad ecosystem support make it the de facto standard for drone telemetry. However, it is less suited for high-bandwidth data such as video or large sensor streams and may require companion protocols or additional modules for advanced swarm logic or complex data fusion tasks.

Criteria	Zigbee	APRS	MAVlink
Range (km)	0.3–1.0	100+	5–15
Data Rate (kbps)	250	1.2	57600
Power Consumption (W)	0.2	1.0	2.5
Frequency	915 MHz / 2.4 GHz	144.39 MHz	915 MHz

Table 3.3.3 Telemetry Communication Protocol Selection Chart

By Authors

3.3.2.4 Telemetry Radio Selection

Selecting the appropriate telemetry radio is essential for maintaining reliable, low-latency communication between the drones and the ground control station (GCS). The chosen radio must support the MAVLink protocol, offer sufficient range for beyond-visual-line-of-sight (BVLOS) operations, and operate effectively in environments with potential signal interference. For the

SOAR project, three candidate telemetry radios were evaluated: the HolyBro RFD900x, the Sik Telemetry radio, and the ESP32. Each offers a distinct balance of range, data rate, power consumption, and integration complexity.

HolyBro RFD900x

The HolyBro RFD900x is a high-performance, long-range telemetry radio operating in the 900 MHz frequency band, making it well-suited for expansive search and rescue missions. It supports data rates up to 64 kbps and can achieve ranges of up to 40 km in ideal line-of-sight conditions, ensuring robust connectivity even in remote or obstructed environments. The radio is fully compatible with MAVLink and integrates seamlessly with the Cube Orange flight controller and ArduPilot/PX4 firmware. Additionally, its support for mesh networking allows it to relay data between drones, enhancing swarm coordination. However, the RFD900x is relatively expensive compared to other options and requires careful antenna selection and placement to maximize performance. Its higher power consumption may also impact flight time if not managed with efficient power regulation.

Sik Telemetry Radio

The Sik Telemetry radio, often implemented on platforms such as the 3DR Radio, is a versatile and cost-effective solution for medium-range drone communication. Operating in the 915 MHz band (or 433 MHz in some regions), it provides a reliable link with ranges typically between 1–3 km, which may be sufficient for smaller-scale swarm operations. The Sik radio is lightweight, easy to configure, and widely used within the open-source drone community, offering strong compatibility with MAVLink and popular ground control software like Mission Planner and QGroundControl. Despite these advantages, the Sik radio offers lower maximum data rates compared to the RFD900x, which can limit its effectiveness for high-bandwidth applications. Its range is also more susceptible to degradation from physical obstructions, and it lacks native support for advanced features such as dynamic mesh networking.

ESP32

The ESP32 is a highly versatile system-on-chip (SoC) with integrated Wi-Fi and Bluetooth capabilities, which can be programmed to handle custom telemetry and control tasks. Its main advantages include low cost, widespread availability, and a large developer community, which simplifies prototyping and customization. For short-range applications or within a localized Wi-Fi network, the ESP32 can provide high data rates suitable for video streaming or dense sensor data exchange. However, its effective range is severely limited compared to dedicated long-range telemetry radios, typically extending only a few hundred meters in open environments. The ESP32 also consumes significant power when operating at high data rates, and its performance can degrade rapidly in the presence of RF interference or network congestion. While useful for secondary communication roles or development testing, the ESP32 is not recommended as the primary telemetry solution for SOAR’s long-range swarm missions.

HolyBro RM1

The HolyBro RM1 is a versatile and modern radio that combines a standard SiK-based telemetry link with a low-bandwidth LoRa (Long Range) backup channel. This dual-system approach is highly advantageous for swarm security; the high-speed SiK link handles standard telemetry and commands, while the LoRa channel can transmit critical failsafe messages at extreme ranges with very low power. It is more compact and cost-effective than the RFD900x. The main trade-off is that the primary SiK link's range and data rate are slightly lower than the RFD900x. For SOAR, the RM1 offers a smart balance of performance and robust redundancy, making it an excellent candidate for all swarm members.

Criteria	HolyBro RFD900x	Sik Telemetry Radio	ESP32	HolyBro RM1
Range (km)	40	3–5	0.2	10
Data Rate (kbps)	500	115	2000	250
Power Consumption (W)	1.5	0.8	3.2	1.0
Reliability	99%	92%	80%	97%
Integration Time	15 minutes	10 minutes	25 minutes	12 minutes

Table 3.3.2.4 Telemetry Radio Selection Comparison Chart

By Authors

3.3.4 Video Streaming Protocol

Video streaming is a critical capability as it enables real-time transmission of visual data from drone-mounted cameras, such as RGB, thermal, and EO sensors, to the ground control station (GCS) or between drones. The choice of streaming protocol directly impacts latency, reliability, and overall mission effectiveness, particularly in dynamic search and rescue scenarios where timely visual feedback can influence decision-making. Key considerations include low latency for real-time awareness, resilience to packet loss in unstable RF environments, and compatibility with the onboard computer and communication hardware. The following protocols represent the most viable options for integrating live video into the SOAR communication architecture.

RTSP

The Real-Time Streaming Protocol (RTSP) is a widely adopted network control protocol designed for use in entertainment and communications systems to manage streaming media servers. It establishes and controls media sessions between endpoints, allowing commands such as play, pause, and record. While RTSP supports standard codecs like H.264 and H.265, it typically introduces higher latency (1–5 seconds) and offers limited error resilience, making it less ideal for highly dynamic drone operations where low latency is critical. However, its broad compatibility with existing video management systems and relative simplicity can make it a practical choice for secondary or non-critical video feeds.

WebRTC

Web Real-Time Communication (WebRTC) is an open-source project and set of standards that enables real-time communication of audio, video, and data directly between web browsers and devices without requiring plugins. Designed for peer-to-peer low-latency communication, WebRTC offers sub-500-millisecond latency and strong error resilience, making it well-suited for interactive applications. Its primary drawback is setup complexity, as it often requires signaling servers and network traversal techniques (like STUN and TURN). For SOAR, WebRTC could be valuable for enabling low-latency video interaction between the GCS and drones, especially when integrated into a web-based dashboard.

SRT

Secure Reliable Transport (SRT) is an open-source video transport protocol that excels in maintaining low-latency, high-quality video streams over unpredictable networks, such as the internet or long-range RF links. It uses forward error correction (FEC) and ARQ (Automatic Repeat Request) mechanisms to minimize packet loss, achieving sub-500-millisecond latency even under challenging conditions. SRT supports modern codecs like H.264 and H.265 and is relatively straightforward to set up compared to WebRTC. For SOAR, SRT represents a robust option for reliable drone-to-ground video feeds where both latency and signal stability are important.

TCP

Transmission Control Protocol (TCP) is a core transport-layer communication protocol that provides reliable, ordered, and error-checked delivery of data between devices over an IP network. Unlike UDP, TCP establishes a persistent connection through a three-way handshake and ensures all packets are received in sequence, retransmitting any lost data. This reliability makes TCP ideal for applications where data integrity is more important than speed, such as configuration uploads, log transfers, or cloud-based control interfaces. However, the built-in acknowledgment and retransmission mechanisms introduce additional latency, making TCP less suitable for real-time video streaming or rapid control feedback in dynamic UAV environments. Within the SOAR architecture, TCP may still serve supporting roles in non-time-critical communications, such as remote configuration or mission data synchronization.

UDP

User Datagram Protocol (UDP) is a lightweight, connectionless transport protocol widely used for time-sensitive applications such as live video streaming, gaming, and telemetry. Unlike TCP,

UDP transmits data packets without establishing a persistent connection or waiting for delivery acknowledgments, significantly reducing transmission latency. This design makes UDP highly suitable for drone-based video transmission, where real-time performance is prioritized over perfect reliability. While it lacks built-in mechanisms for error correction or packet reordering, its simplicity enables direct, low-overhead communication between endpoints. In the SOAR system, UDP can serve as the foundational transport layer for custom or hybrid streaming solutions, such as OpenHD, providing a fast and efficient means of transmitting video and telemetry data simultaneously.

Criteria	RTSP	WebRTC	SRT	UDP	TCP
Latency	1-5 s	<500 ms	<500 ms	<100 ms	>500 ms
Error Resilience	Low	High	Very High	Low	Very High
Setup Complexity	Moderate	High	Low	Low	Low
Codec Support	H.264, H.265	H.264	H.264, H.265	H.264, H.265	H.264, H.265

Table 3.3.4 Video Streaming Communication Protocol Selection Chart

By Authors

3.3.5 Video Stream Transmitter Selection

As it directly impacts the quality, latency, and reliability of real-time video feeds from the cameras to the ground control station. The transmitter must efficiently encode and transmit high-definition video from sensors such as RGB, thermal, or EO cameras, while operating within the power and weight constraints of the drone. Key considerations include encoding capability, power consumption, computational resources, and ease of integration with the existing NVIDIA Jetson and communication stack.

Raspberry Pi 4

The Raspberry Pi 4 is a widely adopted single-board computer known for its strong community support and versatile I/O capabilities. It supports H.264 encoding and offers RAM options from 2GB to 8GB, providing sufficient memory for concurrent video processing and transmission tasks. With a power consumption ranging from 3W to 7W, it strikes a reasonable balance between performance and energy efficiency. Its main advantages include widespread availability, extensive documentation, and compatibility with a variety of camera modules. However, its lack

of native H.265 encoding support may limit video compression efficiency compared to more modern alternatives.

Orange Pi 5

The Orange Pi 5 is a more powerful alternative, offering enhanced processing capabilities and support for both H.264 and H.265 video encoding. With RAM configurations ranging from 4GB to 16GB, it provides ample memory for handling high-resolution video streams and additional onboard processing. Its power consumption is higher, typically between 5W and 12W, which must be factored into the drone's power budget. The Orange Pi 5 is well-suited for missions requiring efficient video compression and higher computational throughput, though it may come at a higher cost and with a less mature software ecosystem compared to the Raspberry Pi.

Libre Le Potato

The Libre Le Potato is a low-cost, energy-efficient single-board computer designed for lightweight applications. It supports H.264 encoding and includes 2GB of RAM, making it adequate for basic video streaming tasks. With a power draw of only 2W to 5W, it is one of the most power-efficient options considered, which is advantageous for extending flight time. However, its limited RAM and lack of H.265 support may restrict its use in more demanding video streaming scenarios, particularly where multiple high-resolution streams are required.

NanoPi R6S

The NanoPi R6S is a high-performance computing platform capable of handling demanding video encoding tasks with support for both H.264 and H.265 codecs. It offers 4GB to 8GB of RAM and is designed for applications requiring high data throughput and low-latency processing. However, its power consumption is significant, ranging from 5W to 15W, and it is the most expensive option among those evaluated. While it provides top-tier performance, its cost and power requirements may make it less suitable for cost-sensitive or power-constrained multi-drone deployments.

Criteria	Raspberry Pi 4	Orange Pi 5	Le Potato	NanoPi R6S
RAM	2-8 GB	4-16 GB	2 GB	4-8 GB
Encoding	H.264	H.264, H.265	H.264	H.264, H.265
Power Consumption	3-7 W	5-12 W	2-5 W	5-15 W
Price	$82	$100	$35	$180

Table 3.3.5 Video Streaming Transmitter Selection Chart

By Authors

3.3.6 Drone-to-Drone Communication

Drone-to-drone communication is the foundation for decentralized swarm intelligence, enabling real-time coordination, collaborative task allocation, and shared situational awareness without relying solely on the ground control station; where this link will be primarily established using the 3DR SiK Air telemetry radios; which allows data packets from one to another drones in the swarm to be transmitted, maintaining coordination even if the direct link to the GCS is temporarily lost or if a drone moves behind an obstacle. The communication protocol will utilize MAVLink for transmitting essential, low-latency data. This includes each drone's GPS coordinates, velocity to form a common operational picture. By leveraging the telemetry radios for this purpose, the system maintains a simple, robust, and low-latency communication backbone dedicated to the core functions of swarm coordination, without the overhead of streaming high-bandwidth video between drones.

3.3.7 Drone-to-Base Communication

Drone-to-base communication forms the critical command, control, and feedback loop between the autonomous swarm and the human operators at the Ground Control Station. This bidirectional link is implemented using a dual-channel approach to efficiently handle different types of data based on their bandwidth and latency requirements. The primary channel for command and telemetry utilizes the same 3DR SiK telemetry radios employed in the mesh network. This provides a robust, long-range link for transmitting all essential telemetry data, including drone status and target metadata, such as GPS coordinates and classification confidence of a detected object, from the swarm to the GCS. Concurrently, this channel is used to send mission commands, dynamic tasking updates, and emergency overrides from the operator to the swarm.

The secondary, high-bandwidth channel is dedicated to video streaming, which upon operator request, individual drones will stream live video from their specialized sensors directly to the GCS. This stream will be encoded using the H.264 codec on the companion computer and transmitted using a low-latency protocol like OpenHD, chosen for its resilience on unstable links. This dual-method architecture ensures that the low-bandwidth, high-priority command and telemetry link remains stable and responsive, while the high-bandwidth video feed is activated on-demand to provide operators with visual context for verification and decision-making, without unnecessarily congesting the primary swarm network.

3.4 Software of Drone

The drone software integrates flight control, perception, and swarm coordination into one system. Each drone runs the PX4 flight stack for stability and navigation, while ROS2 and MAVLink handle inter-drone communication and data exchange. Machine learning models optimized for onboard systems like the Jetson Xavier enable real-time object detection and classification using thermal and optical inputs. Through collaborative algorithms, drones share data, plan routes, and adapt to findings in real time. A ground control interface using

QGroundControl and a custom dashboard provides live monitoring and mission management, allowing the swarm to operate autonomously and efficiently in search-and-rescue missions.

3.4.1 Computer Vision Frameworks

For SOAR to achieve the goal of identifying and locating targets such as humans, vehicles, and other objects of interest, the use of computer vision and robust pre-existing frameworks is critical. Our drones will gather sensor input from a variety of payloads—including EO cameras, thermal cameras, and LiDAR—and must be able to process this information with robustness to environmental factors such as lighting changes, shadows, occlusion, and varying altitudes. Below are several computer vision frameworks that can be leveraged on the NVIDIA Jetson Orin Nano platform:

OpenCV

OpenCV is the classic toolkit for image processing, feature extraction, and basic object detection/tracking. It is widely used for pre- and post-processing tasks such as resizing input images, normalization, distortion correction, filtering, and optical flow. OpenCV provides flexibility and integrates well with deep learning frameworks. While traditional OpenCV operations can run on CPU only, modern builds also support CUDA for GPU acceleration. This makes OpenCV valuable for preprocessing and real-time optimization in our drone pipeline, though by itself it is not sufficient for robust target detection.

TensorFlow / TensorFlow Lite

TensorFlow is an open-source deep learning framework developed by Google that provides a comprehensive ecosystem for building, training, and deploying machine learning models. It supports a wide range of applications such as image recognition, object detection, natural language processing, and large-scale distributed training. With the integration of Keras as its default high-level API, TensorFlow offers both ease of use for beginners and the scalability required for advanced research and production systems.

TensorFlow Lite (TFLite) is a lightweight version of TensorFlow designed specifically for deployment on resource-constrained and embedded devices. It provides optimized object detection APIs and pre-trained models such as SSD (Single Shot Detector), MobileNet-SSD, and optimized YOLO variants. TFLite also supports model quantization and hardware acceleration options, enabling lower power consumption and faster inference. This makes it particularly well-suited for onboard inference on platforms like the NVIDIA Jetson Orin Nano, where efficiency and real-time processing are critical.

PyTorch

PyTorch is a modern deep learning framework, developed by Meta’s AI Research lab, is known for high performance and flexibility in model training and inference. It is widely used with state-of-the-art object detection and segmentation models, including YOLOv3, YOLOv4, YOLOv5, and more recent variants. On the Jetson Orin Nano, PyTorch benefits from GPU acceleration, making it effective for running heavier models or fine-tuned custom models trained

for drone-specific perspectives. PyTorch is especially advantageous with custom datasets, retraining, or advanced tasks like multi-modal sensor fusion (e.g., combining EO and thermal imagery).

3.4.2 Computer Vision Algorithms

Computer vision algorithms are designed to enable machines to interpret and understand visual information from the world. In the context of object detection, these algorithms process image or video inputs to identify, classify, and localize objects of interest. This typically involves generating bounding boxes around detected targets (e.g., humans, vehicles, or infrastructure) and assigning class labels that describe what the object is. Different algorithms use different strategies—some prioritize speed and efficiency for real-time applications, while others emphasize accuracy and robustness even under challenging conditions such as occlusion, lighting changes, or high-altitude imagery. For drone-based surveillance like SOAR, these algorithms form the core of target detection, transforming raw sensor data into actionable insights for navigation, tracking, and mission decision-making.

YOLO

YOLO (You Only Look Once) is a family of object detection models designed for real-time inference. Unlike traditional multi-stage detectors, YOLO formulates detection as a single regression problem—directly predicting bounding boxes and class probabilities in one pass. This design enables YOLO to achieve both high speed and strong accuracy, which is critical for SOAR's aerial surveillance, where drones must detect humans, vehicles, and other objects in dynamic environments.

SSD

SSD is a single-stage object detection algorithm. Unlike two-stage methods, SSD detects objects in a single forward pass by predicting bounding boxes and class probabilities directly from feature maps at multiple scales. This makes it fast and efficient, well-suited for real-time applications and embedded devices. However, SSD has a lower accuracy than YOLO or two-stage models, particularly for detecting small or distant objects.

R-CNN

R-CNN (Region-Based Convolutional Neural Networks) are two-stage object detection algorithms. The first stage generates region proposals (candidate areas likely containing objects), and the second stage classifies and refines these proposals. This approach delivers very high accuracy, especially for small, overlapping, or occluded objects. However, the multiple processing stages make R-CNN models computationally heavy and slow, limiting their use in real-time embedded applications like SOAR. They are better suited for tasks where accuracy is more critical than speed.

NanoOWL

NanoOWL is a modern vision-language detection pipeline derived from OWL-ViT (Open-Vocabulary Localization with Vision Transformers) optimized for edge compute, open-vocabulary detection, and real-time performance. Open-vocabulary allows the user to input text prompts and NanoOWL tries to detect the input classes. Additionally, the text prompt is nested, allowing the user to add more detail within the prompt: [detect human[detect red hat, black glove]]. NanoOWL gives the user flexibility with the text prompts, allowing the user to detect new classes without having to retrain the model. However, this flexibility comes at the price of lower accuracy compared to more traditional models like YOLO.

3.4.3 Evolution of Computer Vision Frameworks/Algorithms

OpenCV

OpenCV's first stable release, version 1.0, came out in 2006. Over the following decade, version 2.x introduced Python bindings (a translator that lets Python talk to libraries written in other languages) and machine learning modules, making OpenCV more accessible to researchers and developers. In 2015, version 3.x added the Deep Neural Network (DNN) module and improved GPU acceleration, opening the door to early deep learning integration. By 2018, version 4.x expanded CUDA support and further optimized performance for deep learning backends, solidifying OpenCV's role as a bridge between classical computer vision and modern AI. The latest release, version 5.x (2024), is still in development but builds on version 4.x functionality while preparing for deeper integration with AI workflows.

Version	Pros	Cons	Summary
2.x	Introduced Python bindings, added ML modules	Limited GPU support, weak deep learning capabilities	Major step forward for accessibility and adoption in academia/industry
3.x	Added DNN modules, better GPU acceleration; improved performance	Early DNN support limited compared to PyTorch / TensorFlow	First version to seriously integrate deep learning and GPU acceleration
4.x	Full CUDA support, optimized for modern deep learning backends; better overall performance	More Complex to build and compile, still weaker to specialized DL frameworks	Best blend of classical CV and modern AI tools
5.x	Builds on 4.x with stronger AI integration,	Still under development, not yet widely adopted or	Aims to make OpenCV more AI-centric without

	modernized APIs	stable	losing legacy tools

Table 3.4.3.1 Evolution with Pros and Cons on OpenCV Versions

By Authors

TensorFlow / TensorFlow Lite

TensorFlow's first stable release, version 1.0, was launched by Google in 2015 and introduced static computation graphs, which made it powerful but often complex for developers to use. In 2017, TensorFlow Lite (TFLite) was released to provide optimized inference on mobile and embedded devices. The major leap came in 2019 with TensorFlow 2.0, which simplified the framework by making Keras—a high-level, user-friendly API for building and training neural networks—the default interface. This change made TensorFlow more accessible while retaining its scalability for research and production. Between 2021 and 2023 (versions 2.5–2.14), TensorFlow introduced significant improvements in hardware acceleration, enabling better use of GPUs, TPUs, and specialized inference hardware. The most recent release, TensorFlow 2.15 (2024), continues to build on the 2.x series and is still under active development.

Version	Pros	Cons	Summary
1.x	Highly flexible with static computation graphs	Complex to learn, difficult debugging	Strong in research / industry but difficult to learn
TensorFlow Lite	Lightweight, optimized for mobile / embedded devices	Limited features set compared to full TensorFlow	Specialized for edge inference, not training
2.0	Keras as default, easier debugging, backwards compatible	Migration challenges for older projects with TF 1.x	Major usability leap, approachable to beginners
2.5 - 2.14	Improved GPU / TPU support, better deployment pipelines, hardware acceleration	Incremental improvements, fragments ecosystem	Solidified TensorFlow as a production-scale framework
2.15	Latest release, refined hardware acceleration	Still evolving, some features less mature compared to PyTorch ecosystem	Stable modern version, balances research and production

Table 3.4.3.2 Evolution with Pros and Cons on TensorFlow / TensorFlow Lite Versions

By Author

PyTorch

PyTorch was first released in 2016 and quickly gained popularity in the research community for its flexibility and ease of use. Unlike TensorFLow 1.x, PyTorch used dynamic computation graphs, allowing developers to build and modify models on the fly, making debugging and experimentation much easier. In 2018, PyTorch 1.0 became the first stable release, consolidating features for both research and production. From 2020 to 2022 (versions 1.5 - 1.12), PyTorch introduced incremental improvements such as better ONNX (open format built to represent ML models) export, distributed training, and deployment support. In 2023, version 2.0 released adding TorchDynamo, a graph-capture and compilation system that dramatically boosted performance while keeping the familiar PyTorch workflow. The latest version, PyTorch 2.1, continues to optimize distributed computing, hardware acceleration, and model deployment across a wide range of platforms.

Version	Pros	Cons	Summary
1.0	First stable release, stronger ecosystem	Smaller community than TensorFlow (at the time)	Made PyTorch a serious competitor to TensorFlow
1.5 - 1.12	Added ONNX support, improved distributed training, deployment pipelines	Incremental updates, deployment still lagged TensorFlow	Expanded beyond research into industry applications
2.0	Introduced TorchDynamo (compiler), large performance boost	Some growing pains with new features, ecosystem adapting	Balanced research flexibility with production speed
2.1	Improvements to distributed computing, hardware acceleration	Ecosystem fragmentation	Current stable version, widely adopted across research and production

Table 3.4.3.3 Evolution with Pros and Cons on PyTorch Versions

By Author

YOLO

YOLO's first version, YOLOv1, released in 2016 pioneered the one-pass detection approach. This was followed by YOLOv2 (2017) and YOLOv3 (2018) which improved backbone networks and multi-scale detection. In 2020, YOLOv4 refined accuracy and efficiency, while YOLOv5 (2020, Ultralytics) transitioned the framework into PyTorch with user-friendly training and multiple model sizes. YOLOv7 (2022) pushed accuracy further with efficient training strategies.

The latest version, YOLOv8 (2023, Ultralytics), streamlined deployment and extended YOLO beyond detection into segmentation and classification.

Version	Pros	Cons	Summary
YOLOv3	Simple, fast, low computing cost	Lower accuracy, weaker with small objects	Older model; falls short for robust real-time aerial detection
YOLOv4	Improved accuracy and speed compared to v3	Less optimized compared to newer YOLO models	Proven option for drone detection
YOLOv5	PyTorch implementation; easy to train and deploy	Large models may be too heavy for Orin Nano	Well-rounded, good for real-time aerial detection
YOLOv7	Excellent accuracy and efficient training	Heavier than v5; requiring optimization for embedded systems	Best for accuracy versus speed
YOLOv8	Newest generation; streamlined training and deployment; supports detection, segmentation, and classification	Less UAV-specific research compared to v5	Versatile option, but adoption in aerial platforms still emerging

Table 3.4.3.4 Evolution with Pros and Cons on YOLO Versions

By Author

SSD

SSD is a single-stage object detection model introduced in 2016. Within the next year, SSD were adapted with different backbones and input sizes, such as MobileNet-SSD (2017) for lightweight deployment and SSD512 (2016) for higher accuracy.

Version	**Pros**	**Cons**	**Summary**
SSD300	Fast, efficient, suitable for real-time applications	Lower accuracy, weak with small objects	Good balance of speed/efficiency
SSD512	Higher accuracy than	Slower, more	Better detection, but

	SSD300	compute required	less real-time friendly
MobileNet-SSD	Extremely lightweight, runs on Jetson	Accuracy lower than YOLO	Best for resource - constrained UAVS

Table 3.4.3.5 Evolution with Pros and Cons on SSD Versions

By Author

R-CNN

The R-CNN family of object detectors began with R-CNN in 2014, which achieved strong accuracy for its time but was extremely slow due to its region proposal step. It was followed by Fast R-CNN in 2015, which improved speed by sharing convolutional features across proposals, though it was still limited by the proposal stage. Later in 2015, Faster R-CNN was introduced, which replaced external region proposals with a Region Proposal Network (RPN), greatly improving speed and accuracy, making it a benchmark for many vision tasks. In 2017, Mask R-CNN extended Faster R-CNN by adding pixel-level instance segmentation, enabling detailed scene understanding at the cost of even higher computational requirements. Finally, Cascade R-CNN in 2018 further boosted detection accuracy by refining bounding box predictions in multiple stages, though it increased complexity and computational load.

Version	Pros	Cons	Summary
R-CNN	High accuracy for its time	Extremely slow	Historical baseline, impractical today
Fast R-CNN	Faster by sharing convolutional features	Still bottlenecked by region proposals	Improved but not useful for real-time
Faster R-CNN	Very accurate	Heavy compute, not embedded-friendly	Strong accuracy, but too slow for drones
Mask R-CNN	Adds segmentation	Even slower and heavier	Great for detailed offline tasks
Cascade R-CNN	Better accuracy with multiple stages	More complex, heavier	Research/benchmark use, not real-time UAVs

Table 3.4.3.6 Evolution with Pros and Cons on R-CNN Versions

By Author

NanoOWL

NanoOWL is derived from OWL-ViT (Open-Vocabulary Localization with Vision Transformers) and designed for real-time, prompt-based detection on NVIDIA Jetson hardware using TensorRT acceleration. Instead of detecting only fixed, pre-trained classes, NanoOWL can recognize any object described in text, making it ideal for flexible robotics and perception systems.

Version	Pros	Cons	Summary
NanoOWL - base	Supports open vocabulary detection via text prompts; no retraining needed for new classes	Heavier transformer model; lower accuracy than class specific detectors	Flexible and deployable open vocabulary detector for edge AI
NanoOWL - tiny	Smaller model size; optimized for TensorRT engine; suitable for embedded systems	Slightly reduced accuracy; may miss fine-grain objects	Best for resource limited drones needing adaptable detection
NanoOWL - Tree	Supports hierarchical detect-within-detect structure	More complex to deploy; slower inference	Enables structured, semantically aware detection for advanced perception tasks

Table 3.4.3.7 Evolution with Pros and Cons on NanoOWL Versions

By Author

3.4.4 Computer Vision Framework/Algorithm Comparison

To fully evaluate each computer vision framework for our SOAR project, seven parameters will be used to compare each frameworks: functionality, accuracy / robustness, speed / efficiency, ease of use, hardware optimization, scalability, and ecosystem support. In practice, frameworks are not used in isolation but rather used together depending on the task and deployment environment. For the SOAR project, which requires identifying, classifying, and potentially tracking moving targets, the following framework–algorithm combinations will be evaluated: PyTorch + Algorithm, TensorFlow + Algorithm, TensorFlow Lite + Algorithm, and OpenCV + Algorithm. In each case, the algorithm (e.g., YOLO, SSD, or R-CNN) provides the core object detection capability, while the framework supplies the infrastructure for training, optimization, and deployment. On their own, frameworks cannot achieve our SOAR objectives, but when paired with a strong detection algorithm, they enable improvements in key parameters such as speed, accuracy, hardware optimization, and deployment flexibility. For consistency and fairness, all frameworks in this comparison—PyTorch, TensorFlow/TensorFlow Lite, and OpenCV—are

considered in their most recent stable versions, reflecting the latest performance improvements, hardware support, and community adoption.

Criteria	PyTorch + Algorithm	TensorFlow + Algorithm	TensorFlow Lite + Algorithm	OpenCV + Algorithm
Functionality	Very High	High	High	Moderate
Accuracy / Robustness	Very High	Very High	High	High
Speed / Efficiency	High	High	Very High	Moderate
Ease of Use	High	Moderate	High	Very High
Hardware Optimization	High	High	Very HIgh	Moderate
Scalability	Very High	High	High	Moderate
Ecosystem Support	Very High	High	High	Very High

Table 3.4.4.1 Comparison of Computer Vision Frameworks with YOLO for SOAR AI System

By Author

Version	Accuracy	Embedded	Ease	SOAR Implementation
YOLOv3	Moderate	Very High	High	Low
YOLOv4	High	High	Moderate	Moderate
YOLOv5	Very High	High	Very High	Very High
YOLOv7	Very High	Moderate	High	High
YOLOv8	Very High	High	High	High
SSD300	Moderate	High	High	Moderate
SSD512	High	Moderate	Moderate	Moderate

MobileNet-SSD	Moderate	Very High	High	High
All R-CNN Models	Very High	Low	Low	Low
NanoOWL base	High	Moderate	Moderate	High
NanoOWL tiny	Moderate	Very High	High	Very High
NanoOWL Tree	Moderate	High	Low	Moderate

Table 3.4.4.2 Comparison of YOLO Versions for SOAR AI System

By Author

3.4.5 Summary of Computer Vision Frameworks/Algorithms

After comparing multiple frameworks and object detection models, PyTorch + YOLOv5 was selected as the primary choice for the SOAR project. This combination offers an optimal balance of speed, accuracy, flexibility, and embedded deployment, making it particularly well-suited for real-time aerial surveillance.

PyTorch Advantages

PyTorch brings several critical advantages to the SOAR project, including simplified training and fine-tuning, a strong community ecosystem, seamless deployment integration, and support for modern optimization techniques. Its dynamic computation graph allows researchers to experiment with custom datasets and architectures more easily, which is essential for adapting to unique UAV sensor inputs such as EO and thermal cameras. The large open-source community provides extensive tutorials, pretrained models, and troubleshooting resources, accelerating development and reducing risk. Another key strength is PyTorch's ability to export models to ONNX, which can then be optimized with NVIDIA TensorRT for deployment on Jetson hardware. This enables efficient real-time inference while preserving accuracy. Finally, PyTorch supports mixed precision training, GPU acceleration, and custom loss functions, giving developers flexible tools to build efficient and highly adaptable training pipelines.

YOLOv5 Advantages

YOLOv5 stands out for its well-rounded performance, UAV-specific research base, scalable model sizes, and robustness to environmental variations. It achieves a strong balance between speed and accuracy, making it capable of processing high-resolution aerial imagery at real-time frame rates—an essential requirement for detecting moving targets from drones. Unlike many models, YOLOv5 has been extensively validated in UAV scenarios, including human, vehicle, and object detection at varying altitudes, giving it a proven record of reliability for aerial surveillance. Its range of model sizes allows developers to tailor performance to the constraints

of embedded platforms like the Jetson Orin Nano, trading off between inference speed and detection accuracy. YOLOv5 also demonstrates resilience under challenging real-world conditions, such as fluctuating lighting, shadows, occlusions, and aerial perspectives, which are common obstacles in drone-based operations.

Combination Benefits (PyTorch + YOLOv5)

When paired together, PyTorch and YOLOv5 form a highly effective pipeline for UAV-based object detection. PyTorch simplifies model training, fine-tuning, and experimentation, while YOLOv5 provides the detection backbone optimized for real-time aerial imagery. The ability to export PyTorch-trained YOLOv5 models into ONNX/TensorRT ensures that the system can be deployed efficiently on Jetson Orin Nano hardware without major compromises in speed or accuracy. This synergy allows SOAR to achieve real-time detection, classification, and potential tracking of moving targets, all while remaining computationally efficient. Additionally, the strong research and developer support behind both PyTorch and YOLOv5 ensures long-term maintainability and access to cutting-edge techniques.

Comparison with Alternatives

Compared to alternatives, PyTorch + YOLOv5 strikes the best balance for SOAR. Older YOLO versions such as YOLOv3 and YOLOv4 are lightweight and efficient but struggle with accuracy, particularly for small or distant aerial targets. Newer models such as YOLOv7 and YOLOv8 improve accuracy further but introduce heavier computation demands, making them less practical for embedded deployment without aggressive optimization. Alternatives like SSD models (SSD300, SSD512, MobileNet-SSD) offer speed and efficiency but lack the robustness and accuracy of YOLOv5 in aerial environments. Meanwhile, the R-CNN family delivers excellent accuracy but is computationally too heavy for real-time UAV use. In contrast, PyTorch + YOLOv5 provides flexibility in development, efficiency in deployment, and robustness in performance, making it the most suitable combination for real-world SOAR missions.

Summary

Overall, PyTorch + YOLOv5 is the ideal choice for SOAR because it delivers a robust, real-time, and deployable solution for drone-based target detection while providing a flexible and well-supported development pipeline. It represents the best balance between training flexibility, embedded deployment efficiency, UAV-specific performance, and community support.

3.4.6 Swarm Coordination Ecosystem

For our four-drone SOAR swarm, which must run vision-based detection, identify and possibly track targets, and operate reliably in the field, an effective solution requires a layered stack:

- **Middleware / Coordination Frameworks** for multi-agent behaviors (e.g., ROS2, Crazyswarm)

- **Flight Stacks** for autopilot and low-level control (e.g., PX4, ArduPilot)
- **Communication Protocols** to bridge between layers (e.g., MAVLink, MAVSDK)
- **Simulation Tools** for testing and validation (e.g., Gazebo, AirSim)

This modular approach balances research flexibility with field robustness, allowing us to test swarm behaviors in simulation, deploy them through a reliable autopilot, and coordinate using proven middleware.

Middleware / Coordination Frameworks

ROS2

ROS2 (Robot Operating System 2) is a modern robotics middleware that provides message-passing, lifecycle management, and multi-robot coordination tools.

- It uses a pub/sub (publish–subscribe) model, where processes ("nodes") publish data topics (e.g., drone positions) and subscribe to receive others' data (e.g., target detections).
- Networking is handled via DDS (Data Distribution Service), a protocol designed for reliable, real-time communication across distributed systems.
- ROS2 is widely adopted in research and industry, and projects such as ROS2swarm extend its capabilities to swarm robotics.

Crazyswarm

Crazyswarm is an open-source ROS-based coordination framework originally designed for Crazyflie nano-drones. It simplifies commanding large swarms (up to dozens of drones) for formation control, trajectory planning, and flocking behaviors. While hardware-specific, it is excellent for testing coordination strategies before scaling to larger drones like SOAR.

Flight Stacks (Autopilot Software)

PX4

PX4 is a widely used open-source flight stack that runs on drone flight controllers. It supports offboard control (external commands from ROS2 or custom software) and includes built-in multi-vehicle simulation. PX4 is commonly used in academic and industry swarm projects, making it a strong candidate for SOAR's autopilot layer.

ArduPilot

ArduPilot is one of the most mature and feature-rich autopilot flight stacks, supporting fixed-wing, rotary, and ground vehicles. It has robust multi-vehicle support and integrates with

many Ground Control Systems (GCS). ArduPilot provides built-in methods for swarm-like behaviors and benefits from a large, active community with extensive tutorials.

Communication Protocols

MAVLink / MAVSDK

- **MAVLink (Micro Air Vehicle Link):** A lightweight, standard communication protocol used by PX4, ArduPilot, and many GCSs to send commands (e.g., position setpoints, telemetry).

- **MAVSDK:** A collection of high-level Software Development Kits (SDKs) for languages such as C++ and Python, which wrap MAVLink and make it easier to implement offboard control and multi-drone coordination.

Together, MAVLink/MAVSDK form the bridge between flight stacks (PX4/ArduPilot) and higher-level swarm controllers (ROS2).

Simulation Tools

AirSim

AirSim, built on Unreal Engine (a high-fidelity 3D game engine for realistic graphics and physics), provides photo-realistic environments and robust physics simulation. It supports multi-drone experiments, sensor emulation (cameras, LiDAR), and perception testing – useful for vision-based swarm tasks.

Gazebo

Gazebo is a widely used open-source robotics simulator that integrates seamlessly with ROS2 and PX4. It is ideal for SITL (Software-in-the-Loop) testing, where the autopilot code runs virtually, enabling safe and fast iterations. Gazebo is less graphically realistic than AirSim but excels at control testing, rapid prototyping, and integration with robotics stacks.

3.4.7 Comparing Swarm Coordination Ecosystems

The SOAR swarm requires a carefully chosen coordination stack that can balance research flexibility, real-world reliability, and scalability. The ecosystem of swarm tools can be divided into four categories: middleware (ROS2, Crazyswarm), flight stacks (PX4, ArduPilot), communication protocols (MAVLink / MAVSDK), and simulators (AirSim, Gazebo). Each component plays a distinct role, yet they interact as a pipeline – simulators test swarm logic, middleware orchestrates distributed behaviors, flight stacks ensure stable flight control, and communication protocols provide the glue between layers.

Component	Maturity for Swarms	Ease of Use	Real-World Readiness	Sim Support
ROS2	Very High	High	High	Very High
PX4	Very High	High	Very High	High
ArduPilot	High	High	High	High
MAVLink / MAVSDK	High	Very High	Very High	High
Crazyswarm	High	Moderate	Moderate	High
AirSim	High	High	N/A	Very High
Gazebo	High	High	N/A	Very High

Table 3.4.7.1 Comparison of Swarm Coordination Ecosystem components

By Author

Each tool has unique advantages and limitations. Middleware like ROS2 provides flexible multi-robot coordination but requires steep setup and networking expertise. Flight stacks like PX4 and ArduPilot are production-grade but differ in their community focus and ease of decentralized swarm autonomy. Communication protocols such as MAVLink and MAVSDK are standards for autopilot communication but may not fully support the bandwidth needs of vision-based swarms. Finally, simulators like AirSim and Gazebo are indispensable for safe prototyping, with AirSim excelling in photo-realistic sensor simulation and Gazebo providing deep integration with control frameworks.

Component	Pros	Cons
ROS2	Modern middleware, DS networking, node lifecycle, better for multi-robot; Strong community	Steep learning curve
PX4	Production-grade autopilot, SITL multi-vehicle sim, many offboard control APIs	Requires working with MAVLink and managing telemetry radios / IP networking for multi-vehicle WANs
ArduPilot	Mature docs for multi-vehicle flying, many community	Some herd workflows for true decentralized autonomy can

	tutorials and examples	require custom development
MAVLink / MAVSDK	Standard, well-supported; MACSDK provides easy language bindings	MAVLINK is telemetry-centric; designing high-bandwidth comms for vision data needs separate channels
Crazyswarm	Ready-made swarm behaviors, formation controllers, research demos; good to prototype algorithms	Tied to Crazyflie family; porting ideas to full-size drones is difficult
AirSim	Excels in photorealistic vision testing	Networking multiple drones with independent behaviors can require extra scripting.
Gazebo	Integrates tightly with ROS2 and PX4 for control stacks	Networking multiple drones with independent behaviors can require extra scripting.

Table 3.4.7.2 Swarm Coordination Ecosystem components Pros and Cons

By Author

3.4.8 Summary of Swarm Coordination Ecosystem Components

The SOAR swarm requires a stack that can scale beyond controlled research demonstrations and function reliably in real-world aerial missions. Each component of the ecosystem – middleware, flight stacks, communication protocols, and simulators – brings unique strengths. Middleware like ROS2 enables distributed coordination across drones, while PX4 and ArduPilot provide stable flight control. Communication protocols such as MAVLink and MAVSDK form the backbone of autopilot integration, and simulators like Gazebo and AirSim allow rapid prototyping of swarm strategies without risking hardware.

After comparing maturity, ease of use, real-world readiness, and simulation support, the most pragmatic choice for SOAR is:

- **Middleware: ROS2**, due to its maturity in multi-robot systems, large community, and extensibility (e.g., ROS2swarm).
- **Flight Stack: PX4**, for its proven autopilot performance, extensive offboard APIs, and explicit support for multi-vehicle simulation.
- **Communication: MAVLink with MAVSDK**, as the standardized protocol bridge between PX4 and ROS2, providing reliable telemetry and simple offboard integration.

- **Simulation: Gazebo**, for its deep integration with ROS2 and PX4, allowing SITL-based multi-drone testing before field deployment.

While tools like Crazyswarm and AirSim offer specialized advantages (formation testing, photo-realistic environments), they are best used as secondary research aids rather than the primary stack. For SOAR's operational needs, the ROS2 + PX4 + MAVLink/MAVSDK + Gazebo combination delivers the strongest balance of flexibility, field-readiness, and future scalability. This ecosystem will enable us to design swarm behaviors in simulation, deploy them seamlessly to the real drones, and coordinate vision-driven missions with reliability and precision.

3.4.9 Human Detection Objective

One of the main goals of the SOAR drone program is to achieve robust human detection using three distinct sensing modalities, all processed onboard the NVIDIA Jetson AGX Xavier companion computer. Lockheed Martin has generously provided the SOAR team with both an infrared (IR) camera and an RGB camera (specific models to be identified once confirmed). These sensors will enable the drones to perceive and identify humans under a wide range of environmental conditions, including low-light, high-glare, and thermally cluttered scenes.

To achieve accurate human detection, a high-quality and diverse dataset is essential. The dataset serves as the foundation for the model's learning process—its "lifeline." If the data used for training is biased, inconsistent, or of poor quality, the resulting model will inherit these flaws, leading to unreliable detection performance once deployed on the SOAR drones. Therefore, proper dataset selection, curation, and preprocessing are critical steps in building a model that generalizes well to real-world conditions such as variable lighting, drone altitude, and sensor noise.

Datasets

Primary Dataset — *RGBTDronePerson*

The primary dataset chosen for this project is RGBTDronePerson, a multimodal UAV dataset designed specifically for person detection using RGB and thermal infrared imagery. This dataset is particularly well-suited for SOAR because it provides paired and labeled data from both camera modalities, captured from aerial platforms at multiple altitudes and under various illumination conditions (daytime, nighttime, and mixed lighting).

The RGBTDronePerson dataset will serve as the core training source for two separate YOLOv5 models implemented in PyTorch:

- One model will be trained exclusively on the RGB images, corresponding to the RGB-equipped drone.

