



P.E.G.A.S.U.S.

Pilot Enhanced Guidance and Augmented Sight Utility System

Group 10 Authors:

Luke Anderson
Computer Engineering

Ethan Dill
Optical Engineering

Frank Murphy
Electrical Engineering

Review Committee

Dr. Di Wu	ECE	Assistant Professor
Dr. Enxia Zhang	ECE	Professor
Dr. Shin Tson Wu	CREOL	Professor

Advisors

Dr. Chung Yong Chan	ECE	Senior Lecturer
Dr. Aravinda Kar	CREOL	Professor

Table of Contents

Ch 1: Executive Summary	1
Ch 2: Project Description	1
2.1 Project Background.....	1
2.2 Precedent	2
2.3 Project Goals & Objectives.....	5
2.4 Features and Functionalities.....	7
2.5 List of Requirements and Specifications	8
2.5.2 Key Component Specifications.....	8
2.6 Project Hardware Block Diagram	10
2.7 Project Software Diagram / Flowchart.....	12
2.8 Prototype Illustration & GUI	13
2.9 House of Quality	14
2.10 Test Plan & Live Demonstration.....	15
2.10.1 Test Plan	15
2.10.2 Live Demonstration	15
Ch 3: Research and Investigation	15
3.1 Optics Hardware.....	15
3.1.1 Display Architectures.....	15
3.1.2 Polarization Schemes	21
3.1.3 Collimating Lens System.....	24
3.1.4 Reflective Mirrors.....	28
3.1.5 Aberration Correction Elements	30
3.1.6 Magnifiers.....	33
3.1.7 Self-Dimming Element	36
3.1.8 Optical Combination Technology	38
3.2 Electrical Hardware	41
3.2.1 Power	41
3.2.2 Microcontroller	45
3.2.3 Altimeter	47
3.2.5 Inertial Measurement Unit (IMU).....	50
3.2.6 Global Positioning System (GPS)(Strech Goal)	52
3.3 Software.....	56
3.3.1 MCU Selection from a Software Perspective	56
3.3.2 Software Roles & System Context.....	58

3.3.3 ESP32	59
3.3.4 Raspberry Pi4.....	63
Ch 4: Standards and Design Constraints.....	69
4.1 Optical Standards	69
4.1.1 ISO 10110 – Optical Drawings Standards	69
4.2 Software-Related Standards	70
4.2.1 RTCA DO-178C	70
4.2.2 MISRA-C.....	72
4.2.3 MIL-STD-1472.....	73
4.2.4 I ² C / SPI Communication Protocol Standards.....	74
4.3 Constraints	75
4.3.1 Software Real-Time Performance & Latency	75
4.3.2 Time.....	75
4.3.3 Economic	76
4.3.4 Health and Safety.....	77
Ch 5: Application of ChatGPT and Other Similar Platforms.....	78
5.1 Case Study 1	78
5.2 Software Case Studies	79
5.2.1 AI In the Software Research Section	79
5.2.2 AI in UI Specification Development Using Figma.....	80
5.2.3 AI in Software Design	80
5.3 Hardware Case Study	80
Ch 6: Hardware Design.....	82
Ch 7: Software Design	85
7.1 System Overview.....	85
7.2 ESP32 Software Flow	86
7.3 Raspberry Pi4 Software Flow.....	89
7.4 ESP32 & Pi4 Integration Flow	92
7.5 Software Components	93
7.5.1 Database.....	93
7.5.2 UI Spec	93
7.6 Conclusion	94
Ch 8: System Fabrication/Prototype Construction.....	95
Ch 9: System Testing.....	98
Ch 10: Administrative Content.....	102
10.1 Project Milestones.....	102

10.2 Budget and Funding	103
10.3 Work Distribution	104
<i>Ch 11: Conclusion</i>	<i>105</i>

List of Tables

Table 1: Key Engineering Specifications	8
Table 2: Key Component Specifications	9
Table 3: Optical Component Specifications	9
Table 4: Display Architecture Comparison	19
Table 5: OLED Comparison	21
Table 6: Waveplate Comparison.....	23
Table 7: Polarizer Product Comparison.....	24
Table 8: Collimator Technology Comparison	25
Table 9: Collimator Product Comparison	27
Table 10: Mirror Technology Comparison.....	28
Table 11: Mirror Product Comparison.....	30
Table 12: Mirror Technology Comparison.....	32
Table 13: Mirror Product Comparisons	33
Table 14: Magnifier Technology Comparison.....	34
Table 15: Magnifier Product Comparison	35
Table 16: Self-Dimming Element Technology Comparison	37
Table 17 Self-Dimming Element Product Comparison	38
Table 18: Optical Combiner Technology Comparison.....	39
Table 19: Optical Combiner Product Comparison.....	41
Table 20: List of Components Power Requirements (Made with ChatGPT, 2025)	42
Table 21: Linear vs Switching Voltage Regulators (Made with ChatGPT, 2025)	43
Table 22 5V Regulators Comparison (Made with ChatGPT, 2025).....	44
Table 23: 3.3V Regulators Comparison (Made with ChatGPT, 2025).....	45
Table 24: List of Parts for the Altimeter (Made with ChatGPT, 2025).....	49
Table 25: List of Parts for the IMU(Made with ChatGPT, 2025)	52
Table 26: MCU Combination Comparison [57]	58
Table 27:OS Comparison [58]	59
Table 28: Programming Language Comparison [58]	61
Table 29: IDE Comparison [58]	63
Table 30: Pi OS Comparison [58].....	65

Table 31: Pi Programming Language Comparison.....	66
Table 32: Data Logging Comparison.....	67
Table 33: DO-178 Version History [59]	70
Table 34: Design Assurance Levels [59]	71
Table 35: DO-178C Principles Related to P.E.G.A.S.U.S. [59].....	71
Table 36: MISRA-C Version History [59]	72
Table 37: Relevant MISRA-C Objectives [59].....	73
Table 38: MIL-STD-1472 Version History [59]	73
Table 39: Relevant MIL-STD-1472 Rules [59].....	74

List of Figures

Figure 1: Project Hardware Block Diagram	10
Figure 2: Display Optics Hardware Block Diagram.....	11
Figure 3: Optical Design Schematic	11
Figure 4: Software Diagram.....	12
Figure 5: Prototype Illustration.....	13
Figure 6: Graphical User Interface	13
Figure 7: House of Quality	14
Figure 8: Main PCB.....	82
Figure 9: 12V - 5V Regulator	83
Figure 10: 5V - 3.3V Regulator	83
Figure 11: Altimeter Schematic	84
Figure 12: Inertial Measurement Unit(IMU) Schematic	84
Figure 13: System Flow REMOVE GPS [62]	85
Figure 14: ESP32 Software Flow [62].....	87
Figure 15: Relevant Equations [62]	Error! Bookmark not defined.
Figure 16: CBOR Datagram [62].....	89
Figure 17: Pi4 Software Flow [62]	90
Figure 18: Pi4 Use Case Diagram [62]	91
Figure 19: MCU Integration Flow [62]	92
Figure 20: Database Schema.....	93
Figure 21: Data UI Spec [Figma AI]	93
Figure 22: Camera UI Spec [Figma AI]	94
Figure 23: Project Milestones [13]	103
Figure 24: Bill of Materials [13]	104
Figure 25: Distribution of Workload	105

Appendix A	References
Appendix B	Copyright Permission
Appendix C	Data Sheet (if necessary)
Appendix D	Software Code (if necessary)
Appendix E	ChatGPT Prompts and Outcomes (if necessary)
Appendix F	etc.

Ch 1: Executive Summary

The P.E.G.A.S.U.S [11] project is a student-led senior design project with the goal of creating a portable alternative to current aircraft heads-up displays (HUDs). In most aircrafts, they either come with a HUD or they do not, and aftermarket ones are few and far between. The goal of this project is to create a HUD that provides basic data such as heading, speed, altitude, and an increased FOV to the pilot. The P.E.G.A.S.U.S. project is portable enough to easily transfer between aircrafts and helpful enough to make it worth putting in an aircraft. This project not only fills a gap in current aviation technology but also demonstrates the feasibility of making HUDs accessible to all the aviation community.

Not only does this project fill a gap in the aviation community, but it will also serve as the senior capstone project for our group. Through this project, Ethan will gain hands-on experience with designing, prototyping, testing, and building industry-ready optical devices. Frank will learn how PCBs are designed from the ground up, which is a highly desirable industry skill. Luke will learn how to read flight data and create a GUI on a microcontroller as well as other embedded software skills. All these skills will culminate in a working product that will hopefully help pilots and provide a cheaper alternative for the few products that do exist today.

Ch 2: Project Description

2.1 Project Background

Situational awareness is the primary factor that ensures the safe operation of millions of vehicles around the world each day. In aviation, it is often the case that pilots must make quick decisions during flight based on readings from a complex suite of flight parameter instruments as well as navigational data. This is in addition to understanding their operational environment via visual observation. However, pilots have to increasingly rely on their instrumentation readings as meteorological conditions around them deteriorate and visibility is diminished. It is imperative that these readings are easily visible and clearly displayed, within context, in a cockpit environment.

Currently, there are a variety of ways that pilots are able to stay situationally aware when operating in the sky. These various technologies and systems provide aid in understanding an aircraft's position in the surrounding airspace as well as relative to other aircraft, what the potential nearby hazards could be, and the current and predicted weather conditions for the operational environment. Global Navigation Satellite Systems (or GNSS) provide navigational data and information to pilots and aircraft via satellite communication. These systems allow for an enhanced understanding of the aircraft's position in space as well as its position relative to the desired flight path (Canal Geomatics, 2024). Aircraft flight data monitoring of basic instruments such as airspeed

indicators, attitude indicators, angle of attack indicators, manifold pressure, RPM, and G-indicators all provide vital feedback to pilots that helps them understand and properly manipulate the mechanics of the aircraft to ensure a successful flight (FAA, 2025). Often, these instruments are spread out across the cockpit on multiple digital as well as analog displays. It is typical to have a GPS display configured to the center of the cockpit, while analog instrument readings are spread out panoramically under an airplane's dashboard. The people using these devices often have to divert their attention from the surrounding airspace or wherever they are trying to fly to in order to look around in the cockpit and properly monitor and understand their instrument readings and GPS data.

A Heads-Up-Display (hereon referred to as a HUD) is a technology that solves this issue of diverted attention by being able to project relevant flight information to a pilot in the line of their external forward vision. The data from this display is projected onto a semitransparent display panel, allowing for visualization of flight parameters in the pilot's direct field of view (Skybrary, 2025). With this new capability of enhanced flight parameter view, a pilot does not have to divert their attention multiple times to a flight instrument panel and GPS display. By being able to read flight data in the same perceived plane, the pilot can focus on the surrounding airspace and have visual input in flying the plane while also ready access to this critical and on-demand information.

2.2 Precedent

HUDs are a popular means of projecting information in various types of vehicles and in recent years, the automotive industry has become one of the chief market segments for this kind of technology. As far as past projects are concerned, a UCF team has done a variation of a heads-up display called the "Huddy Buddy". This project aimed to solve a popular issue within driving automobiles, in that smart phone applications are often built to have the smartphone displaying navigational or speed data but are meant to be physically mounted in front of the driver causing complications. The solution to this was for the group to build their own HUD that was integrated with a smartphone to project that data into the same visual plane as the rest of the road (from the driver's perspective) (Growth, et al. 2025). This project helped our team to gain insights regarding what worked for their HUD as well as what the challenges were. These insights helped us to more well-define our portability requirements as well as things like visibility and computing requirements for the system.

In the commercial aviation industry, HUDs are a popular accessory but often come with some limitations. Namely, most HUDs integrated into commercial aircraft come directly from the aircraft manufacturer and directly integrated into the airplane itself. If this is not the case, it is typical that commercial HUDs for aviation are very expensive and require some complex installation work to be successfully integrated into the flight experience. For example, the *SkyDisplay* from MyGoFlight is a HUD that integrates into an aircraft's existing instrument infrastructure (with technologies from companies such as Garmin or HoneyWell). This is a phenomenal piece of equipment but can run at a pretty hefty price tag for the consumer. Installing this system with a common digital Primary Flight Display can cost around \$30,000 and installing it with a system from Aspen Avionics, a

much cheaper option, can still cost upwards of \$15,500. It is for this reason that we aim to develop a lower-cost and more portable unit that fits the needs of a private aviator as opposed to mid-sized commercial or private aircraft (MYGOFLIGHT, 2025).