- The second model will be trained on the infrared images, corresponding to the IR-equipped drone.

Figure 3.4.9.1 RGBTDronePerson Human Detection

By Authors

Using a primary dataset like RGBTDronePerson ensures that both models share consistent annotation formats, synchronized viewpoints, and matched human detection labels. This alignment helps maintain training consistency and simplifies future work on multimodal fusion, where RGB and IR data could be combined or cross-referenced for enhanced detection reliability.

Secondary Datasets — *POP and AU-AIR*

While RGBTDronePerson provides a strong foundation, relying on a single dataset can limit model robustness. Secondary datasets are used to expand the diversity of training data, exposing the model to different environments, camera perspectives, and object appearances. This reduces overfitting to a single dataset's characteristics and improves generalization when the SOAR drones operate in novel conditions.

Two additional datasets will be integrated:

1. POP Dataset (for IR model)
   The POP dataset contains UAV-based infrared thermal imagery focused on partially occluded human detection. By including POP data, the IR model gains exposure to new environments, temperature variations, and human poses that differ from RGBTDronePerson's scenes. This helps the network learn stronger thermal feature representations, particularly useful for real-world thermal noise and partial occlusions during flight.

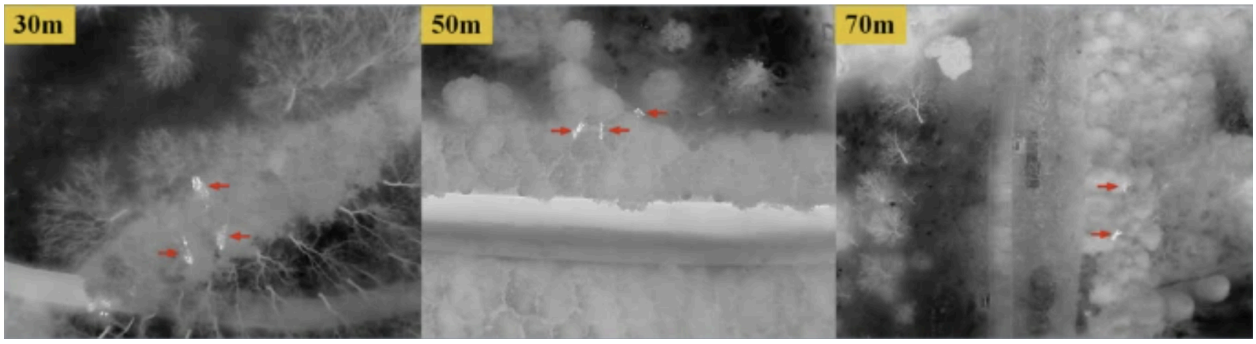


Figure 3.4.9.2 RGBTDronePerson Human Detection Additional Sets

By Authors

2. AU-AIR Dataset (for RGB model)
   The AU-AIR dataset consists of RGB drone footage captured at different altitudes, angles, and urban environments, with detailed object annotations including people. This dataset complements RGBTDronePerson by broadening the RGB model's understanding of lighting variations, background clutter, and altitude-based perspective changes. When trained together, these datasets enable the RGB model to better handle the visual diversity of outdoor and aerial scenes.

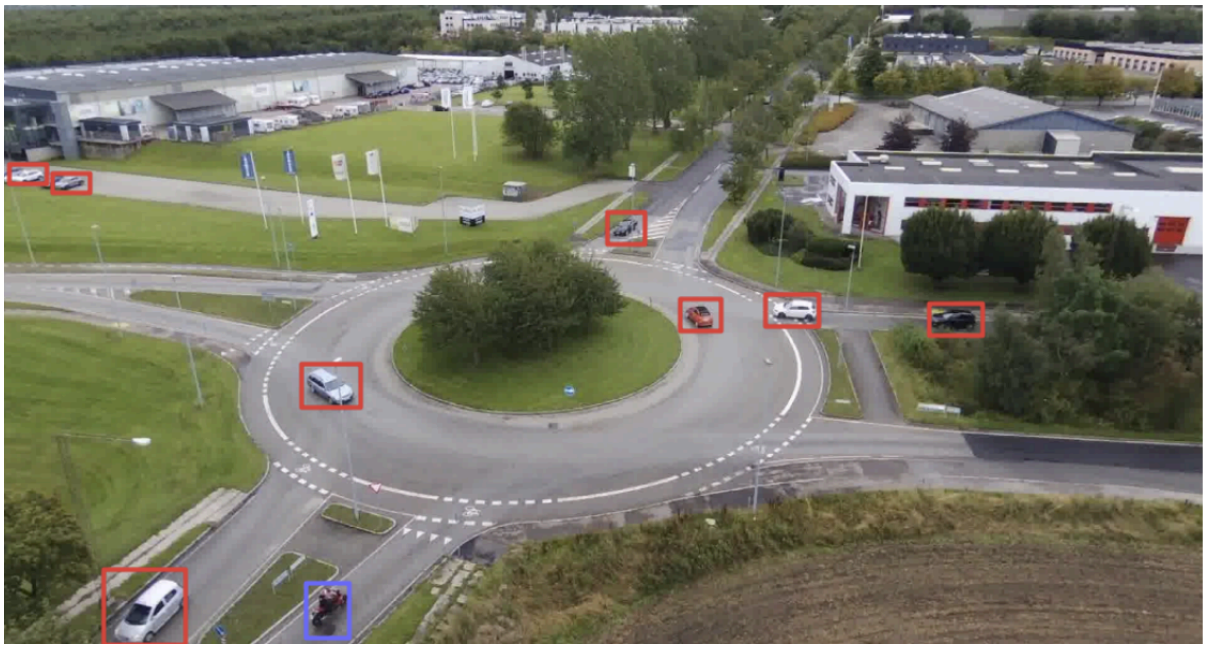


Figure 3.4.9.3 AU-AIR Dataset (for RGB model) Display

By Authors

Integration with PyTorch and YOLOv5

The datasets will be adapted into the YOLOv5-compatible format for training in PyTorch, where each image has a corresponding annotation file defining bounding boxes for human detections.

- Primary training will use RGBTDronePerson as the main data source for both models.
- Secondary fine-tuning will involve integrating POP and AU-AIR samples into the training pipeline—either through data augmentation or mixed-batch learning—to improve generalization and reduce domain bias.

This structured approach allows the SOAR team to train specialized YOLOv5 models optimized for each sensing modality, while still maintaining a unified detection framework deployable on the Jetson AGX Xavier. Ultimately, this multi-dataset, multimodal strategy strengthens the drones' ability to reliably detect humans in diverse environmental and operational conditions.

Data Preprocessing

Before feeding datasets into a model, the data must first be preprocessed to ensure that the model receives a clean, standardized, and properly formatted dataset. Data preprocessing is one of the most critical steps in model development because inconsistencies in format, labeling, or quality can severely degrade performance. Standardization ensures that all input data—whether from RGB or infrared (IR) cameras—follows the same structure, making it easier for the YOLOv5

model to interpret features consistently and learn effectively across different environments. The preprocessing pipeline can be divided into ten key steps:

Step	Purpose
Organize data in YOLO format	Required for YOLOv5
Clean/verify labels	Prevent bad training signals
Standardize channels/resolution	Maintain consistent input
Augment data	Improve generalization
Split data	Evaluate fairly
Create data.yaml	Define paths and classes
Normalize pixels	Handled by YOLOv5
Enhance IR contrast	Improve visibility
Balance datasets	Avoid bias
Sanity check	Ensure all loads correctly

Table 3.4.9.1 Data Preprocessing Pipeline Steps

By Author

Organizing the dataset structure may seem like an intuitive task, but YOLOv5 requires a very specific folder hierarchy to function correctly. Each dataset must contain two main directories: one for images and one for labels. The image folder should include .jpg or .png files, while the label folder should contain corresponding .txt files where each line defines a bounding box using the structure <class_id> <x_center> <y_center> <width> <height>, with all coordinates normalized between 0 and 1. Maintaining this structure ensures YOLOv5 can efficiently pair images with annotations during training and validation. The expected layout can be seen below:

```
datasets/
├── RGBTDronePerson/
│   ├── images/
│   │   ├── train/
│   │   ├── val/
│   │   └── test/
│   ├── labels/
│   │   ├── train/
│   │   ├── val/
│   │   └── test/
│   └── data.yaml
```

Figure 3.4.9.4 Organization of Datasets

By Authors

The input images themselves may vary in channel format and resolution depending on the camera type. RGB images contain three color channels, whereas IR images are often single-channel grayscale. However, YOLOv5 expects all input data to have three channels. Therefore, IR images must be converted by duplicating the grayscale channel into a three-channel format to maintain compatibility. Consistency in image resolution is also vital. Large discrepancies in image size can cause scale imbalances during feature extraction. For this project, the model will be trained on images standardized to a resolution of 1280×720, which provides a good balance between spatial detail and computational efficiency. Maintaining consistent aspect ratios is also important, though YOLOv5 conveniently handles automatic resizing during training to prevent distortion.

Data augmentation is another vital preprocessing step that artificially expands the dataset and improves model generalization. By exposing the model to a wider variety of transformations, it learns to recognize humans under different orientations, lighting conditions, and sensor noise levels. YOLOv5 includes built-in augmentation techniques, but some honorable types can be seen below. These augmentations are particularly beneficial for drone-based datasets where environmental conditions can change rapidly.

Type	Example	Purpose
Geometric	Random flips, rotations, scaling	Simulates different drone angles
Color/Intensity	Brightness, contrast, exposure	Handles lighting variation

Noise	Gaussian noise, blur	Simulates sensor noise and flight vibrations

Table 3.4.9.2 Data Augmentation

By Author

Properly splitting the dataset into training, validation, and test subsets is crucial for achieving a reliable and unbiased evaluation of model performance. If the training set is too large relative to the validation and test sets, the model may appear to perform well during development but fail to generalize to new data. Conversely, if the training set is too small, the model may underfit, failing to learn meaningful patterns. A standard approach is to allocate approximately 70% of the data for training, 20% for validation, and 10% for testing. This division ensures sufficient training examples while reserving unseen data for unbiased evaluation.

```yaml
train: datasets/RGBTDronePerson/images/train
val: datasets/RGBTDronePerson/images/val
test: datasets/RGBTDronePerson/images/test

nc: 1
names: ['person']
```

Figure 3.4.9.5 Splitting of Datasets

By Authors

Creating a data.yaml file is an essential part of YOLOv5 preprocessing. This configuration file tells the model where to find the training, validation, and test images, as well as the number of object classes and their names. For this project, separate YAML files will be used for each modality—for example, rgb_data.yaml for the RGB dataset and ir_data.yaml for the IR dataset. This structure makes it easier to train and evaluate each model independently while maintaining clear data organization.

Normalizing pixel values is another key step to ensure consistent input scaling. YOLOv5 automatically normalizes all image tensors to a range of [0, 1] during training, eliminating the need for manual normalization. However, additional preprocessing may still be required for infrared imagery. IR images typically exhibit lower contrast and narrower dynamic ranges than RGB images, which can make human silhouettes harder to distinguish. Techniques such as histogram equalization, min–max normalization, and filtering are often applied to enhance visibility. Histogram equalization improves contrast by redistributing pixel intensities evenly across the full range, making temperature variations more distinct. Min–max normalization stretches the pixel values between defined minimum and maximum limits, improving consistency between different sensors. Filtering methods, such as Gaussian or median filters, can reduce thermal noise and smooth image gradients, improving feature clarity for the model.

When combining multiple datasets—such as RGBTDronePerson, POP, and AU-AIR—it is essential to balance them carefully to prevent bias toward any one dataset. For instance, if the RGBTDronePerson dataset is significantly larger, the model may overfit to its visual characteristics, reducing performance on other datasets. To mitigate this, sampling weights or proportional batch ratios can be applied during training to ensure that each dataset contributes equally. YOLOv5 conveniently supports multiple dataset entries within the same data.yaml file, enabling flexible integration of diverse data sources.

Finally, performing a comprehensive sanity check ensures that all data is ready for training. This includes verifying that every image has a corresponding label file, checking that there are no broken or unreadable files, and previewing a few samples to confirm that the bounding boxes and color channels appear as expected. Detecting and correcting these issues before training saves significant time and prevents subtle errors that could undermine model accuracy.

In summary, the preprocessing pipeline transforms raw RGB and IR data into a clean, consistent, and well-organized format suitable for YOLOv5 training. Each step—from structure organization to normalization and dataset balancing—plays an integral role in ensuring that the final models can learn effectively and generalize across real-world environments encountered by SOAR's drones.

3.4.10 Ground Control and User Interface

The Ground Control Station (GCS) serves as the central command hub for the SOAR project, enabling mission planning, real-time swarm monitoring, and data fusion from all drone sensors. A critical requirement for the GCS software is its ability to integrate not only standard MAVLink telemetry for command and control but also the live video feeds from the heterogeneous digital video transmitters within the swarm. The ideal solution must present a unified operational picture, combining drone status, swarm coordination data, and live camera footage for effective operator situational awareness and decision-making during search and rescue missions.

Several GCS options are available, each with distinct advantages. QGroundControl is a powerful, open-source option and the standard interface for the PX4 flight stack, offering deep MAVLink integration, robust multi-vehicle support, and cross-platform compatibility. Its built-in video display capabilities can handle standard streaming protocols, though integrating proprietary feeds like from the DJI O3 Air Unit may require additional middleware. Mission Planner serves a similar role as the primary GCS for the ArduPilot ecosystem, providing a mature and feature-rich environment on the Windows platform, though its video integration can be less seamless. For maximum customization, a fully custom web-based interface could be developed using MAVSDK for swarm communication and a WebRTC gateway to unify the various video streams into a single browser-based dashboard, offering a tailored user experience at the cost of significant development effort.

However, a hybrid approach is recommended to balance development time, reliability, and functionality. The primary flight control and mission planning should be handled by QGroundControl, leveraging its proven and robust capabilities for managing the PX4-based swarm. To address the specific need for integrated video monitoring, a complementary custom web dashboard should be developed. This secondary interface would connect to a video gateway

server to display all live drone camera feeds in a single, cohesive view. This strategy effectively separates the complex tasks of flight control and video synthesis, allowing the team to use a battle-tested GCS for core operations while building a mission-specific application for the video-centric tasks that are fundamental to successful search and rescue operations.

Chapter 4 - Standards and Design Constraints

4.1 Standards

4.1.1 Drone Flight Standards

Since the SOAR platform involves autonomous drone flight, adherence to federal aviation laws is fundamental to ensuring safe and lawful testing. All system demonstrations and field evaluations must comply with the United States Federal Aviation Administration (FAA) regulations for Small Unmanned Aircraft Systems (sUAS). For our team to legally perform any non-recreational drone activity associated with this engineering project, a certified FAA Part 107 Remote Pilot is required to supervise the operation, and the aircraft must be registered with the FAA prior to flight. These requirements ensure that the operator has a strong understanding of airspace structure, emergency procedures, and aviation safety expectations.

The University of Central Florida also maintains its own drone operation policies, which include a campus-specific flight approval process intended to coordinate airspace use and minimize risks around academic facilities. Because of the increased risk and insurance limitations associated with complex autonomous aircraft, the university strongly encourages that test flights occur off campus. To meet these guidelines while also gaining access to a large controlled flying environment, our team intends to conduct most testing at Moonport Modelers Field located at 4950 Penrod Rd, Titusville, FL 32780. The facility is frequently used for model aircraft operations and provides a safe open-air flight zone away from pedestrian traffic or high-density infrastructure. The organization that operates the field also enforces safety protocols that align closely with FAA regulations, supporting responsible test activities for projects such as ours.

The FAA imposes additional operational limits that shape the physical design and software behavior of our drone. The aircraft must weigh under fifty-five pounds at takeoff, including the battery, propulsion system, onboard computers such as the Jetson Xavier, mounted camera payloads, and structural hardware. This weight constraint is a key design driver in component selection, as larger batteries or metal hardware could easily exceed allowable limits. The drone must also remain below four hundred feet above ground level during flight unless operating within a defined radius of a structure. The operator must be able to maintain constant visual line of sight with the drone without the assistance of binoculars or FPV video goggles. This requirement ensures collision avoidance and informs our decision to treat computer vision and LiDAR sensing as supplemental safety layers rather than replacements for human observation.

Other regulatory expectations include restrictions on flying during nighttime hours unless the drone is equipped with anti-collision lighting, as well as prohibitions against flying in controlled or restricted airspace without explicit authorization. With the recent implementation of Remote ID requirements in 2023, the drone must continuously broadcast its identity, position, and control

station location to support situational awareness for nearby aircraft. Compliance with Remote ID compatibility influenced our flight controller and telemetry selections.

The academic nature of our project does not reduce the need for full legal compliance. These aviation standards are critical constraints that directly affect the direction of our engineering work and limit certain design freedoms that might otherwise be feasible. As federal aviation guidance continues to evolve to support the rapidly growing UAS industry, our project remains committed to staying informed and compliant throughout development and testing.

4.1.2 Battery and Power Safety Standards

The SOAR drone depends on a high-capacity lithium polymer (LiPo) battery to supply all propulsion and onboard electronic power. While LiPo batteries are widely used for drone applications due to their high energy density and lightweight characteristics, they also introduce safety concerns including thermal runaway, fire risk, internal shorting, and voltage instability. As a result, our design must adhere to several safety standards governing both the physical handling of lithium-based cells and the electrical architecture that distributes power throughout the system.

These standards influence various decisions in our hardware layout. The battery must be mounted securely inside the airframe with vibration-isolating material to prevent mechanical damage while the drone is in flight. Power lines from the battery to the T-Motor F66A 4-in-1 ESC must be sized to accommodate high current loads without overheating. Additionally, the Jetson Xavier AGX and Raspberry Pi 4 require clean, regulated voltage separate from the propulsion bus, leading to the inclusion of a 19 V regulator and 5 V regulator that ensure stable power delivery to computing and avionics systems.

Battery safety standards also dictate how we handle charging and storage throughout the development lifecycle. The battery must be charged in controlled environments using certified LiPo balancing chargers and stored in fire-resistant enclosures during transportation. As operational heat conditions can degrade cell performance, the mounting location must also provide adequate airflow and separation from the ESC heat sinks.

Adherence to these lithium battery standards not only protects the aircraft from electrical failure but also ensures that all project activities remain compliant with industry safety practices during testing, transportation, and operational deployment.

Key standards and regulations include:

● **UN 38.3**: This standard establishes the testing requirements for safe transportation of lithium batteries—including vibration, altitude simulation, forced discharge, external short circuit, and thermal shock. Compliance ensures that the LiPo batteries can be legally transported by air, ground, or sea without presenting an unacceptable safety risk.

● **IEC 62133**: An international standard defining requirements for the safe operation of rechargeable lithium batteries. It includes tests for mechanical durability, charge retention, and protection against electrical abuse to ensure reliability during repeated use.

● **UL 1642**: A North American lithium battery safety standard focused on preventing catastrophic failure under fault conditions. It includes evaluations for puncture, external short, internal short, crush, abnormal charging, and forced discharge—tests that are especially relevant for drone-mounted batteries.

● **ASTM F2663**: This standard provides guidelines for lithium-ion battery design and testing in high-demand mobile applications, such as electric vehicles. Its emphasis on reliability and abuse protection aligns well with the power requirements of multirotor aircraft.

● **ANSI C18.3M**: A standard specifying performance and safety requirements for portable lithium batteries, including dimensional consistency, discharge properties, and failure mode protection.

● **EN 62133**: The European counterpart to IEC 62133, ensuring consistent safety certification across global electronics markets and providing requirements for lithium batteries used in consumer and industrial devices.

Lithium battery safety is guided by a range of international standards designed to prevent dangerous situations like overheating, fires, and explosions. One of the most important rules is UN 38.3, which is required for transporting lithium batteries. It involves a series of demanding tests that check how well the batteries can handle different environmental and physical stresses. Another widely used guideline is IEC 62133, and its European equivalent EN 62133. These focus on the safe design and performance of rechargeable batteries, paying close attention to how sturdy they are and how well they operate electrically.

In North America, the UL 1642 standard plays a major role in ensuring lithium battery safety by testing for things like internal and external short circuits, improper charging, and crushing forces. For electric vehicles, ASTM F2663 sets expectations for how lithium-ion batteries should be designed and tested to ensure they are both reliable and safe while in use. ANSI C18.3M defines the requirements for portable primary lithium batteries, including their size, performance, and safety characteristics.

Battery handling also matters. For example, battery terminals must be protected from short circuits so they don’t accidentally come into contact with metal objects. This can be as simple as keeping batteries in their original packaging, covering the terminals with tape, using a protective battery case or sleeve, or storing them in a plastic bag or pouch.

When all of these standards are followed, lithium batteries in everything from everyday electronics to advanced electric vehicles are far less likely to pose a safety risk. Compliance is important not only for manufacturers and distributors but also for users who handle these batteries in real-world settings. For most projects, it is recommended to keep each battery under 100 watt-hours of total energy capacity, since rechargeable lithium-ion batteries with ratings above that threshold come with additional safety concerns and restrictions.

4.1.3 Communication Standards

Effective communication between the various subsystems of the SOAR drone is essential to achieving autonomous flight, real-time situational awareness, and reliable interaction with mission software. Because our design incorporates multiple processors operating at different levels — the Cube Orange flight controller, Jetson Xavier AGX companion computer, and Raspberry Pi 4 — a combination of communication standards is required to balance robustness, bandwidth, latency, and compatibility.

These standards ensure that the connections between flight-critical components operate consistently under electromagnetic interference, mechanical vibrations, and battery voltage fluctuations that are inherent to multirotor platforms. They also guarantee that our drone adheres to industry-accepted message formats and hardware interoperability expectations, allowing us to integrate commercially available modules while maintaining a professional engineering workflow.

The communication protocols incorporated into the SOAR are detailed below.

UART -MAVLink Telemetry and ESC Feedback

UART is the most widely used serial communication method in small unmanned aircraft due to its simplicity and reliability. On the SOAR platform, UART links are used for:

• Cube Orange ↔ Jetson Xavier (primary autonomy data channel)
• Cube Orange ↔ 3DR SiK Telemetry Radio (ground control communication)
• ESC Telemetry TX → Cube Orange GPS2 RX (health status feedback)

UART communication requires only a TX (transmit) and RX (receive) line and does not depend on a shared clock signal, which simplifies wiring and reduces cost. Reliable UART communication is governed by several configurable parameters shared by both devices:

- Baud Rate determines how quickly data is transmitted. The Cube Orange commonly uses 57600 or 921600 bps depending on mission needs.
- Data Bits define the length of each data unit, typically 8 bits in MAVLink framing.
- Parity Bit enables basic error checking inside each data packet to detect transmission corruption.
- Stop Bits mark the end of a packet, ensuring proper synchronization.

More advanced features such as flow control (RTS/CTS) may be used to prevent buffer overflow during high-frequency telemetry bursts, especially when transmitting large MAVLink state messages such as GPS, attitude, and battery status at high rates.

Because UART is a point-to-point interface, we deliberately avoid sharing UART buses across multiple endpoints to prevent data collisions and signal contention. The decision to route ESC telemetry through a dedicated UART channel ensures that real-time propulsion data — RPM,

motor temperature, and current draw — is received without interfering with UAV command communication.

MAVLink, implemented on top of the UART transport layer, provides a rich set of safety functions including:

• Heartbeat messages to detect link loss
• CRC checksums for message integrity
• Standardized navigation and status messages
• Remote command acceptance rules (arming, RTL, landing)

This makes UART + MAVLink the backbone of safe, certifiable communication between onboard autonomy controllers and the flight-critical MCU.

CAN Bus - GPS Navigation and Compass Data

CAN Bus is chosen for communication between the Cube Orange and the Here3 GPS module due to its superior resistance to EMI and ability to support multiple smart avionics devices on a single network. Drone power systems produce high amounts of electromagnetic noise, especially from the ESC and brushless motors, which can disrupt low-voltage digital signals. CAN prevents this by using differential signaling, allowing valid data to be recovered even when noise is present.

The SOAR drone uses UAVCAN, a drone-specific CAN-based standard that ensures:

• Low latency positioning updates
• Network prioritization — urgent messages are automatically transmitted first
• Built-in broadcasting, scalable for any future sensor additions
• Automatic device addressing and hot-swapping
• Firmware upgrades through the bus

These capabilities provide greater reliability and functional safety when compared to serial GPS modules that rely on UART or I2C. Accurate pose data from the GPS and internal compass is essential for the Extended Kalman Filter (EKF3) inside the Cube Orange, which fuses IMU, barometer, and GNSS data to generate stable flight performance.

Using CAN also future-proofs the airframe — additional UAVCAN-compliant sensors (airspeed tubes, rangefinders, barometers) could be integrated with minimal wiring modification.

Ethernet + ROS 2 DDS – High-Bandwidth Distributed Computing

The SOAR drone incorporates advanced perception systems that require the transmission of large data streams such as:

• Real-time RGB video frames
• LiDAR environmental point clouds
• Thermal imaging for survivor identification

To handle these high-bandwidth data sources, the Jetson Xavier AGX and Raspberry Pi 4 are networked via Gigabit Ethernet. This interface provides orders of magnitude greater throughput than UART or CAN and enables true distributed computing capability onboard.

Data sharing and synchronization between the Jetson and Raspberry Pi are implemented using the ROS 2 DDS middleware. DDS provides the following benefits:

- Deterministic communication for robotic control loops
- Publish/Subscribe architecture enabling modular node design
- Quality of Service (QoS) policies for prioritizing urgent command data
- Automatic discovery and failover support
- Scalability to multiple compute nodes and edge devices

The Jetson Xavier subscribes to topics such as camera imagery and local obstacle mapping and publishes navigation decisions back to MAVLink control nodes. This arrangement allows deep-learning inference and mission planning to run independently of the flight-critical automation executed on the Cube Orange.

Ethernet + DDS therefore serves as the high-level autonomy “nervous system” while UART + MAVLink remains the real-time flight “spinal cord.”

4.1.4 Autonomous System Safety Standards

Autonomous flight introduces both technical and ethical responsibility, especially when an aircraft is expected to operate with limited human oversight. For the SOAR platform, the autonomy stack must comply with established standards that govern how intelligent systems are developed, validated, and operated in real-world safety-critical environments. These standards help ensure that perception, decision-making, and actuation functions behave predictably, even when faced with unfamiliar hazards or partial system failures.

One of the most influential guidelines for autonomous robotics is ISO 12100, which addresses safety principles for machinery equipped with automated control. Although originally designed for industrial equipment, its framework is directly applicable to unmanned aircraft. It emphasizes hazard identification, risk reduction, and fault monitoring. In practice, this standard motivates our inclusion of real-time system status checks, including telemetry heartbeats between the Jetson Xavier and Cube Orange, continuous monitoring of ESC motor diagnostics, and fusion of multiple environmental sensors to minimize the likelihood of single-point failures. Ensuring that the autonomous decision software cannot issue impossible or unsafe flight commands reinforces these same principles.

Software integrity and reliability are further guided by recommended practices from organizations such as SAE and IEEE that encourage traceability between mission expectations and programmed behaviors. This is particularly important for our perception algorithms that

interpret RGB and thermal imagery to detect victims or obstacles. Misclassification can lead to dangerous navigational decisions, which is why the system is structured so that autonomy recommendations are validated by the Cube Orange's flight control logic before being translated into actual motor outputs. The flight controller retains final authority regarding attitude stability and collision avoidance, reflecting widely accepted autonomous safety doctrine that no high-level perception engine should compromise core flight safety.

Another emerging regulatory influence is the FAA's Remote ID requirement for all autonomous aircraft that operate outside controlled indoor environments. Remote ID can be considered part of a broader safety standard for autonomous systems because it permits accountability and airspace situational awareness for both operators and commercial air traffic. Our decision to utilize the Cube Orange, which supports Remote ID compliance through telemetry integration, ensures that the aircraft's identification and location are continuously broadcast to nearby entities who may need to respond to its presence.

Autonomy in aviation also requires safeguards that protect humans, property, and the aircraft itself whenever mission goals cannot be met safely. To address this requirement, our system incorporates fail-safe transitions including Return-to-Launch (RTL) behavior if communication to the Jetson is lost, automated landing if battery levels degrade below a safe threshold, and manual override options that allow a certified pilot taking control instantly if an unexpected hazard develops. These behaviors align with the operational risk-reduction concepts found in ASTM standardized flight performance testing for unmanned aircraft systems.

Like many search-and-rescue focused platforms, SOAR processes data that may involve individuals in distress. For this reason, our autonomy design also acknowledges the ethical elements included in modern AI safety standards. Thermal imagery and camera data are processed only for mission-critical analysis and are not stored or transmitted without operator supervision to protect privacy and human dignity. This consideration reflects a growing standardization of responsible data handling in autonomous robotics.

Altogether, the SOAR autonomy framework is built around the principle that automation must always enhance flight safety rather than supersede it. Through compliance with industry-recognized safety standards, preservation of supervisory human authority, and protective data handling practices, the SOAR system achieves a responsible balance between advanced machine independence and strict operational safety expectations.

4.1.5 Mechanical Standards and Regulations

When developing a quadcopter for use in search and rescue missions, the case flying within a limited area (150 × 120 m), at altitudes below 500 m, and potentially over humans, vehicles and populated areas, both the mechanical design of the aircraft and the operational practices must be engineered in compliance with current regulatory standards and industry guidelines. This ensures safety of people on the ground, reliability of the system, air-space integration, and liability mitigation. For operation and ability to swarm, the drones must have a certain center of gravity (CG), weight limit and flight time ability. The following sections describe the key regulatory frameworks, mechanical/airworthiness design standards, and operational constraints that must be incorporated.

Some common regulations and standards that exist include ISO 2768: This standard specifies apply-related dimensional deviations regarding linear dimensions and angles in manufactured parts. Being within special tolerances helps keep the center of gravity (CG) in check, which is important for even flight and ability for electronic hardware to work properly. The next regulation is ASTM F2910. This standard outlines the design and requirements for small unmanned aircraft systems (sUAS) using aerodynamic principles, weight balances to reduce drag and increase flying capacity.

4.1.6 Flying over Humans and Vehicles

The FAA Part 107 is a UAS rule that oversees operations of small drones. This license is typically held by a drone pilot, and allows them to operate drones for non commercial use. These limitations include keeping the unmanned aircraft in visual line of sight (VLOS), flying in daylight or twilight with appropriate lighting, flying at or below 400 ft above ground level (unless certain conditions are met), operating in Class G airspace unless authorization is obtained, and yielding right of way to manned aircraft.

While part 107 is an important factor for flying drones such as the SOAR drones, this is not needed for flight in our case. Under USC 44809, exceptions for limited recreational operations of Unmanned Aircraft waves the need for a Part 107 license. The flyer has to pass the recreational UAS Safety Test (TRUST) to ensure that drone basics are covered and understood. \

Beyond operational regulations, there are industry standards for the design and construction of UAS. For example, ASTM International Standard F3298-24 “Standard Specification for Design and Construction of Lightweight Unmanned Aircraft Systems (UAS)” establishes baseline design, construction, and verification requirements. Although it primarily addresses fixed-wing UAS, analogous principles apply to multirotor systems. Compliance with recognized standards supports airworthiness, safety, and regulatory acceptance.

4.1.7 Mechanical Design Regulations

For a quadcopter intended for search and rescue, mechanical design must ensure the airframe (for example composed of hollow carbon-fibre tubes) can handle dynamic loads (takeoff, landing, gusts, manoeuvres), vibration, payload variations (sensors, cameras, thermal imaging), and rapid mission cycles. Stiffness is critical to maintain stable flight and sensor platform stability; strength and fatigue resistance ensure durability in repeated operations.

Since search and rescue may involve flights in challenging environments (wind-gusts, low light, debris, obstacles such as trees and vehicles), mechanical design must account for environmental exposures: dust, moisture, temperature extremes, shock/impact (e.g., landing rough terrain). Material selection, protective coatings, and robust mechanical joints matter. Modularity and ease of maintenance also enhance operational readiness.

Pre-flight test operations are also critical for design operations and maintenance standards. The remote pilot on command is responsible for ensuring that the drone is “condition for safe operation” per FAA regulations 107.15. This includes motor/esc verification, propellor life check, battery cycle count, and preflight inspection checklist.

The preflight inspection checklist for SOAR will include the following mechanical checkouts, which will occur prior to each flight. Fastener security, landing gear verification, laceration protection measure in place, and housing integrity must be checked to ensure that all flight operations can be performed safely. In addition, the emergency procedures must be verbally run through, in case of damage, crash or test failure.

4.1.8 Material Standards

Drones are designed with flight and operation in mind. For the SOAR drones, the current materials include carbon fiber and MPV wood, with harnessing and screws adjacent. In order to sell, use and fly materials, they must be able to pass certain standards defined by national agencies. This projects main regulation refers back to drones flying over humans. Currently, the Federal Aviation Administration (FAA) does not prescribe specific materials but requires that the structure be capable of safely supporting expected loads and flight conditions. Therefore, teams must demonstrate that chosen materials meet structural integrity, impact resistance, and fatigue life requirements under anticipated flight and environmental conditions. Verification typically involves documenting material properties (density, tensile strength, modulus) and validating them through manufacturer datasheets, mechanical testing, or simulation results that align with ASTM or ISO standards.

For small unmanned aircraft systems (sUAS), material verification can reference **ASTM F3201** (Standard Practice for Design of sUAS) and ASTM D3039 (tensile testing of polymer matrix composites). These standards guide how to test and certify composite and lightweight materials like carbon fiber laminates and balsa wood cores used in structural or aerodynamic components. Designers are expected to confirm consistency between theoretical design models and physical properties through sample testing or finite element analysis (FEA). The SOAR team will follow all verification procedures through ASTM F3201, which outlines standards of practice for small unmanned aircraft.

4.2 Design Constraints

Engineering a fully autonomous aerial search-and-rescue platform requires balancing desired performance with practical limitations that come from physical laws, available technology, regulatory compliance, and project resources. Each subsystem in the SOAR drone — propulsion, sensing, computing, communication, and power — introduces constraints that directly affect overall system capability. Understanding these limitations is critical not only for making responsible design decisions but also for ensuring that the aircraft can function safely within FAA requirements and the performance expectations of a real rescue scenario.

The major design constraints that shaped the development of the SOAR platform include weight limits set by aviation regulations, budget constraints associated with specialized components, power consumption and heat management requirements due to high-performance onboard computing, environmental and testing constraints driven by operational conditions, manufacturability and assembly difficulty associated with compact airframes, and finally ethical and safety considerations because the system interacts with human-critical rescue operations. The following subsections describe each of these constraints in detail and explain how they impacted the engineering process.

4.2.1 Weight and Structural Constraints

Weight is one of the most influential design constraints for any multirotor system and especially for ours. FAA regulations classify small unmanned aircraft systems (sUAS) as drones weighing less than 55 pounds at takeoff. Because our platform relies on dense electronics — including a 19-volt regulated NVIDIA Jetson Xavier AGX, multiple high-resolution cameras, a LiPo battery capable of supporting intense propulsion draw, and a robust carbon-fiber frame — weight budgeting became a driving force in the component selection process.

Every component added to the aircraft imposes a cascading cost on flight endurance. Higher mass requires larger motors and more powerful ESC channels to maintain lift, which in turn increases required battery size and therefore total aircraft mass. To prevent this "spiral of escalating weight," the team prioritized lightweight structural elements, aluminum and 3D-printed mounting solutions, and carefully selected sensing hardware. The T-Motor MN3510-KV360 motors and F66A 4-in-1 ESC were chosen specifically for their high thrust-to-weight performance ratio, enabling the drone to carry computational payloads without exceeding thrust margins needed for stable, wind-resistant flight.

Weight distribution is just as critical as overall mass. The Cube Orange flight controller must remain near the center of gravity to ensure accurate inertial measurements, and batteries must be positioned to avoid front- or rear-heavy imbalance that could stress the motors during maneuvering. The requirement to keep the system structurally rigid while minimizing material use led to iterative modeling in CAD followed by prototype balance testing. This constraint influenced nearly every engineering trade-off, from the gauge of wiring used in the power system to the number of peripheral sensors mounted for perception.

The 55-pound regulatory limit became a strict upper boundary that shaped every selection, packaging decision, and testing requirement throughout the project lifecycle.

4.2.2 Budget and Resource Constraints

Financial constraints played a significant role in shaping the hardware and integration strategy behind the SOAR drone. Our team was allocated a budget of $5,000 for the design and development of the system. Although this amount is more generous than many academic projects receive, we recognized early on that complex autonomous aerial systems can quickly exceed cost expectations due to high-performance electronics, specialized sensors, and aviation-grade mechanical components. Instead of attempting to exhaust the full budget, we placed priority on efficiency by selecting components that maximize reliability, performance, and integration value, while leaving a reasonable margin for unexpected expenses such as part failures, spares, field-testing consumables, and replacement batteries.

A major factor that allowed us to remain well within budget is the partnership with Lockheed Martin, which provided access to high-end resources that would have otherwise consumed most, if not all, of the funding. Items such as the NVIDIA Jetson Xavier AGX, various sensor platforms, and professional-grade modeling and testing support are typically cost-prohibitive for student teams. Leveraging these resources enabled us to allocate more of our spending toward

aerospace-reliable propulsion components, carbon-fiber structural assemblies, and long-range communication equipment.

Even with sponsor support, balancing performance against cost remained an ongoing engineering challenge. Advanced sensor packages such as gimbal-stabilized thermal cameras were evaluated but ultimately excluded, as their expense and weight conflicted with both financial and flight constraints. Instead, a lighter and more affordable USB-based thermal imager was selected, ensuring mission functionality without unnecessary overspending. Similar trade-offs guided the decision to use a Raspberry Pi 4 as a secondary processing unit rather than a second Jetson Xavier, as the Pi sufficiently handles auxiliary video processing and ROS 2 communication duties while significantly reducing system cost and power draw.

Budget limitations also affect project risk and development scheduling. Each major purchase had to be justified not only for performance benefit but also for supply availability, replacement cost, and integration complexity. The strategy of staying well below the maximum budget while using sponsor-provided resources wherever possible ensures that our design remains financially responsible and resilient to hardware setbacks during the testing phase.

Ultimately, our budget and resource plan ensures that SOAR can be fully functional, operationally durable, and mission-capable without excessive expenditure — demonstrating responsible engineering practice aligned with realistic aerospace development cycles.

4.2.3 Power Efficiency and Thermal Constraints

The SOAR drone integrates advanced computing systems and a high-performance propulsion architecture that together impose demanding power requirements. The main LiPo battery must simultaneously supply large bursts of current to the brushless motors through the T-Motor F66A 4-in-1 ESC while also powering the Jetson Xavier AGX, the Cube Orange flight controller, and the connected sensing peripherals. Managing this power distribution effectively is a core constraint because excessive load or voltage sag can destabilize flight control or trigger onboard computer resets.

The Jetson Xavier in particular is a major power consumer, capable of drawing more than 30 watts during heavy GPU-accelerated computation such as object detection and image processing. To ensure clean and stable energy delivery, the drone includes a dedicated 19-volt voltage regulator separate from the propulsion circuit. Even with this solution, thermal buildup becomes a major engineering consideration. The Jetson’s processing hardware generates substantial heat that, if not adequately managed, can induce thermal throttling and severely degrade autonomy performance during real-time rescue missions.

Propulsion system thermals are equally critical. ESCs and motors can reach high temperatures when sustaining heavy loads, especially during hovering, rapid climbs, or strong wind compensation. This risk is mitigated through the placement of the ESC and battery in high-airflow regions of the frame, while motor telemetry is continuously monitored via a UART feedback link to detect and respond to heat-related issues before they escalate.

Since flight endurance is directly constrained by available energy, every nonessential gram of weight and every watt of consumption influences operational viability. Choosing a more efficient Pi 4 instead of a second Jetson Xavier is one example of a compromise made to control power draw. The autonomy software must also operate efficiently, ensuring computational tasks do not consume more energy than justified by their mission contribution.

These power and thermal constraints guide not only hardware mounting but also flight behaviors, ensuring that energy management remains a priority from takeoff through mission completion.

4.2.4 Environmental and Testing Constraints

Because SOAR is intended to perform real-world search and rescue missions, it must be designed to withstand unpredictable environmental conditions. Wind, airborne debris, sunlight glare, GPS multipath interference, and temperature variations all impose design limitations and influence how the aircraft is tested and evaluated. A drone that performs well in controlled lab settings but collapses under modest wind conditions would fail to demonstrate mission readiness.

Our primary testing environment at Moonport Modelers Field in Titusville, Florida provides a safe open-air space suitable for regulatory compliance and flight evaluation, but also introduces environmental variability that requires careful preparation. Coastal wind gusts common to the region challenge the stability of the propulsion system and influence propeller and frame selection. Humidity and heat introduce additional thermal load on electronics, strengthening the need for cooling strategies around the Jetson Xavier and ESC.

Sensor performance also depends heavily on environmental context. GPS accuracy may fluctuate depending on atmospheric conditions or the drone's proximity to metal structures, directly affecting navigation confidence. Likewise, the thermal camera may struggle to differentiate human body signatures when reflective heat sources are present in the background, such as vehicles or machinery warmed by the sun.

LiPo batteries exhibit reduced power output in low temperatures and degrade faster in high temperatures, so operational planning must account for weather in order to preserve both safety and performance. During early testing phases, autonomy capabilities are limited to controlled environments where a human pilot can intervene immediately if environmental conditions exceed safety thresholds.

Environmental constraints therefore extend beyond simply surviving outdoor operation — they require that the drone's sensing, communication, and stability mechanisms can continue functioning reliably as conditions shift. These realities make robust testing essential, ensuring that the SOAR platform is not only functional under ideal circumstances but also capable of performing during the real-world unpredictability typical of search and rescue scenarios.

4.2.5 Manufacturability and Assembly Constraints

Many of the engineering decisions for the SOAR platform were heavily influenced by how the drone would be physically assembled, serviced, and iterated upon during development. Because the system includes multiple processors, distributed sensors, communication modules, power

regulation hardware, and a compact propulsion system, the overall complexity of the wiring and mounting architecture is significant. Ensuring that all components remain easily accessible for maintenance and modifications became an essential design constraint.

While carbon fiber is the primary frame material due to its excellent strength-to-weight performance, its rigidity and limited adaptability required careful planning for sensor brackets, power distribution, and component positioning. To accommodate the evolving hardware layout and minimize excess structural weight, the team relied on 3D-printed mounts for the Jetson Xavier, camera modules, and telemetry components. These additively manufactured parts enabled rapid cycling of design improvements and helped maintain precise alignment of components relative to the center of gravity. However, 3D-printed plastics introduce their own constraints, particularly regarding thermal tolerance, vibration resilience, and long-term fatigue. Each mount design underwent revision to ensure that it could survive high-frequency vibration generated by the motors and propellers without cracking or deforming.