Existing and Past Projects

There are many different heads up displays that are already in use, especially in cars, but there is a lack of them in planes, particularly in cheaper private planes like the Cessna 172. This is one of the most used planes and it lacks some of the special features large airliners and military level aircraft have. While it does show the altitude and speed that this project will display it presents it through dials spread all across the pilot's cockpit. If all of the most used measures can be displayed into one central spot so the pilot does not need to scan multiple gauges this can improve safety and the pilot's overall ability to focus on flying. While there are some after-market options that are offered, they are very expensive. While a more affordable option is preferred, the reliability of a plane's HUD must be accurate and constantly updated in real time to avoid any safety concerns and is one of the main reasons for its hefty price tag.

Heads up displays (HUDs) are becoming increasingly popular for displaying data due to their low profile and ability to relay important data. Because of this many production cars come with already built in HUDs. While they are able to give important information, they don't take away from the driver's field of view or make the vehicle more unsafe to drive. It really does the opposite by putting all of the data that the driver would look down on the dash for directly in front of their vision of the road, so there is no shift in focus to check speed or other data on the dash like the RPMs.

The newer cars having these features are a good safety feature, but most of the cars on the road do not have this capability, and it really can be added through an aftermarket product. Amazon sells multiple different options that offer the same features. The only requirement is a compatible car outlet for the product. This will give older cars the same features as the newly manufactured ones and provide a good alternative as opposed to buying a new car. The Amazon products have multiple options and are small, compact products with no difficult install. They stick onto somewhere on the dash and either have a flip up glass pane or have a screen saver that goes onto the window that shows a projection.

MOMAN - Heads Up Display for Cars

This is one of the middle-priced HUD for a car. It cost \$49.95 pre-taxes and it is manufactured by MOMAN, a photo, video and audio company. They make many different aftermarket electrical products for automobiles. This specific HUD is mounted to the dash of a vehicle and uses a vertical piece of glass attached to a base to present its data. It is connected using a 5V usb cable or connects to a cigarette lighter. The features are extensive with speed, time, driving distance, date, altitude, compass, driving direction, speeding alarm, and fatigue driving. MOMAN claims to have no delays and functions built in to prevent glare and night and day settings for clear visibility no matter what time of day. MOMAN provides a good example of a product that is an affordable

and reliable option with many different features and a good display, which are all of the desired traits of a good HUD (MOMAN Amazon, 2025).

KUOWEIHUDD - Heads Up Display for Cars

This is one of the cheapest options found on amazon for HUD. It costs \$13.99 before taxes and fees and is manufactured by KUOWEIHUDD. They specifically specialize in making cheap HUDs for cars. This product in specific is only able to connect using a USB cable and it utilizes the windshield to project its image. It does not use a glass panel like the last product and rather just shines the data up onto the glass. With the cheaper price tag comes much fewer features. The only bit of data that this device displays is speed with a maximum speed at 115mph. It uses a GPS to track speed and claims a max of 1.8mph of error. While speed is a good bit of data to display it is the only piece of data and is the reason the HUD is so cheap. If speed is the only thing that is needed, then it would be a good choice (KUOWEIHUDD Amazon, 2025).

HUDWAY - Drive

This HUD is on the higher end of the price range for cars, costing \$329 before taxes and fees. It is manufactured by HUDWAY, which is another company that specializes in making HUDs for cars. With this being the most expensive out of the three car options it does have the most features. While the other displays have a set of data that is displayed, the HUDWAY Drive has the capability to customize the data shown. The screen has a high resolution and has the capability of displaying navigation, phone calls, music, notifications, and OBD-2 data. OBD-2 Data is required for special installation, but it makes the HUD capable of displaying data coming directly from the car and makes it very reliable due to its use of the car's instruments and not other outside devices. It can display speed, RPM, temperature, time, fuel capacity, battery level, and really any other data received from the OBD-2. While this HUD costs multiple times more than any of the other HUDs it could be worth the price for the extra features it offers. It allows the user to display the data they want and with high reliability with the use of the OBD-2 data. The only other downside besides the price is the extra installation time to plug into the OBD-2 (Hudway, 2020).

GARMIN - GHD 2100 Head-Up Display

This HUD is designed for planes and is an optional addition when purchasing an aircraft. It is manufactured by GARMIN, and it is only purchasable through a dealer and according to an article on AIN written by Matt Thurber the option to include this is \$668,700. This makes it a serious financial purchase and could be too expensive an option for some buyers. This display is installed from the top of the cockpit instead of the dash and hangs in the pilot's field of view. It can show flight path markers, flight director, acceleration error, speed error, and other quantitative measures such as elevation and pitch. This is one of the only real options for plane HUDs and its expensive price tag could make it unaffordable. The price isn't unjustified, with the installation being part of the plane and using expensive instruments for very high reliability the price reflects a reliable product. There is little room for errors in flight HUDs with little differences

meaning there could be a crash or rough landing. While GARMINS HUD is expensive there is no need to worry about the information being wrong. With this in mind it is a good product with an expensive but fair price tag (Thurber, 2024).

Military Heads-Up Display

The F-18 is one of many military planes that has been utilizing HUDS for the last 50 years. They were manufactured by multiple different companies going all the way back to the 70s with two of the most notable manufacturers being L-3 and Rockwell. The public has never really gotten access to products from these companies, with most of the technology being contracted out by the government. This is why commercial planes did not catch up and started using their own HUDS until the 80s. All the military technology remains ahead, with the newer HUDs being built directly into the headset instead of having it on a glass panel on the dash. The headset for the F-35 still displays all the same flight data important to the pilot, but now when the pilot can be scanning the skies and not looking forward, the data will still be visible with the use of the headset. Some of the features of the headset are airspeed, altitude, heading, weapon status, and target designation, as well as real-time infrared imagery from the aircraft's Distributed Aperture System (DAS) to give the pilot a 360-degree, "see-through" view of their surroundings. This is by far one of the most advanced HUDs ever made but it is not a realistic option for residential use..

Lessons From These Products / Projects

Upon reviewing the existing products, we have determined that the typical heads-up display for the aviation industry is an expensive one with proprietary software as well as hardware integration into the airplane. These systems often project flight parameters. Our solution would allow for a portable and accessible HUD module that not only enables enhanced visibility of key flight parameters, but also additional visibility in terms of an increased field of view for the pilot. Because of this, we are paying close attention to the increased visibility in meeting parameters such as brightness, screen resolution, and contrast.

2.3 Project Goals & Objectives

The overall goal for this project is to develop a self-contained and portable heads-up display for aircrafts. Currently, there are no affordable options for any aircraft in the residential sector, and we plan to change that. Our product will have a box that houses all sensors and other electrical components as well as lenses for the optical display. There will be a piece of glass that will stick out of the box or be attached at the end that will be used to display the GUI. The idea behind this project is to create a portable, affordable, and self-contained heads-up display to fill a gap in the personal aviation industry.

Listed below are the P.E.G.A.S.U.S. goals and objectives as determined by the engineering team. The goals are used to define the intended outcomes this project aims

to achieve, while objectives specify the concrete deliverables. There are also advanced goals and objectives which are feasible if time permits. Alongside advanced goals and objectives, stretch goals and objectives are listed as items that are beyond the scope of our implementation but would be very beneficial to this project if added.

Goals

Basic Goals:

- Develop a portable standalone Heads-Up-Display (HUD) for aircraft.
- System will display essential flight parameters
- System will incorporate a Graphical User Interface (GUI).
- System will have debugging capabilities for the engineering team.
- Good HUD visibility in variable lighting conditions / polarizations.
- LCD clarity maintained throughout system.
- Portable optical components.

Advanced Goals:

- Live system will increase the pilot's peripheral field of view.
- Incorporate an adaptive lighting system so that HUD can be viewed comfortably in any visibility conditions.
- Optimized eye-box virtual image

Stretch Goals:

- Read, calculate, and display ground and air speed data.
- Implement a GUI navigation feature.
- Overlay pilot's flight path on to their FOV from navigation software.
- Head / Eye tracking for dynamic eye box.
- Variable Focus of system optics.

Objectives

Basic Objectives:

- The P.E.G.A.S.U.S. must be roughly the size of a briefcase.
- House all optical components and sensors in a single housing.
- The device will be fully powered by a 12V outlet inside the plane.
- P.E.G.A.S.U.S. will read and compute sensor data in real time.
- Implement an Inertial Measurement Unit (IMU) to calculate three-dimensional attitude data and compass heading data.
- Implement an altimeter for altitude data reading.
- Incorporate one microcontroller unit (MCU) designated to real-time sensor reading.
- Implement a dedicated MCU for data formatting and GUI display control.
- The GUI will display attitude (orientation in space) data, altitude, and compass heading.

- The MCU will periodically log flight data onto a local storage device for debugging purposes.
- HUD Visibility Above Threshold through polarized and unpolarized lenses.
- LCD resolution still clear at virtual image plane because of optical component optimization.
- Optical Components within physical constraints of package (package dimensions: 304mm x 127mm x 254mm).

Advanced Objectives:

- Cameras will have their own dedicated housing stuck to the pilot's side windows.
- System displays data for >80% of total flight time with restart technology if an unexpected shut off occurs.
- Increase blind-side FOV with use of lateral cameras bringing up FOV to full 210° of human eyesight.
- Adaptive Visibility of System based on ambient lighting.
- Correct for system aberrations and distortion specifically localized within target eye box area.

Stretch Objectives:

- The P.E.G.A.S.U.S. parses GPS data for accurate route-planning through a routing algorithm to allow for a general flight direction for the pilot.
- Display two kinds of route overlays in the form of a full navigation overlay and a minimal navigation overlay to give the pilot more versatility in their routing display options.
- Integrate cameras with a computer-vision model to allow for runway detection and overlay for increased landing safety in poor conditions.
- Mechanically adjust optics to shift virtual image plane based on user tracking.
- Add tunability to collimating optics for varying user focus conditions.

2.4 Features and Functionalities

Project Capabilities

There are four different devices for peripherals used for data acquisition for the display. There will be a camera, altimeter, gyroscope, and GPS. With these four devices an array of different data can be displayed to a pilot in real time to help navigate an aircraft in a safe manner. Each of these four different peripherals serves a different purpose and is important to the design. All of the data will be displayed onto a glass screen and none of the parts will need to be bolted or screwed down. None of the pieces are made to fit a specific type of plane and will be universal for any plane that has the necessary power required. The glass window is located on the dash in front of the pilot.

The screen is meant to be minimalistic and will require two different settings, one where the screen is being used to display the feed from the camera and another where it will

display the numbers measured from the other devices. The screen is small and these two different settings will make it so the screen is able to properly convey information without the pilot straining to see the display. The camera is more ideally used while on the ground for parking or for maneuvering through tighter spaces similar to the functionality of a backup camera. The other setting for the screen is meant for while the plane is in the takeoff, flight and landing stages.

The other peripherals are the more important flight data measures, and they will all be displayed on the flight setting. The hardware will have enough power to be able to rapidly get all measured data and display and update values quickly. The values displayed will be the speed, location, altitude, and the pitch. These are all of the important measures that could be helpful during the flight.

2.5 List of Requirements and Specifications

Listing clear qualitative and quantitative requirements for this project is crucial to its success. By mapping out what system requirements are, designing will be simpler due to understanding what range certain components need to be in for use as a viable option. By defining these measures, we also ensure that the project meets certain practical benchmarks as a useful tool for pilots. Below are the two tables for system and component specifications for use across this project.

2.5.1 Seven Key Engineering Specifications with Quantitative Measures:

Table 1: Key Engineering Specifications

Parameter	Specification	Description	Priority
Accuracy	~10%	Ensures data is accurately displayed t pilot	High
Total System Power Consumption	<21W	Critical for aircraft electrical load margin	High
Display Brightness	>999 nits	Ensures HUD visibility in cockpit daylight	High
System Boot Time	< 1 minute	Pilot needs minimal delay at startup	Med
Attitude Measurement, Processing and Display Time	<200ms	Ensures prompt reading to ensure accurate display of data	Med
IMU Polling Rate	>9.9 Hz	Smooth sensor readings	Med
GUI Refresh Rate	> 30 FPS	Ensures a responsive GUI	Med
Total System Weight (including housing)	<11 lbs	Keeps system portable and minimizes aircraft load	Low
Horizontal Field of View Overlay*	>39°	Provides useful HUD coverage	Low

* - Indicates an advanced goal

2.5.2 Key Component Specifications

Table 2: Key Component Specifications

Component	Specification	Quantitative Target	Notes
ESP32-S3 MCU	Max I ² C/SPI bus rate	I ² C @ 400 kHz, SPI @ 10 MHz	Handles multiple sensors without bottleneck.
ICM-20948	IMU data accuracy	±2000 °/s, 16-bit	Detects fast aircraft IMU changes.
BMP388	Altitude resolution	±0.1 m (24-bit oversampling)	Needed for flight altitude awareness.
GPS (PA1616S)*	Position accuracy	< 3 m CEP, update rate 10 Hz	Sufficient for navigation overlay.
Raspberry Pi 4	Processing power	Quad-core 1.5 GHz, ≥4 GB RAM	Ensures GUI rendering + camera feed overlay.
Pi Camera (160° FOV)	Resolution & FPS	8 MP @ 30 FPS	Provides wide visual for HUD overlay.

* - Indicates an advanced goal

Table 3: Optical Component Specifications

Optical Component	Parameter / Specification / Units
Micro-OLED Display	Size: 8.6mm x 4.8mm Resolution: Full HD (1920 x 1080) Brightness: (>1000 nits) Contrast Ratio (1000:1)
λ/4 Waveplate	Type: Achromatic Wavelengths: 380-700nm Polarization: Linear to Circular Conversion
Collimating Lens	Lens Type: Achromatic Doublet Lens Diameter: >18mm Focal Length: 12mm Material: Optical Grade Glass (BK7, Silica) or Acrylic
Flat Mirror	Size: 40mm square Reflectivity: 90% Material: Optical Reflective Coating (Silver, Aluminum)
Aspherical Mirror	Diameter: 40-50mm Thickness: 10-20mm Effective Focal Length: 50-100mm

Magnifying Lens	Lens Type: Aspheric Singlet Lens Diameter: >25mm Focal Length: 18-24mm Material: Optical Grade Glass (BK7, Silica) or Acrylic (anti-reflective)
Electrochromic Glass Element	Size: 6.5'' x 6'' Thickness: >5mm Transparency: Variable based on ambient light
Optical Combiner Film	Size: 6.5'' x 6'' Thickness: 0.1-0.5mm Transparency: 40-60% Reflectivity: 40-70% Material: Holographic Optical Film or Dielectric Coated Film