Cable routing also represented a significant manufacturability challenge. The drone requires a balance between clean organization and minimal excess wiring that would add unnecessary mass or risk interfering with moving parts. Wires carrying high-current propulsion power needed thicker insulation and greater physical separation from low-voltage digital signal lines to prevent electrical noise and data corruption. This led to a modular wiring strategy, where propulsion, power regulation, and avionics harnesses were grouped separately to simplify troubleshooting and reduce EMI.

Additionally, each avionics module uses a different connector type — including JST-GH, DF13, USB-C, and XT60 — making connector standardization difficult. Secure mounting and connector locking mechanisms were prioritized to prevent mid-flight disconnections due to vibration. Because the drone must remain field-serviceable during rapid deployment testing, the internal layout was designed so that critical components such as the battery, ESC, Cube Orange, and Jetson Xavier could be accessed without complete disassembly.

Manufacturing constraints ultimately shaped the physical architecture of the SOAR system, ensuring that assembly remained manageable, aesthetics remained clean and professional, and field maintenance could be conducted quickly and safely.

4.2.6 Ethical and Safety Constraints

Safety is the most important condition governing the SOAR project, not only because the aircraft operates in public airspace, but also because the mission objectives involve assisting in emergency and rescue situations. For this reason, system design must ensure that autonomy does not compromise the well-being of people on the ground or the integrity of the airspace. The drone includes several layers of safety redundancies, including return-to-launch failsafes, manual override capabilities, telemetry heartbeat monitoring, and propulsion feedback diagnostics that allow the aircraft to respond gracefully to failure events.

A key ethical responsibility relates to the drone's use of cameras, particularly the thermal imaging sensor intended to detect human heat signatures during search-and-rescue missions. These sensing capabilities carry obligations to protect privacy and prevent misuse of collected

information. To address this, image and thermal data are only captured and processed for mission-critical analysis, and no personally identifiable information is recorded without operational justification. Data handling practices align with emerging responsible robotics standards emphasizing transparency, privacy protection, and minimizing unnecessary data retention.

The autonomy included in SOAR requires careful boundaries. Rather than granting unrestricted control to artificial intelligence algorithms, the autonomy layer provides recommendations while the Cube Orange flight controller remains the final authority for navigation and safety-critical movement. This separation of responsibilities ensures that flight stability, obstacle avoidance, and emergency responses remain deterministic and tested under controlled conditions. Human supervision is always available through a certified remote pilot who can intervene immediately if conditions become unsafe or autonomy begins to deviate from expected behavior.

Another ethical constraint relates to acceptable operational environments. The drone must not be flown near crowds, occupied buildings, busy roadways, or areas where system failure could cause injury. These restrictions influence both test planning and demonstrated mission scenarios. The team will continue to comply with U.S. regulations and University of Central Florida policies to protect both the public and project personnel during every stage of development.

By prioritizing operational safety, privacy awareness, and human-supervised control, the SOAR platform reflects a responsible engineering mindset in line with both professional ethics and modern autonomy safety standards.

4.2.7 Vibrational and Shock Constraints

When designing and testing a quadcopter, it is important to understand how vibration and shock affect both the mechanical structure and onboard electronics. Vibrations are unavoidable in any multirotor system because of the high-speed rotation of propellers and motors, aerodynamic disturbances, and sudden changes in flight dynamics. If not properly controlled, these vibrations can lead to mechanical fatigue, sensor errors, and unstable flight performance. Shock loads, on the other hand, occur during hard landings, collisions, or rapid maneuvers, and can cause structural failure or damage to delicate components such as cameras and flight controllers.

The main sources of vibration in a quadcopter are motor imbalance, propeller imperfections, and resonance in the frame. Each motor-propeller pair generates cyclic forces at the motor's rotational frequency (and its harmonics). If these frequencies align with the frame's natural frequency, a resonance condition can amplify vibrations throughout the structure. Even small vibrations can interfere with sensors such as gyroscopes and accelerometers, leading to inaccurate flight control inputs or oscillatory behavior during hovering.

To mitigate vibrations, students must consider both material selection and structural layout during the design phase. Using carbon fiber tubes or composite plates provides high stiffness with minimal weight, which helps raise the frame's natural frequency above the operating vibration range of the motors. Structural joints should be tight and free from play, and components such as arms and landing gear should have consistent stiffness to avoid asymmetrical vibrations. Additionally, vibration isolators, such as rubber dampers or silicone

mounts, can be used to decouple sensitive electronics (like the flight controller or camera) from the main frame.

Shock loading occurs primarily during landing, impact, or sudden thrust changes. To protect the quadcopter, landing gear and motor mounts should be designed to absorb and distribute impact energy without transmitting large loads to the frame. Using materials with slight flexibility, such as nylon, TPU, or composite laminates, can help dissipate energy rather than fracture. In carbon fiber structures, layering or reinforcing high-stress areas (like arm roots and motor mounts) with additional plies or aluminum brackets can prevent cracking under shock loads.

Managing vibration and shock is essential for ensuring flight stability, structural integrity, and component longevity in a quadcopter. For student projects, incorporating these considerations early in the design process promotes better performance and more reliable flight results. By understanding how vibrations propagate through the structure and how shocks affect the system during landing or impact, students can design quadcopters that are not only lightweight and efficient but also robust and safe to operate.

4.2.8 Time and Project Management Constraints

Time was one of the most influential constraints throughout the development of the SOAR system, as the project needed to progress through design, integration, and testing phases within a single academic year. Autonomous aerial systems require careful sequencing of hardware procurement, software development, and field testing, and each stage depends heavily on the successful completion of the previous one. For example, autonomy software cannot be properly validated until the aircraft is airworthy, and high-speed communication links such as Ethernet-based ROS 2 networking require extensive tuning before full sensor fusion can take place.

Component lead times and availability introduced additional scheduling risk. Many flight-critical components, including motors, ESCs, and GPS modules, can have unpredictable shipping windows due to supply chain fluctuations. The team proactively made purchase decisions early in the semester to avoid delays that could impact later integration milestones. Despite this planning, time still limited the complexity of onboard perception and constrained the number of full-scale outdoor test flights we could conduct before final demonstration.

Software development also had to be carefully managed to match the hardware readiness timeline. Testing autonomous navigation algorithms inside simulation environments such as Gazebo helped accelerate progress, but ultimately those algorithms needed validation through actual flight trials at the Moonport Modelers Field. Because each outdoor test requires licensed supervision, weather compliance, and coordination with facility operators, the number of available flight days is finite. Any significant crash or hardware failure could have consumed valuable time that would otherwise be spent demonstrating mission success.

These tight deadlines required disciplined task prioritization and team coordination. By breaking the project into distinct engineering deliverables and maintaining weekly progress goals, the team ensured that major subsystems such as power distribution, telemetry integration, and perception algorithms matured in parallel rather than sequentially. Time remains a critical

constraint in fine-tuning system performance, reinforcing the importance of early testing, incremental feature deployment, and contingency planning.

4.2.9 Summary of Design Constraints

The design of the SOAR platform has been shaped by a combination of regulatory, physical, financial, ethical, environmental, and scheduling constraints that collectively define what is feasible within the scope of this project. FAA regulations set hard boundaries on aircraft weight, operational limits, and safety expectations, while power and thermal constraints dictate how high-performance computing and propulsion must be integrated. Budget limitations drive smart resource usage, taking advantage of Lockheed Martin sponsorship to access advanced technology without exceeding cost thresholds. Environmental constraints require rugged hardware that can withstand unpredictable real-world rescue conditions, whereas manufacturability constraints influence component placement, wiring clarity, and repairability.

Most importantly, ethical and safety considerations ensure that the drone operates responsibly, protecting both people and property while collecting only the data required for mission operations. Finally, the time available for drone development necessitates aggressive planning and efficient engineering practices to achieve autonomous functionality within a fixed academic schedule.

Recognizing and responding to these constraints made it possible to design a drone system that is both technically innovative and practically achievable. These limitations did not hinder creativity but instead guided decisions that led to a more reliable, realistic, and mission-focused solution.

Chapter 5 - Case Study

5.1 Case Study One - Belle Perry

Prompt: “What factors should be considered when determining how to manufacture a quadcopter with carbon fiber?”

In reference to the prompt given to three different AI softwares (refer to Appendix E), each software created a different outcome, varying in responses.

These three AI softwares all answered the questions in different ways. To begin, Copilot, Appendix E, had the most concise answer. Offering the most high level summary, Copilot opted to list five parameters to consider when designing a carbon fiber quadcopter. Copilot is currently free for users, and is integrated into the search platform of the chrome tab. The copilot offers an initial summary, with the option to expand. The overall copilot response gave simplistic ideas for designing the drone using carbon fiber.

ChatGPT lists six parameters, then goes into specific detail for each parameter. While not represented in this paper, ChatGPT also utilizes emojis, most likely to appeal to the visual learner. Emojis provide context to generations who grew up using them, and provide an alternative to reading the entire thing. ChatGPT is the most common AI platform for students, and can be upgraded to the paid version, ChatGPT 5. This AI response uses the free version, since the other AI platforms were also free.

Lastly, Google Gemini uses the most detail, and goes into significant detail for each parameter it outlines. It lists both parameters and sub parameters, providing enough detail to continue research individually. Google Gemini also auto pops up when something is searched, and currently pops up as “google summary”. Google Gemini is integrated both into the search engine, but can also be used through the link. Its significant detail was not specifically prompted, and gave good bulleted ideas and topics that can be explored further.

Overall, all AI forms followed the same structure: Summarizing the questions, providing a list of parameters/solutions, and then summarizing the solutions. The type of research being done for the SOAR project can be significantly improved by utilizing AI softwares, but the vast range of answers makes it unpredictable to determine the validity of each response. By incorporating AI into research and development, new ideas can be considered. However, due to the nature of AI softwares, it is vital to cross reference information to ensure that the team has all information necessary to build and produce the drones.

5.2 Case Study Two - Adrian Castillo

Prompt: “What design practices help reduce risk of propulsion failure on autonomous quadcopters used in search-and-rescue operations?”

Each language model provided a technically relevant answer to the question of reducing propulsion failure risk in autonomous search-and-rescue drones, but each did so with a slightly different focus.

ChatGPT emphasized practical engineering considerations, such as thermal management, motor telemetry, ESC selection margins, and maintenance routines. Its response was grounded in real-world system integration and flight safety, making the advice directly applicable to our own SOAR propulsion architecture.

Google Gemini approached the question from a higher-level systems engineering perspective, highlighting topics such as rotor redundancy, dual power sources, and fault-tolerant control behaviors. It also included physical robustness of materials and structured emergency procedures like parachute deployment. While some of the concepts suggested larger multi-rotor systems beyond quadcopters, the response demonstrates a strong understanding of aviation reliability requirements for mission-critical operations.

Copilot provided a more forward-looking angle by introducing advanced control strategies and predictive diagnostic technologies, such as reinforcement learning-based failure recovery and AI-driven component health models. The inclusion of rigorous simulation and environmental resilience demonstrates an appreciation for the complexities involved in operating drones in unpredictable search-and-rescue environments. However, some recommendations (like FPGA-based hardware redundancy) may exceed the practicality or budget constraints of smaller drone systems like ours.

In combination, the three models addressed unique dimensions of propulsion safety: ChatGPT focused on grounded engineering practices, Gemini emphasized redundancy and mission safety frameworks, and Copilot introduced innovative control logic and predictive maintenance capabilities. By comparing these responses, it becomes clear that different AI models excel at different aspects of engineering problem solving, and using multiple tools together can help identify a more comprehensive solution space than relying on a single model alone.

5.3 Case Study Three - Robert Hagerty

Prompt: “What specifications and design parameters should be considered when selecting propellers for a quadcopter?”

The three language models all provided exceptional answers for the specifications and design parameters of propellers for a quadcopter. Each model elaborated on what it thought were the most essential design considerations for a propeller. The following is a breakdown on what each specific language model discussed.

To begin, ChatGPT initially gave eighteen different parameters that should be considered when selecting a propeller for a quadcopter. This was more in depth than any of the other models and gave a quick little snippet explaining each specification. Following this, ChatGPT gave some guidance on selecting the proper propeller for a drone based on some of these specifications. ChatGPT gave quick recipes, how to pick and test a propeller, and safety tips. Unlike the other models, ChatGPT then gave formulas and example calculations to aid in finding the right propeller for the project.

When asking Google Gemini the same prompt, it broke the question down into three sections. It first provided some insight on geometry and specifications of a propeller. This included diameter, pitch, blade count, and airfoil. Gemini then gave information focused on the physical properties of the propeller blade like material selection, weight, and mounting. Lastly, Gemini provided performance parameters to keep in mind when selecting a propeller. Gemini’s response was very simple and concise. It didn’t provide much specific detail for the various information it provided but it did well to summarize the various parameters it was asked of.

Lastly, similar to Gemini, Copilot broke its reply into different main categories and elaborated further to answer the question. Copilot explained the different specifications of propellers and then went into explanations of performance and design considerations. Unlike the others, Copilot

gave a little segment on optimization techniques which covered ways to pick the best propeller possible for the application.

All three models provided insightful and easy to understand information on propellers and what parameters should be considered in the selection of a propeller for a quadcopter. ChatGPT focused on the calculation and formula side of propeller selection, Google Gemini provided concise user friendly information on general propeller requirements and considerations, and Copilot gave similar results to Google Gemini with an added look into optimization. Asking each of these language models shows that each model provides its own unique answer to a question. These models in combination can provide the user with a better understanding of the prompt due to the different insights the models provide.

5.4 Case Study Four - Kristopher Tellez

Prompt: “What is NanoOWL and how does it compare to more traditional AI algorithms like YOLOv5?”

Each language model provided relevant details when summarizing NanoOWL and comparing both algorithms, but each did so with a slightly different focus on depth of detail or relevant comparable features.

ChatGPT summarized NanoOWL as a modern vision-language with open-vocabulary detection, allowing users to simply enter whatever object(s) they want to detect within a prompt without having to retrain the model. NanoOWL offers real-time performance when hardware requirements are met such as memory and GPU availability. ChatGPT quickly summarized YOLOv5 as part of a family of object detection algorithms that is widely used for real-time object detection optimized for speed and high accuracy on fixed class sets. When comparing both algorithms, ChatGPT created a table comparing each algorithm features such as flexibility, performance, model architecture, and accuracy. Finally, ChatGPT gave the user considerations for use cases like hardware compatibility, latency, and flexibility before answering which model it would use; YOLOv5 for fixed classes or NanoOWL for flexible, prompt-based detection.

Google Gemini approached the question at a higher-level, summarizing NanoOWL as an open-vocabulary detection algorithm with real-time optimization and tree detection pipeline. Next, Gemini compared NanoOWL to YOLOv5 by comparing features such as detection type, training requirements, deployment focus, and object classes. Finally, Gemini provided the user with its consideration; YOLOv5 for fixed sets of known objects or NanoOWL for flexibility and versatility.

Copilot provides a more “to-the-point” approach, summarizing NanoOWL with key features like open-world detection, real-time performance, nested detection, and flexibility. Copilot then compares both algorithms by their features such as detection scope, architecture, speed, and flexibility before telling the user to choose YOLOv5 for high-speed detection or choose NanoOWL for flexible, prompt-driven detection.

In combination, the three models addressed the functionality of both YOLOv5 and NanoOWL as well as providing the user which algorithm to choose after the comparison. ChatGPT focused on providing the user with important details about both algorithms before continuing to algorithm comparison and which model to choose. Gemini did not go into as much detail about NanoOWl compared to ChatGPT and did not provide a summary for YOLOv5, but still came to the same comparison results and algorithm selection. Copilot provided the user with the needed information (nothing more, nothing less) to summarize NanoOWL and offer which algorithm to implement after comparison. By comparing these responses, it becomes clear that different AI models take different approaches to explain and compare software algorithms, but in the end all had the same explanation on when to use YOLOv5 / NanoOWL.

5.5 Case Study Five - Michael Palin

Prompt: “What project parameters will improve the video stream on an autonomous quadcopter?”

Asking ChatGPT, GoogleAI and Copilot the same question prompted a unique response, and additional parameters the team had not considered. Each response addressed the problem of improving video streaming on an autonomous quadcopter with a technically valid yet distinct emphasis.

ChatGPT focused on foundational engineering and system optimization, emphasizing communication protocols, antenna design, encoding efficiency, and onboard processing. It provided a structured breakdown of parameters affecting latency and bandwidth, offering practical insights into hardware selection, video compression standards, and network tuning. This approach was grounded in the physical and electronic constraints of real-world quadcopter systems, making it directly applicable to improving link stability and overall stream reliability.

Google Gemini adopted a more configuration-driven and performance-oriented perspective, presenting a detailed checklist that balanced theory with hands-on tuning. It focused on the precise calibration of the entire pipeline, going from camera exposure and encoder settings to RF transmission and ground-station decoding. By offering specific numeric baselines, it provided a ready-to-implement framework for achieving low latency and maintaining stream integrity in dynamic flight environments. This response stood out for its methodical approach to system integration and iterative testing, making it highly practical for experimental tuning and optimization.

Copilot broadened the focus to include high-level autonomy and environmental interaction. It emphasized adaptive bandwidth management, flight stability, and onboard intelligence through edge computing and predictive control. By incorporating techniques such as model predictive control, sensor fusion, and real-time video quality monitoring, it highlighted the importance of coupling communication performance with flight dynamics and AI-based decision-making. This systems-level approach reflected a more forward-looking vision of autonomous aerial video systems, aligning streaming quality with situational awareness and control stability.

Taken together, the three responses cover complementary dimensions of video-stream optimization: ChatGPT grounded the discussion in core engineering and hardware choices; where Gemini refined execution through precise parameter tuning and empirical testing; and Copilot extended the problem space toward intelligent automation and environmental adaptation. Collectively, they illustrate how combining low-level optimization, configuration discipline, and adaptive autonomy can yield a robust and efficient video transmission architecture for autonomous quadcopters.

5.6 Case Study Six - Cristian Gomez

Prompt: “What is ArduPilot and does it compare to PX4 autopilot for swarming and search and rescue operations?”

While all three provide informative content, their tone, depth, and focus vary based on their design and purpose. ChatGPT tends to emphasize detailed technical reasoning, Gemini leans toward innovation and trend awareness, and Copilot focuses on clarity and practicality. Comparing their outputs helps reveal how each model suits different audiences and objectives in technical research and project development.

ChatGPT’s response stands out for its depth and structured explanation of the differences between ArduPilot and PX4. It provides strong technical coverage, likely including topics such as firmware architecture, community support, mission planning, and multi-UAV operations. This makes it particularly valuable for engineers or students seeking a comprehensive understanding. However, its weaknesses include occasional overgeneralization and a tendency to produce long, information-heavy answers that may overwhelm readers seeking concise summaries.

Google Gemini takes a broader and more forward-looking approach to the same question. Its response likely emphasizes emerging trends in drone autonomy, AI-assisted swarming, and the future of search-and-rescue applications. This forward-thinking perspective is useful for innovation and conceptual planning. Yet, its weakness lies in its lack of technical depth and specific operational comparisons, making it less ideal for users needing implementation-level insight.

Microsoft Copilot offers the most concise and practical response among the three. It focuses on summarizing key decision points, such as reliability, mission readiness, and hardware compatibility, in a clear and business-oriented tone. This makes it especially effective for quick professional briefings or project updates. Its limitation, however, is that it sacrifices technical nuance and analytical detail for simplicity and speed.

In conclusion, each AI model demonstrates strengths aligned with its purpose: ChatGPT excels in depth and technical explanation, Gemini inspires innovation and broader thinking, and Copilot delivers clarity and efficiency. Together, their differences show how AI tools can complement one another across stages of research and project work. For tasks like evaluating ArduPilot versus PX4 in swarming and search-and-rescue contexts, ChatGPT offers the most technical depth, Copilot the most clarity, and Gemini the most vision.

Chapter 6 - Mechanical Modeling

6.1 Overview and Initial Modeling

Computer-Aided Design (CAD) is a powerful design and visualization tool widely used across engineering disciplines. It enables designers and engineers to create precise 2D drawings and 3D models that represent complex systems or components before they are physically produced. CAD is essential for projects involving 3D printing, detailed design reviews, or any engineering application that demands a high level of accuracy and visualization of the final product. Through CAD, teams can efficiently iterate on their designs, identify potential issues early in the process, and ensure that all parts integrate seamlessly within an assembly.

For the SOAR project, CAD plays a critical role in the development process. The design will undergo several stages of refinement as the project evolves, particularly as computer hardware, aerodynamic considerations, and additional system requirements are defined. The current model builds upon the foundation established by the previous senior design team, incorporating lessons learned and introducing several modifications to enhance performance, manufacturability, and integration. This baseline design serves as a starting point for more advanced iterations throughout the semester.

As the project progresses, the CAD model will be continuously updated with greater detail and functionality. Future revisions will include refined subassemblies, improved component fit, and optimized structural configurations to support the overall mission objectives. Specific improvements will focus on propeller integration, enhanced aerodynamics, and a more robust and functional landing gear system. These updates will reflect ongoing research, testing outcomes, and team collaboration to ensure the SOAR design is both innovative and feasible for final implementation.

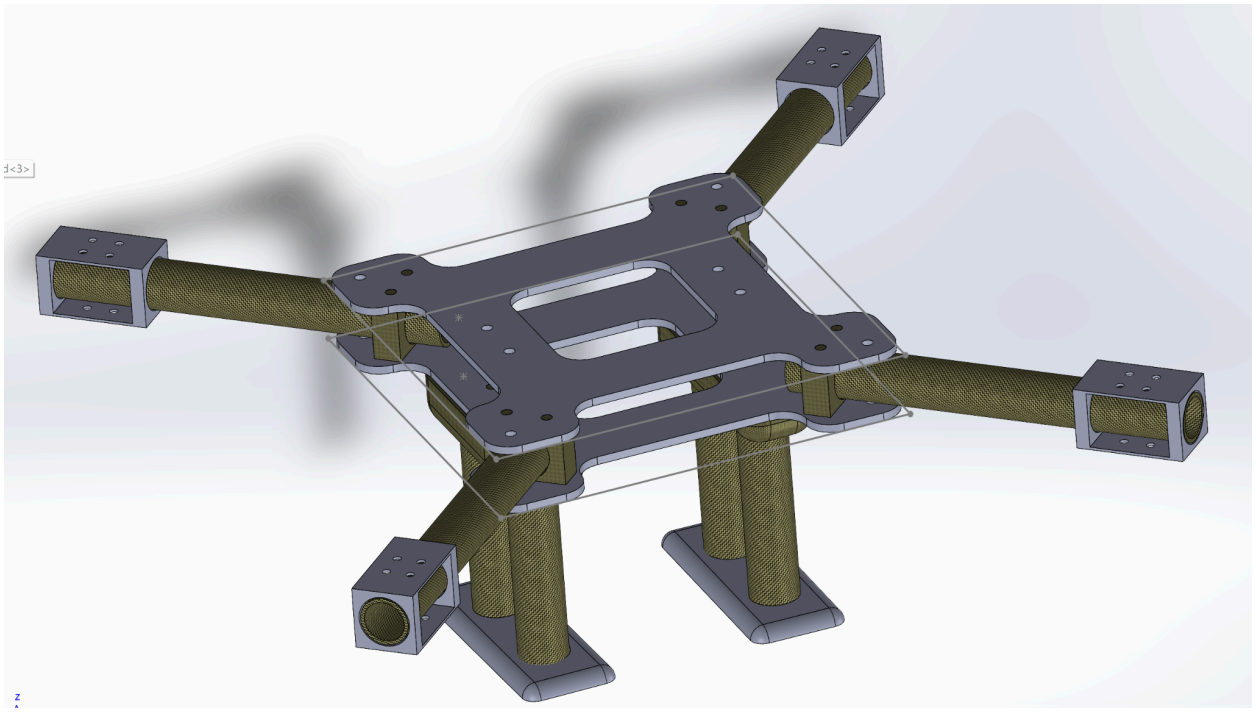


Figure 6.1 Initial CAD Model

By Authors

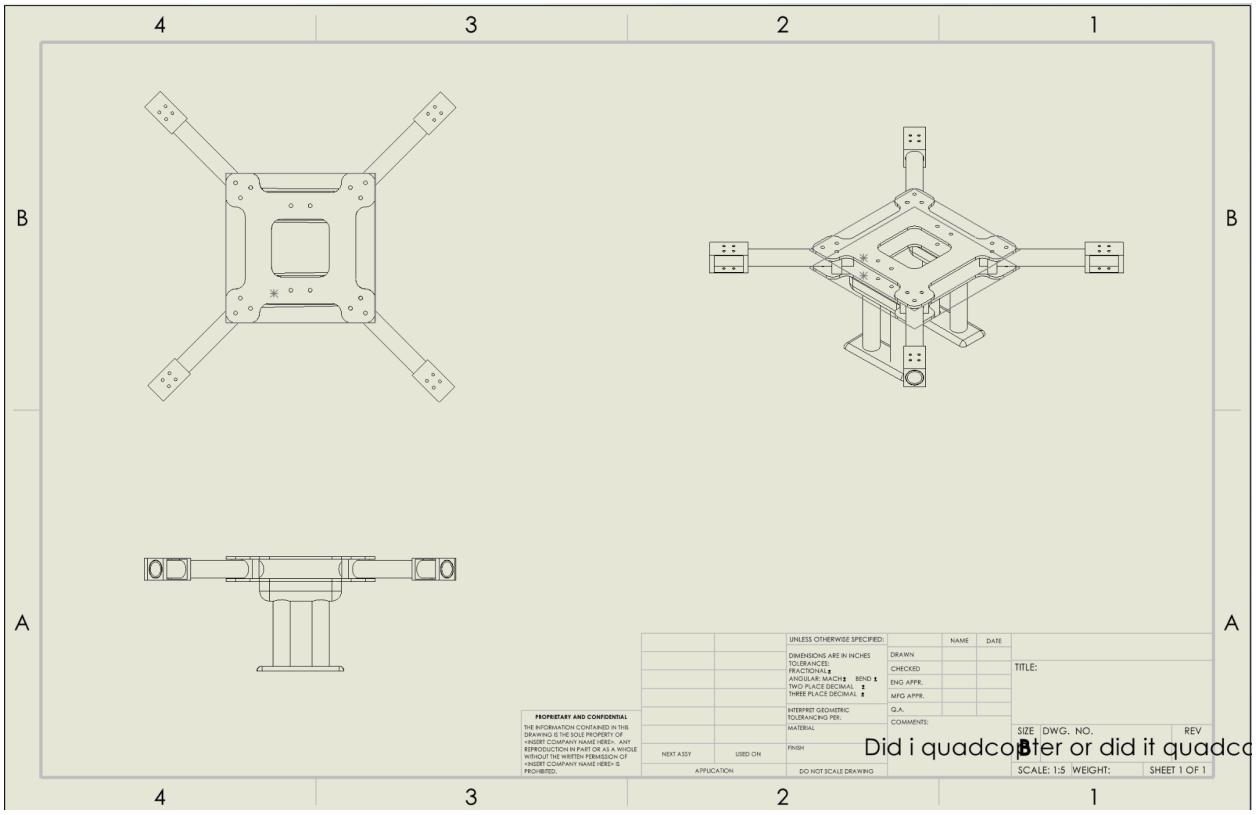


Figure 6.2 Initial CAD Drawing

By Authors

This initial design incorporates ideas from the previous SOAR team, as well as upgrades and things this team aims to achieve. By incorporating various materials, the goal is to optimize the weight and functionality of the drone, and be able to incorporate all components. The main frame is designed using wood, being flexible and sturdy. The support arms and landing gears are designed out of carbon fiber. These hollow tubes are designed to have low weight, high stiffness and durability for repetitive flights and missions.

Initial designs include two plates, one on the top and bottom of the quadcopter tubing. This design lacks the ability to easily install, maintain and replace electronic components with ease and user in mind. As this develops, the goal is to create a housing that can be removed with ease to maintain and install components.

This CAD lacks the propellers. The propellers that the team has selected are the SpeedyFPV 14x5.5 propellers. These propellers will be added to the drone frame in order to provide the most accurate CFD and FEA analysis. This CAD lacks the landing gear. The landing gear that the team will be using is the landing gear that is provided in the selected drone kit. This landing gear is of the skid-type. Enhancements may be made to this landing gear in order to increase stability or shock and vibration mitigation if necessary. Once the CAD for this component has been finalized it will be added.

It should also be noted that the design team will elect to make a unique housing structure for the on board components that will promote quick and simple access to the electronics or other systems. This structure also must be able to keep these systems intact during drone operation.

Communication with the sponsor will be essential before finalizing the design. In addition, budgeting and current supply should be considered as the design progresses.

6.1.1 Updated Design

Based on feedback from the customer, current supply, and other design factors, the team has shifted to using the X4-500 Quadcopter drone kit. The X4-500 Quadcopter Frame is a lightweight yet durable drone frame designed for stability and adaptability, making it an excellent choice for search and rescue applications. Constructed primarily from carbon fiber, the frame provides a strong, rigid structure capable of supporting high payloads and operating in challenging environments while keeping overall weight low. Its 500mm motor-to-motor span offers a balanced platform for both maneuverability and endurance, allowing the drone to cover large search areas efficiently.

One of the X4-500's standout features is its modular housing design, which can be easily manipulated or customized to accommodate different mission needs. This flexibility allows engineers to integrate specialized payloads. For each drone, a different set of electronics will be integrated. The open-frame architecture also simplifies maintenance and quick component replacement in the field. Overall, the X4-500's combination of strength, modularity, and adaptability makes it a reliable foundation for building a versatile, mission-ready search and rescue drone.

Figure 6.3 X4-500 Quadcopter

By Authors

This design uses the carbon fiber frame from the original kit. Using ¼ inch MDF wood, the original carbon fiber was replaced. From research and feedback from both the sponsor and contributors, the new design includes a removable top piece that allows for maintenance and installation of components.

By utilizing a hinge, the casing can both stay on the drone at all times, and mitigate any scratching or handling wear down. However, hinges add weight, and will have to be considered when doing thrust and motor load calculations. To combat the associated shock with having a removable lid, the lid will have a sealing point on the exterior. A lock/nudge mechanism will be put in place in order to secure this lid from moving.

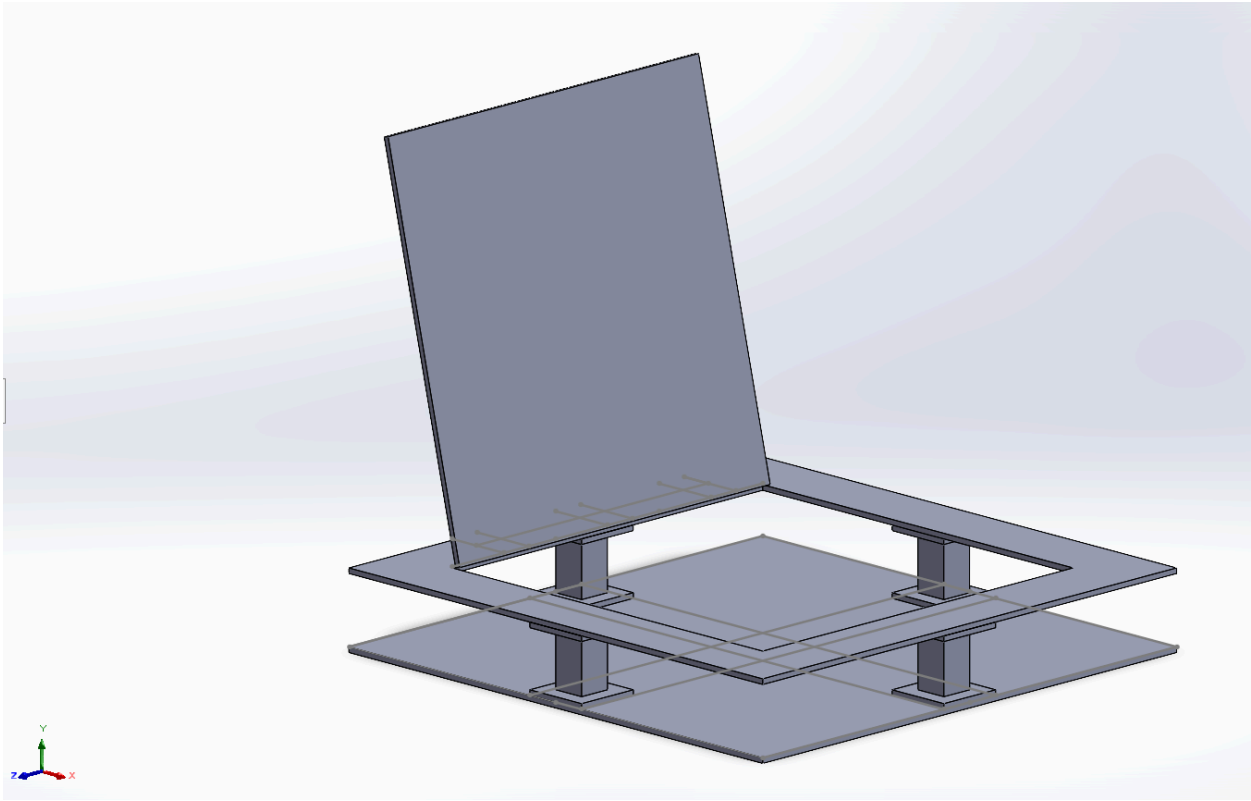


Figure 6.4 Initial Design of Modified Central Drone Body

By Authors

The design of the central drone body has been and will continue to be optimized to promote easier maintenance and improved accessibility to the Jetson Xavier, Cube Black flight controller, and all other electronic components housed between the top and bottom platforms. To support this, the door system, originally implemented with three hinges for access, will be refined and integrated using carbon-fiber materials in the final drone to increase durability while minimizing added weight. This hinge mechanism will undergo thorough testing to ensure that its placement and movement do not negatively impact flight performance or stability.

Additionally, the central body will maintain mid-section supports along its sides to reduce vibration and shock loads. These structural supports help protect sensitive onboard electronics and ensure long-term reliability throughout operation.

6.1.2 Flight Performance Hand Calculations

The drones current estimated weight is 9 pounds. Based on the current weight and specifications of the drone at the end of the course, the hand calculations have been done to ensure flight can occur steadily for the requirements outlined by the project. The motors, as outlined in chapter 5 and 7, have been selected based on the overall weight of the drone. The next important factor is endurance. This is based on the battery, which will power the entire drone. The current selection is a 16S battery. This will be split into two 8S batteries.

Hover Flight Time (*min*) = (*Battery Capacity* (*mAh*) / *Total Current* (*A*)) × 60

Battery capacity = 9000 (mAh)

Total Current = 60 (A)

Hover Flight Time (*min*) = 9/ 60 × 60 = 9 (*min*)

Total Hover Flight Time (*min*) = 4. 5 × 4 = 36 (*min*)

Based on this initial calculation, the drone swarm will be able to perform for 36 minutes, higher than the requirement. This calculation does not include any extra components, including higher travel speeds, more component draw, etc.

6.2 FEA Analysis

6.2.1 Overview

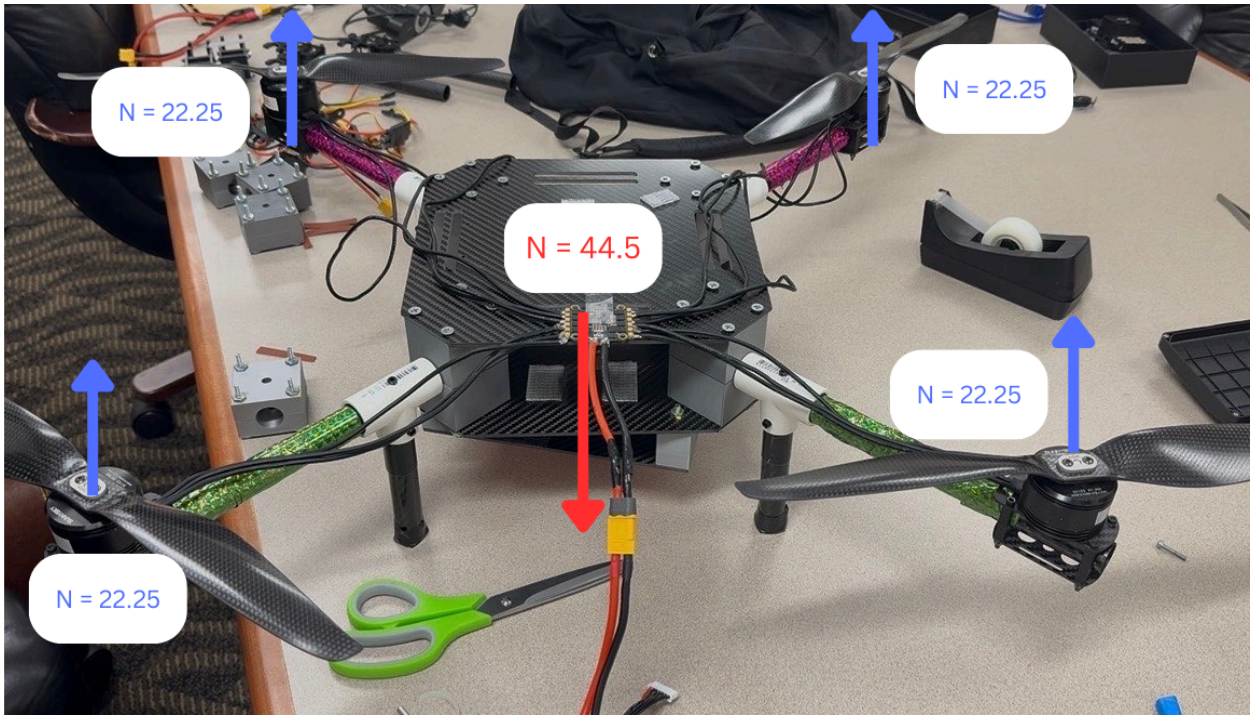


Figure 6.2.1.1 Loads and Forces in a Drone Frame

By Authors

Based on the above figure, the forces that are acting on the drone frame are able to help the team visualize what stresses and forces the drone may encounter during flight. The four motors each will generate an upwards force of 22.25N. This force comes from the T-Motor MN3510 in partnership with the EOLO 15x5.5 Carbon Fiber Propellers which will need to spin at roughly ~5965 rpm which is well within the operating range for these motors. These motors will generate

this thrust on each drone arm to produce a total upward thrust of 89N. This upward force is desired to counteract the weight of the drone which is roughly 10lbs or 44.5N; being made up of the electronics, payload, and structural components onboard the drone. This 44.5N downward load is to be centered on the drone frame for the most stable and controlled flight. The upward thrust counters the drone weight by a 2:1 ratio which is the desired Thrust to Weight Ratio needed to ensure stable flight. This force diagram provides insight on to the loads the drone frame will experience during flight and what must be considered when attempting to limit stress or deformation during operation.

6.2.2 Deformation

Deformation is also known as the bending deflection which is the lateral displacement of a beam caused by an external load. Deflection is calculated using the following formula:

$$\delta = \frac{FL^3}{3EI}$$

In this formula, F is the load, L is the length of the arm, E is the elastic modulus (Young's Modulus), and I is the moment of the inertia of the arm cross section. Having too high of a bending deflection can lead to motor misalignment, increased vibration, contact between moving parts, and even failure. These issues can cause issues in some of the onboard systems and physical connections of the drone. All of this can lead to a severe hindrance in operational ability and lead to an inability to complete the mission. To reduce deflection, arm length can be minimized, larger diameter arms, increased stiffness, and hollow arms. All of these techniques can be implemented or utilized if bending deflection is deemed to be too high for effective operational ability.

Deformation occurs when a cyclic load is repeated. Drones experience this deformation as it withstands flight. For the quadcopter outlined in the chapter, the analyses will focus primarily on hinge regions, clamps, and fastener holes. These locations are the typical indicators for damage. As the drones are being manufactured and assembled, they will be reviewed and analyzed, to ensure that each fitting is up to the team's standards. Where the screws and electronic components are present, a preload is modeled to understand the realistic bearings that will be experienced when the drone is assembled. During the prototyping phase, the team will ensure that the locations of the hardware aligns with both the mission profile, and will not cause any deformation. Prototyping will be backed by analysis. Using Ansys, the team will be able to predict the locations of deformation, and apply mitigation techniques prior to testing. Static loading will give an accurate estimate of the drones capabilities.

Due to deformation being caused by cyclic load, it can be predicted that deformation will be majorly present at the end of the drone arms due to the proximity to the motors and propeller blades which move in a spinning nature. As stated above, hinges, clamps, and fastener holes are more affected by cyclic load and thus are more prone to deformation, which is how the motors and propellers also attach to the drone arm. The spinning of the motors and the manor in how they are mounted cause the ends of the drone arms the hot spot for deformation. The team aims to counteract these effects and to ensure that these loads are evenly distributed across the drone arm to make sure failure does not occur.

6.2.3 Stress and Strain

Stress is the force applied to an area. If the force over a certain area is too great, failure can occur. Stress is calculated using the following formula:

$$\sigma = \frac{Mc}{I}$$

In this formula, M represents the bending moment at the section (N * m), c is the distance from the neutral axis to the outermost fiber (m), and I is the second moment of area (m^4). For a drone, the components that are to assume the most stress will be the drone arms as the moment is the thrust multiplied by the length of the arm (M = T*L). This stress that occurs must be considered in order to ensure that the drone is to not undergo failure during operation. To reduce stress, hollow drone arms should be employed, shorter drone arms, larger arm diameter, and a stiffer material all will help to reduce stress if it is deemed significant enough to potentially cause failure during operation.

Stress and strain are common attributes that occur when drones go through mission profiles. Specifically at joints and union locations. For the quadcopter, the connections from the housing to the motor arms, and the connection from the housing to the landing gears experience the most stress and strain. Picking materials that can withstand these loads are beneficial, as the drones will be able to withstand the mission objectives outlined in the requirements.

Stress and strain analysis is essential for ensuring that the quadcopter can withstand the loads it has been designed to hold. The drone faces stress and strain during flight, takeoff and landing, and maneuvering. In analysis procedures, the stresses are used to evaluate the durability of the materials. Strain will indicate how much the structure deforms. Based on the previous SOAR design, it was clear that there was significant stress sustained at the mounting point between the landing gear and the central frame of the drone body. There was clear warping of the MDF board and cracking and splitting was also visible. The team will focus on ensuring that the stress and strain impacting the landing gears are mitigated and focused on in the current design implementation. One method in doing so is having a landing gear leg below each drone arm as these points are the most structurally intact and thus the least likely to undergo failure due to stress.

The team plans to mitigate stress and strain in the drone body to avoid failure and maintain normal flight operations. If failure is to occur during flight or landing operations, then the mission may be compromised or unable to be fulfilled to the required standard. Ansys simulations will be used to simulate the potential stress or strain that will be seen during operation. Once these simulations have been completed, the team will assess the capability of the drone in handling these loads and will work to counter any significant stress or strain that may occur.