2.6 Project Hardware Block Diagram

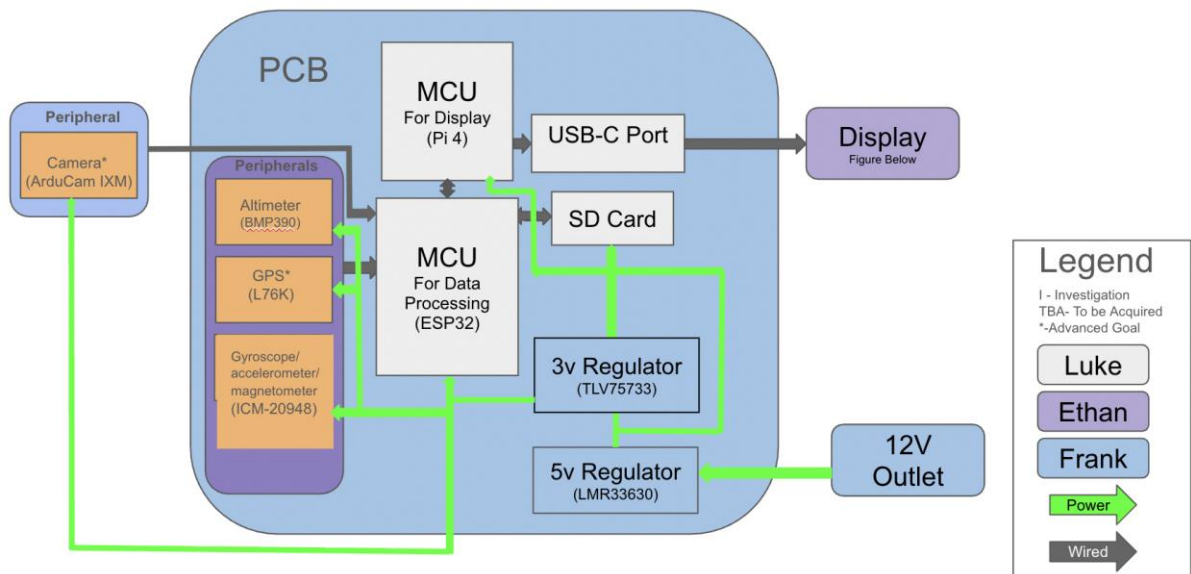


Figure 1: Project Hardware Block Diagram

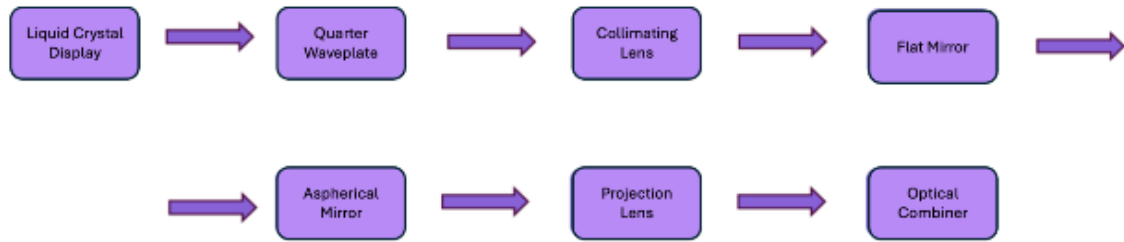


Figure 2: Display Optics Hardware Block Diagram

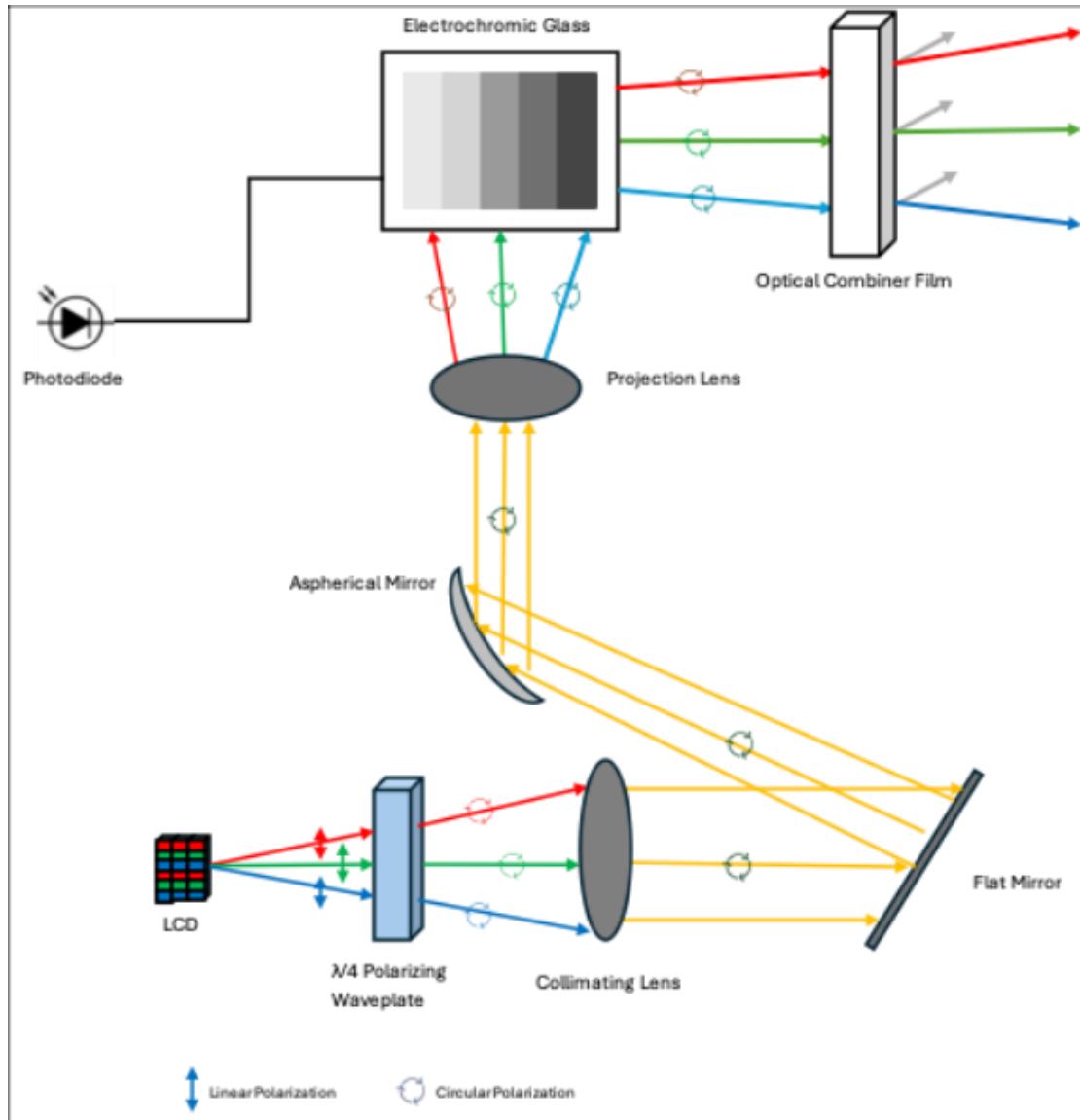


Figure 3: Optical Design Schematic

2.7 Project Software Diagram / Flowchart

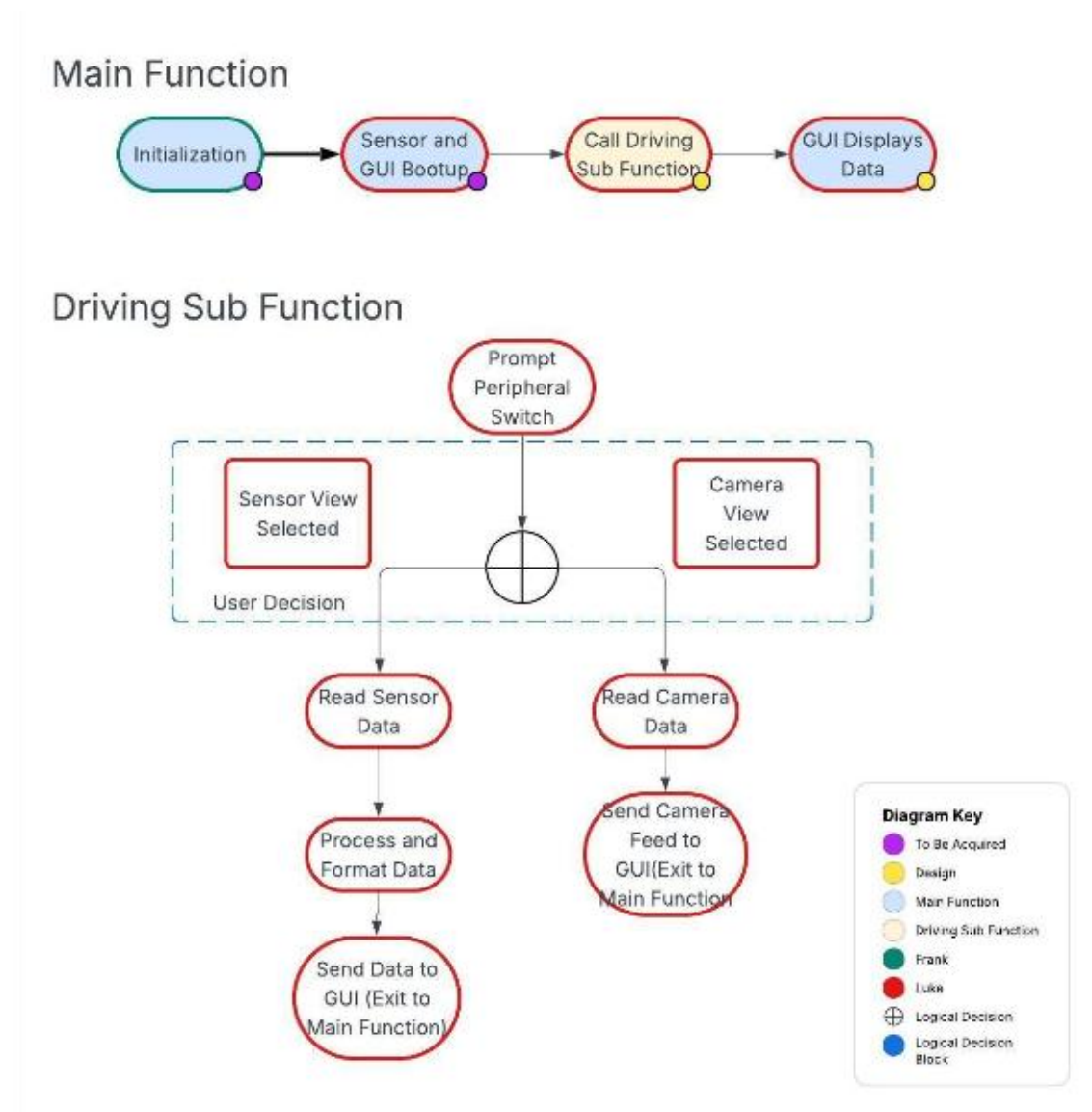


Figure 4: Software Diagram

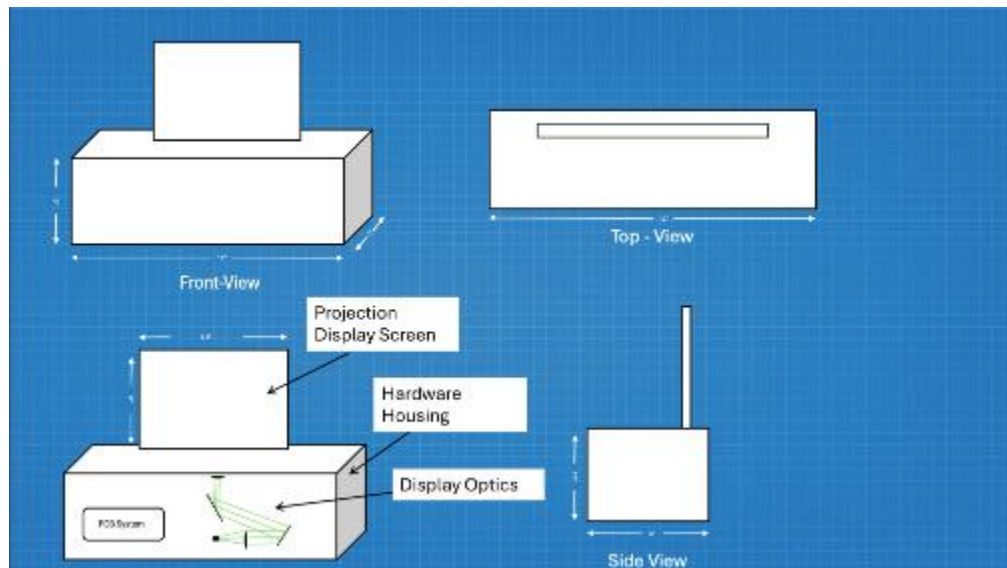


Figure 5: Prototype Illustration

2.8 Prototype Illustration & GUI



Figure 6: Graphical User Interface

2.9 House of Quality

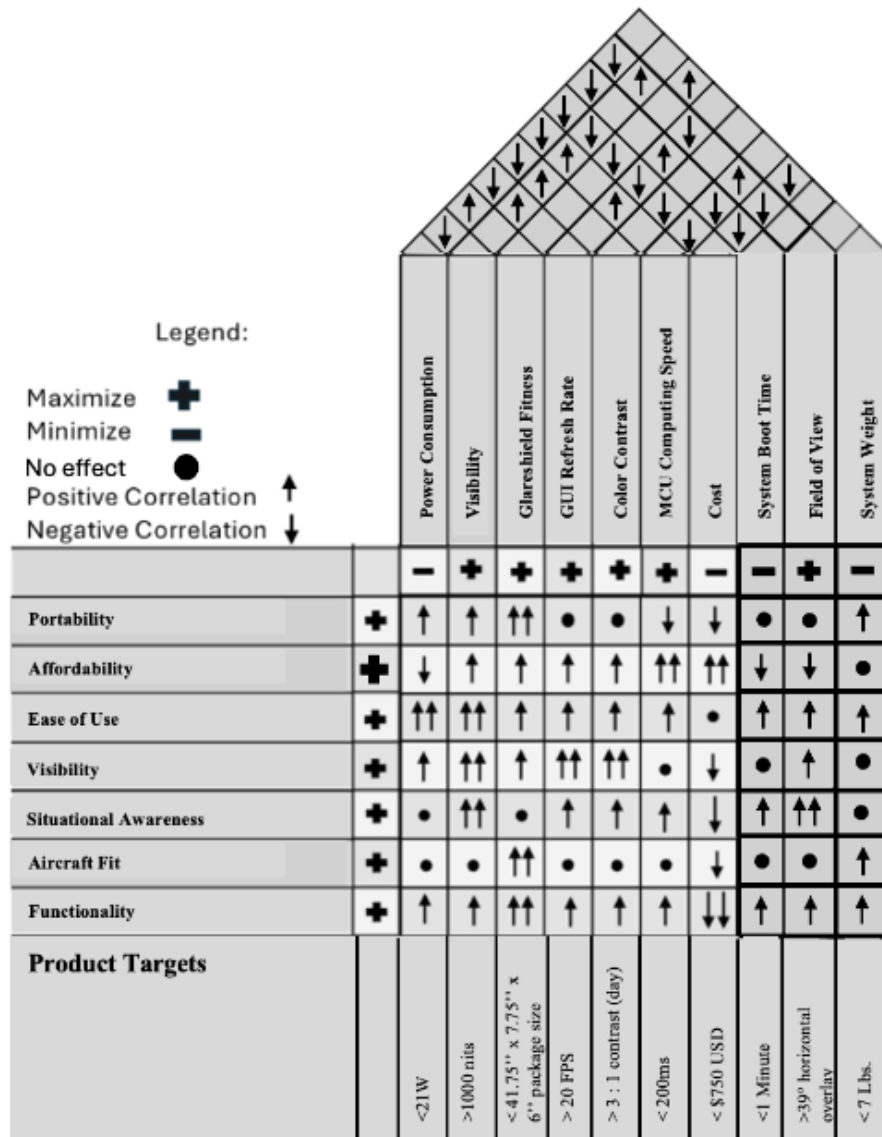


Figure 7: House of Quality

2.10 Test Plan & Live Demonstration

2.10.1 Test Plan

The testing of a heads-up display for an airplane requires the use of an airplane for some data pieces, but many pieces can be tested on the ground or in a car. Our group plans to test the ground speed, compass heading, pitch heading, and camera view on the ground. We plan to test ground speed by driving in a car down the road and comparing it with the speedometer in our vehicle. The compass heading is also easy to test simply by comparing the display to a compass. Pitch heading will also be easy to test by comparing it to a level and rotating the device around the respective axis. Camera view does not require being in an aircraft; we simply ensure the camera feed is sent to the HUD through the data lines. For altimeter testing, we plan to use a Cessna 172 if available and if not available, we will find small hills and go to the top and bottom of these hills for altitude readings if needed. Altitude readings could also be done by going to the roof of a parking garage and comparing them to ground readings or more accurate altimeters [16].

2.10.2 Live Demonstration

For our basic goals of three-dimensional orientation data, as mentioned in the test plan we are able to pick the HUD up and rotate it around each axis to demonstrate the changes. We can even bring a level to compare accuracy. For the goal of HUD visibility in all weather conditions, we can open and close the blinds in our demo room or go from inside to outside to display that the HUD is bright enough. The live camera feed is very demonstratable, simply select that view and it will appear on the HUD. The advanced goals may require a field trip by walking around the building or outside or up and down an elevator. For altitude, we can go up and down an elevator to the top and bottom floors of buildings to demonstrate that feature. For ground speed, we can walk around the building or if noise interferes with that, we can walk around outside the building to demo that. While we cannot bring our reviewers into an airplane, we can demonstrate working components and software to them by doing these basic tests.

Ch 3: Research and Investigation

3.1 Optics Hardware

3.1.1 Display Architectures

In order to project our desired visual output to a pilot, our project and desired optical design dictate that we incorporate a display to be projected throughout the system. During the discovery process, several display architectures presented themselves as potentially feasible options. These were notably a standard liquid crystal display (LCD), OLED displays, and compact LED displays (Micro/Mini LEDs). Each one of these options provides a unique set of advantages in areas such as cost, resolution, power consumption, output brightness, and more. This being said, the display architectures that we will review do come with some tradeoffs.

3.1.1.1 Liquid Crystal Displays (LCD)

Liquid Crystal Displays are a prominent optoelectronic display technology that has been around for decades. LCDs take advantage of liquid crystals that are placed between various polarizing filters. These crystals do not emit their own light, but instead the system takes advantage of a backlight, typically an LED. To produce images, the crystals modulate the backlight by changing their orientation as a response to an input electric current. (citation 17). To add the complexity of colors to images, color filters are typically added to the system. Each pixel on an LCD is made up of three sub-pixels that include a red, blue, and green filter respectively (citation 18).

Liquid crystals are substances that achieve the fluidity and physical properties of liquids but maintain a crystalline molecular structure. When an electric current is applied to this structure, the orientation of the full structure changes and because of this, the way that the structure interacts with incoming and outgoing light also changes. By modulating the current that is driving these liquid crystals, we can also control the amount of light that passes through them, effectively allowing for a degree of freedom with respect to dimness.

The backlight is the fundamental lighting element that allows for visibility of the LCD system. This process is initiated by a series of LED lamps that are diffused across the screen by a diffuser panel. This diffused light is what will pass through the liquid crystal structure as well as a set of polarizing filters to eventually produce the desired images at the output of the system. (citation 18). To create pixels to form more precise and color-accurate images, a combination of red, blue, and green sub-pixels are used. These sub-pixels are controlled by an array of thin-film-transistors that are modulated and allow for varying levels of light to pass through based on their on or off states. The alignment of the liquid crystalline layer, which is determined by incoming voltages applied to that layer, determines the amount of light reaching each color-filtered sub-pixel. This allows for a full set of colors to be produced by the system as a combination of red, green, and blue (citation 18).

The three most prominent types of liquid crystal display technology are twisted nematic (TN), in-plane-switching (IPS), and vertical alignment (VA).

TN LCDs were the first commercially successful LCD technology. TN displays use a 90-degree twist of nematic liquid crystal fluid to modulate the amount of light passing through the system. This amount of light is based on changes in applied voltage to the liquid crystal fluid. This voltage, depending on what it is, will change the “twist” or nematic orientation of the fluid which is what results in varying levels of light transmission through the system. There are varying subsets of TN displays such as hyper TN and super TN that apply the same principle but instead of modulating at discrete 90-degree intervals, they modulate nematic phases at varying angles between 90 and 180 as well as 210 and 270 respectively (citation 19).

Because of their prevalence, TN LCDs are typically the most cost-effective liquid crystal display option. In addition, these screens have the highest refresh rate and response times. That being said, these systems suffer from limited viewing angles and limited brightness.

Average viewing angles for TN LCDs vary from 45-65 degrees and TN screens are typically not bright enough for outdoor applications (citation 20).

Vertically Aligned LCDs are different to TN architectures as the liquid crystals involved are aligned vertically with respect to the display surface as opposed to being aligned parallel to the surface. This means that when a voltage is applied, the liquid crystal molecules lay flat which provides the distinct advantage of having a dark black-state with minimal leakage of light. This then allows for each image element to have assigned to it its respective color or black (citation 19).

Displays with vertically aligned liquid crystal architectures allow for wider viewing angles, better color contrast, and better color reproduction than traditional TN LCDs. That being said, they typically suffer from limited brightness and higher power consumption (citation 20).

In-Plane-Switching LCD systems work by applying an electrical signal to only a single substrate and an electrode as opposed to electrodes on multiple substrates. Image quality is improved as the liquid crystal inside of the display is being directly acted upon. Because of this, the viewing angle of this type of LCD is much wider, and the color clarity at wider viewing angles is maintained and uninterrupted, unlike traditional LCDs (citation 19).

IPS Liquid Crystal Displays have in recent years become more cost-effective than traditional TN LCDs. They also offer the widest viewing angles of any LCD, the most accurate colors, and the brightest backlights of any LCDs. That being said, IPS systems have much slower refresh rates and response times than traditional TN LCD screens (citation 20).

Overall, LCDs are low-cost, lightweight, and energy efficient systems. Because our HUD system is required to be compact, low-cost, and energy efficient in order to successfully operate in an aviation environment, these are distinct advantages to our team. That being said, it is evident from discovery that LCDs have relatively slow response and refresh times, limited viewing angles, lower contrast from other displays, lower resolution than other displays, as well as the added need for an LED backlight. Because no LCD solution is able to solve all of these problems in one system, this proves to be a distinct disadvantage when it comes to our project. Our HUD needs to have a rapid response and refresh rate as well as a display that has an initial high resolution in order to accurately display the wide variety of information to the user.

3.1.1.2 Organic Light Emitting Diode (OLED) Displays

Organic LEDs are a more novel display technology than traditional liquid crystal schemes. OLEDs are solid state devices that consist of thin, carbon-based semiconductor layers that exhibit spontaneous emission when driven by electricity in the form of electrical contact to transparent electrodes. The intensity of the light that is emitted is modulated by the incoming electrical current into the transparent electrodes. In addition, the color emitted by the OLED is determined by the specific type of material used to emit

that light, whether it be structural or pigmented (citation 21). Three prominent OLED technologies are Passive-Matrix OLEDs, Active-Matrix OLEDs, and Micro-OLEDs.

Passive-Matrix OLEDs, or PMOLEDs, consist of strips of cathodes as well as carbon-based layers and strips of anodes. These strips are orthogonal to each other on the display, and the points at which the cathodes and anodes meet are what make up the pixels of the system. A circuit applies the input current to various selections of strips on the display, and this is what determines what specific locations of pixels emit or don't emit light. In this system, pixel brightness is a function of input current (citation 22). PMOLEDs are best suited for smaller screen applications such as those included in portable and compact devices requiring screens. That being said, these systems typically consume more power than other OLED schemes.

Active Matrix OLEDs, or AMOLEDs, consist of full layers of cathodes and anodes placed on a display. That being said, the distinction is that the anode layer of the system overlays a thin-film-transistor array. This TFT matrix is what drives the pixel locations that get activated in order to form an image for the end user. AMOLEDs have more optimized electronic circuits than PMOLEDs with respect to power consumption. This is because the TFT matrix requires less power than individually powering various strips of cathodes and anodes (citation 22). These systems typically have faster refresh rates that is better for video and moving image applications. This is why they are often used in greater-size and less-compact technologies such as computer monitors, televisions, and even electronic billboards.

Micro-OLED is a display architecture that is made for hyper-compact applications, such as AR/VR Headsets, Electronic viewfinders in cameras, and smart glasses. These displays typically consist of a layer of carbon-based light emitters on a small and thin silicon or glass substrate. Because typical AMOLED and PMOLED systems are printed on glass substrates, the pixel density in these purely glass-based systems is limited which in turn limits resolution. When incorporating TFT matrices for AMOLEDs and strip arrays for PMOLEDs onto a silicon substrate, the TFT circuitry becomes much more precise in terms of being able to localize pixels. This enables a much higher pixel density in these systems, which allows for much higher resolutions at smaller sizes (citation 23). In addition, because of this higher resolution, these systems offer exceptional sharpness, and high contrast. In addition, the optimized TFT circuitry offers lower power consumption for these systems.

Compared to the older technology of liquid crystal displays, OLED displays have improved image qualities, better contrast in images, increased brightness, a wider and more vibrant color range, as well as lower power consumption and increased refresh rates. These system also have the notable advantage of being able to be incorporated in more complicated geometries, such as curved and foldable displays or wearable displays. Compact OLEDs also have orders of magnitude better resolution than traditional LCD displays (citation 24) . OLEDs are generally more expensive than LCD displays because of the complexities and less mature nature of the manufacturing infrastructure of these systems.

3.1.1.3 Micro Light-Emitting Diode (μ LED) Displays

Micro-LEDs are another more novel display architecture, being invented and popularized in the early 2000s. This technology is newer, and thus the infrastructure and manufacturing processes to make these devices are much younger than their LCD and OLED counterparts. Micro-LEDs, or μ LEDs, are flat and square or rectangular light sources that can be placed into arrays or matrices in a manner that enables them to exhibit very high contrast. Each single light-emitting diode in a μ LED array or display acts as its own individual pixel as they each have the distinct ability to display red, green, or blue colors as well as combinations of all three (citation 25). This omits the necessity for the implementation of things like color filters as well as different pigmented layers on top of the lighting layer. Because of their compact and precise nature, Micro-LED displays are promising for applications such as near-field displays in wearables, ultra-large televisions, and thin-panel displays.

μ LEDs work differently than traditional liquid crystal displays as LCDs require a backlight, a polarizer, a series of films, and a liquid crystal layer that is excited in different orientations to emit certain constructions of images. With a micro-LED, each individual pixel, or light-emitting diode, can emit its own light, thus creating much more crisp and precise images with the added advantage of having more localized light control as well as contrast. Micro-LEDs are manufactured using what is called a 'pick-and-place' transfer method. This method involves manufacturing the different colored LEDs directly onto a semiconductor wafer, combining this wafer with a backplane containing the electronic infrastructure/circuitry to support the light modulation/generation, and then separating these individual devices and placing them directly onto a display.

Micro-LEDs and OLEDs are similar technologies, with the main distinction being that μ LEDs are manufactured from inorganic materials and compounds as well as their smaller size. This non-reliance on inorganic materials gives them performance advantages as well. Also, Micro-LEDs are manufactured with the pick-and-place, wafer-backplane method. That being said, μ LEDs prove to be superior in various respects as opposed to their organic counterparts. Micro-LED displays have superior power efficiency, contrast, color fidelity, response time, resolution, sunlight visibility, and size flexibility. That being said, the viewing angles are still high, just like with OLED displays, and the fact that only lit pixels are the ones that are drawing power from the system remains consistent (citation 26). Micro-LEDs do, however, have their drawbacks. Because of the fact that the technology is not as mature as LCDs and OLEDs, the infrastructure to support the mass commercialization of μ LEDs is also lacking. This means that the supply of these devices is still limited, and the costs associated with buying them are still high. These devices are also not known to be as physically flexible as OLEDs primarily because of the fact that most of the flexible display market relies on OLED technologies to build their products.

Table 4: Display Architecture Comparison

Architecture	Resolution at smaller sizes	Power Efficiency	Cost	Sunlight Visibility / Brightness
--------------	-----------------------------	------------------	------	----------------------------------

LED-Backlit LCD	Low	Low	Low	Medium
OLED	High	Medium	Medium	Low/Medium
Micro-LED	Very High	High	High	High

3.1.1.4 Display Architecture Technology Selection

After careful consideration of all of the technologies available to us for our display scheme, it is clear to us that the OLED, specifically the micro-OLED is the superior choice for our HUD. This is for a couple of reasons: First of all, we can get the highest resolution out of one of these displays without having to pay an outrageous cost. In addition, with a micro-OLED, we have increased resolution at compact sizes, and can tailor our optical design as well as the optics in our system to this compact design. This helps in making the optics smaller, and thus cheaper, and also adhering to our keeping all of our components contained within our package.

3.1.1.5 OLED Display Part Selection

After selecting the micro-OLED display for use in our HUD, we can now begin to understand what the best commercially available displays are for our system. In order to do this, we must consider a number of criteria, chief among them are:

Cost: As low as possible

Size: Compact (0.39" diagonal)

Resolution: The highest possible (720p>)

Alibaba AMOLED MicroDisplay 1920 x 1080 (HXO-19201080FM)

This option from Alibaba comes in at around \$84 and comes equipped with a 0.39" AMOLED screen. In addition, it boasts a high resolution at 1920x1080p and this resolution will be suitable for our optical design. It is important that we have a high resolution like this as it is possible to inherently lose some resolution throughout the optical system. The outline dimensions of this particular display are 11.1 x 9.5 x 1.35mm and because of this, we can magnify them effectively with compact optics.

SONY High Brightness AMOLED Small Micro Display (ECX334C)

This transmissive AMOLED micro display from SONY boasts a higher brightness than its previously mentioned counterpart and comes in at a price tag of around \$117. In addition, it meets our size requirements perfectly at being a 0.39" diagonal screen. That being said, the resolution for this display is 1024 x 768p, but it also does have a brightness of 1000nits which would be perfect for our system.

DisplayModule Micro OLED Display (DM-OLED039-694)

This option from DisplayModule is a bit more expensive at \$139, but is an AMOLED. In addition, this display supports much higher resolution at 1920 x 1080 full HD and has

modes for high contrast and multiple interfaces (namely I2C and MIPI). In additionm the typical brightness for this module is at about 1500nits or 50% more than the previous module. It also boasts a high contrast ratio of >10000:1.

Table 5: OLED Comparison

Display	Cost	Resolution	Brightness	Size
Alibaba HXO-19201080F	\$84	1920 x 1080	N/A	0.39” diagonal
SONY ECX334C	\$117	1024 x 768	1000 nits	0.39” diagonal
DisplayModule DM-OLED039-694	\$139	1920 x 1080	1500 nits	0.39” diagonal

3.1.1.6 OLED Display Product Selection

After reviewing the three options previously mentioned, it is clear that the best OLED display for our purposes here is the DisplayModule Micro-OLED (DM-OLED039-694). This is for a variety of reasons. First of all, it boasts the highest resolution and in addition to that, it also boasts the highest listed britghness of all three products. Because of this as well as the fact that it has superior integration with the user interface using both I2C and MIPI interfaces, it presents itself as the superior of the three choices. That being said, it still is an expensive choice but the quality of this system justifies the choice.

3.1.2 Polarization Schemes

3.1.2.1 Importance of Polarization

Once the light output from our user interface is successfully emitted from our display, our system dictates that we must polarize it. This is because one of our goals includes being able to view the system from various polarized perspectives. The practical application of this being that the aviator is able to view this display through polarized sunglasses, a popular wearable with civil aviators. The ability to view the output of the system through various polarizations will maintain situational awareness in flexible viewing environments.

Polarization is an inherent property of electromagnetic waves that affect their performance in optical systems, especially display technologies. This property of polarization describes the behavior of the oscillating electric and magnetic fields in these EM waves and is typically described in three distinct types: linear, circular, and elliptical. Linear polarization dictates that the electric field component is confined to a single plane along the propagation path. Circular polarization dictates that the electric field of the EM wave consists of two perpendicular linear components that have a $\pi/2$ phase difference, resulting in a circular rotation of the e-field along the propagation path. Elliptical polarization dictates that the electric light field describes an ellipse with a phase difference that is not $\pi/2$ (citation 27).

When discussing light from the various previously mentioned display architectures, it is important to note that the light from these displays are already linearly polarized. When you view linearly polarized light through a polarizing filter, such as a set of polarized sunglasses, the polarizing elements interfere with each other and appear to have the effect of “cancelling each other out”. This is why monitors or televisions often appear darker or even completely blacked out when viewing them through these types of sunglasses. Because of this fact, we must ensure that our system display output is circularly polarized. When viewed through a linearly polarizing filter, there will be no interference of polarizations and light from the display will be maintained and visible to the user.

In order to convert linearly polarized light to a circularly polarized format in order to be viewed in various visibilities/polarizations, we must utilize what is known as a quarter-wave-plate, or a $\lambda/4$ waveplate. These waveplates take incoming linearly polarized light and retard their velocities of one of the polarization components (in the x or y direction) one quarter of a wave out of phase from the other component. This modification in the electromagnetic wave results in circular polarization (citation 28). These systems are often used to improve the viewing capabilities of emissive displays, especially in bright light environments.

3.1.2.2 Zero-Order Waveplates

With zero-order waveplates, the total retardation of the light is the desired value with no excess, as opposed to the leakage that a multi-order waveplate has. This is to say that the net retardation equals exactly one-half or one-quarter of the wavelength, but the thickness of the optic involved is only just greater than the retardation needed for the application (citation 29). Because of this fact, these waveplates can be made with much less thick materials and can be made to be much more stable and robust systems. In addition, these waveplates are much less susceptible to deteriorations in performance due to temperature variations as well as wavelength fluctuations. It is for this reason that they are often employed in precision applications such as high powered lasers as well as advanced polarization techniques. This precision/consistency across temperature and wavelength ranges makes zero order waveplates much more expensive than their multi-order counterparts.

3.1.2.3 Multi-Order Waveplates

With multi-order waveplates, these systems are engineered to introduce an amount of retardation equal to the desired parameter plus a discrete integer value of full wavelengths. This process results in a shift in phase of the output light as well as a thicker system and substrate but also results in the same or comparable level of sought after polarization. In addition, because of this thicker structure and the higher-order nature of these systems, they can exhibit higher sensitivity to broad changes in wavelengths. They can also introduce higher system drift over a broad range of spectra (citation 29). That being said, the greatest advantage to using a multi-order waveplate is the fact that they are much more cost effective than their zero-order counterparts. Applications using these polarization schemes may have to sacrifice the range of wavelengths that are visible in the system, even potentially opting to polarize discrete wavelengths as opposed to broad

ranges. That being said, they enjoy the added benefit of dramatically lower overall costs to their systems.

Table 6: Waveplate Comparison

Waveplate Type	Thickness	Polarization Precision	Cost	Wavelength Sensitivity
Zero-Order	Thin	High	Very High	Low
Multi-Order	Thick	Moderate	Medium	High

3.1.2.4 Polarizer Technology Selection

After careful consideration of the various parts and options presented to us, it is clear that the better option of the two is to incorporate the use of a multi-order quarter ($\lambda/4$) wave plate. The reasoning is twofold. First of all, these systems are highly cost-effective and more widely available than their zero-order counterparts. In addition, to achieve circular polarization in our system, we must employ the use of the $\lambda/4$ wave plate. Also, the fact that the polarization precision is not as high over a range of wavelengths, but is still relatively good over discrete wavelengths is good enough for the scope of this project. Had the price differential not been many hundreds of dollars for this single system, it may have been more feasible to include in our low-cost heads-up display.