6.3 Flight Plan

6.3.1 Overview

With the occurrence of unmanned flight, it's important that a flight plan is outlined to ensure testing and operation will align with the testing and operational goals. The components mounted on the drones are purchased by Lockheed Martin and make up a significant cost. With that consideration in mind, each flight, test or demonstration will have an outlined flight plan. During the initial phases, this will primarily include a verbal announcement of the testing sequence, while connecting the drone to the remote flight control system. As autonomous flight operations are integrated into the drone, the mission planner will identify the course of action, and include crash landing and return home cases.

6.3.2 Assembly

Preparation for flight will include the build up and mounting of electronic components. Prior to this assembly, all electronic components should and will be tested at a component level, to ensure that there are no issues prior to mounting.

As the team begins to assemble, design and build the components, inventory is extremely important. As our material is lended from Lockheed, it's essential that all components are counted for each week to ensure that nothing has been misplaced. Major components include ESC’s, motors, LiDAR, cameras (RGB, IR, and visual), GPS, flight controller, and Jetson Xaviers. These components not only are essential to the design, but make up a majority of the cost.

The initial steps to building the drone will revolve around the construction of the frame since all components need to fit within it. Since the frame will be entirely 3D printed for all of the prototyping stages, there is a minimal amount of prepping needed other than possibly sanding any rough surfaces that can be seen on the outside and possibly prepping any screw holes needed to mount the processing board and motors. Smoothing the outer surfaces of the design will allow the prototype to be more aerodynamically efficient and prepping the screw holes to verify a secure connection of the motors. The next step would be to install the motors and ESCs. For quadcopter drones, the motors must be oriented in a way that eliminates the unwanted torque provided by the rotation of the propellers. This is done by adding an equal and symmetrical amount of counter-clockwise (CCW) and clockwise (CW) rotations. Going clockwise from the top left arm of the drone, the order should be clockwise, counter-clockwise, clockwise, counter-clockwise, or vice versa. The ESC should then attach the motors to the PCB, allowing the speed of each motor to be controlled based on the variable values sent from the flight controller. The wiring to each of the components must be neatly placed and bound to avoid any issues with moving parts.

After all of the hardware is finished being assembled, the drone's software has to be configured for the operation. The first step would be to install the firmware for the flight controller, for the SOAR project this would be ArduPilot. This firmware implements fundamental features like navigation, stabilization, and waypoint planning. Once installed, the user can input any

parameters needed such as frame type, motor layout, control preferences, etc. Afterwards, the program can be connected to the flight controller via a USB cable.

6.3.3 Calibration

One of the most important steps when it comes to software setup is calibration. The sensors like the IMU, gyroscope, and accelerometer must be calibrated to create valuable flight data. Without doing these calibrations, the data will be skewed and ultimately be rather useless, wasting a lot of time and effort during the test flight. The calibrations need to be in a controlled environment where there will be a minimal amount of interference. Furthermore, a calibration of the ESCs would be necessary to mesh the motor responses with the throttle inputs sent from the flight controller, enabling a smooth flight with minimal self-caused stability issues.

6.4 Structural Fatigue Analysis

Structural fatigue is a prevalent problem in any mechanical design that undergoes a constant cyclic loading, with quadcopters being no exception. Fatigue happens as a result of repeated loading that does progressive damage until the eventual failure, even if the cyclic stress that is occurring is well below the material's yield strength. In the case of quadcopters, the analysis of fatigue is especially critical since the frame, arms, mounts, and landing gear are in a constant state of cyclic stress. Understanding how fatigue occurs in mechanical systems and knowing how to mitigate it is crucial in keeping the durability and safety of the drone at an adequate level.

A few design criteria have already been considered for the swarm drones. The location of the landing gears, and its mounts are designed to be able to survive the cyclic loading.

6.4.1 Fatigue Mechanisms and Behavior

Fatigue is known as the progressive failure of a material that is under repeated loading, even if the repeated stress caused by the loading is below the material’s yield strength. For drones, the understanding of this is critical to ensure the longevity of the vehicle. Fatigue failure can be split into three separate stages: crack initiation, crack propagation, and final fracture. Crack initiation begins in the spots of the design with the highest stress concentration. These are often found near sharp corners, holes, or any geometric irregularities in the cross-sectional geometry. For 3D-printed parts, they are usually made out of plastic like ABS or PLA. These types of materials have crack invitations that happen often after the layer boundaries due to the interlayer adhesion being weaker than if the material was made from bulk rather than sliced sheets. Rough finishes, voids, or other surface imperfections that are common during printing can further worsen this issue, serving as places where the stress can concentrate onto a single area, accelerating the timeline of fatigue.

6.4.3 Mitigation Strategies for Fatigue

The first place to address fatigue is with material selection. While ABS beats PLA in factors like toughness and fatigue resistance, further optimization is available if the printing parameters such as layer height, infill density, and print orientation are better enhanced. For example, if the printed layers of the print are aligned in the same direction that all primary loads will occur,

fatigue resistance will greatly improve. Geometry optimization is also another possibility. Adding fillets/rounds to any sharp corners or sudden changes in geometry will reduce concentrations of stress on the design, and delay crack initiation. As long as smooth transitions are maintained, stress will naturally distribute more evenly, improving the life of the components. Surface treatments post-print can also play a vital role in fatigue mitigation. For ABS, a common post-processing method would be acetone vapor smoothing, effectively minimizing surface roughness and eliminating any micro-defects that might act as catalysts for crack initiation sites. For PLA, a similar type of acid or coating can also be applied to warrant the same response, protecting against environmental degradation over time.

An indirect method of reducing fatigue damage would be load reduction while indirect is still very effective at reducing damage done. If the propulsion system of the drone can be developed to reduce the vibrations or possibly lower the weight of the components, the load can be reduced in the critical components. This means that the use of any vibration-damping mounts or shock-absorption landing gear will additionally reduce fatigue.

Chapter 7 - Electrical Hardware Design

7.1 Overview of the Electronic System

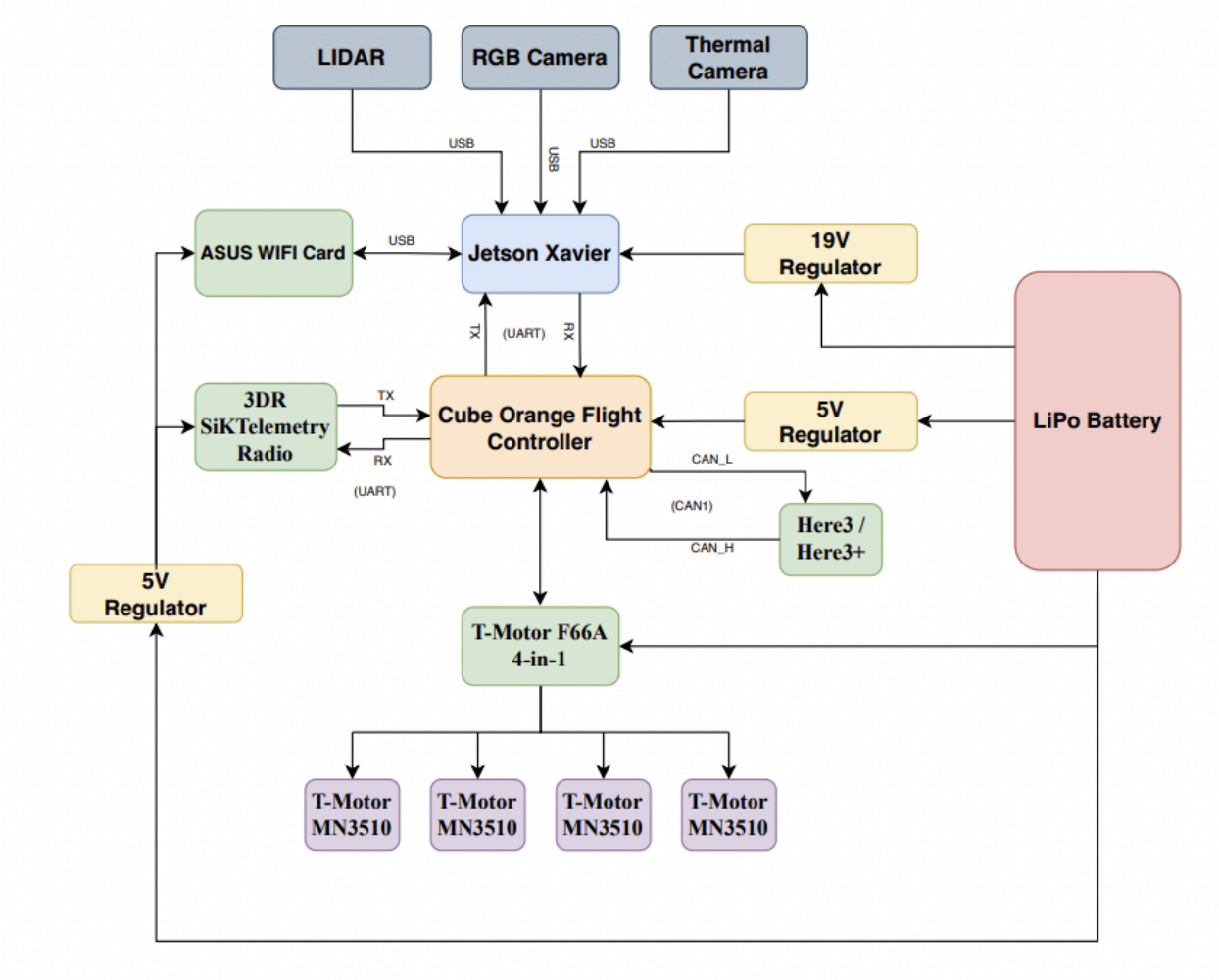


Figure 7.1 SOAR System-Level Electronic Architecture

By Authors

The electronic system of the SOAR platform serves as the backbone for autonomous flight, communication, and onboard data processing. Each electronic subsystem has been carefully selected and integrated to ensure efficient performance, reliability, and scalability during autonomous rescue operations. The system architecture is designed around the coordination between the Cube Orange flight controller, Jetson Xavier companion computer, T-Motor F66A 4-in-1 ESC, a range of sensors, and the supporting power regulation hardware.

At the highest level, the Cube Orange acts as the real-time flight control unit, handling low-level stabilization, sensor fusion, and control of the propulsion system. The Jetson Xavier functions as

the high-level computational hub responsible for artificial intelligence tasks such as target recognition, obstacle avoidance, and multi-sensor data processing. The T-Motor ESC and MN3510 motors provide propulsion, while the Here3 GPS, LIDAR, RGB, and thermal cameras enable environmental awareness and navigation.

The system's communication network connects all these modules using a combination of UART, CAN, Ethernet, and USB interfaces. Power is distributed from a high-capacity LiPo battery through dedicated voltage regulators that deliver stable 5V and 19V outputs to the Cube Orange, Jetson Xavier, ESC, and other subsystems. This hierarchical architecture ensures that high-current components like the motors are electrically isolated from the low-power logic circuits while maintaining a common ground reference for data integrity. Overall, the design enables the drone to operate autonomously and communicate seamlessly across all onboard electronic modules.

7.2 Flight Control System

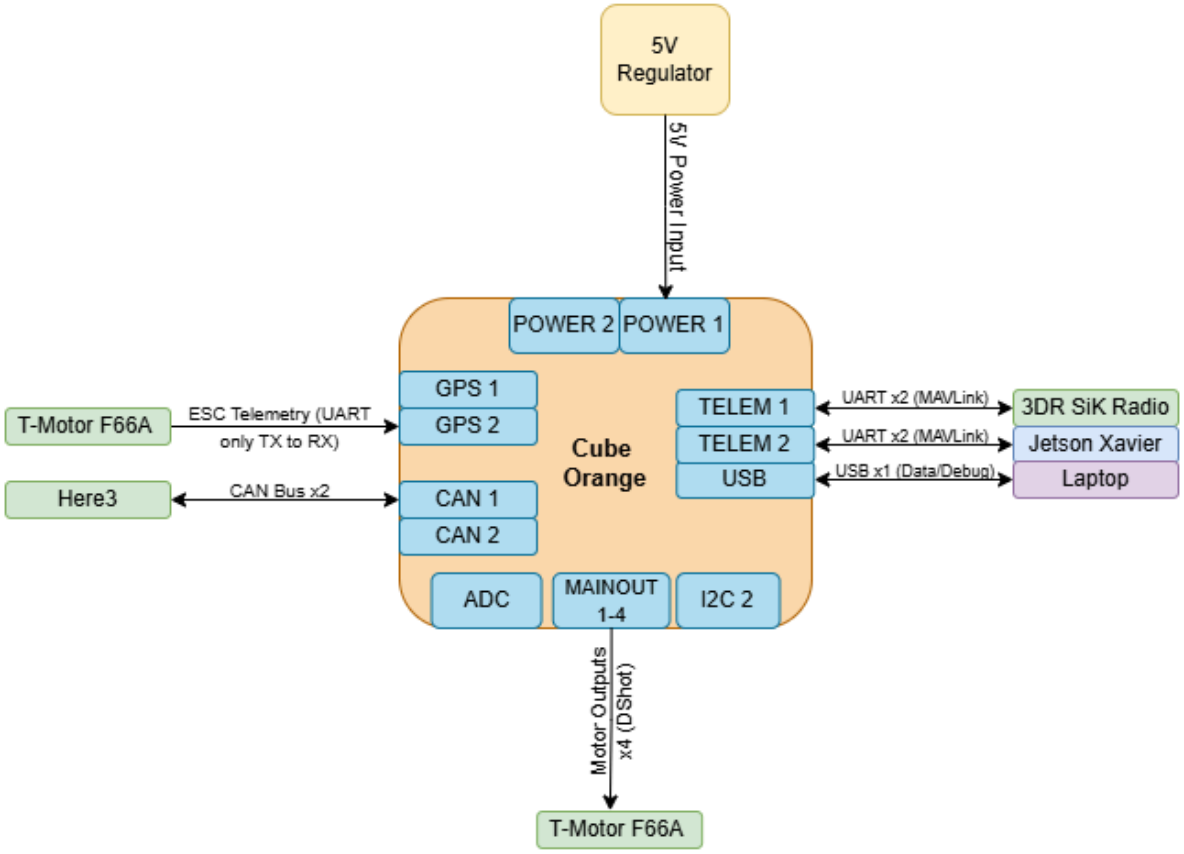


Figure 7.2 Cube Orange Flight Controller Connection Diagram

By Authors

The Cube Orange flight controller serves as the central processing unit for the drone's flight operations. Built on an STM32H7 microcontroller architecture, it is designed for high reliability and precision in multirotor control applications. The Cube manages flight stabilization, motor signal generation, telemetry communication, and data acquisition from the various sensors connected across its interfaces.

In the SOAR system, the Cube Orange communicates with several major components. Its first telemetry port (TELEM1) is dedicated to the 3DR SiK Telemetry Radio, which transmits MAVLink data to the ground control station for real-time monitoring and command input. The second telemetry port (TELEM2) is linked to the Jetson Xavier companion computer, enabling bidirectional communication through MAVLink. This connection allows the Jetson to send high-level navigation commands and receive telemetry data for AI-based decision-making.

The Cube's CAN1 port interfaces with the Here3 GPS module via a dual-wire CAN bus (CAN High and CAN Low). This digital interface provides robust and noise-resistant data transmission for GNSS positioning and magnetometer readings. For propulsion control, the Cube's MAINOUT 1–4 ports deliver DShot digital signals to the T-Motor F66A 4-in-1 ESC. The use of DShot ensures fast and accurate motor control without the signal drift or jitter that can occur with analog PWM.

An additional connection exists between the Cube's GPS2 port and the ESC's telemetry output. This one-way UART line (TX from ESC to RX on Cube) provides real-time feedback on motor speed, current, voltage, and temperature, allowing the Cube to perform health monitoring and respond to potential failures. Power to the Cube is supplied through the POWER1 and POWER2 ports, both regulated to 5V from the onboard power distribution module. Together, these connections enable the Cube Orange to act as a stable and intelligent flight management system, ensuring precise control and reliable data exchange throughout the aircraft.

7.3 Jetson Xavier Companion Computer

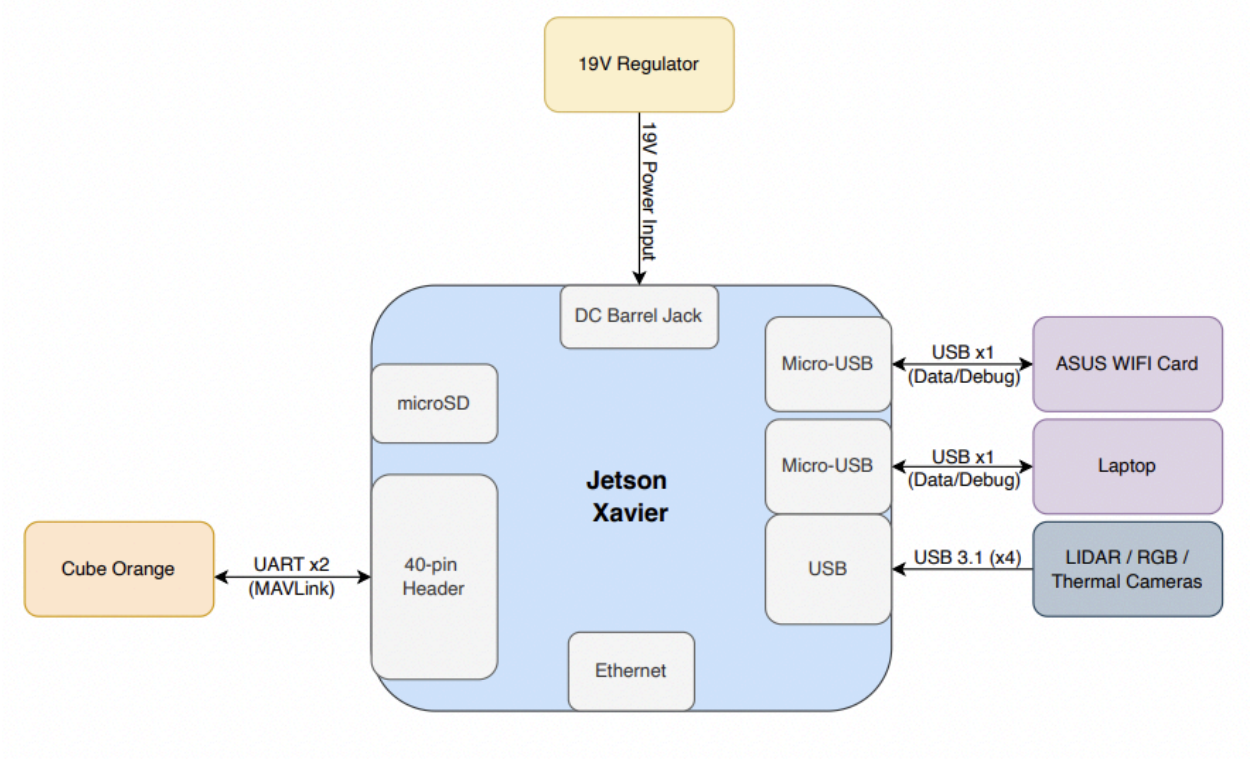


Figure 7.3 Jetson Xavier Connectivity Diagram

By Authors

The Jetson Xavier serves as the high-level companion computer, responsible for running computationally intensive algorithms that support autonomous operation and perception. Unlike the Cube Orange, which focuses on flight control, the Jetson is designed for advanced onboard processing tasks such as machine learning, image recognition, and multi-sensor data fusion. It operates under the NVIDIA JetPack environment, running ROS 2 as the middleware to manage distributed communication between sensors and the flight controller.

The Jetson Xavier interfaces with several critical components in the system. It communicates directly with the Cube Orange through a UART connection on the 40-pin header, utilizing the MAVLink protocol for telemetry and high-level command exchange. This link allows the Jetson to receive navigation data and flight states from the Cube while sending mission-level commands, such as waypoints or object-following directives. For video and environmental data, the Jetson connects to the LIDAR, RGB, and thermal cameras through the USB 3.1 ports, which provide the necessary bandwidth for high-resolution and low-latency data streaming.

A Gigabit Ethernet connection links the Jetson Xavier to the wifi card, which serves as a dedicated camera processing and communication node. The Ethernet interface operates using ROS 2 DDS, enabling synchronized message passing between the two computing platforms. This setup allows the Jetson to capture and pre-process video streams before transmitting the

data to the Jetson for object detection and scene understanding. The use of Ethernet ensures a high-speed and reliable communication channel that can support both image data and ROS topics simultaneously.

Power to the Jetson Xavier is supplied through a 19V regulator connected to its DC barrel jack. This regulator converts the main LiPo battery voltage to the precise input level required by the Jetson's onboard power management system. A micro-USB port is also available for system debugging, flashing, and terminal access, while a microSD card can be used for additional storage or operating system expansion. Altogether, the Jetson Xavier acts as the "brain" of the SOAR system, managing intelligent behavior, coordinating sensors, and interfacing seamlessly with the flight controller to achieve full autonomous operation.

7.4 Electronic Speed Controller

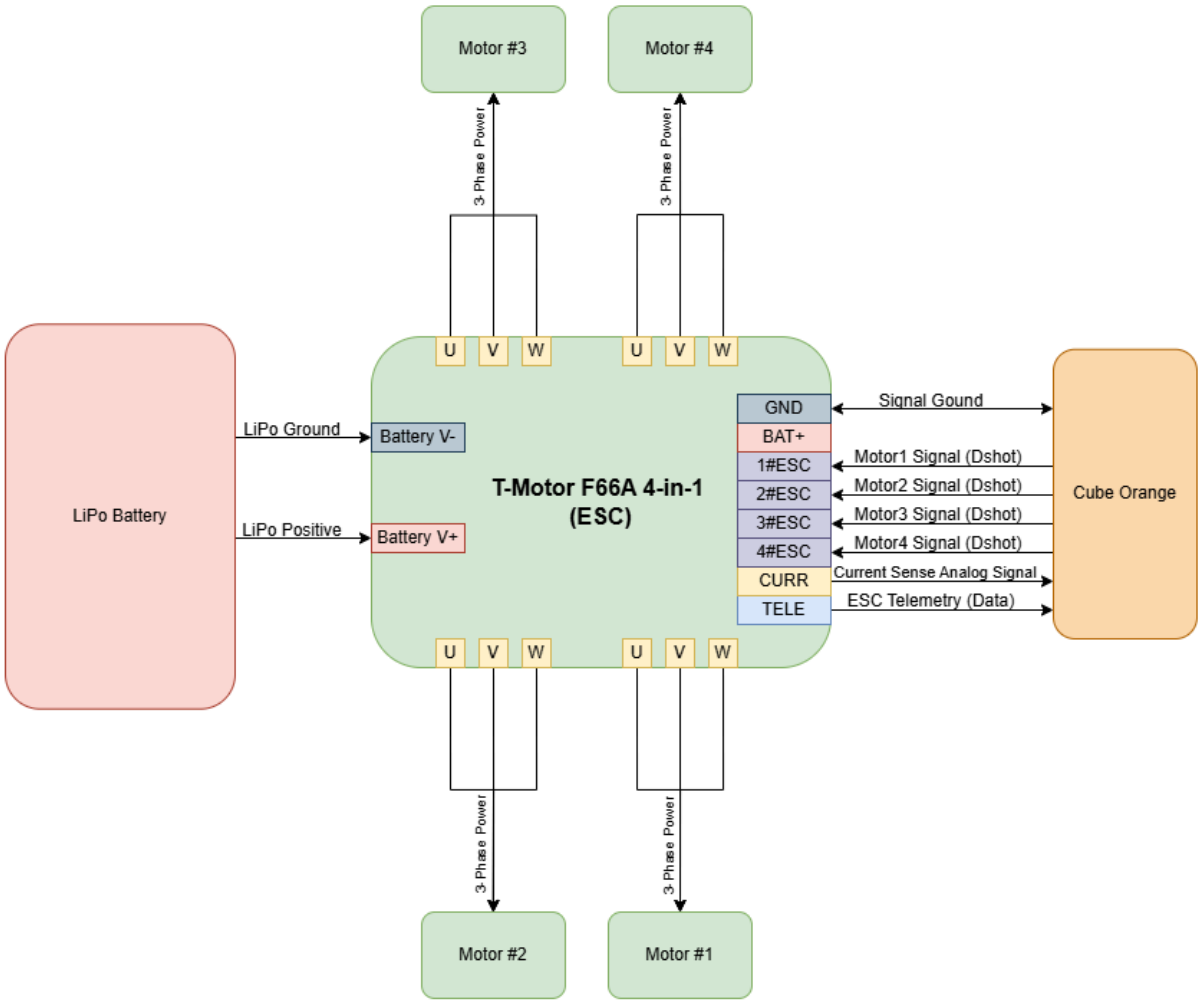


Figure 7.4 T-Motor F66A 4-in-1 ESC Wiring and Signal Flow

By Authors

The T-Motor F66A 4-in-1 Electronic Speed Controller (ESC) is responsible for translating the Cube Orange's digital throttle signals into three-phase power for each of the four MN3510 brushless DC motors. This ESC was chosen for its high reliability, integrated current sensing, and robust telemetry capabilities, all essential for stable multirotor operation in demanding flight environments.

The F66A ESC is powered directly by the LiPo battery through two main terminals labeled BAT+ and BAT−. These lines supply the raw DC voltage to the ESC, which then regulates and switches power to each motor through its high-efficiency MOSFET stages. Each of the four ESC channels outputs three-phase signals (U, V, W) to its corresponding motor. This architecture ensures synchronized operation across all motors and provides precise thrust control necessary for stable flight.

Signal communication between the Cube Orange and the ESC occurs through the four signal input lines labeled 1#ESC to 4#ESC. These signals are transmitted using the DShot digital protocol, which provides faster and more accurate control than traditional PWM. Unlike PWM, DShot encodes throttle values digitally, eliminating calibration drift and improving response time. A shared signal ground ensures consistent reference levels across all channels, preventing desynchronization during high-load operations.

In addition to motor control, the F66A supports ESC telemetry output through its TELE port. This line transmits one-way UART data (TX only) to the Cube Orange's GPS2 RX pin. The telemetry includes motor speed (RPM), temperature, voltage, and current information, allowing the flight controller to perform health diagnostics and detect anomalies in real time. The ESC also includes a current sense analog output (CURR), which can be used for direct current monitoring or power estimation if required. By combining high current capacity (up to 66A per channel) with telemetry feedback, the T-Motor F66A 4-in-1 ensures efficient and safe propulsion control for the entire aircraft.

7.5 Sensors and Peripheral Modules

The sensor suite of the SOAR system provides the essential perception and navigation data required for autonomous flight, object detection, and environmental mapping. Each sensor is carefully selected and connected to the appropriate processing unit based on bandwidth, protocol, and computational demand.

The Here3 GPS module connects to the Cube Orange via the CAN1 port, utilizing the UAVCAN protocol for robust and noise-tolerant communication. This connection provides high-precision positioning data, compass orientation, and altitude readings. Because CAN is differential and electrically resilient, it performs reliably even in the presence of electromagnetic noise from the motors and ESC.

The LIDAR sensor interfaces directly with the Jetson Xavier through a USB 3.1 connection. The LIDAR provides detailed distance and spatial mapping data used for obstacle detection and terrain awareness. The RGB camera, also connected through USB 3.1, captures high-definition imagery for computer vision tasks such as target recognition and object tracking. Complementing these is the thermal camera, which detects heat signatures and is used primarily for identifying human bodies during search and rescue missions. The use of the Jetson's USB 3.1 ports ensures sufficient data throughput for simultaneous operation of multiple high-bandwidth sensors.

The Jetson acts as an auxiliary processing node for video handling and lower-level communication. Connecting to the Jetson Xavier through the USB, which enables high-speed data transfer while maintaining synchronization. This modular division of sensing and processing responsibilities allows the Jetson to focus on real-time AI and decision-making, while it handles video preprocessing and compression tasks.

Collectively, these sensors and peripherals give the SOAR platform multi-modal situational awareness, enabling it to perceive its environment in both the visible and thermal spectrum, accurately navigate through GPS data, and perform intelligent path planning in dynamic conditions.

7.6 Power Supply and Regulation

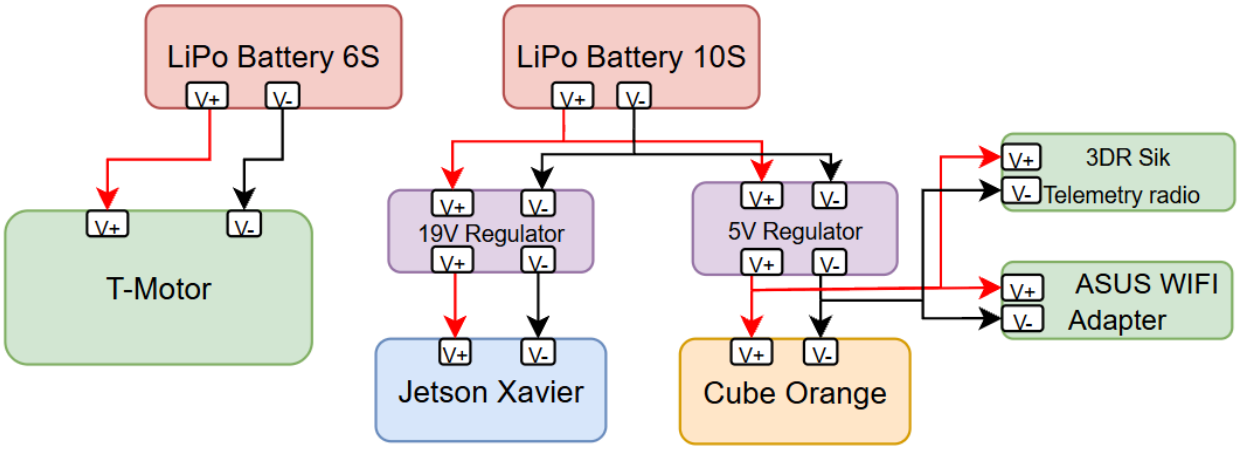


Figure 7.6 Power Supply Wiring

By Authors

The power system is designed to deliver stable and isolated power to all electronic subsystems while maximizing efficiency and safety. The main power source is a high-capacity LiPo battery, which provides sufficient voltage and current to drive the motors, flight controller, and onboard computing units simultaneously. From this primary source, the voltage is distributed through multiple DC voltage regulators tailored to the needs of each subsystem.

The 19V regulator supplies power to the Jetson Xavier via its DC barrel jack input. The Jetson requires a stable 19V input to support its onboard voltage regulation circuitry, which powers the CPU, GPU, and peripheral interfaces. This regulator is rated to handle the high current demands of the Xavier, especially during peak processing loads when all cameras and sensors are active.

A separate 5V regulator provides power to the Cube Orange flight controller, the Here3 GPS module. This ensures that sensitive logic components receive clean, noise-free power isolated from the high-current motor systems. The same 5V bus also powers the 3DR SiK telemetry radio and provides a regulated supply to the T-Motor ESC's logic circuits. By isolating high-power and low-power subsystems through individual regulators, the design minimizes the risk of voltage fluctuations affecting flight-critical components.

Grounding is shared across all major components to maintain consistent reference levels for communication signals. However, power routing is carefully managed to prevent ground loops and electromagnetic interference, particularly between the ESC and sensor systems. The power system architecture also facilitates modular testing, allowing each subsystem to be powered independently for diagnostics or software development.

In summary, the power distribution system efficiently converts and delivers energy from the LiPo battery to all critical subsystems. This ensures stable operation, electrical isolation, and reliability throughout the drone's mission lifecycle.

7.7 Communication Architecture

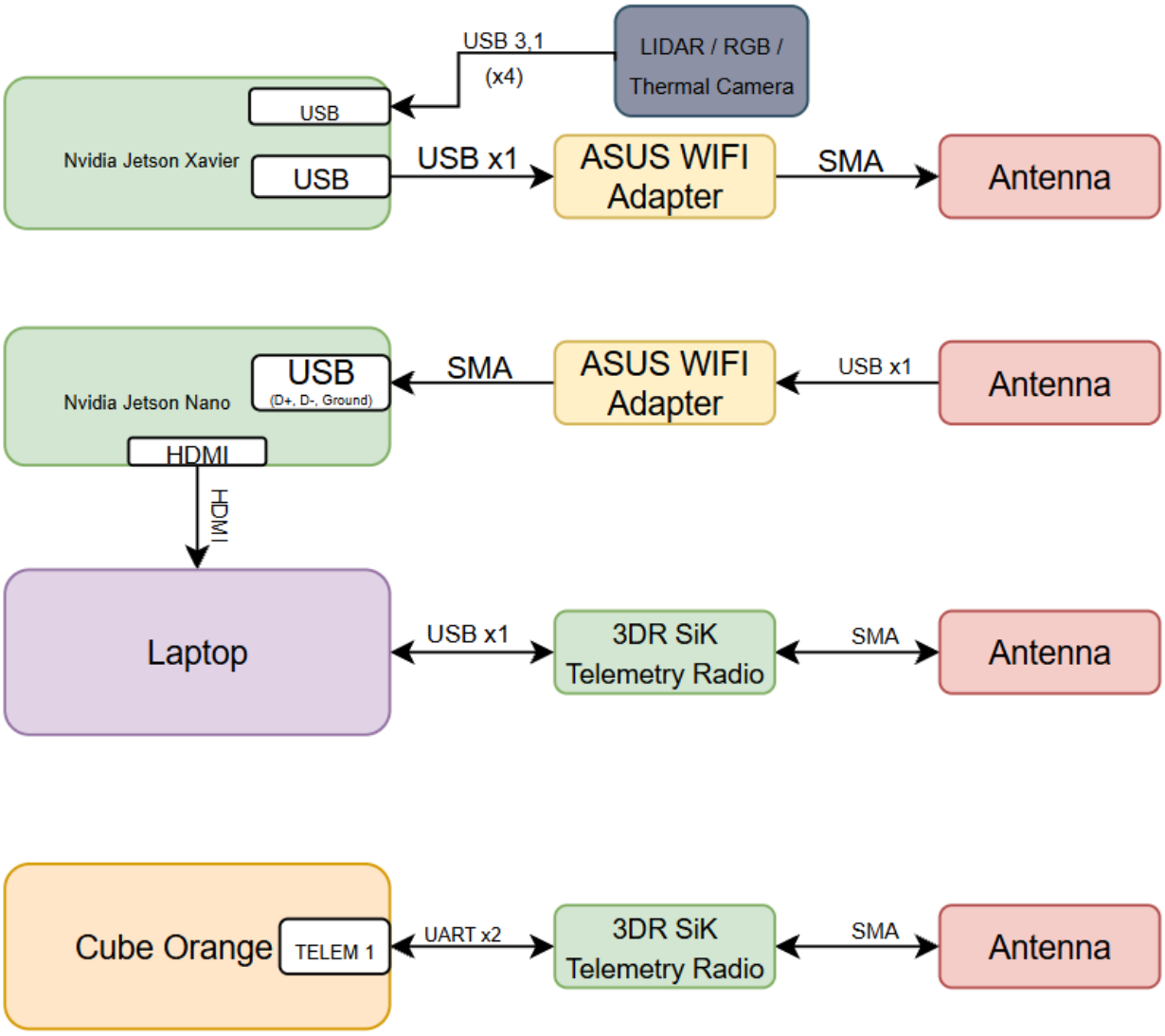


Figure 7.7 Communications Wiring and Signal Flow

By Authors

The communication architecture of the SOAR system is designed to support reliable, high-speed data exchange between the flight controller, companion computer, sensors, and peripheral modules. This network of communication interfaces forms the digital backbone that allows the drone to operate autonomously while maintaining coordination across all onboard subsystems.

At the heart of the communication framework is the MAVLink protocol, which enables structured data transfer between the Cube Orange flight controller and the Jetson Xavier companion computer. This communication occurs through a UART connection on the Jetson's 40-pin header. MAVLink allows the Jetson to receive flight telemetry, attitude data, and sensor information from the Cube while simultaneously sending high-level mission commands and

navigation inputs. The same protocol is used by the 3DR SiK Telemetry Radio, connected to the Cube's TELEM1 port, to transmit flight data to the ground control station. This ensures that mission operators can monitor system health and flight parameters in real time.

For higher-level communication and distributed computing, the Jetson Xavier communicates over USB to the radio; which allows to exchange sensor streams, image data, and control topics efficiently; then interprets and translates the results into actionable flight commands via MAVLink. The use of Ethernet provides the necessary bandwidth for transmitting uncompressed or lightly compressed video data, which is essential for tasks such as real-time object tracking and environmental mapping.

The CAN bus network connects the Cube Orange to the Here3 GPS module. This two-wire differential system (CAN High and CAN Low) provides reliable communication even in electrically noisy environments. The CAN protocol's robustness ensures that critical positioning and orientation data are transmitted without interference from the high-current ESC or motor lines.

For peripheral sensors such as the LIDAR, RGB camera, and thermal camera, communication occurs through the USB 3.1 interface on the Jetson Xavier. These connections deliver high-speed data transfer for simultaneous sensor operation. The USB ports are directly managed by the Jetson's internal controllers, allowing each device to be enumerated and synchronized in the ROS 2 framework.

The ESC telemetry link provides another essential feedback path. It uses a one-way UART line from the ESC's TX pin to the Cube Orange's GPS2 RX port, transmitting real-time motor diagnostics such as RPM, current, and temperature. This information enhances flight safety by allowing the Cube to detect anomalies or inefficiencies in the propulsion system.

Overall, the SOAR communication architecture integrates multiple bus standards; UART, CAN, Ethernet, and USB. Each is selected for its strengths in bandwidth, robustness, and latency. This layered communication model ensures that time-critical control signals are prioritized while allowing high-bandwidth sensor data to flow concurrently across the system.

7.8 Integration and Safety Considerations

System integration and safety form the foundation of the SOAR platform's electronic design. Given the combination of high-current propulsion components, sensitive signal lines, and advanced computing hardware, special care was taken to ensure electrical isolation, thermal protection, and fault tolerance across all subsystems.

One of the primary safety measures is electrical isolation between the power and logic domains. The Cube Orange, Jetson Xavier, and peripheral sensors are powered through regulated 5V and 19V supplies, while the motors and ESC operate directly from the LiPo battery. This separation prevents voltage fluctuations from the propulsion system from propagating into the sensitive control electronics. Common grounding is maintained throughout the system to ensure consistent

reference levels for all communication signals. To further reduce noise and interference, all high-current wires, particularly those between the ESC and motors, are routed away from low-voltage data lines and are twisted to minimize electromagnetic emission.

The connectors used across the system were chosen based on reliability and ease of maintenance. DF13 and JST-GH connectors are used for UART, CAN, and telemetry connections, providing secure locking mechanisms to prevent disconnections during vibration. XT60 connectors are used for the main LiPo battery and ESC power lines due to their high current handling capacity and low resistance. Each connector is labeled and color-coded to simplify integration and field servicing.

Thermal management is another critical aspect of system safety. The Jetson Xavier is equipped with an active cooling module that includes a heatsink and fan assembly, ensuring stable operation even under intensive AI workloads. Similarly, the ESC features an integrated aluminum plate that acts as a heat spreader, dissipating thermal buildup from the MOSFETs. During operation, the ESC's temperature is continuously monitored through its telemetry feedback, and thermal warnings are sent to the Cube if thresholds are exceeded.

Redundancy and fault detection are built into the communication network. The Cube Orange maintains redundant sensors internally through its triple IMU system, and the Here3 GPS provides real-time correction data for increased navigation accuracy. ESC telemetry and current sensing allow the system to detect faults such as motor stalls or power loss early.

Cable management and harness design were also optimized for flight reliability. Wires are bundled and secured using heat shrink and cable sleeves to minimize movement during flight, reducing the risk of wear or accidental disconnection. The wiring layout maintains short signal paths to minimize latency and noise pickup, particularly between the Cube, ESC, and Jetson.

Collectively, these integration and safety measures ensure that the electronic hardware not only performs reliably during standard operation but can also handle abnormal conditions such as power surges, overheating, or signal interference. This careful engineering integration allows the SOAR platform to achieve stable, repeatable, and safe performance during autonomous missions.

7.9 Summary

The electronic hardware design of the SOAR system integrates multiple high-performance subsystems into a cohesive and reliable architecture optimized for autonomous aerial operations. Each component including the Cube Orange flight controller, Jetson Xavier companion computer, T-Motor F66A 4-in-1 ESC, sensor suite, and power regulation network, plays a vital role in ensuring that the platform performs safely and intelligently during complex rescue missions.

The Cube Orange manages low-level flight control, stabilization, and sensor fusion while maintaining direct communication with the ESC and telemetry radio. Its integration with the Jetson Xavier via UART allows for a clear separation of responsibilities: the Cube handles

real-time control, while the Jetson executes high-level AI and perception tasks. This division not only enhances reliability but also leverages the strengths of each processor to achieve an efficient balance between computational power and deterministic response.

The Jetson Xavier, acting as the system's central processor, interfaces with the wifi card, LIDAR, RGB, and thermal cameras to process data using advanced machine learning algorithms. Through MAVLink, the Jetson translates this intelligence into actionable flight commands for the Cube, creating a closed-loop control system that blends autonomy with precise execution.

The T-Motor F66A ESC bridges the digital control and physical motion layers, converting DShot commands into motor actuation signals. Its telemetry feedback provides valuable diagnostic data that enhances the Cube's situational awareness, allowing for fault detection and preventive responses. The integration of CAN-based GPS communication further ensures robust and noise-immune navigation even in electromagnetically active environments.

Power distribution across the system is managed by dedicated regulators that provide clean and stable voltage levels to all components. The separation between the high-current propulsion lines and low-power logic circuitry minimizes interference, ensuring reliable communication and consistent operation. Each power path and connector was chosen to optimize efficiency, safety, and serviceability under the demanding conditions of field deployment.

Overall, the SOAR electronic hardware architecture represents a carefully engineered balance of performance, modularity, and robustness. The combination of intelligent control, high-speed communication, and precise power management allows the drone to operate autonomously and safely while performing complex search and rescue missions. This integrated design establishes a solid foundation for future system expansion, whether through the addition of new sensors, more powerful compute modules, or improved autonomy algorithms.

Chapter 8 - Software Design

8.1 Object Detection and Obstacle Avoidance

Using LiDAR's 2D output, the system employs a proximity filtering algorithm that prioritizes the nearest object per 45° sector around the drone, similar to the proximity sensor framework in ArduPilot's avoidance library. Implementation for obstacle avoidance is explained below.

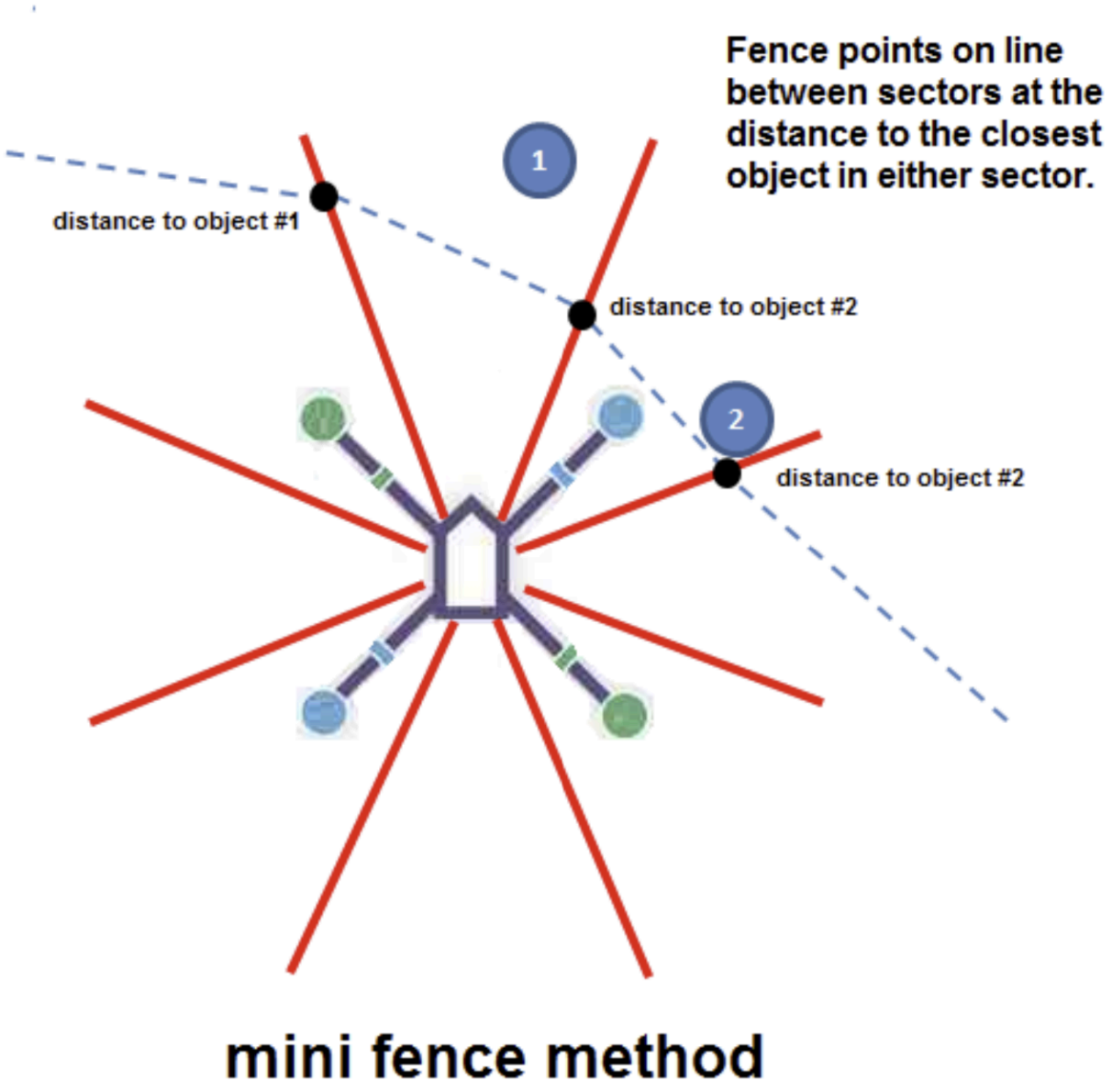


Figure 8.1.1 ArduCopter Sensor Coverage Diagram

By ArduPilot Team, permitted via open-source non-commercial policy

The obstacle avoidance system operates entirely onboard the Jetson AGX Xavier, utilizing data from the RPLiDAR C1 sensor to perceive the drone's surroundings in real time. The RPLiDAR C1 communicates with the Xavier using the UART protocol over a USB-to-USB-C interface, and its data stream is accessed using the rplidar_ros2 package within the ROS 2 framework. Once initialized, the rplidar_ros2 node publishes the LiDAR scan data to the "**/scan**" topic, which provides a 2D polar representation of the environment, including both distance and intensity information for each detected point.

After receiving this scan data, a preprocessing step divides the 360° LiDAR field of view into eight equal sectors, each spanning 45 degrees. This sectorization simplifies obstacle analysis by allowing the system to evaluate the environment in directional regions rather than individual points. During this stage, the software identifies the minimum distance to an obstacle within each sector, effectively mapping out where potential collisions might occur.

Next, the processed sector data is passed into the obstacle avoidance module, which uses a two-part strategy combining the Mini Fence Method and A* path planning. The Mini Fence Method acts as a reactive layer, it monitors each sector's proximity data and generates immediate avoidance vectors to steer the drone away from nearby obstacles. If the obstacle cannot be avoided with simple local maneuvers, the A* algorithm is engaged to compute a new global path around the obstacle. This ensures that the drone not only avoids collisions but also maintains an efficient and navigable trajectory toward its goal.

If new avoidance vectors or path updates are generated, the Xavier transmits these updated flight commands to the Cube Black flight controller using the MAVLink protocol over a UART connection. The Cube Blackthen executes the updated trajectory by adjusting the drone's motor outputs accordingly. If no avoidance vectors are required, the Xavier continues to monitor LiDAR data in real time, looping continuously to ensure dynamic obstacle detection and avoidance throughout the flight.

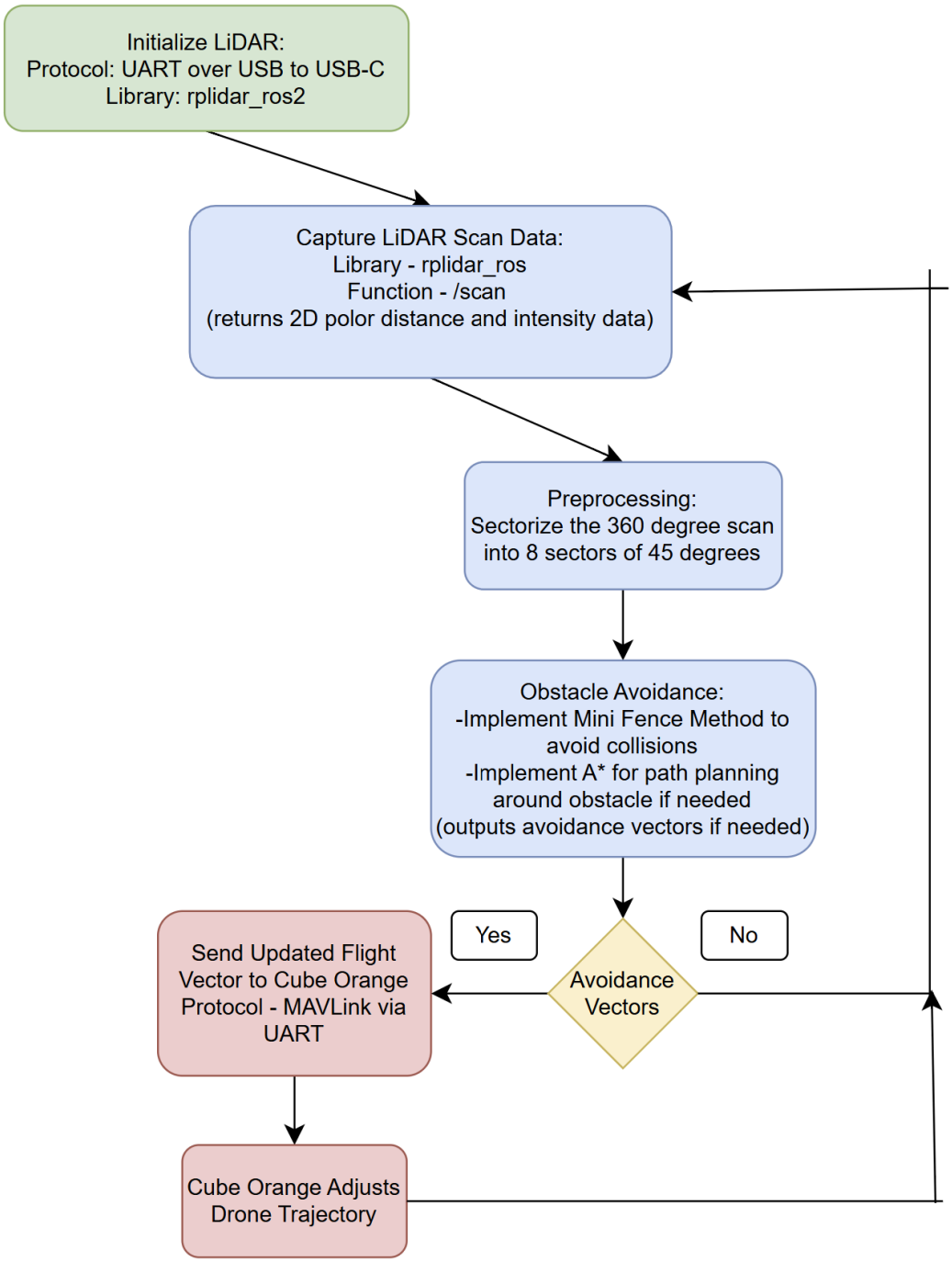


Figure 8.1.2 Obstacle Avoidance via LiDAR

By Authors

The object detection subsystem operates fully on the Jetson AGX Xavier and is responsible for identifying people using visual data collected from the drone's onboard camera. The camera connects to the Xavier via USB-C, using the USB UVC protocol for device communication and the V4L2 (Video for Linux 2) library to detect and register the camera as a video input source. Once initialized, GStreamer is used to manage the continuous capture of the camera's video stream. Specifically, the v4l2src element in GStreamer retrieves the raw video frames from the camera and passes them downstream for preprocessing.

Before inference, the Xavier preprocesses each incoming frame using GStreamer's nvvideoconvert and nvvidconv plugins, which handle necessary color-space conversions, resizing, and format optimization to prepare the data for efficient GPU processing. After preprocessing, the video stream is passed to the TensorRT inference engine, where the trained deep learning model—either YOLOv5 or NanoOWL—is executed. Each drone in the swarm uses a slightly different model configuration depending on its camera modality: Drone 1 uses an RGB-trained YOLOv5 model, Drone 2 uses an infrared (IR) version of the same architecture, and Drone 3 runs a NanoOWL or YOLOv5 model once its third sensor modality is confirmed.

Once inference is complete, the postprocessing stage interprets the model's output by drawing bounding boxes and labeling detected humans within the video frames. This ensures that detections are clearly visualized in the encoded video stream. When a human is detected, the system triggers an additional process: the Xavier sends a MAVLink message over UART to the Cube Black flight controller containing the GPS coordinates of the detected person. The Cube Black then relays this information back to the ground station, reporting the precise location of the detected individual.

In parallel, the Xavier encodes the processed video stream using GStreamer's NVENC hardware encoder, which compresses the output using H.264 or H.265 standards for efficient transmission. The final encoded video stream is then broadcast to the ground station using the RTSP (Real-Time Streaming Protocol). All steps within the dashed boundary—frame capture, preprocessing, inference, postprocessing, and encoding—operate continuously in a closed loop to maintain real-time performance during the mission.

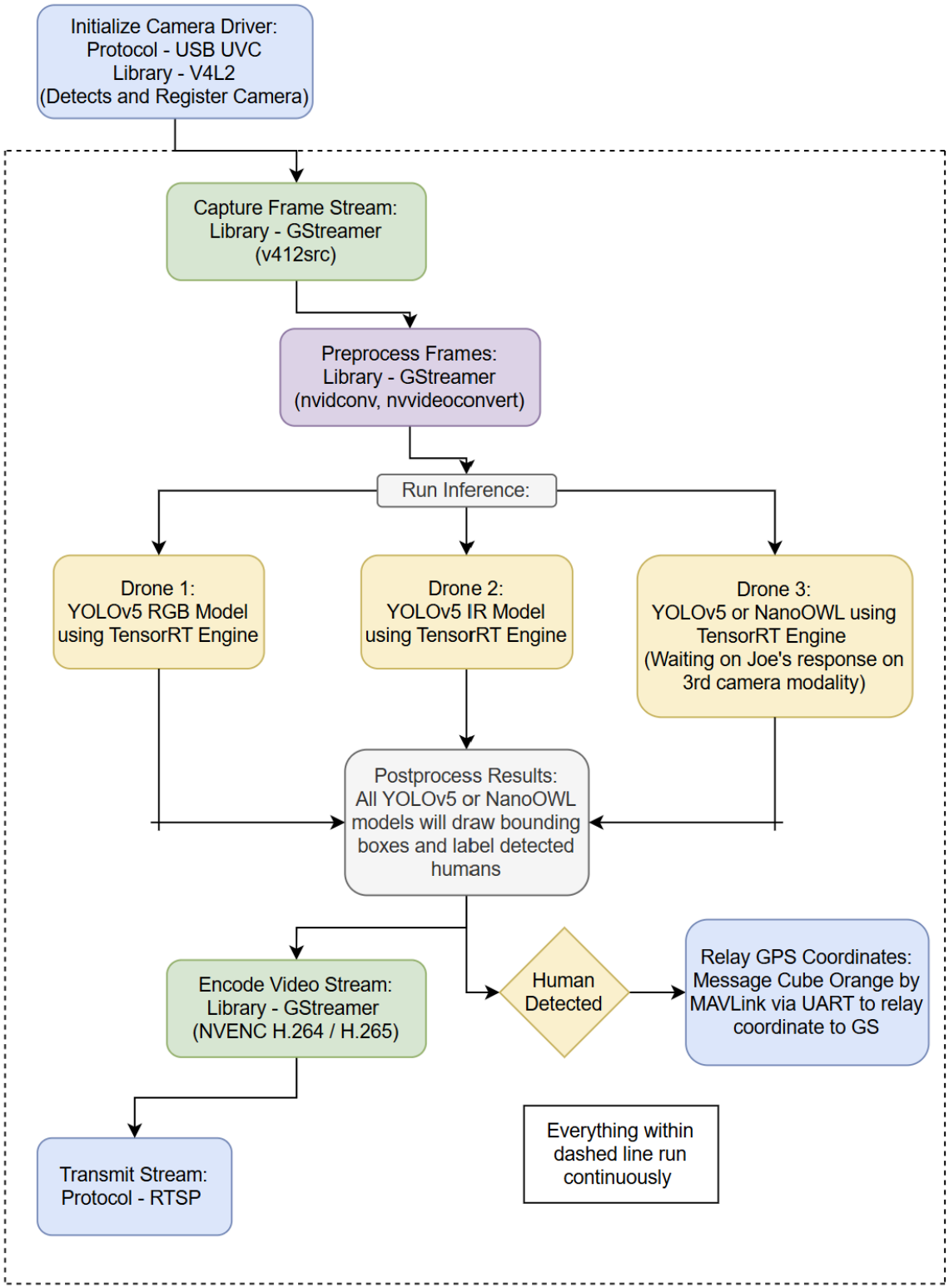

Initialize Camera Driver:
Protocol - USB UVC
Library - V4L2
(Detects and Register Camera)
Capture Frame Stream:
Library - GStreamer
(v412src)
Preprocess Frames:
Library - GStreamer
(nvidconv, nvvideoconvert)
Run Inference:
Drone 1:
YOLOv5 RGB Model
using TensorRT Engine
Drone 2:
YOLOv5 IR Model
using TensorRT Engine
Drone 3:
YOLOv5 or NanoOWL using
TensorRT Engine
(Waiting on Joe's response on
3rd camera modality)
Postprocess Results:
All YOLOv5 or NanoOWL
models will draw bounding
boxes and label detected
humans
Encode Video Stream:
Library - GStreamer
(NVENC H.264 / H.265)
Human
Detected
Relay GPS Coordinates:
Message Cube Orange by
MAVLink via UART to relay
coordinate to GS
Transmit Stream:
Protocol - RTSP
Everything within
dashed line run
continuously

Figure 8.1.3 Human Detection via onboard camera processed through Jetson AGX Xavier

By Authors

The initialization flowchart outlines the startup sequence executed by the Jetson AGX Xavier before any main operations such as object detection or obstacle avoidance begin. Upon power-up, the Xavier boots its operating system and initializes all necessary hardware drivers. This includes those for the RGB camera, LiDAR, and flight controller. Once hardware is detected, the system loads essential software frameworks such as ROS 2, GStreamer, and TensorRT, and establishes communication links through protocols like USB UVC for the camera, UART for the LiDAR, and MAVLink for the Cube Black flight controller. The initialization process also involves verifying sensor data streams, loading pretrained YOLOv5 or NanoOWL inference models, and setting system parameters such as scan rates or detection thresholds. Once all devices and libraries are successfully configured and verified, the system reports a ready status, signaling the transition into the main operational loop where real-time human detection and obstacle avoidance are carried out.

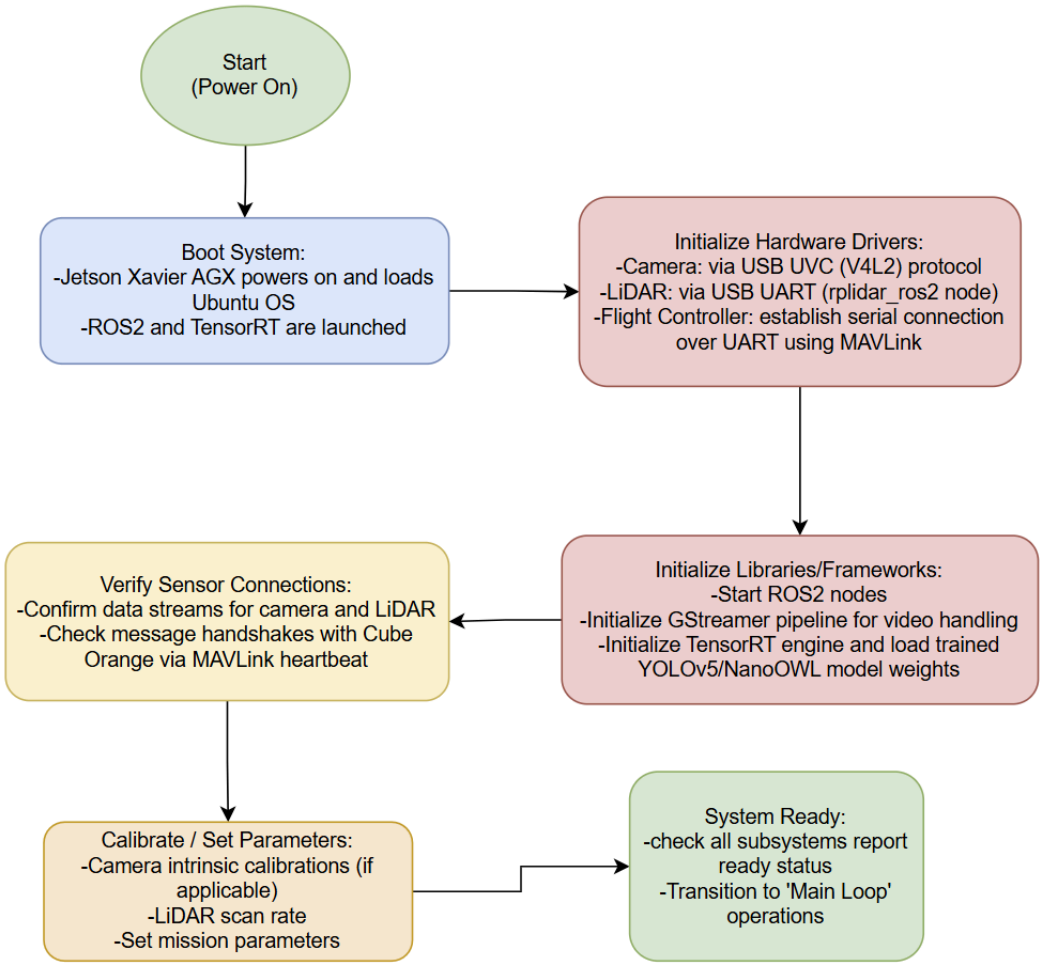


Figure 8.1.4 Initialization Process for Jetson Xavier AGX

By Authors

8.2 Communication and Ground Control

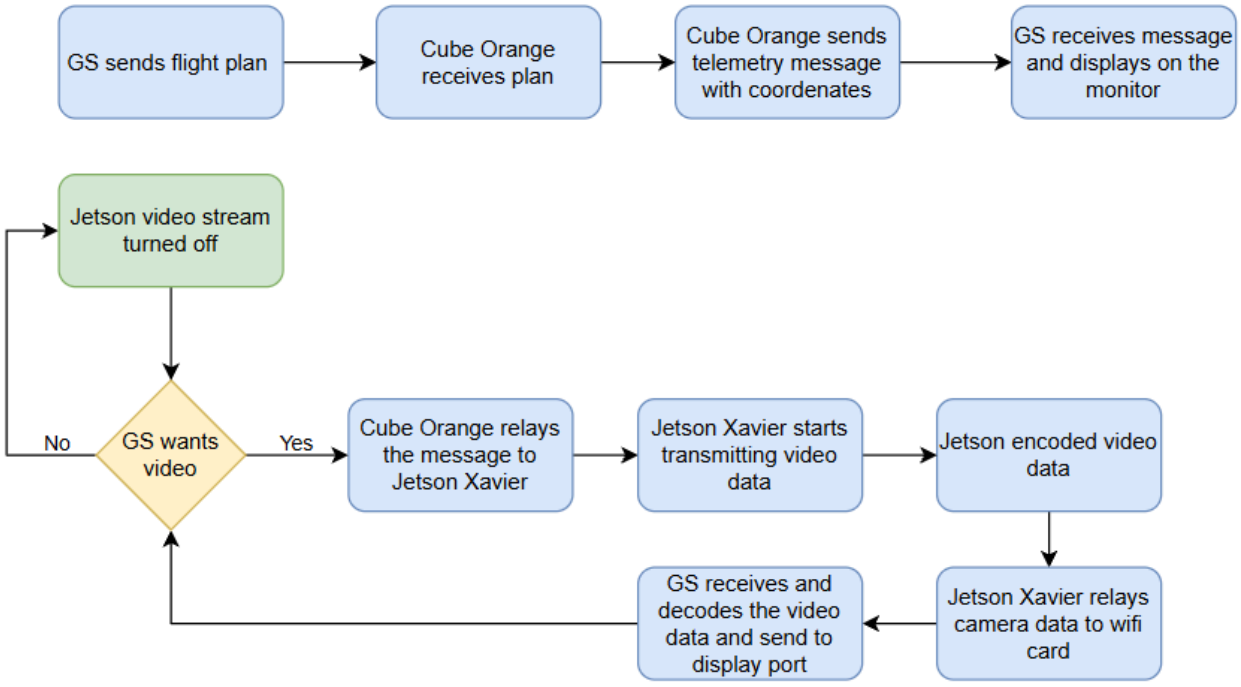


Figure 8.2 Communication Process for Jetson Xavier

By Authors

The communication and ground control system serves as the operational link between the drone swarm and the Ground Control Station (GCS). Each drone transmits telemetry data, sensor feedback, and mission updates to the GCS through a reliable wireless network, ensuring seamless coordination during autonomous missions. The system utilizes a mesh-based communication structure that allows both drone-to-drone and drone-to-ground data exchange, maintaining connection integrity even in challenging environments. The GCS provides real-time visualization of flight paths, sensor readings, and mission progress, enabling the operator to oversee the swarm's activities and intervene when necessary.

This setup ensures efficient command distribution and situational awareness across all swarm units. If a communication interruption occurs, the onboard Jetson module and Cube Black flight controller work in tandem to maintain autonomous function, preserving mission continuity.

Commands from the GCS, such as new waypoints or mission abort signals, are sent through MAVLink protocols, while telemetry and video feeds are returned via low-latency radio or Wi-Fi links. This integration of ground control oversight and autonomous resilience creates a fault-tolerant communication network that ensures both operator safety and mission reliability.

8.3 Swarm Operations

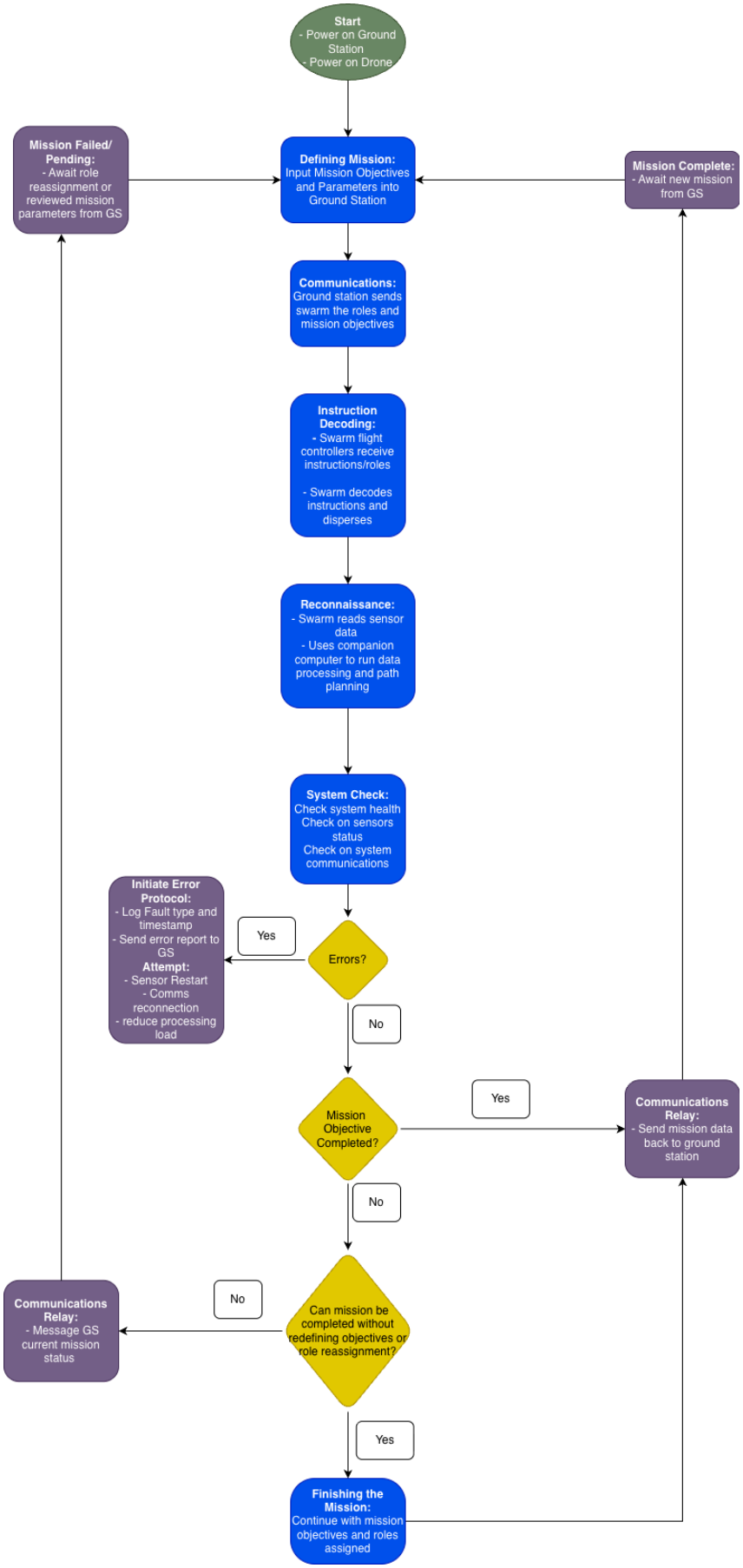

Start
- Power on Ground Station
- Power on Drone
Mission Failed/ Pending:
- Await role reassignment or reviewed mission parameters from GS
Defining Mission:
Input Mission Objectives and Parameters into Ground Station
Mission Complete:
- Await new mission from GS
Communications:
Ground station sends swarm the roles and mission objectives
Instruction Decoding:
- Swarm flight controllers receive instructions/roles
- Swarm decodes instructions and disperses
Reconnaissance:
- Swarm reads sensor data
- Uses companion computer to run data processing and path planning
System Check:
Check system health
Check on sensors status
Check on system communications
Initiate Error Protocol:
- Log Fault type and timestamp
- Send error report to GS
Attempt:
- Sensor Restart
- Comms reconnection
- reduce processing load
Yes
Errors?
No
Mission Objective Completed?
Yes
Communications Relay:
- Send mission data back to ground station
No
Communications Relay:
- Message GS current mission status
No
Can mission be completed without redefining objectives or role reassignment?
Yes
Finishing the Mission:
Continue with mission objectives and roles assigned

Figure 8.3 Swarm Algorithm Software Diagram

By Authors

During swarm operations, the process begins with system initialization, where both the ground station and aerial units are powered on and synchronized. The operator inputs the mission parameters and objectives into the ground station, which then assigns roles and operational zones to each drone based on mission requirements and sensor capabilities. After receiving and decoding their instructions, the swarm disperses and begins executing assigned tasks while continuously performing system health checks. The companion computer and flight controller jointly manage two parallel decision sequences: system integrity and mission progress. The system integrity sequence monitors sensor performance, communication stability, and subsystem health, activating an error-handling protocol if faults are detected. This protocol includes fault logging, timestamping, reporting to the ground station, and executing recovery procedures such as sensor restarts, communication reconnection, or computational load redistribution to maintain stability. Simultaneously, the mission progress sequence determines whether objectives have been met; if so, the drone transmits mission data back to the ground station and awaits further instructions. If objectives remain incomplete, the system evaluates whether they can be achieved with the current configuration, continuing if feasible or reporting to the ground station for reassignment if not. This closed-loop framework enables the swarm to operate autonomously with adaptive reconfiguration, fault tolerance, and reliable coordination across dynamic mission environments.

8.4 Landing Sequence and Failsafe Behavior

When landing conditions are met (low battery, mission completion, or manual override), the Jetson sends a “Land” command to ArduPilot. The flight controller transitions to *LAND_MODE*, reducing throttle gradually while monitoring barometric altitude. If communication is lost, the Cube autonomously executes a *Return-to-Launch* (RTL) failsafe.

8.5 Summary

The SOAR 2025 software stack achieves modular, resilient, and intelligent control of autonomous drone flight. By combining the robustness of ArduPilot, the computational power of Jetson Xavier, and the network efficiency of ROS 2 DDS, the team has implemented a system that mirrors aerospace-grade autonomy while maintaining open-source adaptability.

Chapter 9 - Prototyping and Fabrication

9.1 Drone Body

After conferring with customers, it was encouraged that the SOAR project team is to use a drone kit (Figure 9.1.1) for the frame design of the drones used in the SOAR project. These drone kits offer affordability and reliability as they are substantially more cost effective than manufacturing a unique drone design and since this drone kit has been used in previous implementations of the SOAR Project.

Figure 9.1.1 Initial Drone Kit

-From Amazon Webstore

In order to fit all the on board electronics and components on the drone body, the team elected to employ a custom design that implements a hatch system so these components are also easy to access for modification or maintenance. The new body plates will be constructed using ¼ inch Birch wood using a laser cutter from the UCF TI Lab. This material selection is largely due to the visible stress that occurred on the MDF wood used in the previous team's implementation of the central drone body. Also, different screws will be necessary for the mounting of the drone arms due to the change in thickness of the central drone frame.

To select the most stable design, the components and their placement will be determined with the assistance of simulations in order to find the placements with the most desirable center of gravity in mind. These components that will be onboard each drone include the battery, flight controller, ESC, transceiver, along with wiring and other needed components.

In an additional attempt to reduce shock, it has been determined that beams will be implemented on the middle of each side of the drone body in order to reduce the stress that may occur when the landing gear comes in contact with the ground. In the original kit design, the landing gear attaches to the central drone body at the weakest point on the lower central body. In order to reduce the stress that may occur here when encountering shock, I-shaped beams will be placed here in order to prevent the wood body from yielding. As other failures and issues become

apparent, further implementations will be made in order to produce the most effective and stable product.

9.2 Drone Construction

The current construction guide for the drone is to follow a bottom to top approach of construction. The team first plans to gather and procure all necessary parts for the drone body and frame. Next, before the drone frame is assembled, to mount the electronic components to their selected positions and establish their necessary connections. Following this, the drone frame will be put together ensuring that all connections and wirings are secure. This process is very basic due to the hatch design that the team has decided to implement. The hatch allows for the simple altering of on-board components and allows the SOAR team to quickly check on the electrical and physical connections of these devices.

Chapter 10 - System Testing and Evaluation

10.1 NVIDIA Jetson TX Implementing Human Detection

Models:

I plan to train and compare several models to determine the best-performing architecture for each sensing modality and scenario:

- Model 1: Trained on RGBTDronePerson (RGB data only)
- Model 2: Trained on RGBTDronePerson (IR data only)
- Model 3: Trained on POP (IR data)
- Model 4: Trained on AU-AIR (RGB data)
- Model 5: Trained on AU-AIR + Canny Edge Detection (RGB data)
- Model 6: Trained on a combination of RGBTDronePerson (RGB) and AU-AIR datasets to detect both humans and vehicles (an experimental extension)
- Model 7: Fine-tune POP's best weights on FLIR_ADAS_v2 (IR data only)
- Model 8: Fine-tuned RGBT* + AU-AIR's best weights on FLIR_ADAS_v2 (RGB data only)

The goal is to compare detection accuracy, robustness, and real-time inference capability to identify which configuration performs best for onboard Jetson deployment.

10.1.1 Human Detection Model Evaluation

Definitions:

- **Recall**: measures the model's ability to find all actual positive instances

$$\frac{True\ Positives}{True\ Posities\ +\ False\ Negatives}$$

- **Precision**: measures how accurate the model's predictions are

$$\frac{True\ Positives}{True\ Positives\ +\ False\ Positives}$$

- **mAP@0.5:** computes the average precision for a class and averages them. A prediction box counts as correct if the Intersection over Union (IoU) with the ground truth is at least 0.5
- **mAP@0.5:95:** the average precision across 10 IoU thresholds from 0.5 to 0.95 with a step of 0.05. This metric is stricter and evaluates localization accuracy and confidence robustness

Evaluation of Models:

- Model 1:

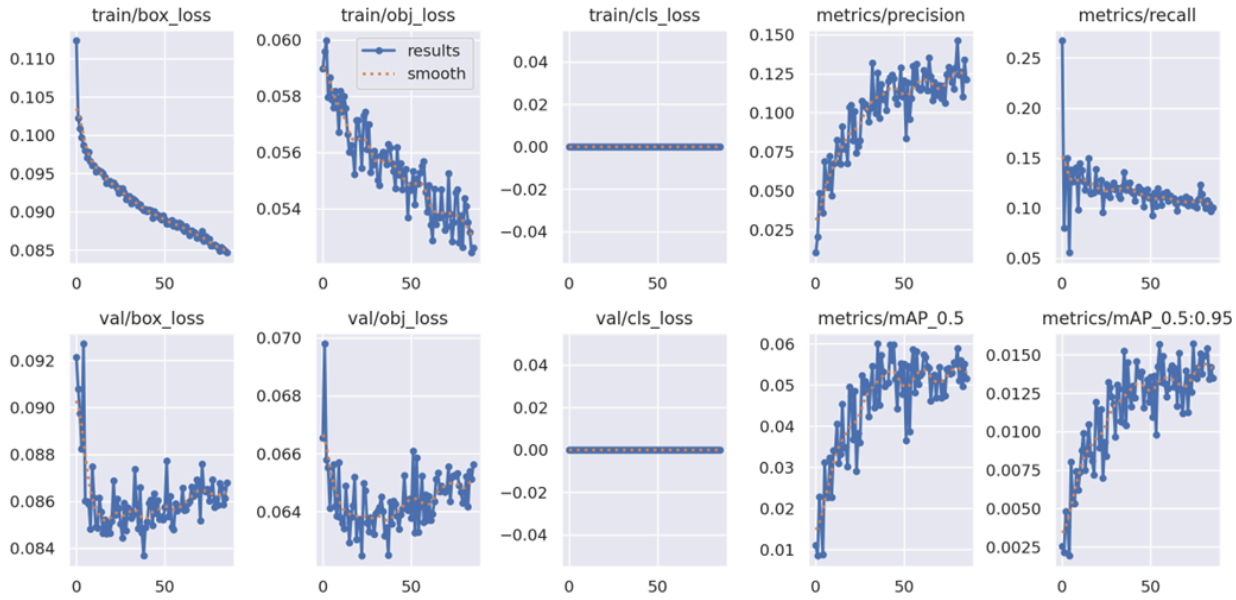


Figure 10.1.1 Model 1 Datasets Performance

By Authors

-Precision is roughly 12.5%

-Recall is roughly 10%

Results:

In theory, Model 1 (human detection using the RGB camera trained on the RGBT* Visible dataset) demonstrates very weak detection performance, achieving approximately 12.5% precision and 10% recall. This means that when the model identifies a human, it is correct about 1 out of 10 times, and it successfully detects about 10% of all humans present in the scene, missing roughly 9 in 10.

- Model 3:

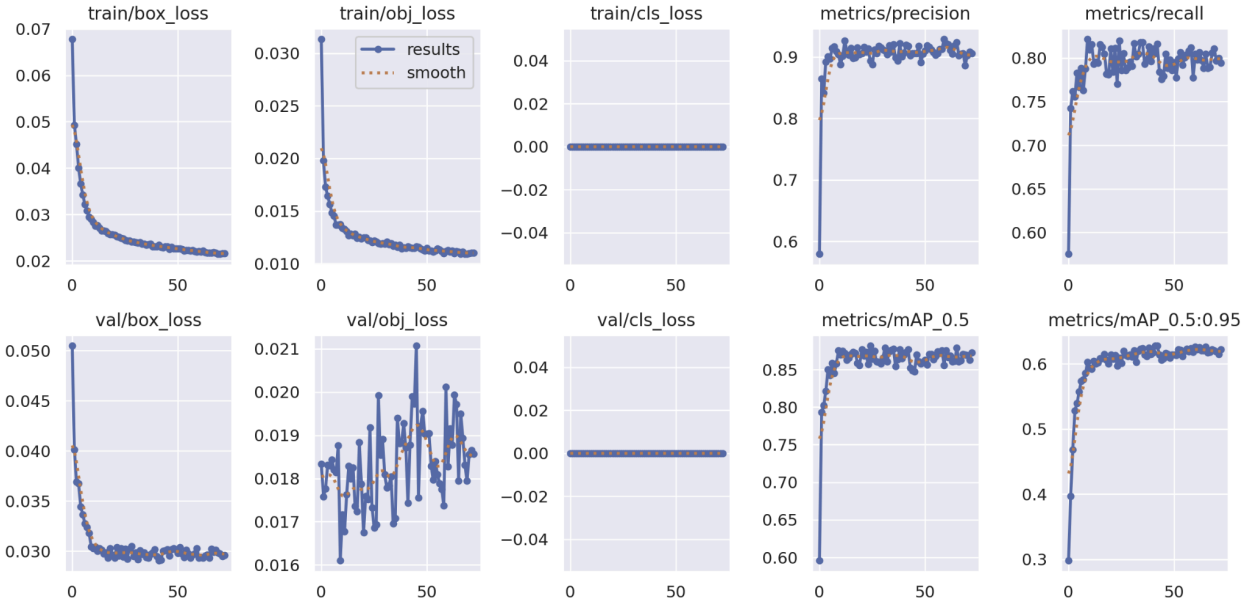


Figure 10.1.3 Model 3 Datasets Performance

By Authors

-Precision is roughly 90%

-Recall is roughly 80%

Results:

In theory, Model 3 (human detection using the IR camera trained on the POP dataset) demonstrates strong detection performance, achieving approximately 90% precision and 80% recall. This means that when the model identifies a human, it is correct about 9 out of 10 times, and it successfully detects about 80% of all humans present in the scene, missing roughly 1 in 5.

- Model 4:

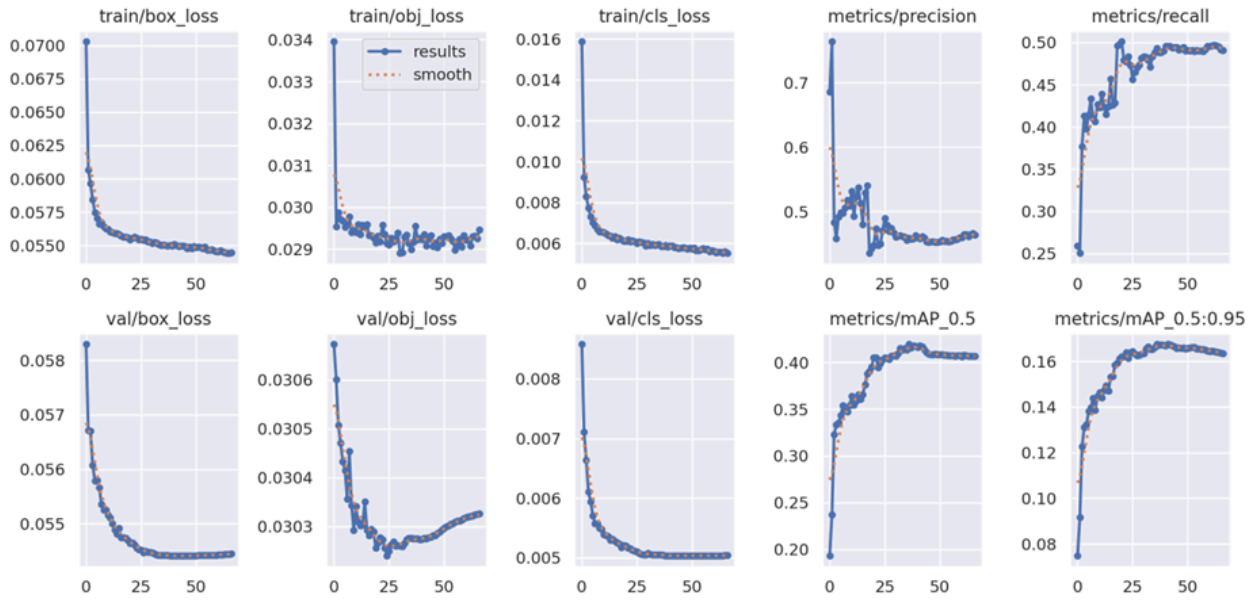


Figure 10.1.4 Model 4 Datasets Performance

By Authors

-Precision is roughly 45%

-Recall is roughly 50%

Results:

In theory, Model 4 (human detection using the RGB camera trained on the AU-AIR dataset) demonstrates decent detection performance, achieving approximately 45% precision and 50% recall. This means that when the model identifies a human, it is correct about 5 out of 10 times, and it successfully detects about 50% of all humans present in the scene, missing roughly 5 in 10.