3.1.2.5 Polarization Part Selection

Now that we have decided to go with the option of a multi-order waveplate, an analysis of the various commercially available products and options can be sufficiently made. When choosing a waveplates specifically, the following criteria are taken into consideration:

Cost: As low as possible

Wavelength Sensitivity: 532nm selective wavelength at minimum

Bandwidth: High bandwidth across visible spectrum

Environmental Factors: Minimal sensitivity to temperature, angular stresses, etc.

Size: Must fit our display output dimensions

Edmund Optics $\lambda/4$ 25mm retarder film (WP140HE):

This quarter-waveplate polarizing film is an inexpensive and yet effective solution to our design requirement of being able to circularly polarize the light output of our display. It is cost effective, at \$17 for a 25mm film. In addition, the thickness is low at 50 microns and the product has a broad bandwidth of 400-700nm of light. The biggest consideration is

that if we chose this product, we would have to design and incorporate some kind of mounting system to include it so that the system could be placed as needed according to our optical design. All things considered, this is a clean and relatively inexpensive solution to the necessity for polarizing our light output.

Band Optics $\lambda/4$ Waveplate (KT2490002)

This multi-order quarter-waveplate is made out of N-BK7 glass and offers circular polarization at a low cost. Coming in at \$19.50, this optical element is a 12.7 (or 0.5”) plate that comes pre-mounted into a fixture. This provides a distinct advantage as there is no additional hardware or mounting infrastructure that needs to be developed for a product like this. Integration would be easy. That being said, this product does have the major drawback of only offering $\lambda/4$ circular polarization at one discrete wavelength (532nm). This is not a huge problem, but is less than ideal as for our system, we would prefer a wider bandwidth of polarization.

Band Optics $\lambda/4$ Waveplate (KT2490005)

This multi-order quarter-waveplate is similar to the one mentioned previously. That being said, this product offers a few distinct advantages to the aforementioned waveplates. For \$27.50, this waveplates offers a larger working diameter of 25.4mm, allowing for greater flexibility in polarizing screen or display output. In addition, it offers customizable wavelength ranges between 200nm and 2100nm. It also comes pre-mounted and no additional mounting infrastructure would have to be developed or built around it.

Table 7: Polarizer Product Comparison

Polarizer	Cost	Wavelength Sensitivity	Bandwidth	Integration Challenges
WP140HE	Low (\$17)	Low	400-700nm	Moderate
KT2490002	Low (\$19)	High	532nm	Minimal
KT2490005	Moderate (\$27)	High	200-2100nm Selective	Minimal

$\lambda/4$ Quarter Waveplate Product Selection

After careful consideration of the various different commercially available products reviewed, the best option has presented itself in the form of the Edmund Optics $\lambda/4$ 25mm retarder film (WP140HE). This decision was made on the basis that this system has the highest wavelength range of any of the products reviewed and does not limit itself to only polarizing one discrete wavelength. Although we will have to make some mounting infrastructure to house this product, we have increased flexibility in how we do so, allowing us to better accommodate the needs of our HUD system.

3.1.3 Collimating Lens System

3.1.3.1 Collimators in Projection Systems

The end user for this system is a human being viewing our projected HUD through a screen in free space. Thus, when engineering around human eyes, one needs to take into account their limitations. The human eye can only focus on a single point at a time and it is because of this that HUD systems need to provide collimated images. That is, HUD systems need to provide images in which the rays of light appear parallel to each other out to infinity. When an image is not collimated, the rays of light are not parallel and they either converge or diverge. When this is the case, images appear to be in focus only at discrete points in a display system. This is not practical for heads up displays, especially in vehicles as these displays need to have the ability to be viewed in an “infinite” range of planes (or an infinite plane). In order to solve this problem, collimators are used in projection systems in order to fix the light at infinity. In addition, they provide the added benefit of being able to manipulate that light with minimal deterioration. For example, one can reflect, refract, and magnify collimated images with much greater ease than non-collimated images (citation 30). Because these systems allow for the human eye to not have to constantly refocus when viewing a collimated display, they are much better for applications where situational awareness and speed of awareness are the primary concerns, such as aviation information projection.

3.1.3.2 Types of Collimating Lenses

Some popular types of collimating lenses include aspheric, achromatic, reflective lenses, and refractive lenses. Aspheric lenses have non-spherical geometries, which enables better reduction of aberrations as well as better control over collimation compared to their spherical counterparts. Achromatic lenses are specifically designed to correct for chromatic aberrations, a type of distortion that occurs when lenses cannot focus various wavelengths at the same point. Reflective Collimators work by reflecting light rays off of a series of curved mirrors in a manner in which they leave the system parallel to each other. This is similar to cavity operation in laser applications. Refractive lenses are the most common variation of collimator and simply bend light as it passes through the lens. They can bend converging or diverging light rays to ensure a parallel ray output (citation 31).

Table 8: Collimator Technology Comparison

Collimator Type	Aberration Correction	Wavelength Range	Cost	Quality of Collimation
Aspheric	Spherical	Limited Narrow Band	Lower (for discrete λ 's)	High
Achromatic	Spherical & Chromatic	Wide (Broad-Band)	Medium (Higher for more precision correction)	Very High
Reflective	Spherical & Chromatic	Very Wide (Ultra Broad-Band)	High	Very High
Refractive	None	Limited Narrow Band	Low (widely available)	Medium

3.1.3.3 Collimating Lens Technology Selection

After careful consideration of the various collimating lens types considered, the achromatic collimator presented itself as the best option for this type of project. This is because of the fact that the achromat can collimate over a wide and broad band of varying wavelengths and this will be highly important as we aim to project colors across the visible spectrum. In addition, this helps correct for spherical and chromatic aberrations which would present themselves in a much more dramatic manner at the end of the system if they were not accounted for and corrected at the beginning of the system. In addition, this system would work better than a reflective collimator as it is a much more cost effective and practical solution. There is also no cumbersome alignment associated with this type of system.

3.1.3.4 Collimating Lens Part Selection

Now that a decision has been made regarding what type of collimation technology to incorporate into our HUD system, an investigation into the various commercially available product options can be made. When assessing collimators, it is important to be considerate of the following criteria:

Cost: As low as possible

Focal Length: Small enough to fit into package relative to other components

Diameter: Large enough to collimate full display output

Material: BK7 Glass, Fused Silica

Wavelength Range: Visible (400-700nm)

OptoSigma Achromatic Doublet Lens (DLB-20-40PM)

This collimating lens from OptoSigma comes in at a price tag of around \$150. It boasts the correct visible wavelength range of collimation of 400-700nm that is required for this project. In addition, it has a 20mm diameter and a 40mm focal length. With this diameter, we would be able to collimate the full size of our display output and with this focal length, integrating this system into our package would be feasible based on the relatively small length relative to our physical package. The primary drawbacks of this system are the high relative cost for such a small component as well as the fact that the system does not come pre-mounted. Because of this, our group would have to design or source a mounting system, and this would present extra engineering challenges. The lens is currently in stock according to OptoSigma.

Edmund Optics Achromatic Doublet Lens (#49-955)

At \$113.40, this collimating lens from Edmund Optics presents itself at a slightly cheaper price than its counterpart. In addition, it boasts a wider collimation wavelength range of 400-1000nm and a diameter and focal length of 18mm and 22.5mm respectively. While the increased collimation wavelength range is an advantage, our project requires just the visible band of light to be collimated. With such a small focal length, this lens presents the clear advantage of being able to be integrated into our package while taking up much less space than the previous achromatic doublet. The diameter is slightly smaller than its counterpart but should still be enough to encompass the entirety of our polarized screen display output. This doublet also has the advantage of having an anti-reflection coating already on the surface of it. That being said, the cost is still high relative to what we want for our components, and the lens does not come pre-mounted, so we would have to design and build a custom mount for it. The lens is also currently in stock according to Edmund Optics.

OptoSigma Achromatic Doublet Lens (S-DLB-20-50PM)

This achromatic collimating lens, also from OptoSigma, has a significantly cheaper price tag of \$61 but comes with a few distinct differences. First off, the wavelength range is 486nm center λ , 546nm center λ , and 656nm center λ . Because of this, the collimation won't be as wide ranging as the previous doublets mentioned as it is not over a range but focuses on discrete wavelengths (mainly blue, green, and red respectively). That being said, the lens does boast a diameter of 20mm and a focal length of 50.2mm. With these dimensions, it would successfully collimate the entirety of our display as well as be compact enough in terms of focal length to integrate into our system without being too large. The lens is made out of BK7 glass which is ideal for our HUD application. The lens is not immediately available as of this moment, but is available within a couple of weeks according to OptoSigma.

Table 9: Collimator Product Comparison

Collimating Lens	Cost	Collimation Wavelengths	Focal Length	Diameter
OptoSigma DLB-20-40PM	\$150	(400nm-700nm) range	40mm	20mm
Edmund Optics #49-955	\$113	(400-1000nm) range	22.5mm	18mm
OptoSigma S-DLB-20-50PM	\$61	(486nm, 546nm and 656nm) discrete center λ 's	50mm	20mm

3.1.3.5 Collimating Lens Product Selection

After consideration of the aforementioned three achromatic doublet options for our heads-up-display system, we have decided to go with the Edmund Optics #49-955 achromatic doublet lens. The reasoning for this is threefold: First of all, the product offers

a wide range of collimation wavelengths at 400-1000nm wavelengths. Our application requires a wide range of wavelengths because our display outputs an entire spectrum of colors as opposed to a simple set of discrete colors. In addition, this product is advantageous as it offers the physical dimensions in diameter and focal length that we require for our display collimation and packaging requirements. Also, it offers similar advantages to the OptoSigma doublet while being significantly cheaper. We will still have to design a custom mounting system for this lens, but it remains the superior option.

3.1.4 Reflective Mirrors

After the polarized output from our Display system is collimated with a collimating lens, we must employ reflective optics to manipulate the image so that it can adequately pass through our system and successfully to the end user. This is done with reflective mirrors in a fold configuration. The main reasoning behind this is that our image cannot travel in a straight line and is also constrained by the physical dimensions of our package. In addition, mirrors will allow us to take our collimated light rays and manipulate them with aspheres and a magnifier system. This will allow for aberration correction as well as the magnification in order to properly project a crisp image onto the optical combining system.

3.1.4.1 Types of Mirrors

There are three primary types of mirror geometries relevant to a project like this: plane (or flat) mirrors, concave mirrors, and convex mirrors. Plane mirrors, also known as flat mirrors, are the most common type of mirror used in everyday life. These mirror systems reflect virtual images to their normal proportions, functioning as a reflective guide for light without providing any major distortions, shrinkages, or enlargements. In addition, these mirrors have no focal point making them highly flexible in their applications as they have no hard physical limit on where their output light can be manipulated from. Concave mirrors are spherical mirrors that curve inward. These mirrors create the illusion of largeness and focus collimated light rays inward to a single focal point. Their images can be real or virtual, inverted or upright, and magnified or reduced depending on the distance of the object. Because these systems have a real focal point, the uses of their proceeding optics are restricted by this constraint. Convex mirrors are also spherical mirrors that distort reflected images and make them smaller. They tend to diverge collimated light rays and their images are always virtual, upright, and smaller than the object whose light they are reflecting (citation 32). In addition, these systems have the constraint of a virtual focal point behind the mirror where light appears to diverge from. This can make optical designs with these systems more difficult. Flat mirrors are typically the least expensive and most widely available. Concave mirrors are typically the most expensive and convex mirrors are moderately priced relative to the other two

Table 10: Mirror Technology Comparison

Mirror Type	Reflections	Focal Point Location	Cost
-------------	-------------	----------------------	------

Flat	Uniform	No Focal Point	Low
Concave	Converging Rays to single focus	In front of mirror where convergence occurs	High
Convex	Diverging Rays from single focus	Behind mirror where divergence originates	Medium

3.1.4.2 Reflective Mirror Technology Selection

After consideration all of the options of various mirror types for the reflective optics of this project, the clear optimal choice presented to us was to employ the use of a flat or plane mirror system. Because of the fact that this hardware is widely available, the reflections dictate the fact that the image maintains its original proportions throughout the system, and there is no focal point constraint, we see clear advantages to using this kind of system. The main thing to consider is that the mirror that we use to reflect our collimated and polarized light coming from the display has to be able to encompass the entire image that is created from this system.

3.1.4.3 Reflective Mirror Part Selection:

Now that we have decided that the flat, or plane, mirror system is the optimal solution for being sure that our reflective optics can properly manipulate the light in our HUD system while also staying true to packaging constraints, we can do an analysis of the various commercially available options. When deciding on reflective optics, the primary criteria to be considered are as follows:

Cost: As low as possible

Diameter: Large enough to reflect full collimated display output

Shape: Round or Rectangular

It should be noted that the shape is not of critical importance but must be large enough to encompass our full collimated display output. Elliptical options are not practical for this solution.