- Model 5:

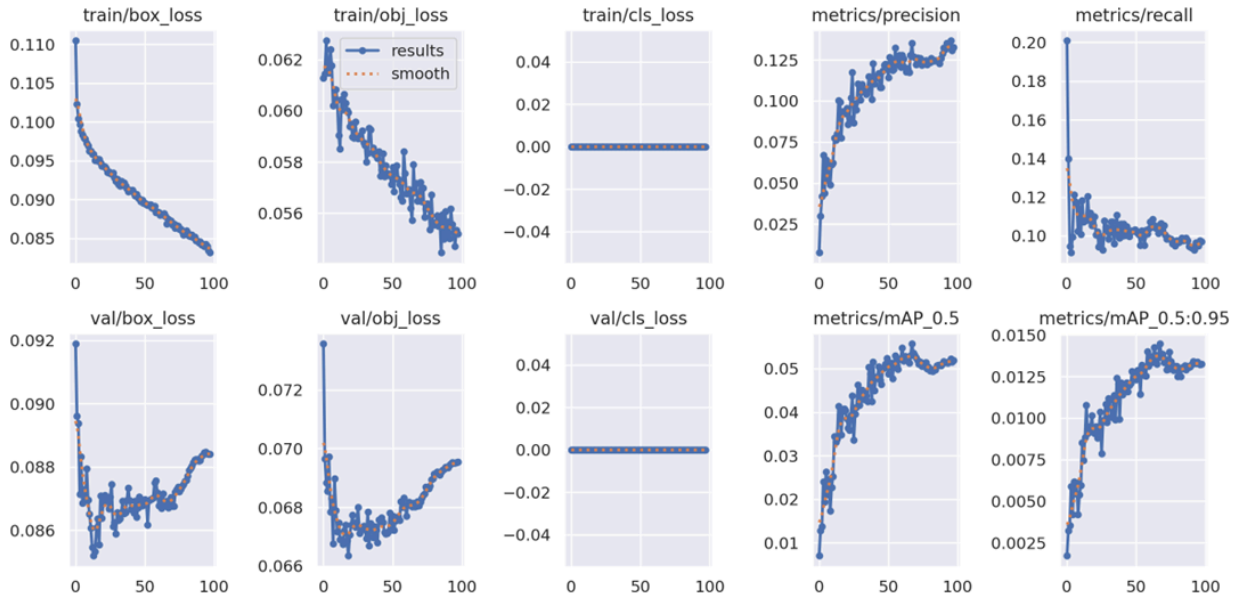


Figure 10.1.5 Model 5 Datasets Performance

By Authors

-Precision is roughly 12.6%

-Recall is roughly 10%

Results:

Since AU-AIR produced the best results out of the two RGB datasets, Canny edge detection was added to this dataset. In theory, Model 5 (human detection using the RGB camera trained on the AU-AIR + Canny edge detection dataset) demonstrates weak detection performance, achieving approximately 12.6% precision and 10% recall. This means that when the model identifies a human, it is correct about 1 out of 10 times, and it successfully detects about 10% of all humans present in the scene, missing roughly 9 in 10.

- Model 6:

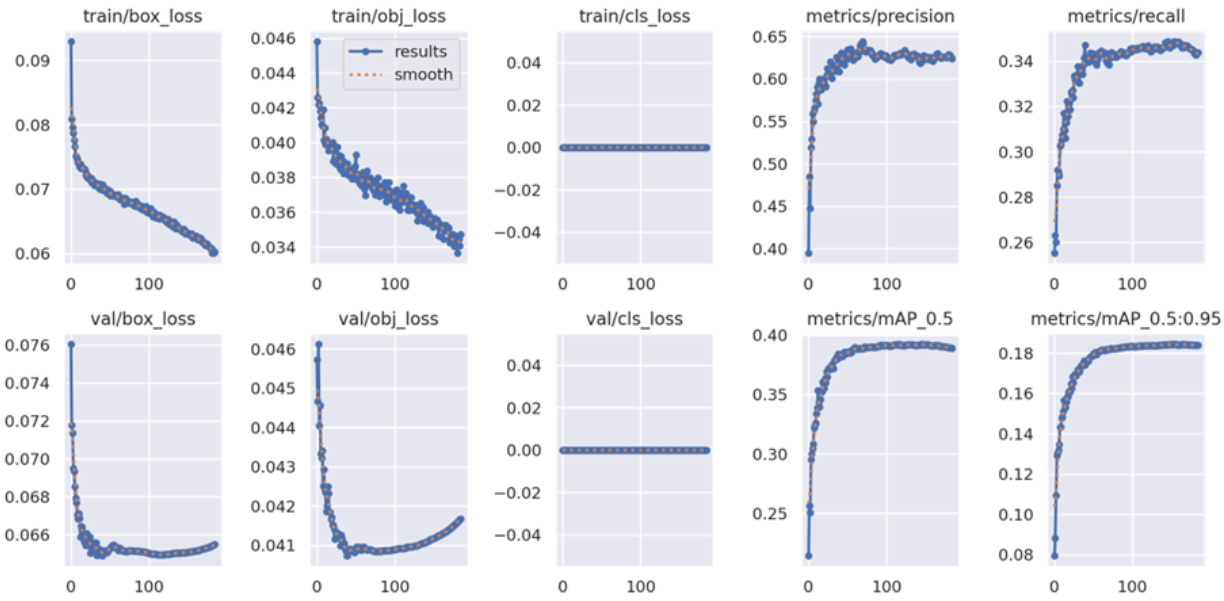


Figure 10.1.6 Model 6 Datasets Performance

By Authors

-Precision is roughly 62%

-Recall is roughly 34%

Results:

Since adding Canny edge detection significantly decreased the models performance, I decided to combine both RGB datasets to increase the data used for training. In theory, Model 6 (human detection using the RGB camera trained on the AU-AIR + RGBT* Visible dataset) demonstrates decent detection performance, achieving approximately 62% precision and 34% recall. This means that when the model identifies a human, it is correct about 6 out of 10 times, and it successfully detects about 34% of all humans present in the scene, missing roughly 7 in 10.

- Model 7:

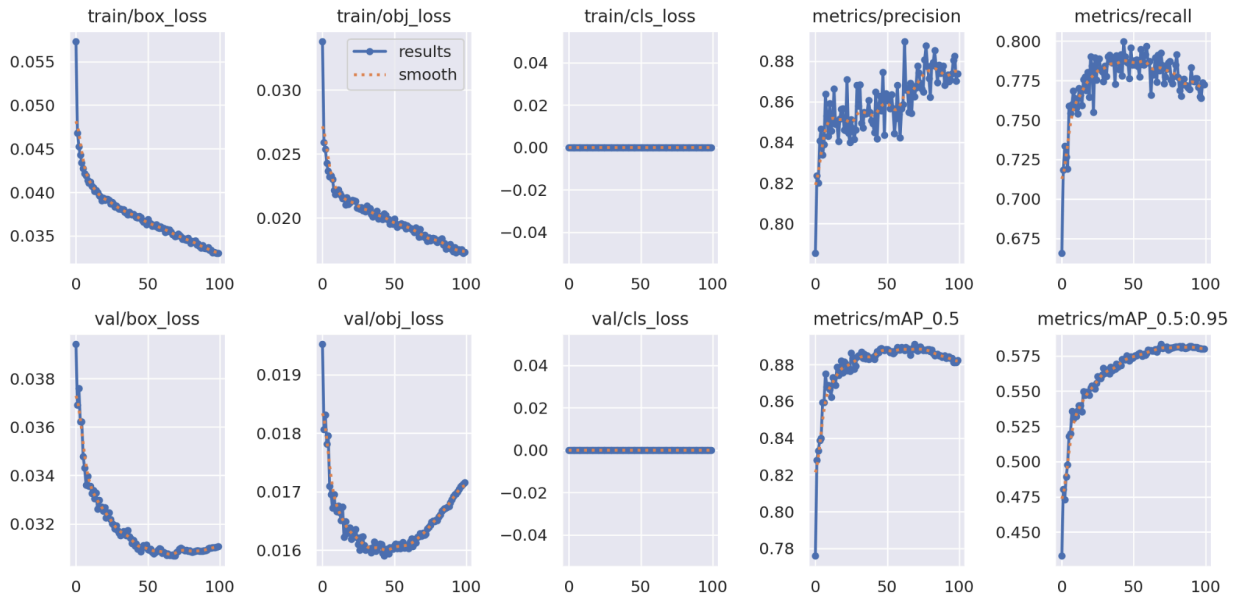


Figure 10.1.7 Model 7 Datasets Performance

By Authors

-Precision is roughly 88%

-Recall is roughly 78%

Results:

In theory, Model 7 (human detection using POP dataset's best weights fine-tuned with FLIR thermal dataset) illustrates a significant increase in detection performance and decrease in false positives, achieving approximately 88% precision and 78% recall. The FLIR dataset is a series of images created from a video. This simulates live thermal video input. An improvement of roughly 10% for precision and roughly 14% recall from beginning to end shows the importance of fine tuning a model to a video feed. This means that when the model identifies a human, it is correct about 9 out of 10 times, and it successfully detects about 78% of all humans present in the scene, missing roughly 8 in 10.

10.2 Mechanical Integration Testing

As the drones are integrated with electrical and mechanical components, each individual unit will have a slightly different center of gravity due to variations in wiring, battery placement, and component tolerances. To ensure that all drones can maintain stable and symmetrical flight during swarm operations, a series of integration and verification tests will be conducted. These tests will confirm that electrical subsystems function properly in conjunction with the mechanical structure, while also validating that autonomous flight algorithms can compensate for small

deviations in weight distribution. This stage is essential to ensure coordinated and reliable performance across the entire swarm.

Since all electrical components will undergo bench-level testing prior to full integration, the focus during this phase will shift toward verifying that component placement, wiring layout, and mass distribution allow the drone to execute mission operations safely and efficiently. Proper routing of wiring will be examined to minimize electromagnetic interference and prevent obstruction of moving parts. Early in Senior Design 2, the team will finalize the mounting of electronic systems, battery placement, and quick-release mechanisms that allow for safe disassembly or replacement in the event of a crash or emergency. Documentation of each integration step will ensure repeatability and traceability for future testing or manufacturing.

A final verification step will include a motor and propeller ground test, serving as a comprehensive start-up and functional check of the entire drone system. This test will verify motor synchronization, thrust balance, and system responsiveness to control inputs. During this process, the drone's stability and vibration levels will be monitored to identify any imbalances caused by component placement or assembly variations. Successful completion of this test will confirm that the drone's electrical, mechanical, and software systems operate harmoniously—ensuring the airframe is flight-ready and capable of performing its autonomous and swarm missions safely and effectively.

10.2.1 Flight Testing

A major component of senior design 2 for the aerospace team will be flight testing. While the autonomous test will be a key factor for our electrical team, it will be important for inspection and maintenance as each flight will give data on the structural integrity of each drone. Based on FEA analysis, the team will identify "weak" points, which will be monitored during the flight test. Based on previous design, these points will most likely be the location where the housing integrates with the arm support. These "weak" points could be the location where snapping may occur. Preventive maintenance, such as shock mitigation has been put in place to increase the usability of the drones.

Flight testing will not only test the drones in real life applications, but it will also allow for initial issues to be identified on a software side. The first round of tests will be conducted at about one second intervals, aiming to get the drone a few feet off the ground. This short duration test will allow all operational components to be turned on and verified, while not compromising the drone in its entirety.

Once initial testing is complete, the drones will be able to perform small autonomous operations. These will be 20-30 second tests, with a maximum height of 20 feet. This will provide the computer engineers the ability to test swarm operations, while the aerospace team ensures the drones are capable of flight at higher altitudes and longer time intervals.

Chapter 11 - Administrative Content

11.1 Budget

The SOAR project is supported by Lockheed Martin and the University of Central Florida (UCF), with a projected total budget of $5,000. We plan not to use the full funding and hope to use as minimal as spending as possible. This funding will be used strategically to cover both hardware and software resources necessary for the development, testing, and demonstration of the SOAR drone swarm.

The budget will be primarily allocated to the following key components:

- Drone Frames and Flight Controllers: Includes the base hardware for each of the four drones, such as ESCs, motors, propellers, and flight controllers (e.g., Pixhawk or STM32-based).
- Sensor Packages (1 per drone):
    - EO (RGB) Camera: For real-time visual feedback and target recognition.
    - Thermal (IR) Camera: For detecting human heat signatures during SAR.
    - LIDAR Module: For environmental mapping and obstacle detection.
    - Ultrasonic Sensor: For short-range proximity awareness in cluttered environments.
- Microcontrollers and Embedded Systems: Microcontrollers (e.g., STM32, ESP32) will handle sensor interfacing, real-time control, and communication.
- Communication Modules:
    - Wi-Fi mesh transceivers for drone-to-drone and drone-to-ground networking.
    - Optional LoRa or APRS modules for long-range GPS data transmission.
- Power Systems: Batteries, BMS (Battery Management Systems), and charging hardware to support extended operations and safety compliance.
- Development Tools: Software licenses (if needed), soldering kits, cabling, and 3D-printed mounts for hardware integration.
- Spare Components & Contingencies: A reserve of approximately 10–15% of the total budget will be set aside for unexpected hardware replacements or upgrades during development and testing.
- 3D Print Material: Carbon Fiber and Wood for support and stability

We plan to take advantage of Lockheed Martin's sponsorship to source advanced sensors (e.g., thermal and LIDAR) directly through their procurement channels, reducing cost burdens and ensuring the use of industrial-grade components. The budget will be tracked throughout the semester using a shared financial spreadsheet, and any major purchase decisions will be made collectively by the team and approved by the faculty advisor.

Sub-System	**Estimated Budget**
Microcontroller Units (MCUs)	$300

Communication Modules (e.g., Wi-Fi, APRS)	$600
EO (RGB) Cameras	$200
Thermal Cameras (IR)	$200
GPS Modules	$150
Additional Sensors (LIDAR, Ultrasonic)	$600
Motors & ESCs	$600
Power Systems (Batteries, Power Mgmt)	$300
Drone Frames & Assembly	$400
Licensing & Registration	$400
Frame (Material)	$900
Harnessing (Screws, additional Materials)	$150
Propellor	$100
Total	**$4900**

Table 11.1 Estimated Budget

By Authors

11.2 Distribution of Work

Below is the distribution of primary and secondary tasks assigned to each group member. Since the success of the project relies on the integration of all components—where one system depends on another—effective collaboration among team members is essential. This collaborative approach ensures that all systems are designed to be compatible and non-interfering with one another. As a result, every group member remains actively involved and informed throughout the design and implementation phases of the project.

Name/Major	Primary Responsibilities	Secondary Responsibilities
Adrian Castillo Computer Engineering	• Project Lead • Motors and Motor	• Power Supply Unit (PSU) Integration

	Control • Component Installation • GPS Selection and Implementation	• PCB Design and Implementation • Antenna and Communication Setup • Accelerometer Setup
Kristopher Tellez Computer Engineering	• Camera Selection and Implementation • Sensor Selection and Implementation • Website Design and Management • AI Software Implementation	• MCU Implementation • Barometer Integration • Peripheral Software Implementation
Michael Palin Computer Engineering	• Power Supply Unit (PSU) Integration • PCB Design and Implementation • Antenna and Communication Setup • Accelerometer Setup	• Motors and Motor Control • Component Installation • GPS Selection and Implementation
Cristian Gomez Computer Engineering	• MCU Implementation • Barometer Integration • Peripheral Software Implementation	• Camera Selection and Implementation • Sensor Selection and Implementation • Website Design and Management • AI Software Implementation
Belle Perry Aerospace Engineering	• Structural Analysis • Integration & Test • Quadcopter Design • Systems Verification • CAD Airframe	• Material Selection • FEA Analysis • Structural Integrity • Material Testing • Vibration/Shock Analysis • Flight Profiling • Environmental Testing • Hardware and Frame Verification
Robert Hagerty	• Lift/Drag Analysis	• FEA Analysis

Aerospace Engineering	• Aero Design • Propellors • Flight Dynamics • CAD Airframe	• Deformation • Stress and Strain • Propeller Calculation • Weight Distribution Calculation • Lift Design • Propulsion • Stability

Table 11.2 Distribution of Work

By Authors

11.3 Milestones

The development of this project relies on regular meetings and events to strengthen teamwork and guide progress toward success. At the start, new members are welcomed into the group, and during the first week of Senior Design I, brainstorming sessions are held to exchange ideas and lay down a clear plan for moving forward.

Project Initialization: Senior Design 1				
Start Date	Planned End Date	Required End Date	Task	Description
08/18/2025	08/18/2025	08/18/2025	Recruiting	Members Recruited: Adrian Castillo (CpE) Cristian Gomez (CpE) Kristopher Tellez (CpE) Michael Palin (CpE)
08/18/2025	08/24/2025	08/24/2025	Brainstorming	Meeting in person to decide and pick a project idea for the group
08/24/2025	09/04/2025	09/05/2025	Initial Document	First 10 Pages of Divide and Conquer task
08/24/2025	10/30/2025	10/31/2025	60-Page Document	First half of the assignment
08/24/2025	11/24/2025	11/25/2025	Final Document	Final Report for Senior Design I
08/24/2025	11/24/2025	11/25/2025	First Prototype Completion	Construct and test major hardware and software modules (drone movement, data collection, data transmission)
OCT.	OCT.	OCT.	Requirements	Agreeing on Requirements with Stakeholder
09/04/25	9/21/2025	9/22/2025	AE Documentation	AE Divide and Conquer Task
9/16/2025	9/21/2025	9/22/2025	Chapter 3	First 5 Pages (ea.) of Hardware Technologies and Component Selection
9/23/2025	9/29/2025	9/30/2025	Chapter 3	Second 5 Pages (ea.) of Hardware Technologies and Component Selection

9/30/2025	10/08/2025	10/7/2025	Chapter 3	Last 5 Pages (ea.) of Hardware Technologies and Component Selection
OCT.	OCT.	OCT.	Designing	CAD initial design developed and reviewed
OCT.	OCT.	OCT.	Designing	CAD Optimization and beginning FEA Analysis
NOV.	NOV.	NOV.	Prototype	Printing PLA Frame for Initial testing and sizing
NOV.	NOV.	NOV.	Prototype	Testing Initial Design and Determining Final Improvements
Q4	Q4	Q4	FEA Analysis	Using Ansys to run simulation flights using drag, thrust, weight, and selected materials

Table 11.3.1 Project Initialization: Senior Design 1 Milestones

By Authors

Project Initialization: Senior Design 2				
Start Date	Planned End Date	Required End Date	Task	Description
TBD-SD2	TBD-SD2	TBD-SD2	Testing Development	Define test cases for each subsystem; perform initial tests and collect data
TBD-SD2	TBD-SD2	TBD-SD2	Design Iteration and Refinement	Based on test results, improve design: fix bugs, optimize performance, improve reliability
TBD-SD2	TBD-SD2	TBD-SD2	Full System Integration	Combine all subsystems into a working drone swarm

TBD-SD2	TBD-SD2	TBD-SD2	Final Testing	Fun full mission tests that are as close as possible to real conditions
TBD-SD2	TBD-SD2	TBD-SD2	Final Report	Compile and submit the full written report; Final report for Senior Design 2

Table 11.3.2 Project Initialization: Senior Design 2 Milestones

By Authors

Project Fabrication				
Start Date	Planned End Date	Required End Date	Task	Description
08/18/2025	09/26/2025	11/25/2025	Component Selection	Make a tentative budget and BOM for the core components
08/18/2025	09/26/2025	11/25/2025	System Design	Design the overall schematic for the project
09/26/2025	11/24/2025	11/25/2025	Subsystem Testing	Test individual components / subsystems and compatibility within the project
TBD-SD2	TBD-SD2	TBD-SD2	Subsystem Integration	Combine all subsystems into a working drone swarm
TBD-SD2	TBD-SD2	TBD-SD2	Final Prototype Completion	Complete Project

Table 11.3.3 Project Fabrication Milestones

By Authors

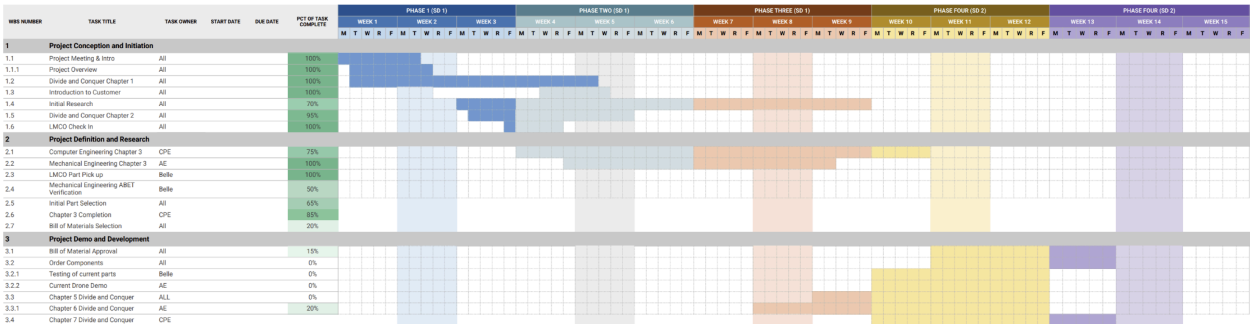
The Gantt chart for the LMCO SOAR project outlines the team's progress across multiple phases of their senior design schedule, spanning from project conception to final presentation. The chart shows that the team has successfully completed the initial project setup and research foundation. Tasks such as the introductory meetings, project overview, customer introductions, and the initial

chapter submissions have been completed at 100%. Early research and documentation, including the initial literature review and Chapter 2 work, are nearly complete with progress percentages between 70% and 95%. These early accomplishments demonstrate strong organization and collaboration among the team, ensuring that foundational requirements are met on time.

Moving into the development and testing phases, the chart reveals that the team is transitioning from research and planning toward practical implementation. Progress on the "Computer Engineering" and "Mechanical Engineering" sections is strong, with significant completion in both. However, areas such as component ordering, part testing, and demo preparation remain in the early stages, with completion percentages ranging from 0% to 20%. The Bill of Materials and quality assurance tasks are also underway but require more attention. Overall, the Gantt chart indicates that the group has effectively met its early milestones and is well-positioned to begin hands-on development, though increased focus on procurement, testing, and system integration will be essential for maintaining progress into the later phases of the project.

Figure 11.1 Gantt Chart

By Author



Bill of Materials

Bill of Materials						
Category of Item	Component	Quantity	Description	Price of Each	Price	Discipline Related to Item
Flight Controller and Components connected to	The Cube Orange	4	Flight controller	$385 (Price is combined with the carrier	$1540	Computer Engineering

it				board)		
	ADS-B	4	Carrier Board for the Flight Controller	$385 (Price is combined with the flight controller)	$1540	Computer Engineering
Ground Station Components	Nvidia Jetson Xavier	4	GPU for Computer Vision on the Ground Station	$1399	$1399	Computer Engineering
Motors and Propellers	T-Motor F66A	3	4-in-1 ESC	$139	$417	Computer Engineering
	T-Motor MN3510	13	Drone Motors	$79.90	$1038.70	
	EOLO 15x15.5	4	Carbon Fiber Propellers	$17.99	$71.96	Aerospace Engineering
Drone	X-4 500 Carbon Fiber Kit	3	Drone kit used for base of drone	$89	$267	Aerospace Engineering
	Birch Wood	1	Wood plate for designing housing			Aerospace Engineering
	Filament	3	Filament for 3D printing housing structure			Aerospace Engineering
Companion Computer and Sensors	NVIDIA Jetson TX	3	Companion Computer	$1399	$1399	Computer Engineering
	RPLiDAR C1	3	LiDAR	$67.99	$203.97	Computer Enginnering

	EMEET SmartCam C970L	1	RGB Camera	$49.99	$49.99	Computer Engineering
	(IR Camera)	1	IR Camera	~5k	~5k	Computer Engineering
	StereoLabs ZED 2i	1	RGB Stereo Camera	$519	$519	Computer Engineering
Transmitter	RDF900ux	6	Telemetry Radio	$83	$498	Computer Engineering
	ASUS USB-AC56	3	Video Radio	$72.99	$219	Computer Engineering
Antennas	915MHz Antenna	6	Telemetry Antenna	$18	$108	Computer Engineering
	TrueRC Singularity 5.8	4	Video Antenna	$20.99	$84	Computer Engineering
Power	Power Module V1.0 for Pixhawk	3	Cube Black Power Module	$13.89	$41.67	Computer Engineering
	DC-DC Regulator Module (19V)	3	NVIDIA DC-DC Regulator Module	$24.65	$74	Computer Engineering
	Matek 12S Pro BEC 12S	3	RFD900ux BEC Module	$24.99	$75	Computer Engineering
	Matek FCHUB-12 S V2	3	Power Distribution Board	$38.99	$116.97	Computer Engineering
	HRB 6S 5000mAh Lipo	3	Battery	$70	$210	Computer Engineering

	Battery					
	SKYRC B6Neo Balance Lipo Battery Charger	3	Battery Charger	$37.99	$114	Computer Engineering

Table 11.3.4 Bill Of Materials

By Authors

Appendices

Appendix A – References

[1]. M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo,
"Swarm robotics: a review from the swarm engineering perspective,"
Swarm Intelligence, vol. 7, no. 1, pp. 1–41, 2013.
https://doi.org/10.1007/s11721-012-0075-2

[2]. Pixhawk Project Team,
"Pixhawk 6x Autopilot Documentation,"
https://docs.px4.io/

[3]. K. M. Passino,
Biomimicry for optimization, control, and automation,
Springer, 2005.

[4]. APRS Protocol Reference,
"APRS101: APRS Protocol Reference,"
http://www.aprs.org/doc/APRS101.PDF

[5]. D. Floreano and R. J. Wood,
"Science, technology and the future of small autonomous drones,"
Nature, vol. 521, pp. 460–466, 2015.
https://doi.org/10.1038/nature14542

[6]. Teledyne Optech,
"Understanding LIDAR for Drones,"
https://www.teledyneoptech.com/en/lidar-for-uav/

[7]. MaxBotix Inc.,
"MB1240 XL-MaxSonar-EZ4 Datasheet,"
https://www.maxbotix.com/documents/MB1240-Datasheet.pdf

[8]. Espressif Systems,
"ESP-NOW and ESP-MESH Overview,"
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/esp-now.html

[9]. FLIR Systems,
"FLIR Lepton Integration Guide,"
https://www.flir.com/products/lepton/

[10]. MAVLink Developer Team,
"MAVLink Micro Air Vehicle Protocol,"
https://mavlink.io/en/

[11]. OpenAI,
ChatGPT (Sept 4 Version), OpenAI, 2025. [Online].
Available: https://chat.openai.com/

[12]. Alva Industries,
"Motors for drones and robotics,"
Unmanned Systems Technology, 2023.
Available: https://www.unmannedsystemstechnology.com/expo/drone-motors/

[13]. T-Motor,
"UAV motors and propulsion systems for industrial drones,"
T-Motor Official Website, Accessed Sep. 2025.
https://store.tmotor.com

[14]. DJI,
"ESCs and propulsion systems: How drone motors work,"
DJI Tutorials and Documentation, 2022.
https://www.dji.com/newsroom/news/esc-and-drone-motor-guide

[15]. Drone Nodes,
"How to choose drone motors: KV, thrust, efficiency,"
Drone Nodes Guides, 2021.
https://dronenodes.com/choose-right-motors-for-drone/

[16]. QuadMeUp,
"Drone Motor Basics: What is KV rating?,"
QuadMeUp Blog, 2020.
https://quadmeup.com/understanding-kv-and-motor-specs/

[17]. Oscar Liang,
"Best Drone Motors and How to Choose Them,"
OscarLiang.net, Aug. 2023.
https://oscarliang.com/drone-motors/

[18]. GetFPV Learn,
"Brushless Motors for FPV Drones,"

GetFPV Learn Center, 2021.
https://www.getfpv.com/learn/fpv-drone-motors/

[19]. DroneU™,
"Thrust-to-weight ratio in drones: What it means,"
Drone U Blog, 2022.
https://www.thedroneu.com/blog/drone-thrust-to-weight-ratio-explained/

[20]. T. Pham and Y. Liu,
"Power-efficient motor systems for long-endurance UAVs,"
IEEE Aerospace Conference Proceedings, 2020.
https://doi.org/10.1109/AERO47225.2020.9172734

[21]. Al-Haddad, L. A., Jaber, A. A., Giernacki, W., Khan, Z. H., Ali, K. M., Tawafik, M. A., & Humaidi, A. J. "Quadcopter Unmanned Aerial Vehicle Structural Design Using an Integrated Approach of Topology Optimization and Additive Manufacturing" *Designs, MPDI*, 2024.
https://doi.org/10.3390/designs8030058

[22]. R. Singh, R. Kumar, A. Mishra, A. Agarwal, "Structural Analysis of Quadcopter Frame," *Materials today Proceedings,* 2020.
https://www.sciencedirect.com/science/article/pii/S2214785320320757

[23]. M. Eryildiz, "Design and Analysis of a Topology-Optimized Quadcopter Drone Frame," *ResearchGate,* 2023.
https://www.researchgate.net/publication/379362481_Design_and_Analysis_of_a_Topology-Optimized_Quadcopter_Drone_Frame

[24]. E. Kuantama, R. Tarca, "Quadcopter Body Frame Model and Analysis," *ResearchGate*, 2016.
https://www.researchgate.net/profile/Endrowednes-Kuantama-2/publication/304574753_QUADCOPTER_BODY_FRAME_MODEL_AND_ANALYSIS/links/577e415008aeaa6988b096b2/QUADCOPTER-BODY-FRAME-MODEL-AND-ANALYSIS.pdf

[25]. A. Abaid, N. Parveen, B. Parveez, S. Khan, "Reviews on Design and Development of Unmanned Aerial Vehicle for Different Applications," *Journal of Mechanical Engineering Research and Developments,* 2022.
https://www.researchgate.net/profile/Abdul-Aabid/publication/360210389_Reviews_on_Design_and_Development_of_Unmanned_Aerial_Vehicle_Drone_for_Different_Applications/links/62687e561b747d19c2ab89dc/Reviews-on-Design-and-Development-of-Unmanned-Aerial-Vehicle-Drone-for-Different-Applications.pdf

[26]. Lockheed Martin, "Desert Hawk, Enhancing Warfighter Capabilities," *Lockheed Martin*, 2019.
https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/desert-hawk/Desert_Hawk_Brochure.pdf

[27]. Smithsonian, "Lockheed Martin/ Boeing RQ-3A Dark Star," National Air and Space Museum.

https://airandspace.si.edu/collection-objects/lockheed-martin-boeing-rq-3a-dark-star/nasm_A20070230000

[28]. K. Shenoy, “Design and development of a novel triphibian quadcopter,” International Journal of Engineering and Technology. https://www.academia.edu/81665798/Design_and_development_of_a_novel_triphibian_quadcopter

[29] L. Lakhdhar, A. Bouabidi, and A. Snoussi, “Impact of blade number on the aerodynamic performance of UAV propellers: A numerical and experimental study,” *Aerospace Science and Technology*, vol. 166, p. 110594, Jul. 2025, doi: https://doi.org/10.1016/j.ast.2025.110594.

[30] Liam P. Hanson, Kabilan Baskaran, Shaun F. Pullin, Beckett Yx Zhou, Bin Zang and Mahdi Azarpeyvand. "Aeroacoustic and Aerodynamic Characteristics of Propeller Tip Geometries," AIAA 2022-3075. *28th AIAA/CEAS Aeroacoustics 2022 Conference*. June 2022. https://doi.org/10.2514/6.2022-3075

[31] Kenneth W. Van Treuren, Ricardo Sanchez, Brett Bennett and Charles Wisniewski. "Design of Propellers for Minimum Induced Drag," AIAA 2021-2225. *AIAA AVIATION 2021 FORUM*. August 2021. https://doi.org/10.2514/6.2021-2225

[32] M. Podsędkowski, R. Konopiński, D. Obidowski, and K. Koter, “Variable Pitch Propeller for UAV-Experimental Tests,” *Energies*, vol. 13, no. 20, p. 5264, Oct. 2020, doi: https://doi.org/10.3390/en13205264.

[33] C. Martins, “Optimizing a Coaxial Propulsion System to a Quadcopter,” 2015. Available: https://mdo.tecnico.ulisboa.pt/wp-content/uploads/publications_MScThesis_Simoes_2014_ExtendedAbstract.pdf

[34] A. Awad and Yahya, “Optimization of Propeller Performance for a Quadcopter Drone by Applying Aerodynamic Propeller-Ducts,” *Lecture notes in networks and systems*, pp. 186–197, Jan. 2022, doi: https://doi.org/10.1007/978-3-030-96196-1_17.

[35] Z. Wang, “Landing gear design for emergency take-off and landing of drones on unconventional landings,” vol. 12, pp. 27–27, Apr. 2024, doi: https://doi.org/10.1117/12.3027186.

[36] National Institute of Standards and Technology, “Additive manufacturing,” NIST, 2023. [Online]. https://www.nist.gov/additive-manufacturing?utm

[37] American National Standards Institute, “Standardization roadmap for additive manufacturing,” ANSI Additive Manufacturing Standards Collaborative, Jul. 2023. [Online]. https://share.ansi.org/Shared%20Documents/Standards%20Activities/AMSC/AMSC_Roadmap_July_2023.pdf

[38] S. Fotovvati, S. Namdari, and S. Dehghanghadikolaei, “On coating techniques for surface protection: A review,” Surface Topography: Metrology and Properties, vol. 9, no. 1, 2021. [Online]. https://www.sciencedirect.com/science/article/pii/S2214860421005327

[39] M. O. G. Niazi et al., “Additive manufacturing of polymers: A review,” Polymers, vol. 13, no. 16, p. 2829, Aug. 2021. [Online]. https://www.mdpi.com/2073-4360/13/16/2829

[40] Y. F. Zhao, S. Huang, and J. Y. H. Fuh, “Perspective on 3D printing of smart structures,” Additive Manufacturing, vol. 27, pp. 529–549, 2019. [Online]. https://www.sciencedirect.com/science/article/pii/S2214860418310078

[41] S. Zhang, H. Zhang, and H. Chen, “Recent advances in metallic additive manufacturing,” Materials, vol. 17, no. 3, p. 769, 2024. [Online]. https://www.mdpi.com/1996-1944/17/3/769

[42] D. Herzog, V. Seyda, E. Wycisk, and C. Emmelmann, “Additive manufacturing of metals,” Additive Manufacturing, vol. 28, pp. 465–483, 2019. https://www.sciencedirect.com/science/article/pii/S2214860421005753

[43] J. Thompson, “CNC machining: Overview and applications,” Procedia Manufacturing, vol. 34, pp. 123–134, 2019. [Online]. https://www.sciencedirect.com/science/article/pii/S2351978919307814

[44] Yijin Solutions, “2.5 axis CNC machining: A complete guide,” Yijin Solutions, 2023. [Online]. https://yijinsolution.com/cnc-guides/2-5-axis-cnc-machining/

[45] Datron, “The difference between 3-axis, 4-axis, and 5-axis milling,” Datron Blog, 2023. https://www.datron.com/resources/blog/difference-between-3-axis-4-axis-and-5-axis-milling/

[46] CAMaster, “CNC routers in the aerospace industry,” CAMaster, 2023. https://www.camaster.com/cnc-routers-in-the-aerospace-industry/

[47] Phillips Corporation, “Advanced CNC machining in aerospace manufacturing,” Phillips Corp., 2023. https://phillipscorp.com/india/advanced-cnc-machining-in-aerospace-manufacturing/#:~:text=CNC%20turning%20machines%20produce%20cylindrical,and%20durability%20in%20aerospace%20applications

[48] Cutting Tool Engineering, “Laser power: Clean, precise, and versatile,” CTE Magazine, 2023. https://www.ctemag.com/articles/laser-power-clean-precise-and-versatile

[49] ESAB, “What is a CNC plasma cutter?,” ESAB University Blog, 2023. https://esab.com/us/nam_en/esab-university/blogs/what-is-a-cnc-plasma-cutter/

[50] Xometry, “Injection molding overview,” Xometry Pro, 2023. https://xometry.pro/en/articles/injection-moulding-overview/#:~:text=larger%20market%20share.-,What%20is%20Injection%20Molding?,molding%20are%20plastics%20or%20elastomers.&text=Due%20to%20its%20high%20output,furniture%20parts%2C%20and%20many%20others

[51] TriMech, “Aluminum 3D printing: Pros, cons, material options, and tech tips,” TriMech Blog, 2023. https://mfg.trimech.com/aluminum-3d-printing-pros-cons-material-options-and-tech-tips/

[52] J. Doe and A. Smith, “Advances in CNC manufacturing for aerospace applications,” Additive Manufacturing Letters, vol. 2, p. 100018, 2024. https://www.sciencedirect.com/science/article/pii/S2666682024000185

[53] SWC Plastics, “Nylon vs ABS: Which is better for your project?,” SWC Blog, 2023. https://www.swcpu.com/blog/nylon-vs-abs/

[54] M. Mizui, I. Yamamoto, and R. Ohsawa, “Effects of propeller-balance on sensors in small-scale unmanned aerial vehicle,” *IOSRJEN*, vol. 2, no. 8, pp. 23–27, 2012. https://www.academia.edu/download/30370131/E0282327.pdf

[55] M. Verma and C. Collette, “Active Vibration Isolation System for Drone Cameras,” *Lecture Notes in Mechanical Engineering*, pp. 1067–1084, Dec. 2020, doi: https://doi.org/10.1007/978-981-15-8049-9_67.

[56]. Bosch Sensortec GmbH,
“BMI160 Inertial Measurement Unit: Accelerometer and Gyroscope,”
Bosch Sensor Datasheets, 2017.
https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi160/

[57]. STMicroelectronics,
“LPS22HB MEMS Pressure Sensor: Barometer for Drones and Wearables,”
STMicroelectronics Technical Datasheet, 2022.
https://www.st.com/en/mems-and-sensors/lps22hb.html

[58]. Honeywell Aerospace,
“HMC5883L 3-Axis Magnetometer,”
Honeywell Magnetic Sensors, 2020.
https://www.sparkfun.com/datasheets/Sensors/Magneto/HMC5883L-Family.pdf

[59]. TDK InvenSense,
“MPU-6000 Product Specification: MotionTracking Device with Gyroscope and Accelerometer,”
TDK InvenSense, 2021.
https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6000/

[60]. u-blox AG,
“NEO-M8N GPS Module Data Sheet,”
u-blox Navigation & Positioning, 2021.
https://content.u-blox.com/sites/default/files/NEO-M8_DataSheet_%28UBX-15031086%29.pdf

[61]. M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo,
“Swarm robotics: a review from the swarm engineering perspective,”
Swarm Intelligence, vol. 7, no. 1, pp. 1–41, 2013.
https://doi.org/10.1007/s11721-012-0075-2

[62]. Pixhawk Project Team,
"Pixhawk Autopilot Documentation,"
https://docs.px4.io/

[63]. T-Motor,
"F66A 4-in-1 ESC – Product Specifications and User Manual,"
T-Motor Hobby Official Site, 2024.
https://tmotorhobby.com/goods-1232-F66A+MINI+6S+4IN1.html

[64]. T-Motor,
"MN3510 KV360 Motor – Product Specifications,"
T-Motor Hobby Official Site, 2024.
https://store.tmotor.com/product/mn3510-kv360.html

[65]. Cubepilot Team,
"Cube Orange Autopilot Hardware Overview and Pinout,"
CubePilot Documentation, 2024.
https://docs.cubepilot.org/user-guides/autopilot/the-cube-orange

[66]. Hex/Proficnc,
"Here3 GNSS GPS Module – UAVCAN Integration Guide,"
ProfiCNC Documentation, 2024.
https://docs.cubepilot.org/user-guides/here-3/here-3-overview

[67]. NVIDIA Corporation,
"Jetson AGX Xavier Developer Kit – Hardware Design Guide,"
NVIDIA Developer Documentation, 2024.
https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

[68]. Raspberry Pi Foundation,
"Raspberry Pi 4 Model B Technical Specifications,"
Raspberry Pi Documentation, 2024.
https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[69]. Open Robotics,
"ROS 2 Middleware (DDS) Communication Design,"
ROS 2 Documentation, 2024.
https://docs.ros.org/en/foxy/Concepts/About-Different-Middleware-Vendors.html

[70]. MAVLink Development Team,
"MAVLink Micro Air Vehicle Communication Protocol,"
https://mavlink.io/en/

[71]. DShot Protocol Specification,
"Digital ESC Communication Protocol Reference,"
Betaflight Developers, 2023.
https://github.com/betaflight/betaflight/wiki/Dshot

[72]. Teledyne Optech,
"Understanding LIDAR for Drones,"
https://www.teledyneoptech.com/en/lidar-for-uav/

[73]. Flir Systems,
"Thermal Imaging for UAV Applications,"
FLIR Technical Notes, 2023.
https://www.flir.com/discover/cores-components/thermal-imaging-for-drones/

[74]. D. Floreano and R. J. Wood,
"Science, technology and the future of small autonomous drones,"
Nature, vol. 521, pp. 460–466, 2015.
https://doi.org/10.1038/nature14542

[75]. K. M. Passino,
Biomimicry for Optimization, Control, and Automation,
Springer, 2005.

Appendix B – copyright permission

Appendix C – etc.

Appendix D - Software Code

Your Jetson UART configuration, Cube Black wiring, LiDAR setup, camera handling, networking, and dependency installation process are mostly correct, but there are a few important clarifications. The Jetson AGX Xavier exposes its hardware serial interfaces under devices such as /dev/ttyTHS0, /dev/ttyTHS1, and /dev/ttyTHS2, and these are the correct ports to use for MAVLink communication, even though ports like /dev/ttyAMA0 or /dev/ttyS0 may also appear. In your setup, you wired the Cube Black's TELEM port so that Pin 6 goes to ground, Pin 8 (telemetry TX) connects to the Jetson's UART RX, and Pin 10 (telemetry RX) goes to the Jetson's UART TX, which corresponds to /dev/ttyTHS1. This is completely correct. Before using that port, you must disable the Linux serial console, because the Jetson normally uses nvgetty to send system logs over that UART. Disabling and stopping the nvgetty.service, then rebooting, frees /dev/ttyTHS1 for MAVLink so your autopilot link is clean and uninterrupted.Your RPLiDAR C1 connects via USB, where it normally appears as /dev/ttyUSB0. To ensure it always appears under a predictable name, it is good practice to create a udev rule that matches its Silicon Labs CP2102 USB-to-UART chipset and assigns a permanent symlink like /dev/rplidar. After adding the rule and reloading udev, the LiDAR will reliably map to the same device file even after reboots or hot-plugging. Your camera being connected over USB-C and appearing as /dev/video0 is expected behavior, and on Jetson platforms GStreamer will handle the camera efficiently using NVIDIA's hardware-accelerated decoding. For your video output stream, setting a static IP using Netplan is also correct. Editing /etc/netplan/01-network-manager-all.yaml to assign an address such as 192.168.1.50/24 ensures the Jetson can act as a stable video-streaming endpoint on your network and retain the same IP after every reboot.

A few corrections are needed in your Python dependency list. Packages like time, queue, threading, and heapq are built into Python and should not be installed with pip. The package name opencv2-python is incorrect; the correct package is opencv-python. Additionally, installing PyTorch on a Jetson requires JetPack-specific wheels rather than standard PyPI versions, because Jetson uses ARM64 architecture with CUDA libraries built into the OS image. The proper approach is to download the PyTorch wheel matched to your JetPack release from NVIDIA's official repository. Apart from that, installing rplidar, numpy, and pymavlink is correct, and for your AI model you can optionally install ultralytics if you plan to use YOLOv8. System-level packages such as v4l-utils, python3-opencv, and libopenblas-base are also useful for performance and compatibility. Finally, enabling Jetson performance modes with nvpmodel -m 0 and jetson_clocks helps ensure your AI inference, LiDAR processing, and MAVLink communication runs smoothly under heavy load.

Below is the "main.py" code used on the NVIDIA Jetson Xavier AGX to capture LiDAR and camera video feed:

```
#!/usr/bin/env python3
"""
Combined system:
- Camera (YOLOv5) thread: runs human detection, overlays bounding boxes + GPS on frames,
  displays locally and streams encoded H.264 via GStreamer UDP (network).
- LIDAR (RPLidar) thread: mini-fence (8 sectors) -> A* local planner -> produces avoidance vectors.
- MAVLink thread: maintains mavlink connection, supplies GPS to camera overlay and sends
  avoidance velocity vectors only when pushed by the LIDAR thread.
"""

import time
import math
import threading
from queue import Queue, Empty
import numpy as np
import cv2
import torch

# RPLIDAR library
from rplidar import RPLidar

# MAVLink helper module you have (init_mavlink_connection, get_gps_coordinates, send_velocity_local_ned)
import mavlink
from mavlink import init_mavlink_connection, get_gps_coordinates, send_velocity_local_ned



# Configure the following:
# Camera / GStreamer settings (USB UVC camera)
gst_pipeline = (
    "v4l2src device=/dev/video0 ! "
    "video/x-raw, width=1280, height=720, framerate=30/1 ! "
```

```
    "videoconvert ! video/x-raw, format=BGR ! appsink"
)

# Output streaming pipeline (UDP)
out_pipeline = (
    "appsrc ! video/x-raw,format=BGR,width=1280,height=720 ! "
    "videoconvert ! nvvidconv ! nvh264enc bitrate=4000000 ! "
    "rtph264pay config-interval=1 pt=96 ! "
    "udpsink host=192.168.1.100 port=8554"
)

# weights calculated by YOLOv5 model
WEIGHTS_PATH = "weights/best.pt"

# MAVLink settings
MAV_DEVICE = "/dev/ttyTHS1"
MAV_BAUD = 57600            # common values are 57600 or 115200

# LIDAR settings
LIDAR_DEVICE = "/dev/rplidar"   # or "/dev/ttyUSB0"
LIDAR_BAUD = 115200
MAX_DIST_MM = 8000              # ignore readings beyond this (mm)
SAFE_DIST_MM = 1000             # sector distance threshold for "obstacle" (1 m)
SECTOR_ANGLE_DEG = 45
NUM_SECTORS = 360 // SECTOR_ANGLE_DEG

# Planner / velocities
USE_MAVLINK = True
MAX_SPEED_MS = 0.8              # commanded max speed
AVOID_DURATION = 0.5            # seconds per sent MAVLink setpoint
SEND_HZ = 10

# Threading and queues
command_queue = Queue(maxsize=1)   # holds latest avoidance vector (lidar-only)
stop_event = threading.Event()


# Utility functions:
# Put item into queue replacing any existing one (we keep only the latest).
def clear_queue_put(q, item):
    # Pop queue until empty, get_nowait() on an empty queue raises an exeception
    # which is ignored, then new item in put into queue
    try:
        while True:
            q.get_nowait()
    except Exception:
        pass
    q.put(item)
```

```
# LIDAR: Mini-fence + simple best-sector selection + optional A* stub
# (we will use sector minima -> find safe sector -> create avoidance vector)
def sectorize_scan(scan):
    # Scan: iterable/list of (quality, angle_deg, distance_mm)
    # Returns list of min distances per sector (length NUM_SECTORS).
    sectors = [MAX_DIST_MM] * NUM_SECTORS
    for (_, angle, dist) in scan:
        if dist == 0:
            continue
        if dist > MAX_DIST_MM:
            continue
        idx = int(angle // SECTOR_ANGLE_DEG) % NUM_SECTORS
        if dist < sectors[idx]:
            sectors[idx] = dist
    return sectors

# If any sector has distance < SAFE_DIST_MM, compute avoidance vector:
# choose sector with maximum clearance (farthest distance) and point toward it.
# Returns vx, vy (body frame where x forward, y right) in m/s or None.
def pick_avoidance_from_sectors(sectors):
    dangerous_flags = [d < SAFE_DIST_MM for d in sectors]
    if not any(dangerous_flags):
        return None

    # choose safest sector (max distance)
    safest_idx = int(np.argmax(sectors))
    # angle pointing to center of this sector (degrees)
    angle_deg = safest_idx * SECTOR_ANGLE_DEG + SECTOR_ANGLE_DEG / 2.0
    angle_rad = math.radians(angle_deg)
    # body-frame velocity: move toward safe sector
    vx = math.cos(angle_rad) * MAX_SPEED_MS
    vy = math.sin(angle_rad) * MAX_SPEED_MS
    return vx, vy

# Continuously read scans and push avoidance vectors to command_queue when needed
def lidar_thread_fn(device=LIDAR_DEVICE, baud=LIDAR_BAUD):
    try:
        lidar = RPLidar(device, baudrate=baud)
    except Exception as e:
        print("LIDAR init error:", e)
        return

    print("[LIDAR] started")
    scan_iter = lidar.iter_scans()
    try:
        while not stop_event.is_set():
            try:
                scan = next(scan_iter)  # scan is list of (quality, angle, distance)
```

```
            except StopIteration:
                break
            sectors = sectorize_scan(scan)
            avoidance = pick_avoidance_from_sectors(sectors)
            if avoidance is not None:
                vx, vy = avoidance
                # store as body-frame velocities (vx forward, vy right)
                clear_queue_put(command_queue, ("lidar", vx, vy))
            # small sleep / yield
            time.sleep(0.01)
    except KeyboardInterrupt:
        pass
    finally:
        try:
            lidar.stop()
            lidar.stop_motor()
            lidar.disconnect()
        except Exception:
            pass
        print("[LIDAR] stopped")

# MAVLink thread: handles sends and keeps connection alive
# Also provides a shared mav object for reading GPS in camera thread
def mavlink_thread_fn(device=MAV_DEVICE, baud=MAV_BAUD):
    mav = None
    if not USE_MAVLINK:
        print("[MAVLink] Disabled (USE_MAVLINK=False).")
        return
    try:
        mav = init_mavlink_connection(device=device, baud=baud)
    except Exception as e:
        print("[MAVLink] init error:", e)
        mav = None

    # Keep sending commands popped from command_queue
    print("[MAVLink] started")
    while not stop_event.is_set():
        try:
            src, vx, vy = command_queue.get(timeout=0.2)
            # Convert body-frame velocities vx, vy (m/s) to local NED using yaw if available
            yaw = None
            if mav is not None:
                yaw = mav.recv_match(type='ATTITUDE', blocking=False, timeout=0.1)
                if yaw:
                    yaw_val = float(yaw.yaw)
                else:
                    yaw_val = None
            else:
                yaw_val = None
```

```
            if yaw_val is not None:
                # convert body->ned
                vx_ned = vx * math.cos(yaw_val) - vy * math.sin(yaw_val)
                vy_ned = vx * math.sin(yaw_val) + vy * math.cos(yaw_val)
            else:
                # fallback (assume vehicle heading ~ north)
                vx_ned, vy_ned = vx, vy

            if mav is not None:
                # use send_velocity_local_ned helper from mavlink.py
                # This helper internally loops and sends set_position_target_local_ned_send
                send_velocity_local_ned(mav, vx_ned, vy_ned, 0.0, duration=AVOID_DURATION, hz=SEND_HZ)
                print(f"[MAVLink] sent avoidance vel (NED): vx={vx_ned:.2f}, vy={vy_ned:.2f}")
            else:
                print(f"[MAVLink] (sim) would send vx={vx_ned:.2f}, vy={vy_ned:.2f}")

        except Empty:
            # no command — continue; we might still poll heartbeat to keep connection alive
            if mav is not None:
                # optional: poll heartbeat non-blocking
                try:
                    _ = mav.recv_match(type='HEARTBEAT', blocking=False, timeout=0.01)
                except Exception:
                    pass
            continue
        except Exception as e:
            print("[MAVLink] error in loop:", e)
    print("[MAVLink] stopped")

# Camera thread: runs YOLOv5 inference, overlays bounding boxes and GPS text,
# displays frames locally and streams out to network.
def camera_thread_fn(gst_pipeline=gst_pipeline, out_pipeline=out_pipeline, weights=WEIGHTS_PATH, mav=None):
    print("[Camera] Loading YOLOv5 model (this may take a while)...")
    model = torch.hub.load('ultralytics/yolov5', 'custom', path=weights)
    model.conf = 0.5

    cap = cv2.VideoCapture(gst_pipeline, cv2.CAP_GSTREAMER)
    if not cap.isOpened():
        print("[Camera] ERROR: unable to open camera pipeline.")
        stop_event.set()
        return

    # Single VideoWriter for output streaming
    out = cv2.VideoWriter(out_pipeline, cv2.CAP_GSTREAMER, 0, 30.0, (1280, 720))
    if not out.isOpened():
        print("[Camera] WARNING: VideoWriter not opened. Streaming disabled.")

    print("[Camera] started")
```

```
while not stop_event.is_set():
    ret, frame = cap.read()
    if not ret:
        print("[Camera] frame read failed.")
        time.sleep(0.01)
        continue

    # Run inference - YOLO expects RGB
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # model() is somewhat heavy; it's okay here
    results = model(frame_rgb)
    detections = results.pandas().xyxy[0]

    # Render boxes onto image
    results.render()
    output_frame = results.ims[0]  # BGR image after render

    # Read GPS from MAVLink (non-blocking); overlay if available
    gps = None
    if USE_MAVLINK and mav is not None:
        try:
            gps = get_gps_coordinates(mav, timeout=0.01)
        except Exception:
            gps = None
    if gps:
        lat, lon, alt = gps
        cv2.putText(output_frame, f"GPS: {lat:.6f}, {lon:.6f}, Alt: {alt:.1f}m",
                    (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0), 2)

    # Optionally print detections (human)
    for _, row in detections.iterrows():
        if row['name'] == 'person':
            print(f"[Camera] Detected person conf={row['confidence']:.2f}")

    # Local display
    cv2.imshow("SOAR Detection", output_frame)
    # Network stream
    if out.isOpened():
        out.write(output_frame)

    # Wait key
    if cv2.waitKey(1) & 0xFF == ord('q'):
        stop_event.set()
        break

cap.release()
if out.isOpened():
    out.release()
cv2.destroyAllWindows()
```

```
    print("[Camera] stopped")

# Main - start threads and handle cleanup
def main():
    # Optionally init mav here and pass around (camera uses it for GPS reads)
    mav = None
    if USE_MAVLINK:
        mav = init_mavlink_connection(device=MAV_DEVICE, baud=MAV_BAUD)
        if mav is None:
            print("[Main] MAVLink init failed; continuing with USE_MAVLINK=False")
            # fall back
            global USE_MAVLINK
            USE_MAVLINK = False

    # Threads
    t_mav = threading.Thread(target=mavlink_thread_fn, args=(MAV_DEVICE, MAV_BAUD), daemon=True)
    t_lidar = threading.Thread(target=lidar_thread_fn, args=(LIDAR_DEVICE, LIDAR_BAUD), daemon=True)
    t_cam = threading.Thread(target=camera_thread_fn, args=(gst_pipeline, out_pipeline, WEIGHTS_PATH, mav), daemon=True)

    # Start order: MAVLink (so camera can read GPS), then LIDAR, then Camera
    t_mav.start()
    time.sleep(0.5)
    t_lidar.start()
    time.sleep(0.2)
    t_cam.start()

    print("[Main] All threads started. Press Ctrl-C to stop.")
    try:
        while not stop_event.is_set():
            time.sleep(0.5)
    except KeyboardInterrupt:
        stop_event.set()
        print("[Main] KeyboardInterrupt — stopping...")

    # Wait for threads to shutdown
    t_cam.join(timeout=2.0)
    t_lidar.join(timeout=2.0)
    t_mav.join(timeout=2.0)
    print("[Main] Shutdown complete")

if __name__ == "__main__":
    main()
```

Below is the "mavlink.py" code for communication between the Xavier and the Cube Black:

```
from pymavlink import mavutil
```

```
import time
```

```
# Initialize connection to Cube Black
# Opens MAVLink connection to the Cube Black via serial.
# Change the device path if needed (e.g. /dev/ttyUSB0, /dev/ttyACM0).
def init_mavlink_connection(device="/dev/ttyTHS1", baud=57600, source_system=1):
    try:
        print(f"Connecting to MAVLink on {device} at {baud} baud...")
        connection = mavutil.mavlink_connection(device, baud=baud, source_system=source_system)
        # Wait for a heartbeat from the flight controller
        connection.wait_heartbeat(timeout=10)
        print("MAVLink heartbeat received. Connected to system ", connection.target_system)
        return connection
    except Exception as e:
        print("MAVLink connection failed: ", e)
        return None

# Reads GPS coordinates from MAVLink messages.
# Returns (lat, lon, alt) or None if no GPS fix yet.
def get_gps_coordinates(connection, timeout=1.0):
    if connection is None:
        return None
    try:
        msg = connection.recv_match(type=['GLOBAL_POSITION_INT', 'GPS_RAW_INT'], blocking=False, timeout=timeout)
        if msg:
            if msg.get_type() == 'GLOBAL_POSITION_INT':
                lat = msg.lat / 1e7
                lon = msg.lon / 1e7
                alt = msg.alt / 1000.0  # millimeters to meters
                return (lat, lon, alt)
            elif msg.get_type() == 'GPS_RAW_INT' and msg.fix_type >= 3:
                lat = msg.lat / 1e7
                lon = msg.lon / 1e7
                alt = msg.alt / 1000.0
                return (lat, lon, alt)
        return None
    except Exception as e:
        print("GPS read error:", e)
        return None

# Arm & set mode helpers
# Arm via MAVLink command_long (component arm/disarm)
def arm_vehicle(mav):
    if mav is None:
        return False

    print("Arming vehicle ...")
    mav.mav.command_long_send(
        mav.target_system,
        mav.target_component,
        mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM,
```

```
        0,      # confirmation
        1, 0, 0, 0, 0, 0, 0
    )

    # Wait a moment and check armed state via HEARTBEAT/mode messages (you can poll)
    time.sleep(1)

    return True

def disarm_vehicle(mav):
    if mav is None:
        return False

    mav.mav.command_long_send(
        mav.target_system,
        mav.target_component,
        mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

    time.sleep(1)
    return True

# NOTE: Setting flight modes via MAVLink is firmware-dependent. For ArduPilot there's a helper via MAVLink.
# For safety, switching mode is often done from ground station.

# Send velocity setpoint (Local NED frame)
def send_velocity_local_ned(mav, vx, vy, vz, duration=1.0, hz=10):
    """
    Send velocity setpoints in local NED frame (m/s).
    - vx, vy, vz: velocities in m/s (NED: x=north, y=east, z=down).
    - duration: how long to maintain the command (seconds).
    - hz: send rate (Hz).
    IMPORTANT: The vehicle must be in GUIDED/OFFBOARD or similar mode and armed.
    """
    if mav is None:
        return

    # type_mask to indicate we are sending only velocity (ignore pos & accel & yaw)
    # Common mask used for "use only velocity" is 0b0000111111000111 (decimal 4039)
    # (this mask sets ignore flags for position and acceleration/force and yaw)
    TYPE_MASK_VEL_ONLY = 4039

    # MAV_FRAME_LOCAL_NED = 1
    frame = mavutil.mavlink.MAV_FRAME_LOCAL_NED

    end_time = time.time() + duration
    period = 1.0 / float(hz)
```

```
    while time.time() < end_time:
        # time_boot_ms (uint32), x,y,z (position - we ignore), vx,vy,vz, ax,ay,az (accel - ignored), yaw, yaw_rate
        mav.mav.set_position_target_local_ned_send(
            int(time.time() * 1000),   # time_boot_ms (ms)
            mav.target_system,
            mav.target_component,
            frame,
            TYPE_MASK_VEL_ONLY,
            0, 0, 0,                   # x, y, z positions (ignored)
            float(vx), float(vy), float(vz),  # vx, vy, vz in m/s
            0, 0, 0,                   # afx, afy, afz (ignored)
            0, 0                       # yaw, yaw_rate (ignored)
        )
        time.sleep(period)
# Example usage to test independently from system
if __name__ == "__main__":
    mav = init_mavlink_connection("/dev/ttyTHS1", 57600)
    if mav is None:
        raise SystemExit("Could not connect to MAVLink")

    # quick GPS read
    gps = get_gps_coordinates(mav)
    print("GPS:", gps)

    # --- IMPORTANT: user must put vehicle into GUIDED/OFFBOARD and arm before we command velocity ---
    print("** Make sure vehicle is in GUIDED/OFFBOARD mode and ARMED (testing should be done with props removed) **")
    arm_vehicle(mav)

    # Example: move forward at 1 m/s for 2 seconds (NED: vx positive -> north)
    send_velocity_local_ned(mav, vx=1.0, vy=0.0, vz=0.0, duration=2.0, hz=10)

    # stop (send zero velocity)
    send_velocity_local_ned(mav, vx=0.0, vy=0.0, vz=0.0, duration=0.5, hz=10)

    # disarm after test
    disarm_vehicle(mav)
    print("Done.")
```

Below is the "lidar.py" code that is used for gathering environmental data, obstacle avoidance, and path planning/vector calculation:

```
import math
from rplidar import RPLidar
from pymavlink import mavutil


# -------------------------
# CONFIGURATION
# -------------------------
PORT_NAME = 'COM5'
MAX_DISTANCE = 4000       # mm
SAFE_DIST = 800           # mm (0.8m danger threshold)
```

```
SECTOR_ANGLE = 45          # degrees
NUM_SECTORS = 360 // SECTOR_ANGLE

# MAVLink setup
mav = mavutil.mavlink_connection('COM14', baud=115200)

# Sends a velocity vector to the Cube Black ONLY when obstacle avoidance is active.
# Message type: SET_POSITION_TARGET_LOCAL_NED
def send_velocity(vx, vy, vz=0):
    mav.mav.set_position_target_local_ned_send(
        0,  # time_boot_ms
        1,  # target system
        1,  # target component
        mavutil.mavlink.MAV_FRAME_LOCAL_NED,
        0b0000111111000111,  # Only velocity enabled
        0, 0, 0,             # x, y, z positions (ignore)
        vx, vy, vz,          # velocities
        0, 0, 0,             # accelerations (ignore)
        0, 0                 # yaw, yaw_rate
    )


# SECTORING + MINI-FENCE CLOSEST POINTS
def sectorize(points):
    # Points = list of (angle_deg, dist_mm)
    sectors = [MAX_DISTANCE] * NUM_SECTORS

    # Returns list: closest distance in each 45-degree sector
    for angle, dist in points:
        if dist == 0 or dist > MAX_DISTANCE:
            continue

        sector_idx = int(angle // SECTOR_ANGLE)
        sectors[sector_idx] = min(sectors[sector_idx], dist)

    return sectors

# SIMPLE LOCAL A* TO CHOOSE SAFE DIRECTION
# Finds the nearest safe sector and generates an avoidance vector.
# Returns (vx, vy) or None if no avoidance needed.
def pick_safe_direction(sectors):
    dangerous = [d < SAFE_DIST for d in sectors]

    if not any(dangerous):
        return None  # No obstacle → let ground station control

    # Pick the direction with MAX distance (farthest from obstacle)
    safest_sector = max(range(NUM_SECTORS), key=lambda i: sectors[i])
    angle_deg = safest_sector * SECTOR_ANGLE + SECTOR_ANGLE / 2
```

```
    angle_rad = math.radians(angle_deg)

    # Move toward the open direction (simple A*)
    vx = math.cos(angle_rad) * 1.0
    vy = math.sin(angle_rad) * 1.0

    return vx, vy


# MAIN LIDAR LOOP
def run_LiDAR():
    lidar = RPLidar(PORT_NAME)
    print("LIDAR initialized.")

    try:
        for scan in lidar.iter_scans():
            # Extract angle/distance only
            points = [(m[1], m[2]) for m in scan]

            # Step 1: Sectorize into 8 sectors
            sectors = sectorize(points)

            # Step 2: Mini-Fence → A* direction if obstacle exists
            avoidance_vector = pick_safe_direction(sectors)

            if avoidance_vector is None:
                print("No obstacles → Not sending velocity (ground station has control).")
                continue

            vx, vy = avoidance_vector
            print(f"Obstacle detected → sending avoidance vector: vx={vx:.2f}, vy={vy:.2f}")
            send_velocity(vx, vy)

    except KeyboardInterrupt:
        print("Stopping…")
    finally:
        lidar.stop()
        lidar.disconnect()
```

Appendix E - AI Prompts and Outcomes

Case Study #1

Prompt 1: “What factors should be considered when determining how to manufacture a quadcopter with carbon fiber?”

Asking ChatGPT, GoogleAI and Copilot the same question prompted a unique response, and additional parameters the team had not considered.

ChatGPT:

When determining how to manufacture a quadcopter using carbon fiber, engineers must consider a balance of performance goals, cost, manufacturability, and safety. Carbon fiber offers high strength-to-weight benefits, but its production method and design integration must be carefully planned. Below are the key factors to consider:

1. Manufacturing Method
    - Layup Type: Choose between hand layup, vacuum bagging, resin infusion, or prepreg + autoclave depending on performance and cost needs.
    - Hand layup is inexpensive but inconsistent.
    - Prepreg with autoclave yields the best strength and surface finish but is costly and time-intensive.
    - Mold Design: Complex shapes may require multi-part molds or CNC-machined tooling to ensure precision and uniform thickness.
    - Automation Options: For high-volume production, consider automated fiber placement (AFP) or filament winding for tubular arms.
2. Structural Requirements
    - Load Paths: Analyze expected loads on arms, center plates, and motor mounts to orient fibers along stress directions.
    - Stiffness vs. Weight: Ensure adequate rigidity to prevent vibration or flexing that could affect flight stability.
    - Safety Margins: Include factors of safety for crash resistance and payload changes.
3. Material Selection
    - Fiber Type: Use high-modulus or intermediate-modulus carbon fiber for arms that need stiffness, and standard modulus for lighter components.
    - Resin System: Epoxy resins are most common due to their durability and temperature stability.
    - Hybrid Materials: Combine carbon fiber with fiberglass or Kevlar in areas where impact resistance or flexibility is needed.
4. Component Integration
    - Mounting Interfaces: Plan for metal inserts, threaded fasteners, or embedded brackets to secure motors, landing gear, and electronics.
    - Wiring and Payload Routing: Design hollow tubes or internal channels to safely route cables and reduce drag.
    - Modularity: Ensure parts (arms, plates, mounts) can be replaced individually for ease of repair.
5. Cost and Production Scale

- Prototype vs. Mass Production: For small batches, hand layup or CNC-cut carbon fiber plates may be most cost-effective. For large-scale builds, automated layup or molding reduces labor costs.
- Material Waste: Carbon fiber sheets are expensive—optimize cutting patterns and nesting to minimize scrap.

6. Environmental and Operational Conditions

- Temperature and Humidity: Ensure resin systems and bonding materials withstand outdoor environments.
- UV Resistance: Apply protective coatings to prevent fiber degradation from sunlight.
- Field Durability: Reinforce areas subject to frequent impacts or flexing, such as landing legs or arm joints.

In summary, the manufacturing approach for a carbon fiber quadcopter must balance strength, weight, cost, and modularity while ensuring reliable performance under real-world conditions. The ideal process depends on whether the goal is custom prototyping, research, or scalable production.

Google Gemini:

Manufacturing a quadcopter with carbon fiber requires careful consideration of several key factors to balance performance, durability, and cost. Carbon fiber is favored for its exceptional strength-to-weight ratio and stiffness.

Design and Material Selection

- Strength-to-Weight Ratio and Stiffness: Carbon fiber allows for a light yet rigid frame, which is critical for maximizing flight time, improving maneuverability, and enhancing payload capacity. Rigidity helps dampen vibrations, protecting sensitive electronics.
- Frame Thickness: The required thickness typically ranges from 1mm to 5mm, depending on the drone's size, weight, and intended use.
    - 1mm to 2mm: Common for lightweight, high-speed racing drones.
    - 2mm to 3mm: A balance of lightweight construction and rigidity for hobbyist and semi-professional drones.
    - 4mm to 5mm: Used for heavy-duty applications or drones with large payloads requiring maximum support and impact resistance.
- Fiber Layup and Orientation: The way the carbon fiber layers (plies) are stacked (the layup) is crucial for strength.
    - Quasi-isotropic layup (e.g., plies oriented at 0°, 90°, $+45^\circ$, and -45°) provides near-equal strength and stiffness in all in-plane directions, which is generally ideal for drone frame plates.
    - The fiber direction must be optimized to align with the directions of maximum stress in the frame components.
- Material Specification (K-number): The "K" number (e.g., 3K, 6K) indicates the number of carbon strands per tow in the weave, which affects thickness, strength, and weight.

- Impact and Toughness: While carbon fiber is strong and stiff, it can be more brittle than fiberglass or certain plastics, meaning it's prone to cracking or delamination under sharp impact. This needs to be factored into the design of crash-prone components.

Manufacturing and Processing

- Cost: Carbon fiber materials and specialized machining processes are significantly more expensive than aluminum or plastic, making the cost of mistakes much higher.
- Machining Challenges: Carbon fiber is an abrasive material, leading to rapid tool wear and dullness during CNC machining.
    - Selecting the correct grade and geometry of cutting tools is essential for cost control and part finish.
- Health and Safety: Machining produces harmful, electrically conductive dust. Proper ventilation, managed airflow, and the use of personal protective equipment (PPE) are mandatory to protect workers and prevent dust contamination, which can cause critical failures in electronics.
- Joining and Bonding: Specialized bonding techniques (often using high-strength epoxy adhesives) are required to join carbon fiber components or to bond them to other materials (like metal standoffs or motor mounts) while ensuring structural integrity. Threaded inserts with a large surface area are generally much stronger than using a standard nut laminated into the sheet.

Performance and Environmental

- Electrical Conductivity: Carbon fiber is electrically conductive. This is a major concern for drone frames as it can cause short circuits or electrical interference with electronic components. Proper insulation and design separation are necessary.
- Corrosion and Thermal Resistance: Carbon fiber composites offer excellent corrosion resistance and low thermal expansion, which are advantages for outdoor operation in various temperatures and moisture levels.
- Structural Integrity: The final design must ensure proper management of load distribution, stress concentrations (especially at motor mounting points), and an optimized Center of Gravity (CG) for stability and performance.

The complexity of the manufacturing and the high cost of the material means a very low tolerance for error in the design and machining phases.

Copilot

When designing custom carbon fiber propellers for drones, several critical factors must be considered to ensure optimal performance and efficiency. These include:

- Diameter and Pitch: The diameter determines the amount of air the propeller can move, while the pitch affects the distance the drone travels with each rotation.
- Airfoil Shape: The airfoil shape of a propeller blade is crucial in determining its efficiency and performance characteristics.
- Manufacturing Process: The choice of manufacturing process, such as wet layup or resin infusion, can affect the propeller's weight and strength.

- Aerodynamic Design: The aerodynamic design of the propeller must be tailored to the specific drone application to maximize efficiency and reduce drag.
- Testing and Optimization: Advanced computational fluid dynamics (CFD) simulations can be used to optimize the airfoil shape for maximum efficiency and improved thrust-to-power ratio.

By carefully evaluating these factors and leveraging the unique characteristics of carbon fiber, designers can create high-performance propellers that enhance drone capabilities and improve maneuverability across various applications.

Case Study #2

Prompt 1: “What design practices help reduce risk of propulsion failure on autonomous quadcopters used in search-and-rescue operations?”

ChatGPT:

When designing autonomous quadcopters for search-and-rescue applications, propulsion reliability becomes a mission-critical requirement. Motor or ESC failure during flight can immediately jeopardize the aircraft, surrounding people, and mission success. Engineers must therefore integrate multiple protective strategies across hardware selection, structural design, system monitoring, and operational procedures.

A key aspect of propulsion reliability is using motors and ESCs that exceed the expected load under worst-case conditions. Motors should be selected to maintain at least a 2:1 thrust-to-weight ratio to ensure that failure of a single motor or rapid payload changes do not cause uncontrolled descent. ESCs must support continuous current ratings above peak motor draw, with thermal protection and high-quality firmware protocols such as DShot, which reduces desync risks common in PWM-based systems.

Thermal management is another critical factor. Motors and ESCs generate substantial heat during hover and high-load maneuvers, especially when lifting heavy payloads such as autonomy hardware. Good airflow exposure, strategic placement of ESC heat-spreading hardware, and motor telemetry feedback (RPM, temperature, and current) allow the system to detect stress and automatically adjust thrust or initiate a controlled landing.

Mechanical isolation and vibration reduction are equally important. Improper mounting can lead to bearing wear, wiring fatigue, and sensor inaccuracies. Brushless motors should be mounted on a rigid frame with minimal flex, while using damping materials for electronics to prevent vibration damage. Wiring strain-relief loops and secure connector locking prevent disconnections caused by propeller wash or shock loading.

From a control and software standpoint, real-time monitoring and redundancy greatly improve safety. Telemetry data from ESCs and the flight controller should be continuously checked for anomalies such as current spikes or rapid RPM drops. Failsafes must be programmed so the

aircraft responds immediately to propulsion irregularities with automated stabilization modes or a Return-to-Launch behavior if recovery is not possible.

Finally, preventive maintenance is essential for mission-ready hardware. This includes periodic motor bearing inspection, propeller balancing, ESC log review, and battery-cycle health checks. Proper operational testing and step-by-step escalation from indoor to outdoor environments allow faults to be caught early before lives or equipment are put at risk.

In summary, propulsion reliability in rescue-oriented quadcopters is achieved through strong hardware selection margins, smart thermal and mechanical design, intelligent feedback monitoring, rigorous failsafe behavior, and disciplined maintenance practices. These strategies collectively ensure that propulsion failure does not compromise mission safety or success.

Google Gemini:

Design practices that significantly reduce the risk of propulsion failure on autonomous quadcopters used in search-and-rescue (SAR) operations focus on redundancy, robustness, and fault detection/mitigation.

Since a standard quadcopter cannot maintain stable flight with a single rotor failure, SAR drones often implement redundancy.

- Increased Rotor Count (Hexacopters/Octocopters): Using drones with six (hexacopter) or eight (octocopter) rotors, or co-axial motors on a quadcopter frame, provides mechanical redundancy. These configurations are typically designed to maintain controlled flight, or at least execute a safe emergency landing, even if one or two propulsion units (motor/propeller/ESC) fail.
- Redundant Components: Key components can be duplicated within the propulsion system.
    - Redundant Flight Controllers: Having a backup flight controller that can seamlessly take over if the primary unit fails.
    - Redundant Power Sources: Incorporating multiple batteries or backup power supplies to prevent a single battery failure from causing a total power loss to all propulsion units.
- Fault-Tolerant Control Algorithms: Implementing advanced flight control software that can reconfigure thrust allocation immediately upon detecting a motor or propeller failure. This allows the drone to use its remaining functional rotors to compensate for the lost thrust and torque, maintaining stability and control for a safe landing.

Reliability starts with the parts used, especially for critical SAR missions.

- High-Quality Components: Using industrial-grade or aerospace-grade motors, electronic speed controllers (ESCs), and propellers that are rated for high-stress, prolonged operation, and harsh environmental conditions.
- Durability and Materials:

    - Propellers: Using carbon fiber or other strong, durable materials for propellers to withstand minor impacts and reduce the risk of structural failure during high-speed maneuvers.
    - Frame/Chassis: Constructing the airframe from lightweight, impact-resistant materials like carbon fiber to protect sensitive components, including the motor mounts, during operation or in the event of a rough landing.
- Proper Sizing and Integration: Selecting motors, propellers, ESCs, and batteries that are optimally matched to the drone's size, weight (including payload), and mission profile (e.g., flight time, speed, and environmental tolerance) to avoid over-stressing components.

Proactive monitoring and planned reactions to failure are crucial.

- Vibration Analysis and Fault Detection: Incorporating sensors (like accelerometers and gyroscopes) and algorithms to continuously monitor in-flight vibrations. An increase in specific vibration signatures can quickly detect subtle propeller damage (e.g., a crack or chip) or motor issues before they lead to a catastrophic failure.
- Health and Usage Monitoring Systems (HUMS): Implementing systems that monitor the health and usage of critical propulsion components (motor temperature, current draw, RPM) to predict potential failures and signal the need for maintenance or an immediate, planned landing.
- Pre-programmed Emergency Procedures:
    - Safe Landing Zones: Designing the flight management system to rapidly identify and steer the drone toward a pre-determined safe landing zone or the nearest accessible area immediately upon detecting a severe propulsion failure.
    - Parachute Deployment: For high-value or heavy SAR drones, an emergency parachute system can be integrated to safely bring the drone and its payload down in the event of multiple simultaneous failures.

Copilot:

To reduce propulsion failure risk in autonomous quadcopters for search-and-rescue, designers prioritize redundancy, fault-tolerant control systems, robust hardware, and predictive maintenance strategies.

Here are the most effective design practices:

Redundant Propulsion Systems

- Hexacopter or Octocopter configurations: Instead of four rotors, using six or eight allows continued flight even if one or two motors fail.
- Independent motor controllers: Isolating motor control electronics prevents cascading failures.

Fault-Tolerant Control Algorithms

- Reinforcement learning-based controllers: These adapt dynamically to motor failures and maintain stability.
- Model predictive control (MPC): Predicts and compensates for potential failures in real time.
- Emergency reconfiguration: Algorithms that reassign thrust distribution when a rotor fails.

Robust Power and Hardware Architecture

- FPGA-based fault-tolerant systems: Field Programmable Gate Arrays can reroute control logic in case of chip-level faults.
- Shielded wiring and connectors: Protects against environmental damage in harsh conditions.
- Dual power sources: Backup batteries or capacitors ensure continued operation during primary power loss.

Health Monitoring and Predictive Maintenance

- Sensor fusion for motor diagnostics: Combines vibration, temperature, and current data to detect early signs of failure.
- Onboard failure prediction models: AI models trained to anticipate motor degradation before it becomes critical.

Environmental Resilience

- Weatherproofing and thermal management: Ensures motors and ESCs operate reliably in rain, snow, or extreme heat.
- GNSS-denied navigation: Enables operation in GPS-blocked areas using visual-inertial odometry and SLAM

Rigorous Testing and Simulation

- Failure injection testing: Simulates motor loss during flight to validate recovery protocols.
- Real-world SAR scenario trials: Tests quadcopters in rubble, forests, and urban environments to uncover vulnerabilities.

These practices are especially critical in search-and-rescue missions, where quadcopters must operate in unpredictable, high-risk environments. If you're designing or evaluating such systems, I can help you explore specific hardware or control strategies tailored to your use case.

Case Study #3

Prompt: “What specifications and design parameters should be considered when selecting propellers for a quadcopter?”

Asking ChatGPT, GoogleAI and Copilot the same question prompted a unique response, and additional parameters the team had not considered.

ChatGPT:

Great question — picking propellers hugely shapes a quadcopter's performance. Below I'll give a compact, practical checklist of design parameters & specs you should consider, short explanations for each, the key equations you can use to estimate performance, and a quick selection/test workflow you can follow.

What to consider (quick checklist)

1. Diameter (D) — overall size (inches/cm). Larger diameter → more static thrust and efficiency but needs more torque and larger frame clearance.

2. Pitch (P) — theoretical distance the prop would move in one revolution in an ideal medium (inches). Higher pitch → higher top speed and load on motor, lower acceleration/efficiency at low speed.

3. Blade count — 2, 3, 4+ blades. More blades → more thrust at low RPM and more load/noise/less efficiency per RPM.

4. Blade shape & chord — wider chord increases thrust/drag; planform affects stall and efficiency.

5. Airfoil/profile — influences lift/drag; most hobby props use simple cambered sections optimized for low Reynolds number.

6. Material & stiffness — plastic/nylon (cheaper, forgiving), carbon fiber (stiffer, efficient, more expensive), composite (balanced). Stiffness affects thrust transfer and durability.

7. Mass / rotational inertia — heavier props increase inertia (smoother but slower response) and reduce flight time.

8. Tip speed / compressibility — keep tip Mach low to avoid noise and efficiency loss (rule: keep tip speed < ~0.7× speed of sound for hobby props).

9. Motor match (KV, torque) — prop demands torque; match prop size/pitch to motor Kv and its torque curve so motor isn't overheated or current-limited.

10. ESC & battery current capability — prop + motor combination must not exceed ESC continuous current and battery discharge capability.

11. Static thrust & thrust-to-weight ratio — choose props that provide adequate static thrust so T_total ≥ 2× weight (for good manoeuvrability) or higher depending on use.

12. Efficiency (thrust per watt) — for endurance/long-range optimize for high thrust/W; for agility you may accept lower efficiency.

13. Noise — blade count, sweep, and tip speed affect noise signature.

14. Direction (CW/CCW) and mounting compatibility — make sure prop mounting hole/adapter fits and you order CW/CCW sets.

15. Durability / cost / availability — how often will you break them? are spares available?

16. Flight purpose (use case) — racing/freestyle vs cinematic/long-range dictates different diam/pitch/blade-count tradeoffs.

17. Ground clearance / frame size — ensure props won't clip the frame or each other at max tilt.

18. Balance / vibration — factory balancing or hand-balance; unbalanced props cause vibration/inefficient flight.

Key formulas (simple rotor theory approximations)

Use the standard dimensional relationships for propellers (D in meters, n in rev/s, ρ = air density ≈ 1.225 kg/m³):

- Thrust (order-of-magnitude, using coefficient):
  (T = C_T , \rho , n^2 , D^4)

- Power (shaft):
  (P = C_P , \rho , n^3 , D^5)

- Tip speed:
  (V_{tip} = \omega R = 2\pi n \cdot \frac{D}{2}).
  Tip Mach = (V_{tip}/a) where (a) ≈ 343 m/s at sea level. Keep Tip Mach < ~0.7 for hobby props.

- Advance ratio (useful for matching pitch and rpm):
  (J = \dfrac{V}{nD}) where V is aircraft forward speed. Many prop maps are plotted vs J.

Notes: (C_T) and (C_P) are empirical coefficients that depend on pitch, blade shape, Reynolds number, blade count, etc. Hobby practice is to measure static thrust and current draw on a thrust stand for the most reliable results.

Practical, design-oriented guidance

- Match prop to motor KV and battery voltage: High-Kv motors like 2300–3000KV suit small-diameter, high-RPM props (e.g., 5"×3–5"). Low-Kv motors (600–900KV) pair with large-diameter, higher-pitch props (e.g., 10–14") for long-range, efficient flight.
- Start conservative on pitch: a lower pitch reduces stall and motor current; you can increase pitch if you need speed and have headroom in current/temperature.
- Blade count tradeoffs:
    - 2-blade: highest efficiency and responsiveness (common for racing/FPV).
    - 3-blade: more static thrust and smoother feel, slightly lower efficiency (popular for freestyle).
    - 4+ blade: even more thrust/noise and less efficiency — used when thrust and compact diameter are needed.
- Efficiency vs agility: For long flight times choose larger diameter and lower pitch (lower rpm for same thrust). For quick acceleration and agility choose smaller diameter with higher pitch only if motor/battery can supply current.
- Check motor & ESC current at hover and max throttle: Measure or estimate current draw at the expected throttle for hover (typically ~25–40% throttle for optimized setups) and at full throttle. Ensure ESC and battery C-rating handle peak and continuous draw.
- Thrust-to-weight rules of thumb:
    - Cruising/photography: aim for 2.0–2.5× total thrust / weight.
    - Freestyle/racing: 3× or more for aggressive maneuvers.
    - Heavy-lift: choose larger props and lower pitch to stay efficient.
- Tip clearance & prop overlap: Ensure frame clearance at max tilt; avoid props that will overlap or strike landing gear.
- Noise & legal/urban considerations: If stealth/noise matters (e.g., filming in urban areas), favor lower tip speed, more blades with special blade shaping, and slower RPM operation.

Use-case quick recipes

- 5" freestyle / FPV quad: 5" diameter, 3–5" pitch range, 3-blade common for balance of thrust and control (e.g., 5×4.3 or 5×4.5 tri-blade).

- Cine / smooth flight: 6–9" props with lower pitch (e.g., 6×4 or 8×4), low Kv motors for efficiency and quietness.

- Long-range / heavy-lift: 9–14" props, 2-blade or folding props, low Kv motors, high energy batteries (higher capacity).

- Racing: 5" or smaller with high pitch (but watch current). 3-blade or 4-blade depending on desired handling.

How to pick and test (workflow)

1. Define payload, frame size, flight purpose, battery voltage, and motor KV.

2. Choose a candidate prop family by diameter that fits frame clearance and motor torque capability.

3. Choose blade count and a conservative pitch that your motor/ESC/battery can drive.

4. Check datasheets/specs: motor max current / torque, ESC continuous current, battery C rating. Ensure margins.

5. Throttle curve test on a thrust stand: measure static thrust (N or oz), current (A), and voltage at hover throttle and full throttle for each prop. Record thrust/W and thrust per motor.

6. Evaluate: does total static thrust exceed required thrust-to-weight target? Are currents within ESC/battery limits? Is motor temperature acceptable after a realistic flight profile?

7. Iterate: increase pitch or blade count if you need more thrust and have current margin; reduce pitch/diameter for efficiency if current or temps are high.

8. Balance props and confirm no dangerous vibration. Replace if cracked or delaminated.

Safety & practical tips

- Always use matched CW/CCW props and install them in the correct rotation order.

- Balance new props before flying.

- Monitor motor temps after first flights; overheated motors indicate mismatched prop or insufficient cooling.

- Keep tip speed below recommended limits to avoid inefficiency and audible harshness.
- Have spares — props break.

Short example calculation (estimate tip speed)

Example: 5" prop at 20,000 RPM (typical for small racing motors).

- D = 5 in = 0.127 m. RPM = 20,000 → n (rev/s) = 333.33 rev/s.
- Radius R = 0.0635 m. $\omega = 2\pi n = 2\pi \times 333.33 = 2094$ rad/s.
- $V_{tip} = \omega \cdot R = 2094 \times 0.0635 \approx 133$ m/s. Mach ≈ 133/343 ≈ 0.39 → comfortably below 0.7.
  (So tip compressibility not an immediate issue here — but if you go to large props at high RPM, re-check.)