OptoSigma Round Mirror (S-TFA-30C05-1)

Our first option is a round mirror from OptoSigma that costs around \$18. This flat round mirror has a high reflectance of above 90% with a total diameter of about 30mm. Unfortunately, this specific lens does not come with its own mount or housing, and the mount would have to be custom designed or sourced to fit this solution. In addition, it is available within weeks from OptoSigma

OptoSigma Square Mirror (S-TFA-15S03-10)

This option, also from OptoSigma costs around \$21 and is commercially available in the United States. This mirror is square, as opposed to round which might prove easier to

work with regarding our collimated rectangular display. It boasts a high reflectance of 90% as well and has a 3mm thickness. In addition, this mirror is 15mm x 15mm and would be able to fit the entire breadth of the collimated display output in terms of reflecting it. That being said, it also does not come pre-mounted or with a housing and would require that a housing be designed and fabricated for this specific solution.

Optosigma Square Mirror (S-TFA-20S05-4)

Our final option, also from OptoSigma, is a square mirror that costs around \$25. This mirror is quite similar to the choice previously mentioned and has dimensions of 20x20mm. This would allow for greater freedom in terms of pre-magnification of the display before it was reflected as the surface area is larger than the aforementioned 15x15mm. It also has high reflectivity of >90%, low thickness, and is commercially available in the United States. Although it does boast good specifications, it still does not come pre-mounted or with a native housing, and like with the other two options, would require a custom housing designed and fabricated or sourced for this solution.

Table 11: Mirror Product Comparison

Flat Mirror	Cost	Size	Reflectance
S-TFA-30C05-1	\$18	30mm diameter rounded	>90%
S-TFA-15S03-10	\$21	15x15mm square	>90%
S-TFA-20S05-4	\$25	20x20mm square	>90%

3.1.4.4 Reflective Mirror Product Selection

After reviewing all of the options available to us, it is clear that the best option for our HUD system in terms of a reflective mirror that will allow us to stay true to our packaging constraints is the OptoSigma Square Mirror (S-TFA-20S05-4). This 20x20mm mirror would give us greater flexibility in the surface area of the image we were able to reflect without losing resolution in our system (because of the high reflectance). That being said, we will still have to develop a custom mount / housing for our system with this lens in mind.

3.1.5 Aberration Correction Elements

In addition to requiring mirrors to move and manipulate collimated light throughout this system, there are some mirrors that serve multiple functions. By this, I mean that this project dictates that included are mirrors that can simply reflect the desired image as well as remove/reduce any kind of aberrations that present themselves in the system. Aberrations are inherent results of lens design in the real world that result in light not focusing ideally to what a theoretical system should yield. These imperfections in optical systems cause images to be blurred, distorted, or have color deviations and fringes instead of being perfect reproductions of the images in question. Some common aberrations include spherical aberrations, in which rays focus at different distances depending on where they interact with a lens in question. Astigmatic aberrations that

result in performance reduction as a result of field angles. The performance in the field is reduced compared to the performance perpendicular to the field. Field curvature is an aberration that results in a curved image plane due to imperfections in the design of a lens. Chromatic Aberrations occur when a lens system fails to focus all wavelengths of incoming light rays at a single point. This results in various fringes of color that appear as a spectra (citation 33).

In our heads up display system, it is expected that we will encounter aberrations. This is an inherent issue with many projection systems but does have a solution. The common aberrations for our polarized projection system of an electronic display are spherical aberrations, Astigmatism, field curvature, and chromatic aberrations. Spherical aberrations result as a symptom of spherical optical elements (lenses or mirrors) not bringing light rays to the same focus. Astigmatism occurs as a result of off-axis reflections creating angular inconsistencies in image feedback. Field curvature happens because the image formed by curved mirrors naturally wanting to lie on a curved surface. And chromatic aberrations arise because of natural dispersion in certain refractive elements, such as our multi-order waveplate and the various lenses. In order to correct for these aberrations, we must employ the use of a mirror system to help reduce the amount of image inconsistencies present in our system.

3.1.5.1 Reflective Mirrors for Aberration Correction

When considering what optical elements are best to use in our heads-up-display system for correcting aberrations, the three that present themselves as the most viable options are a series of mirrors. These are parabolic mirrors, spherical mirrors, and deformable mirrors. Deformable mirrors are reconfigurable mirrors that are capable of changing their shape in real time. These systems are dynamic with the mirrors traditionally being physically adjusted by actuator motor setups in real time. Because of this, they are often more expensive as a function of their increased operational complexity (citation 34). That being said, these systems are highly effective and good at correcting for various aberration types. Spherical mirrors have more limited correcting abilities, but are generally less expensive as a result of the fact that they are static systems and their manufacturing infrastructure is much more mature. These systems provide passive correction of some on-axis aberrations. Also, they can only minimize spherical aberrations but not completely eliminate them, as the mirrors themselves suffer from inherent spherical aberrations. Parabolic mirrors also provide passive aberration correction with the added benefit that they can also provide much more robust correction for spherical aberrations. Parabolic mirrors have changing radii of curvature and because of this, can focus collimated light onto a single focal point. While more expensive than spherical mirrors, these systems are much more effective at aberration correction (citation 35). Parabolic mirrors are typically higher cost which is a drawback and additionally they don't do as well in correcting for off-axis errors. That being said, they are still the highest performing standalone system.

Table 12: Mirror Technology Comparison

Mirror	Cost	Type of Correction	On-Axis Aberration Elimination	Off-Axis Aberration Elimination Complications
Spherical	Medium	Passive	Moderate Precision	Moderate
Deformable	Very High	Active	High Precision	None
Parabolic	High	Passive	High Precision	Moderate

3.1.5.1 Aberration Correction Element Selection

After reviewing the various options presented to us, it is clear that the best mirror system to correct for aberrations is a variation of the parabolic mirror. An off-axis parabolic mirror would allow us to correct for traditional aberrations that a parabolic mirror is known to eliminate, such as on-axis spherical aberrations which would provide a sharper focus as well as improved overall resolution in the system. Also, using an off-axis parabolic mirror would allow us to correct for off-axis aberrations like astigmatism in order to ensure that our heads-up display is immune to field angle complications. Even though this system is more costly than a traditional spherical mirror, our system demands a crisp and uninterrupted and uncorrupted image at the visual output for the end user.

3.1.5.2 Parabolic Mirror Part Selection

Now that we have decided on a reflective element to help our system correct for aberrations, we can begin to analyze the various off-axis parabolic mirror options to consider for integration into the HUD. The various relevant criteria for this are as follows:

Cost: As low as possible

Size: Large enough to fit reflected collimated display output

Shape: Round

OptoSigma Off-Axis Parabolic Mirror (OPG-25.4-25.4-A90)

This mirror from OptoSigma is a round OAP mirror with a 25.4mm diameter. In addition, it boasts a high reflectance at >94% and 23mm clear aperture. The item costs around \$237. This item would also require a custom mount or housing to be fabricated in order to properly fit it and integrate it into the system.

Avantier Off-Axis Parabolic Mirror (AVT-OAP-1131)

This option from Avantier Optical Components is significantly lower cost at around \$185 with a reflectance wavelength range of 400-2000nm. It is important to note that the other options are inherently reflective in the visible range. In addition, a custom housing or mount would have to be designed or fabricated around this component in order to successfully integrate it into our system. It is a round mirror with a 25.4mm diameter.

Edmund Optics Off-Axis Parabolic Mirror (#37-234)

This option from Edmund Optics is the most expensive at \$265. Its wavelength range is 250-700nm and has an average reflectance of >85% at this range. Its diameter is also 25.4mm and is a round mirror as well. A custom mount would to be designed and fabricated to successfully incorporate this mirror into our system.

Table 13: Mirror Product Comparisons

Mirror	Cost	Reflectance	Diameter
OptoSigma OPG-25.4-25.4-A90	\$237	>94% 400-700nm	25.4mm
Avantier AVT-OAP-1131	\$185	>90% 400-700nm	25.4mm
Edmund Optics #37-234	\$265	>85% 250-700nm	25.4mm

3.1.5.3 Aberration Corrector Product Selection

After reviewing the various options commercially available to our group for integration and aberration correction into our heads-up-display, it is clear that the superior option is the most cost-effective one. The Avantier AVT-OAP-1131 Off-Axis Parabolic Mirror boasts a high reflectance at the wavelength range that we need for our display (the visible spectrum at 400-700nm) but also has the lowest price tag of any of the options. That being said, we will still have to fabricate or source some kind of mirror mount for this system in order to successfully integrate it into our system.

3.1.6 Magnifiers

Once our system is free of image corruptions due to aberrations, it is time to project our collimated light onto our optical combiner. To do this, we must magnify our small image, one that is small because up to this point, the size of our screen has dictated a relatively tiny image. We must ensure that this image is magnified to fit the dimensions of our combiner film screen in order display a large enough image that the pilot and human eye can make sense of it. We must also be mindful of two things regarding situational awareness. First, the fact that if the image is too large, resolution will be lost in magnification and critical flight information will not be properly interpreted or perceived by the pilot. In addition, we must be mindful of the fact that the image needs to be magnified to such a level that would make sense in the context of a flight HUD. That is, a

system that would not impede the other visual aspects of flying a plane, such as covering precious real estate in the field of view. In order to magnify the system properly from the output of the aberration-correcting mirror system, we must ensure that we use a special type of magnification lens, or in this case, a projection lens, to project the image onto our optical combiner element.

3.1.6.1 Magnifier Lens Systems

When choosing a magnifying lens, there are a number of criteria to be mindful of. For our purposes of a heads-up display, the main factors affecting our system from a magnification point of view are those of working distance, field of view, and actual magnification of the system itself. There are four types of popular magnifying optical systems, those being simple lenses, doublet lenses, achromats, and triplets. Simple lenses are single positive lenses that work for low power magnifiers, such as 2x or 3x. That being said, these lenses are prone to chromatic aberrations. Doublet lenses are two-lens systems that work together but are not physically cemented together. Doublets produce better image quality as they correct for chromatic aberrations. Achromats are positive lenses that are cemented to negative simple lenses. These systems work well at higher powers and correct for chromatic aberrations very well. Triplets are three lens systems that are cemented together. These systems produce better quality images and are used best in applications that require high precision magnification (citation 36).

Table 14: Magnifier Technology Comparison

Magnifier Type	Number of Elements	Chromatic Aberration Correction	Cost
Simple	One	Low	Low
Doublet	Two (not combined)	Medium	Medium
Achromat	Two (combined)	Very High	Medium
Triplet	Three (combined)	Medium	High

3.1.6.2 Magnifier Selection

After carefully reviewing all of the options available to us in terms of magnifiers, one choice has made itself abundantly clear as the best for our application of an aircraft heads-up-display. The achromat is the best choice as it provides superior aberration correction for our system relative to its simplicity and cost. It is also better that we have a combined element system, like an achromat, as opposed to a non-combined element system as this would give us an order of magnitude more complexity in system alignment and add another variable to overall image distortion at the virtual plane of our HUD. The achromat is the superior choice.

3.1.6.3 Magnifying Lens Part Selection

After deciding that the achromatic doublet magnifying projection lens provides the superior aberration correction for our system relative to its simplicity and cost, we can

now conduct a proper analysis regarding what commercially available product is best to suit our needs. The primary criteria to consider for this are:

Cost: As low as possible

Size: >10mm diameter

Focal Length: >60mm but still fitting package dimensions

Integration Flexibility: High

AliExpress LightenOptics 50mm Achromatic Doublet Lens

Coming in relatively inexpensively relative to the other options at \$15, this achromatic doublet lens from AliExpress is a 12.7mm diameter with a 50mm focal length as well. Technically, these options fit our specification but because of the fact that the diameter of the lens is so close to what we would need to magnify in terms of output from the aberration corrective element, it would be a big challenge to precisely align this lens for our system. This may present integration issues in the future.

OptoSigma Achromatic Doublet Lens (S-DLB-20-60PM)

This option from OptoSigma comes in at around \$56 for the lens and would require a housing or mount to be built around it in order to integrate it into our system. That being said, it is a 20mm diameter lens with a 60.2mm focal length which does meet our system specifications and also allows for greater flexibility in alignment and integration.

Edmund Optics Achromatic Doublet Lens (#49-344-INK)

This Achromatic Doublet lens option from Edmund Optics would cost around \$124 and availability is contingent upon contacting the supplier. In addition, the diameter and focal length are 18mm and 60mm respectively, meaning that this system could be integrated into our HUD while still giving us flexibility in terms of our alignment tolerances.