If you want, tell me: frame size, motor KV, battery voltage, and intended use (racing / film / long-range / heavy-lift) and I'll propose 3 concrete prop choices (diameter × pitch × blade count) with estimated thrust/current tradeoffs and whether your ESC/battery would need an upgrade.

Google Gemini:

The specifications and design parameters you should consider when selecting propellers for a quadcopter are critical as they directly influence thrust, efficiency, speed, maneuverability, and flight time.[1]

Propeller Specifications and Geometry

These are the numbers typically marked on a propeller and describe its physical shape and resulting aerodynamic performance:

- Diameter (Size): The first number in a propeller's designation (e.g., the '5' in 5x4x3).[2] It's the maximum circular distance from tip to tip, measured in inches.[3]
  - Larger Diameter: Generates more thrust and is generally more efficient at lower RPMs, ideal for *long-endurance, heavy-lift* applications (cinematic, photography drones). However, it results in a higher moment of inertia, leading to *lower responsiveness*.[4]
  - Smaller Diameter: Allows for faster RPM changes and is more responsive/nimble, suitable for *racing and freestyle* applications.[5]
- Pitch: The second number (e.g., the '4' in 5x4x3), also measured in inches. It represents the theoretical distance the propeller would advance in one complete revolution if moving through a solid.[6]

    - Higher Pitch: Moves more air per rotation, generating greater thrust and enabling *higher top speeds*.[7] Requires more power to spin, which can drain the battery faster.[8]
    - Lower Pitch: Easier to spin, resulting in quicker RPM changes and better handling of turbulent air (propwash), ideal for *precise control and smoothness*.[9]
- Number of Blades: The third number (e.g., the '3' in 5x4x3). Common configurations are 2-blade, 3-blade (tri-blade), and 4-blade (quad-blade).[10]
    - Fewer Blades (2-Blade):[11] Generally most efficient and lightest, with less drag, often used for *long-range* flights.
    - More Blades (3 or 4-Blade): Provides more lift and better control/grip in the air, with reduced vibration and noise, at the cost of lower efficiency and higher power draw.[12]
- Propeller Profile/Shape (Airfoil): The cross-sectional shape of the blade, which is similar to an airplane wing. This affects lift, drag, and noise. Scimitar-like (curved) blades are often used to reduce noise and improve aerodynamic efficiency.

Material and Structural Parameters

The choice of material affects the propeller's durability, weight, and rigidity.

- Material:
    - Plastic/Nylon: Lightweight, inexpensive, and flexible, making them ideal for beginners and less prone to breaking on impact.[13] Their flexibility can cause thrust loss at high RPMs.
    - Carbon Fiber: High strength, lightweight, and very stiff. Provides better aerodynamic performance and less vibration, but is more expensive and brittle, making it prone to breaking clean off in a crash.
    - Polycarbonate/Composite: Offers a balance of flexibility and strength, often the choice for FPV racing/freestyle for durability.
- Weight (Moment of Inertia): Lighter propellers have a lower moment of inertia, allowing the motors to accelerate and decelerate faster, improving responsiveness and agility.
- Blade Root/Hub Design: Must match the motor shaft diameter and mounting style (e.g., T-mount, direct mount with a nut, quick-release).

System and Performance Considerations

These parameters link the propeller to the rest of the quadcopter system and its intended mission.

- Required Thrust: The propeller must be able to generate enough total thrust to lift the quadcopter and its payload. A common design target is a thrust-to-weight ratio of at least 2:1 for stable hovering and basic maneuvering, but often 4:1 or higher for agile or racing drones.
- Motor KV and Battery Voltage: The propeller selection must be compatible with the motor's Kv (RPM per Volt) and the battery's voltage (e.g., 4S, 6S).
    - High Kv Motors typically use smaller, lower-pitch props to prevent over-drawing current and maintain efficiency.

    - Low Kv Motors typically use larger, higher-pitch props to generate more thrust and torque.
- Efficiency: Measured as thrust generated per Watt of power consumed (g/W). This is crucial for maximizing flight time and endurance.
- Maximum RPM / Tip Speed: The propeller's tips should not approach the speed of sound, as this drastically reduces efficiency and increases noise and vibration.
- Frame Clearance: The propeller's diameter must fit within the physical limits of the quadcopter frame, typically determined by the distance between opposite motor mounts.
- Noise Level: For commercial, cinematic, or inspection applications in urban areas, propellers optimized for low-noise operation are often preferred, typically featuring larger diameters, low pitch, and scimitar-like blade shapes.

Copilot:

When selecting propellers for a quadcopter, key specifications include size, pitch, material, blade count, and compatibility with motor and frame. These factors directly affect thrust, efficiency, maneuverability, and flight time.

Here’s a detailed breakdown of the most important parameters to consider:

Core Specifications

- Diameter (Size):
    - Larger propellers generate more thrust but require more torque and reduce responsiveness.
    - Smaller propellers are more agile and better for racing or acrobatic drones.
- Pitch:
    - Pitch refers to the distance a propeller would move in one rotation through a soft medium.
    - Higher pitch = faster forward speed but more power consumption.
    - Lower pitch = better efficiency and control, especially for hovering.
- Blade Count:
    - 2-blade props: more efficient, less drag, longer flight time.
    - 3-blade props: better stability and control, especially in windy conditions.
    - 4-blade or more: increased thrust and grip, but reduced efficiency.
- Material:
    - Plastic: lightweight and inexpensive, but less durable.
    - Carbon fiber: strong and stiff, ideal for performance drones.
    - Composite: balances strength and flexibility.

Design Considerations

- Motor Compatibility:
    - Propeller size and pitch must match the motor's torque and RPM range.
    - Overloading motors with large props can cause overheating or inefficiency.
- Frame Clearance:
    - Ensure propellers don't overlap or hit the frame.
    - Consider diagonal motor-to-motor distance when choosing diameter.
- Thrust-to-Weight Ratio:
    - Aim for a ratio of at least 2:1 for stable flight.
    - Heavier payloads require larger or more aggressive props.
- Flight Style:
    - Racing: small, high-pitch props for speed.
    - Cinematic: larger, low-pitch props for smooth, stable footage.
    - Freestyle: mid-size, mid-pitch for balance between agility and control.

Performance Metrics

- Efficiency:
    - Measured in grams of thrust per watt.
    - A well-matched propeller-motor combo maximizes flight time.
- Noise Level:
    - More blades and higher RPMs increase noise.
    - Lower pitch and larger diameter can reduce sound.
- Vibration and Balance:
    - Poorly balanced props cause vibrations that affect sensors and video quality.
    - Use high-quality or pre-balanced props for precision applications.

Optimization Tips

- Use thrust stand data or simulation tools to test combinations.
- Consider environmental factors like altitude and wind.

- Swap and test different props to fine-tune performance for your specific drone build Tyto Robotics blog.wildair.ch.

Would you like help choosing propellers for a specific drone model or flight purpose? I can walk you through a tailored selection.

Case Study #4

Prompt 1: “What is NanoOWL and how does it compare to more traditional AI algorithms like YOLOv5?”

Each language model provided relevant details when summarizing NanoOWL and comparing both algorithms, but each did so with a slightly different focus on depth of detail or relevant comparable features.

ChatGPT summarized NanoOWL as a modern vision-language with open-vocabulary detection, allowing users to simply enter whatever object(s) they want to detect within a prompt without having to retrain the model. NanoOWL offers real-time performance when hardware requirements are met such as memory and GPU availability. ChatGPT quickly summarized YOLOv5 as part of a family of object detection algorithms that is widely used for real-time object detection optimized for speed and high accuracy on fixed class sets. When comparing both algorithms, ChatGPT created a table comparing each algorithm features such as flexibility, performance, model architecture, and accuracy. Finally, ChatGPT gave the user considerations for use cases like hardware compatibility, latency, and flexibility before answering which model it would use; YOLOv5 for fixed classes or NanoOWL for flexible, prompt-based detection.

Google Gemini approached the question at a higher-level, summarizing NanoOWL as an open-vocabulary detection algorithm with real-time optimization and tree detection pipeline. Next, Gemini compared NanoOWL to YOLOv5 by comparing features such as detection type, training requirements, deployment focus, and object classes. Finally, Gemini provided the user with its consideration; YOLOv5 for fixed sets of known objects or NanoOWL for flexibility and versatility.

Copilot provides a more “to-the-point” approach, summarizing NanoOWL with key features like open-world detection, real-time performance, nested detection, and flexibility. Copilot then compares both algorithms by their features such as detection scope, architecture, speed, and flexibility before telling the user to choose YOLOv5 for high-speed detection or choose NanoOWL for flexible, prompt-driven detection.

In combination, the three models addressed the functionality of both YOLOv5 and NanoOWL as well as providing the user which algorithm to choose after the comparison. ChatGPT focused on providing the user with important details about both algorithms before continuing to algorithm comparison and which model to choose. Gemini did not go into as much detail about NanoOWl compared to ChatGPT and did not provide a summary for YOLOv5, but still came to the same comparison results and algorithm selection. Copilot provided the user with the needed information (nothing more, nothing less) to summarize NanoOWL and offer which algorithm to implement after comparison. By comparing these responses, it becomes clear that different AI

models take different approaches to explain and compare software algorithms, but in the end all had the same explanation on when to use YOLOv5 / NanoOWL.

Case Study #5

Prompt: “What project parameters will improve the video stream on an autonomous quadcopter?”

ChatGPT:

Great question! For a clean, low-latency video link on an autonomous quad, tune parameters in four layers: camera → encoder → transport/radio → ground station. Here’s a practical checklist with sane starting points (works well for 720p30 on Jetson/Pi/OpenHD-style links).

1) Camera & image pipeline

Resolution & FPS: 1280×720 @ 30 fps (good balance). If your link is solid, try 60 fps for smoother control; if marginal, drop to 24–25 fps.

Shutter & exposure: Shutter ≈ 1/(2×fps) (≈1/60 for 30 fps) to reduce motion blur; lock exposure/ISO if lighting is stable to avoid bitrate spikes. Add ND filters in bright sun to keep the shutter reasonable.

White balance & gamma: Fix white balance; avoid auto that hunts. Enable WDR/HDR only if needed—can increase noise/bitrate.

Stabilization: Prefer EIS (camera or software) only if your airframe is very shaky; it costs crop and sometimes latency.

Lens focus & distortion: Manually set focus at flight distances; correct strong fisheye only if the control operator needs rectified horizons (rectification adds compute/latency).

2) Video encoding (Jetson NVENC / Pi hardware encoder)

Codec: H.264 for lowest latency and broad support; H.265 only if bandwidth is tight and latency budget allows.

Bitrate: Start 6–8 Mb/s for 720p30. Go 4–6 Mb/s for long range; 8–12 Mb/s if link is strong.

Rate control: CBR, low-delay. Keep VBV small (e.g., VBV ≈ 1× target bitrate).

GOP / keyframe interval: IDR every 1 sec (≈30 frames at 30 fps). Shorter helps recovery after loss (0.5 s); longer improves efficiency.

B-frames: Disable (0) for latency.

Profile/level: High, CABAC on (unless CPU-constrained decoder).

Intra-refresh / recovery points: Enable periodic intra refresh or frequent IDR to recover from packet loss quickly.

Slices/SEI/SPS-PPS: Insert SPS/PPS in-band and on IDRs; helps decoders re-lock after drops.

Color & sampling: 4:2:0, 8-bit is standard and efficient.

Pre-processing: Mild temporal noise reduction helps compressibility; avoid heavy denoise/sharpen that adds latency.

Example NVENC-style knobs (conceptual):

preset=low_latency, rc=cbr, bitrate=7M, maxbitrate=7M, gop=30, bframes=0, vbv_size=7M, repeat-headers=on, intra-refresh=on, idr-period=30

3) Transport, buffering, and software pipeline

Protocol: RTP/UDP for minimal latency. Add FEC (forward error correction) if you can spare ~10–20% overhead.

Packetization/MTU: Set MTU ≈ 1460–1472 bytes to avoid IP fragmentation over Wi-Fi; align RTP payload accordingly.

Jitter buffers: Keep small: 30–80 ms on ground; tune until stutter disappears.

Queues/buffers: Minimize app queues in GStreamer/FFmpeg; avoid deep OS socket buffers on TX/RX.

Timestamps/clocking: Use a single clock domain; if mixed devices, run NTP/PTP so capture/encode/decode timing stays sane.

Error handling: Enable RTP retransmit (RTX) only if your latency budget allows; otherwise prefer FEC + frequent IDRs.

4) RF link (OpenHD/Wi-Fi in monitor mode style)

Band choice: 5.8 GHz for crowded areas/shorter range, 2.4 GHz for penetration/longer range. Avoid DFS channels mid-flight.

Channel width: 20 MHz for range/robustness; 40 MHz if SNR is excellent and you need throughput.

Data rate (MCS): Fix to a robust MCS (not auto). Start with a mid-rate (e.g., MCS3–MCS5 on 20 MHz) and adjust after field tests.

Guard interval: Short GI only if SNR > ~25 dB; otherwise use normal GI.

TX power & legality: Increase within local regs; avoid power saving modes; lock country code correctly.

Antennas: On the quad, use circular-polarized (RHCP/LHCP) stub with clear LOS. Ground: diversity (omni + directional helical/patch). Match polarization and space antennas apart.

Placement/EMI: Keep VTX/Wi-Fi and antenna leads away from ESCs, power lines, and carbon; add ferrite beads; ensure clean, regulated power.

5) Ground station & decoding

Hardware decode: Use GPU/SoC decoder (VAAPI/NVDEC) to keep latency low and avoid CPU spikes.

Display pipeline: Disable compositor buffering and “video sync” features that add frames; use vsync-off or low-latency modes if available.

OS tweaks: Prioritize the receiver process, performance power governor, and lock CPU frequencies to reduce jitter.

6) Reliability strategies

Dual-stream: Record onboard at high bitrate (e.g., 30–50 Mb/s) while transmitting a lower-bitrate low-latency stream for control/monitoring.

Redundancy: If budget allows, run diverse links (e.g., primary Wi-Fi + backup low-rate analog/digital).

Health telemetry: Overlay minimal OSD (flight mode, voltage, RSSI/SNR, packet loss) but keep it lightweight to save bits.

7) Concrete starting presets (720p30 control link)

Resolution/FPS: 1280×720 @ 30 fps

Encoder: H.264, CBR 7 Mb/s, GOP 30, B-frames 0, High profile, intra-refresh on, SPS/PPS with each IDR

Transport: RTP/UDP, MTU 1472, Jitter buffer 50 ms, optional FEC 10%

RF: 5.8 GHz, 20 MHz channel, fixed mid MCS, short GI off initially, diversity RX

Ground: GPU decode, minimal display buffering

8) How to test & iterate

End-to-end latency: LED timer on camera + photodiode on screen, or film both with a high-FPS phone; aim for <120 ms (excellent), <200 ms (good).

Link quality: Log SNR, PER/packet loss, retransmits; step bitrate/MCS and plot stutter vs. loss.

Quality metrics: Record short clips and compare SSIM/PSNR across settings; check fast-motion scenes and foliage (bitrate killers).

Thermals/power: Monitor encoder and radio temps; throttle will silently add latency/drops.

Google Gemini:

Improving the video stream on an autonomous quadcopter primarily involves optimizing parameters related to latency (the delay between capture and display) and bandwidth efficiency (sending high-quality video with minimal data).

Here are the key project parameters that will improve your video stream:

Communication & Network Parameters

The biggest bottleneck is often the wireless link.

- Wireless Communication Protocol: Choose a robust, low-latency protocol.
    - 5G/4G LTE/Cellular: Offers wider coverage, which is crucial for autonomous or Beyond Visual Line of Sight (BVLOS) flights.
    - Specialized Protocols (e.g., WebRTC, SRT): These are optimized for low-latency, real-time streaming and reliability over unpredictable networks.[1]
- Transmission Power and Frequency:
    - Higher Transmission Power: Generally increases range and signal strength, leading to a more stable connection and fewer frame drops, which reduces perceived latency.
    - Operating Frequency: Selecting a less congested frequency band, or using dynamic channel switching, improves link reliability and data throughput.
- Antenna System: Use directional, high-gain antennas on the ground station and appropriately selected antennas on the drone to maintain a strong link over distance and during quadcopter maneuvers (which can change the orientation of the antennas).
- Multi-Link Bonding (Optional): Integrating a bonded router that combines multiple cellular SIM cards (e.g., 4G/5G) for a more resilient and higher-bandwidth connection.[2]

Onboard Processing & Encoding Parameters

Optimizing how the video is processed and compressed on the drone is essential to maximize bandwidth.[3]

- Video Compression Algorithm: Select a modern, efficient encoder for low-latency encoding.
    - H.264/H.265 (HEVC): Widely used standards, but their implementation must be optimized for speed.
    - Slice-based Encoding: Breaking a frame into multiple slices that are encoded and transmitted in parallel reduces end-to-end latency.[4]

- Compression Rate / Bitrate Management:
    - Adaptive Bitrate Streaming: Use AI-driven or smart compression algorithms that dynamically adjust the video's compression and quality based on the current available network bandwidth.[5] This prevents video drops when the signal is weak and uses the maximum quality when the signal is strong.
- Camera Frame Rate (FPS):
    - Higher Frame Rate (e.g., 60 FPS): Reduces the time taken to capture a single frame, contributing to lower end-to-end latency.[6]
- Resolution and Quality:
    - Resolution vs. Latency/Bandwidth Trade-off: Balance the required clarity (e.g., 1080p for detail) with the available bandwidth. In low-bandwidth situations, dynamically reducing the resolution will maintain a low-latency stream, which is often more critical for autonomous navigation than maximum clarity.
- Edge Computing (AI/ML): Process video data on the drone itself.[7] Instead of streaming all raw video, the drone can run algorithms (e.g., object detection, tracking) and only transmit compressed metadata, alerts, or a highly compressed video stream of the detected area.[8] This drastically reduces the data transmission load.

Hardware Parameters

The physical components chosen for the system have a direct impact.

- Camera System: Choose a camera with minimal inherent capture latency (time from light hitting the sensor to outputting a raw signal).
- Onboard Encoder Hardware: Use a dedicated, low-power, high-speed hardware encoder (often a compact chip or module) on the drone for faster and more efficient compression than general-purpose CPUs.[9]
- Ground Station/Receiver Decoding: Ensure the ground station's decoding hardware (computer/tablet) is powerful enough to quickly decompress and display the stream, as decoding time can become a latency bottleneck.[10]
- Display Refresh Rate: A higher refresh rate on the operator's display will also contribute to lower perceived display latency.

Copilot:

To improve video streaming on an autonomous quadcopter, focus on optimizing bandwidth, latency, camera quality, flight path stability, and onboard processing.

Here’s a breakdown of the most impactful project parameters and how they enhance video performance:

Communication & Network Parameters

- Bandwidth Allocation: Ensure sufficient bandwidth for high-resolution video. Use adaptive bitrate streaming to adjust quality based on available bandwidth.

- Low-Latency Protocols: Implement protocols like RTSP or WebRTC for real-time transmission with minimal delay.
- Signal Strength & Range: Use high-gain antennas or directional antennas to maintain strong connections over long distances.
- Compression Algorithms: Employ efficient video codecs (e.g., H.265) to reduce data load without sacrificing quality.

Camera & Sensor Parameters

- Camera Resolution & Frame Rate: Use cameras capable of 1080p or higher at 30–60 fps for smooth, detailed video.
- Stabilization: Integrate gimbals or electronic image stabilization to reduce motion blur and jitter.
- Field of View: Wide-angle or 360-degree cameras can enhance situational awareness and reduce the need for frequent repositioning.

Onboard Processing & Control

- Edge Computing: Use onboard processors (e.g., Raspberry Pi or Jetson Nano) to handle video encoding and object tracking locally.
- Model Predictive Control (MPC): Implement MPC algorithms to maintain stable flight and predictable motion, which improves video consistency.
- Sensor Fusion: Combine data from GPS, IMU, and visual sensors to maintain accurate positioning and orientation.

Flight Path & Environmental Awareness

- Path Planning: Optimize flight paths to avoid signal dropouts and maintain line-of-sight with the receiver.
- Altitude Control: Maintain consistent altitude to reduce perspective shifts and improve video framing.
- Obstacle Avoidance: Use LiDAR or stereo cameras to prevent sudden maneuvers that disrupt streaming.

Monitoring & Feedback Systems

- Video Quality Monitoring: Implement real-time feedback systems to detect and correct streaming issues like frame drops or latency spikes.
- QoE (Quality of Experience) Metrics: Track user-perceived quality to guide adaptive streaming decisions.

Case Study #6

Prompt: “What is ArduPilot and does it compare to PX4 autopilot for swarming and search and rescue operations?”

ChatGPT’s Response:

Here’s a breakdown of ArduPilot and PX4 Autopilot, with a focus on how they relate to swarming and search-&-rescue (SAR) operations, followed by how they compare and which might make more sense for your senior-design drone swarm project at University of Central Florida (assuming the drone project you mentioned) working on signal-amplifying/detection.

What is ArduPilot

ArduPilot is an open-source autopilot software project (firmware + supporting tools) for autonomous unmanned vehicles. Some key points:

- It supports a very wide range of vehicle types: multirotors (quad, hexa, octo), fixed-wing, VTOL, rovers, boats, submarines. (Wikipedia)

- The software is mature, with strong community support, and has been used in hobbyist, academic, research, and some professional contexts. (ArduPilot.org)

- In terms of features: it supports advanced sensor suites (GPS/RTK, optical flow, lidar/sonar, vision/flow sensors), companion computer integration, mission planning, geofencing, failsafes, etc. (Wikipedia)

- ArduPilot is extremely flexible/versatile. From its website: "one autopilot for any mission" (Copter, Plane, Rover, Sub) (ArduPilot.org)

Relevance to SAR / Swarm

- There are documented efforts using ArduPilot for autonomous search and rescue applications: e.g., a paper "An Autonomous Drone for Avalanche Search and Rescue Integrating ArduPilot …" shows it being used in challenging terrain for SAR. (ResearchGate)

- For swarming: there are community posts showing swarm-type work (e.g., "Introduction to drone Swarming with ArduPilot"). (ArduPilot Discuss)

- Because ArduPilot supports companion computers and complex payloads (which you may need for signal-amplifying drones) it gives you flexibility to implement custom logic (e.g., payload control, signal relaying, coordination among multiple drones).

Strengths for your project

- Good community & documentation; being versatile is helpful if your platform changes (e.g., mixed fixed-wing + multirotor)

- Lower barrier to entry/more "ready" features for many vehicle types

- Good for research/academic settings where custom modifications and experimentation are key

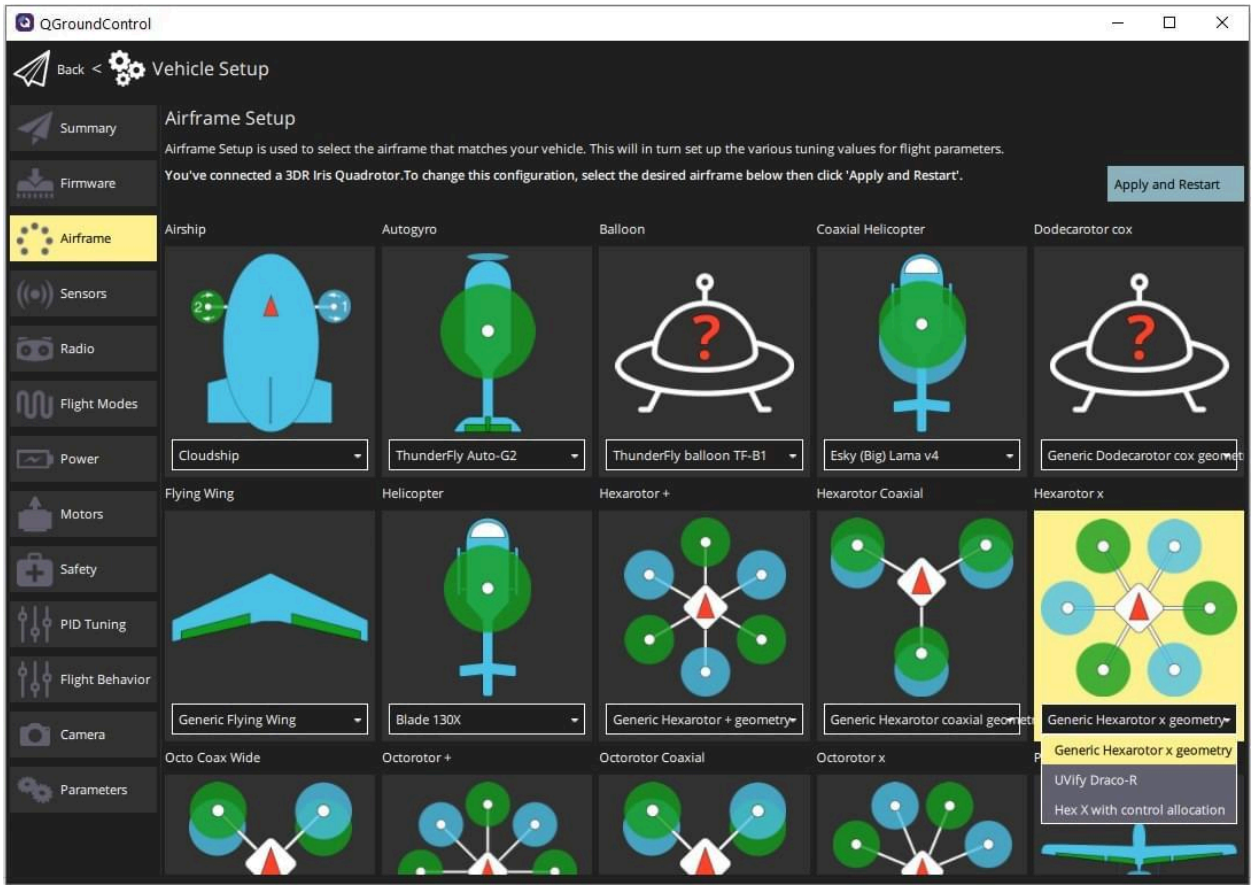
Potential Limitations

- While there is swarm work, swarming might require more custom integration (communication, mission orchestration, distributed control)

- Depending on your architecture (number of vehicles, communication bandwidth, coordination algorithm) you might need to build additional layers on top of ArduPilot

---

What is PX4 Autopilot


QGroundControl
Back
Vehicle Setup
Summary
Firmware
Airframe
Sensors
Radio
Flight Modes
Power
Motors
Safety
PID Tuning
Flight Behavior
Camera
Parameters
Airframe Setup
Airframe Setup is used to select the airframe that matches your vehicle. This will in turn set up the various tuning values for flight parameters.
You've connected a 3DR Iris Quadrotor.To change this configuration, select the desired airframe below then click 'Apply and Restart'.
Apply and Restart
Airship
Cloudship
Autogyro
ThunderFly Auto-G2
Balloon
ThunderFly balloon TF-B1
Coaxial Helicopter
Esky (Big) Lama v4
Dodecarotor cox
Generic Dodecarotor cox geomet
Flying Wing
Generic Flying Wing
Helicopter
Blade 130X
Hexarotor +
Generic Hexarotor + geometry
Hexarotor Coaxial
Generic Hexarotor coaxial geomet
Hexarotor x
Generic Hexarotor x geometry
Generic Hexarotor x geometry
UVify Draco-R
Hex X with control allocation
Octo Coax Wide
Octorotor +
Octorotor Coaxial
Octorotor x

PX4 Autopilot is another open-source flight control/autopilot firmware and ecosystem for unmanned vehicles.

Key points:

- PX4 supports multiple vehicle types (multirotor, fixed-wing, VTOL, rovers, underwater) similar to ArduPilot. (Wikipedia)
- It emphasizes modularity, precision, performance and has a more "industrial/production" orientation in many cases. For example, many commercial systems build upon PX4. (thedroningcompany.com)
- PX4 uses MAVLink protocol, works with QGroundControl as ground station, has strong support for developer integration (ROS, companion computers).
- Swarm support: There are community references to swarming with PX4 (e.g., "Swarm Flight using 100 drones" using Pixhawk + PX4 ecosystem) (PX4 Forum)

Strengths for your project

- If you anticipate scaling up to many vehicles, need industrial reliability, or integration with ROS/companion computer, PX4 might offer good architecture
- If you want to leverage existing swarm frameworks/tools built around PX4 (though still largely research/experimental)
- If you want to commercialize or transition into production (since PX4 may have more "commercial friendly" aspects)

Potential Limitations

- Might have a steeper learning curve, especially if your team is more used to "research/hobby" style flexibility
- Some users report the documentation or "ease of custom mission scripting" might lag compared to ArduPilot in certain use cases. For example, a user commented "with PX4 stack is difficult to know exactly what is supported or well tested since documentation is not as complete as with ArduPilot." (PX4 Forum)
- You may need to build more of the swarm coordination layer explicitly; out-of-the-box swarm support is still a research area.

Direct Comparison: ArduPilot vs PX4 for Swarming + SAR

Here's a side-by-side comparison relevant for your senior design signal-amplifying drone swarm for search & rescue:

Criterion	ArduPilot	PX4 Autopilot
Vehicle type support & flexibility	Excellent: multiple vehicle types, wide hardware support, strong community. (ArduPilot.org)	Also excellent: many vehicle types, modular architecture. (Wikipedia)
Community, documentation, ease of entry	Very good: strong community, lots of tutorials, widely used in research/hobby contexts.	Good but some reports of steeper learning curve or less mature docs in niche areas. (PX4 Forum)
Swarm capability/support	There is documented swarm research (e.g., ArduPilot + Gazebo plugin for swarm robotics) (arxiv.org) Also community posts about swarm leader-follower setups. (ArduPilot Discuss)	Swarm work also present (100-drone flight, swarm packages) (PX4 Forum) But may require more custom setup/communication infrastructure.
Search & Rescue / custom mission suitability	Good: explicit examples of SAR mission with ArduPilot. (ResearchGate)	Suitable, but you may need to build more of the mission logic yourself (payloads, search algorithm, coordination)
Custom payload / companion computer / algorithm integration	Strong: because many users do heavy customization, companion computers, vision, lidar integration.	Also strong: PX4 supports companion computers, ROS2, etc. For advanced autonomy, PX4 may offer more "industrial grade" stack.
Licensing / commercialization concerns	GPLv3 for ArduPilot: means if you modify the code you may need to open-source derived. (ThinkRobotics.com)	PX4 is BSD-3 license: more permissive for businesses/commercial use. (ThinkRobotics.com)
Fit for a senior-design project (research/development)	Very strong: allows experimentation, lots of open research, high flexibility.	Also good, but maybe more "production oriented" or may require more infrastructure; research groups still use it though.

My Recommendation for Your Project

Given your context (senior design project at UCF, signal-amplifying drone swarm, likely research/academic rather than immediate commercial deployment, mixed vehicle type perhaps, custom mission logic), here are my thoughts:

- ArduPilot is likely the "safer" choice for rapid prototyping, research, customization, especially if your emphasis is on mission logic (search & rescue, signal amplification, swarm coordination) rather than on pure production robustness right away. The community, documentation, wide flexibility are major pluses.

- If you plan to scale up the swarm significantly (many drones), or if you see your system evolving into a commercial product, or you intend heavy integration with ROS2, swarm controller frameworks, etc., then PX4 could be very interesting and maybe worth the upfront ramp-up cost.

- Because your project involves "swarming" + "search & rescue" + "signal-amplifying", you may have a custom payload (signal amplifier/receiver), custom coordination algorithm among drones (who goes where, how to cover area, how to relay signal). In that vein, you should ensure whichever autopilot you choose allows:

    - companion computer integration (for your signal logic)

    - custom mission scripting (leader/follower, dynamic re-tasking)

    - reliable communication among drones and GCS

    - ability to integrate sensors/payloads.

- One possible strategy: Choose ArduPilot to get a working prototype / baseline. Then if timeline allows, evaluate PX4 for later enhancement or future work (swarm scaling, more industrial robustness).

If you like, I can pull together a deeper comparison matrix (e.g., communication protocols, swarm libraries/packages available, real-world case studies of each in SAR, latency & bandwidth considerations for swarming, companion computer workflows) tailored to your senior design drone project (signal-amplifying drone swarm). Would you like that?

CoPilot's Response:

Quick comparison table (most relevant attributes)

Attribute	ArduPilot	PX4

Primary design focus	Feature-rich, supports many vehicle types (planes, copters, rovers)	Modular, real-time performance focus for multicopters and fixed-wing
Swarm tooling & frameworks	Community add-ons, MAVLink-based swarm scripts, third‑party swarm frameworks	Native support for companion-computer workflows; used with PX4/NuttX + ROS/px4_ros_comm
Autonomy & mission planning	Powerful mission scripting (Lua), complex mission behaviors, geofencing	Flight stack + companion computer pattern; autonomy commonly implemented offboard
Sensor/airframe support	Very broad hardware support and legacy boards	Strong support on modern flight controllers (PX4 FMUv5 family, etc.)
Simulation & testing	SITL with Gazebo, SITL supports many vehicles	jMAVSim, Gazebo, PX4 SITL, built for CI and rapid testing
Community & ecosystem	Large long-standing community, many field deployments	Large industry adoption, strong corporate backing (Dronecode)
Certification & commercial use	Used in many commercial/academic projects; slower upstream than PX4 for some configs	Widely used in commercial products; clear vendor toolchains
Ease for swarming experiments	Easier to run full autonomy on flight controller (Lua); many community examples	Easier to implement complex offboard swarm algorithms using companion computers and ROS

Sources: Drone Dojo ThinkRobotics.com THE DRONING COMPANY.

What ArduPilot is

ArduPilot is a mature, open‑source autopilot software stack that runs on many flight controllers and supports a wide range of vehicle types (multicopters, planes, VTOLs, rovers, submarines). It emphasizes rich onboard features (advanced flight modes, extensive sensor drivers, mission scripting via Lua, and broad hardware compatibility) and long history of community and field use Drone Dojo.

What PX4 is

PX4 is an open-source flight stack focused on modularity, real-time performance, and integration with companion computers and ROS ecosystems. PX4 is designed to work closely with modern flight controllers (FMUv5 family, Pixhawk variants) and supports outboard autonomy patterns where higher-level algorithms run on companion computers and communicate via MAVLink or ROS bridges ThinkRobotics.com THE DRONING COMPANY.

How they compare for swarming

- **Architecture matters**: ArduPilot often places more autonomy capability directly on the autopilot (Lua scripting, many onboard behaviors), which can simplify small-scale swarms that run distributed logic on each vehicle. PX4 typically expects heavier autonomy to run on a companion computer, which is advantageous for advanced swarm algorithms needing more compute or ROS tooling Drone Dojo ThinkRobotics.com.
- **Communication & protocols**: Both use MAVLink as the primary telemetry/command protocol, so swarm managers and ground/control systems can work with either stack ThinkRobotics.com.
- **Existing swarm efforts**: Research and commercial swarms have been built on both stacks. PX4 + ROS is common for complex, centralized/offboard swarm coordination; ArduPilot examples exist that leverage onboard scripting or companion computers for decentralized behaviors ThinkRobotics.com Drone Dojo.
- **Simulation & testing**: PX4 has strong CI and SITL tooling that integrates easily with ROS/Gazebo for multi-vehicle simulation. ArduPilot's SITL also supports multi-vehicle scenarios and has many community examples for swarm experiments ThinkRobotics.com Drone Dojo.

How they compare for search-and-rescue (SAR)

- **Mission complexity**: SAR missions often need robust RTK/GNSS, obstacle avoidance, vision processing, and dynamic re-tasking. PX4's companion-computer pattern makes it straightforward to run heavy perception (CNNs, SLAM) and task allocation onboard while PX4 handles low-level control THE DRONING COMPANY.
- **Onboard autonomy**: If you prefer simpler field-deployable platforms where the flight controller can execute complex waypoint logic without a companion computer, ArduPilot's built-in mission scripting and mature flight modes can be advantageous Drone Dojo.
- **Safety & redundancy**: Both stacks support GPS failover, EKF, geofencing, and telemetry health checks. Commercial SAR projects often pair either stack with redundant sensors and higher-level verification layers; PX4's vendor toolchains sometimes make integration with certified hardware easier ThinkRobotics.com THE DRONING COMPANY.

- **Ecosystem for perception and mapping**: PX4 + ROS has a slightly smoother path to integrate perception, mapping, and coordination toolchains commonly used in SAR (e.g., ROS packages, mapping stacks). ArduPilot can also integrate these with companion computers but may keep more capability on the autopilot itself ThinkRobotics.com THE DRONING COMPANY.

Practical recommendation (which to pick)

- Choose **ArduPilot** if:
  - You need broad hardware support or want to run richer mission logic directly on the autopilot (Lua scripting).
  - You aim for decentralized swarms with light computation per vehicle and prefer many built-in flight modes and legacy board support Drone Dojo.
- Choose **PX4** if:
  - Your swarm/SAR algorithms rely on heavy perception, ROS integration, or centralized/offboard coordination running on companion computers.
  - You want modern CI, vendor toolchains, and a workflow that separates low-level control (PX4) from high-level autonomy (companion computer) ThinkRobotics.com THE DRONING COMPANY.
- If possible, prototype with both stacks in SITL/Gazebo and a small multi-vehicle simulation to validate the swarm coordination and SAR pipeline you plan to use ThinkRobotics.com Drone Dojo.

Next practical steps

1. Pick a representative mission (e.g., 3-vehicle area search with onboard obstacle avoidance and target reporting).
2. Run SITL multi-vehicle simulations for both stacks (ArduPilot SITL; PX4 SITL + Gazebo) to evaluate ease of integrating your perception/coordination code Drone Dojo ThinkRobotics.com.
3. Test on hardware with a companion computer (Raspberry Pi / Jetson) for PX4; for ArduPilot try both direct Lua mission logic and companion-computer approaches to compare latency and robustness THE DRONING COMPANY Drone Dojo.

If you want, tell me the size of your swarm, whether you plan to use ROS or pure MAVLink, and the onboard compute available, and I'll give a tailored recommendation and a short test plan.

Gemini's Response:

ArduPilot is a mature, open-source, and highly versatile autopilot software suite.[1] Originally developed by hobbyists around 2009, it has evolved into a "battle-tested" and trusted platform

used by hobbyists, researchers, and commercial operators alike.[2] Its main strength is its support for a vast array of vehicle types, including multirotors, fixed-wing planes, helicopters, rovers, boats, and even submarines.[3]

Here's a comparison with the PX4 autopilot, focusing on swarming and search and rescue (SAR) operations.

🚀 ArduPilot vs. PX4: At a Glance

The main difference isn't about which is "better," but which is the right **tool** for the job. Their core philosophies and ecosystems lead to different strengths.

Feature	**ArduPilot**	**PX4**
License	**GPLv3 (Copyleft)**	**BSD (Permissive)**
	Must share modifications.	*Can create proprietary, closed-source products.*
Architecture	Monolithic & Integrated	Modular & "Micro-Kernel" (uORB)
	Often seen as more "battle-tested" out of the box.	*Highly flexible and extensible; "a framework to build on."*
Maturity	Older (c. 2009), extremely mature codebase.	Newer (c. 2012), modern, and research-focused.
Ecosystem	Large hobbyist & DIY community.	Strong institutional & commercial backing (Dronecode).

Best For...	Reliability, wide vehicle support, field-proven performance.	Advanced research, commercial development, robotics (ROS 2).

🐝 Comparison for Swarming

Neither platform has a simple "swarm" button, but they enable swarming in different ways.

- **ArduPilot:** Provides more built-in, high-level features for simple swarming.
  - **Leader-Follower Mode:** ArduPilot has a "Follow Mode" that can be used for basic leader-follower swarms.[4]
  - **GCS Swarming:** The **Mission Planner** ground control station (GCS) has a beta "Swarming" interface that simplifies controlling multiple vehicles in formation.[5]
  - **MAVLink & Companion Computers:** Most advanced ArduPilot swarms use scripts on a companion computer (like a Raspberry Pi) to send commands to multiple drones via the MAVLink protocol.[6] This is well-documented by the community.
- **PX4:** Treats swarming as an advanced robotics problem, providing the low-level tools for developers to build their own robust solutions.
  - **ROS 2 Integration:** This is the key. PX4 is designed for deep integration with the **Robot Operating System (ROS 2)**.[7]
  - **Offboard Control:** Complex swarm logic (e.g., custom formations, decentralized decisions) is run "offboard" on companion computers using ROS 2, which then sends high-frequency commands to the PX4 autopilot.[8]
  - **Simulation:** The PX4 ecosystem has excellent support for advanced simulation in **Gazebo**, allowing developers to test complex swarm algorithms safely.[9]

Swarming Bottom Line: ArduPilot is often faster for getting a *simple* leader-follower swarm in the air. PX4 (with ROS 2) is the more powerful and flexible platform for developing *complex, novel, or large-scale* swarm behaviors for research or commercial products.

⛑️ Comparison for Search and Rescue (SAR)

For SAR, reliability and sensor integration are critical.

- **ArduPilot:** Is a proven choice for SAR, with the documentation explicitly listing it as a primary use case.
  - **Proven Reliability:** Its long history means its flight control is exceptionally robust in real-world, challenging conditions (like wind), which is vital for SAR.
  - **Sensor Integration:** The platform makes it straightforward to integrate and use diverse sensors on a companion computer. For example, researchers have successfully used ArduPilot with a Raspberry Pi to detect avalanche beacons, a perfect SAR application.[10]

    - **VTOL Support:** ArduPilot has excellent, mature support for VTOL (Vertical Take-Off and Landing) fixed-wing planes, which are ideal for SAR as they combine the long endurance of a plane with the hovering capability of a multicopter.[11]
- **PX4:** Is a powerful *base* for building commercial or high-tech SAR solutions, rather than an out-of-the-box SAR tool.
    - **Commercial Foundation:** The permissive BSD license makes PX4 the preferred choice for companies (like Auterion) that build proprietary, high-end SAR drones for professional emergency services.[12]
    - **Advanced Features:** PX4's advanced capabilities (e.g., obstacle avoidance, precision navigation) are essential for SAR, but they are often leveraged by third-party software.
    - **GCS-Dependent:** SAR-specific functions, like pre-programmed search patterns (e.g., expanding square, sector search), are typically a feature of the **ground control software** (like QGroundControl or specialized commercial software) rather than the PX4 firmware itself.

SAR Bottom Line: ArduPilot is an excellent, reliable, and accessible choice for SAR, especially for researchers, volunteer SAR teams, or custom-built systems. PX4 serves as the powerful, stable *core* for advanced, often commercial, SAR platforms.

This video demonstrates a beginner's guide to setting up a drone swarm using ArduPilot's leader-follower configuration.