Table 3.1.12

Table 15: Magnifier Product Comparison

Magnifier	Cost	Dimension Constraints	Integration Flexibility
AliExpress LightenOptics	\$15	12mm diameter 50mm F.L.	Low
OptoSigma S-DLB-20-60PM	\$56	20mm diameter 60mm F.L.	Medium
Edmund Optics #49-344-INK	\$124	18mm diameter, 60mm F.L.	Medium

3.1.6.3 Magnifying Lens Product Selection

After consideration of the various commercial options listed above, we have decided that the best option to integrate into our Heads-Up-Display is the second option, or the 20mm diameter, 60mm focal length magnifying lens from OptoSigma. This is relatively inexpensive compared to Edmund Optics' option and gives us more flexibility in terms of our alignment of the lens to the mirror due to the fact that the lens has an increased diameter.

3.1.7 Self-Dimming Element

One main aspect to consider when thinking about situational awareness in a flight environment is the dynamic ambient lighting environment that the pilot will have to operate in. Because of this consideration, we have engineered our heads-up-display system to operate with a self-dimming display element. This is not to be confused with the initial display system that will have our graphical user interface embedded into it. The self-dimming element will be a dynamic system that changes transparency of an electrochromic glass that we can project onto. This glass will also host our combiner film for our image to be projected onto and will effectively function as the system screen. As the environmental ambient lighting changes, we want to maintain ample contrast so that the pilot can have proper visibility of the information of the heads-up display. For example, the screen should be slightly darker on a bright day to increase contrast and much lighter at night or in overcast weather. This will also serve to prevent eyestrain on pilots during long journeys as low-contrast screens require more work from the brain to constantly monitor and interpret.

3.1.7.1 Self-Dimming Element Systems

For our system to serve this functionality, namely the ability to self-dim as a function of external brightness, there are a variety of commercially available and feasible technologies. The main difference in these technologies is not only cost but also integration complexity and the integration of a system with electronics vs. A system with no electronics involved. The multiple schools of thought are as follows: We can either have a system that reads input from a light-sensitive photodiode, that will take in ambient light readings and distribute electricity to an electrically sensitive self-dimming element, like an electrochromic glass, accordingly. We can alternately have a system that incorporates "dumb" photochromic film that overlays onto it that automatically darkens and lightens as a function of ultraviolet light intensity. In addition, this film has no electronic components to add to it and can be easily integrated into our system. The other technology that we can consider is what is known as a PDLC film, or a Polymer-Dispersed Liquid Crystal film (citation 37). This solution lies somewhere in between the other two in terms of technical complexity and can instantly switch between transparent and opaque states using input electrical current. This would still have the effect of "dimming" or making the system more visible to our pilot, but still being easier to integrate to our system as there are less electronics involved.

Table 16: Self-Dimming Element Technology Comparison

Self-Dimming Element Type	Integration Complexity	Dimming Tunability Degree of Freedom	Cost
E.C. Glass	High	High	High
Photochromic Film	Low	Low	Low
PDLC Film	medium	Low	Medium

3.1.7.2 Self-Dimming Element System Selection

After carefully reviewing the options available to us in terms of self-dimming technologies that serve to increase situational awareness, we have decided that a photochromic film will be the best solution for our system. The reasoning for this goes beyond the fact that this choice would be the most cost-effective. Photochromic film removes a layer of integration complexity in that it does not require the use of additional electronics to be incorporated into our system. The effectiveness of this type of film fits the scope of our project, as we do not necessarily need to have hyper-selective tinting. The system can function properly with dimming occurring as a function of ambient ultraviolet light in the pilot's immediate environment. That being said, we will have to be mindful of how our optical combiner will interface with this system and any distortive effects that arise as a result of that.

3.1.7.3 Self-Dimming Element Part Selection

Now that we have decided that the best self-dimming element for our heads-up-display would be to incorporate a photochromic glass, we can begin to conduct a formal investigation regarding what commercially available products are best to integrate into our final product.

The main criteria to consider are as follows:

Cost: As low as possible

Size: Fitting the dimensions of our end screen

Dimming Range: Light to dark as demanded by ambient lighting

HOHOFILM Photochromic Film Sheet (16-60%)

This product from HOHOFILM comes in sections of 60" x 40" photochromic film that can be resized according to the application needed. The cost for one of these sheets is \$100 and it is available within a few days delivery on Amazon. In addition, the installation method is peel and stick, so it is easy to integrate into our system provides a darker tint for our HUD which can prove advantageous for providing ample display output to increase situational awareness to our end user (pilot).

HOHOFILM Photochromic Film Sheet (45-75%)

This product from HOHOFILM comes in sections of 60” x 20” photochromic film that can also be resized according to the needed application. The cost for one of these sheets is \$40.50 and is available readily on eBay. The installation is also peel and stick, so it is still easy to integrate. It provides a lighter tint than the other options involved.

VLT Photochromic Solar Protective Photochromic Film Sheet (1-50%)

This product from eBay is similar in that it comes in larger sheet sizes that can be resized according to application necessity. The individual sheets come in sizes 50 x 300cm which would be more than enough for our application. They are also low cost at \$15.99. In addition, they offer a darker tint option with a quick response. This solution is peel-and-stick as well and would be easy to incorporate into our system.

Table 3.1.14

Table 17 Self-Dimming Element Product Comparison

P.C. Film	Cost	Size	Dimming Range
HOHOFILM (16-60)	\$100 per sheet	60” x 40”	16-60% tint
HOHOFILM (45-75)	\$40.50 per sheet	60” x 20”	45-75% tint
VLT Solar Protective (1-50)	\$15.99 per sheet	118” x 19”	1-50% tint

3.1.7.4 Photochromic Film Product Selection

It is clear from the research conducted that one option is the optimal solution to fit our need for a self-dimming element in our heads-up display. That is the HOHOFILM 45-75% tint photochromic film. Although this product would not dim as much as the most expensive option for this technology, it would still do well enough relative to the scope of our project in terms of being able to increase contrast of the display and visibility of the system for our end user. In addition, the system would be easy to integrate as it is just a peel and stick element that does not require a ton of complexity in integration. The size that the sheets come in is also plenty in terms of us being able to resize them to fit the needs of our HUD screen.

3.1.8 Optical Combination Technology

One of the most critical portions of our project will be the integration of an optical combiner for the display to overlay onto. In our system, the pilot needs to have visibility not only of the real-world, but also visibility of display output from our HUD. In order to marry these two points of view, we must employ the use of what is called an optical combination technology. This technology serves to manipulate the digital and virtual images in such a way that the image lands on the combiner surface to overlap the real-

world-view (citation 38). In addition, if you simply try to overlay display output from a HUD system onto a regular screen, you will suffer from what is called a “ghost” image or a second image appearing as a result of the inherent optics involved. These reflections can prove problematic in viewing the HUD output (citation 39). In order to eliminate this, it is best to utilize a combiner film.

<https://telepromptermirror.com/glass-teleprompter-mirror/>

3.1.8.1 Various Optical Combination Technologies

The two best optical combiner technologies for use in a HUD are Dielectric partial reflectors, also known as “teleprompter glass”, and HOE combiner film, which stands for holographic optical element. Teleprompter glass is a popular combiner technology typically used in more inexpensive commercially available HUDs. These systems are made of flat fused silica plates with a broadband dielectric stack, meaning that it can accommodate for a wide variety of input wavelengths, something important for dynamic color display modules. In addition, teleprompter glass setups can accommodate large areas of display and these systems work well with PDLC dimming setups (citation 40). HOEs, or holographic optical elements are diffractive holograms that reflect a specific band/angle while passing most other light. This technology acts as a sort of “filter” for input light. These systems typically result in lower-glare systems and are much thinner and lighter than their teleprompter glass counterparts. They are also sensitive to changes in viewing angles as well as the fact that they can shift visible wavelengths (different colors) in a manner such that the original color intended to be viewed is not maintained. They are also harder to source as their manufacturing infrastructure is less mature and as a function of this and the complexity of the technology itself, HOEs are more expensive than traditional teleprompter glass setups (citation 41).

Table 18: Optical Combiner Technology Comparison

Combiner Technology	Effectiveness at different angles / wavelengths	Availability	Cost
Dielectric Partial Reflectors	High	High	Low
Holographic Optical Elements	Moderate	Low	High

3.1.8.2 Optical Combiner Technology Selection

After reviewing all of the options available to us in terms of optical combiner technologies. The two that stood out the most were dielectric partial reflectors, also known as teleprompter glass, and holographic optical elements. It is clear to us that the superior choice of optical combiner for our system is the Dielectric Partial Reflector, or teleprompter glass, for a multitude of reasons. First off, this technology has a mature

manufacturing infrastructure and is part of a broader existing industry of commercially available heads-up displays. This means that costs are lower for these systems and they are proven to work in regards to the scope of a project like this. In addition, Their effectiveness over a range of wavelengths is higher than HOEs and this will prove critical for our project as our displays will be emitting light across the visible spectrum, not only at discrete wavelengths. In addition, there seem to be no issues with integrating dielectric partial reflectors on glass to systems that involved circularly polarized light.

3.1.8.3 Optical Combiner Part Selection

After deciding that the optimal option for our optimal combiner technology for the aircraft HUD is the dielectric partial reflector, or teleprompter glass, we can now look into what commercially available product is the best variation of this that fits our needs. The criteria to consider are as follows:

Cost: As low as possible

Size: Fitting the dimensions of our end screen

Reflectance / Transmittance Ratio: 40-50%R / 50-60%T

ProPrompter 40/60 Beamsplitter Glass

This beamsplitter glass from ProPrompter is designed for commercial teleprompters in mind where a heads-up application is the target application. It is \$100 and comes in a 10.4” diagonal rectangular screen. This would be more than large enough to project our HUD onto and with the 40/60 RT ratio, would be an ample solution for our project.

ProPrompter 10” x 10” Beamsplitter Glass

ProPrompter also sells a 10” x 10” variation of this glass with 60/40 R/T rate optical beam splitting screen incorporated. The 10” x 10” variation of this model is \$180 and while the cost is significantly more, it would provide for greater screen size for our heads-up display. That being said, this is not necessarily practical in the context of our HUD as our system specifications dictate that our screen size must be smaller. One of the engineering challenges associated with this would be re-sizing the teleprompter glass to fit our specifications.

MagiCue Glass Beamsplitter for Presidential Series Teleprompter 17”

MagiCue sells a 60/40 R/T teleprompter glass product with a 17” screen. This product is more expensive coming in at around \$200 and would present some engineering challenges to integrate into our HUD. The most prominent of these challenges would be to re-size the system in order to fit our design specifications as it is too large. That being said, it meets the reflectance and transmittance criteria that we need to successfully project our display to our end user.

Table 19: Optical Combiner Product Comparison

Glass Product	Cost	Size	Reflectance / Transmittance
ProPrompter Small	\$100	10" diagonal	60/40
ProPrompter Large	\$180	10"x10"	60/40
MagiCue Glass	\$199	17" diagonal	60/40

3.1.8.4 Optical Combiner Product Selection

After reviewing all of the options commercially available to us for our optical combiner selection, it is clear that the best choice for our HUD is the smaller of the various options examined from ProPrompter. This 10" diagonal screen has the correct reflectance and transmittance ratio that we need from our system as well as being the correct size. With the 10" diagonal size, we do not need to re-size or re-engineer the screen to fit our specifications. In addition, it is the cheapest of all of the options available.

3.2 Electrical Hardware

Hardware Overview

The system being designed will require multiple different PCBs. The main PCB and the most important will be the one that connects all the peripherals. It will do most of the computing for sensors and handle the bulk of the code. There are three different peripherals that will each require its own PCB and each with its own unique layout and parts to gain the required data. The three peripherals are the altimeter, Inertial Measurement Unit (IMU), and the GPS. These three different peripherals will be all the sensors used in the HUD to provide the pilot with the proper flight information. There are so many different options when it comes to sensor types and PCB layout design to choose from and picking the right one to best fit the overall design is essential in any good product. Along with the three boards required there will also be a need for power regulation to 3.3 and 5 volts which will give a total of 6 individual PCBs with all unique and important functions to the overall circuit.

3.2.1 Power

The first PCBs to design are the regulators. The power source for this project is a 12v outlet which is standard on Cessna 172. While the PCBs job is to provide power the 12v outlet cannot be hooked up directly to the board or it will burn out all of the components. The Cessna 12V power supply does not provide a steady 12V, rather it fluctuates between 10 and 14.4 volts. (citation 41) This can vary depending on factors such as the alternator load and the engine RPM. Other avionics on board can cause spikes and variances as well. Power regulation is required to steady the power supply and make it a constant uniform voltage. The use of a regulator can steady the voltage and both lower the voltage to correct value and remove the spikes. Almost all standard components that

are used on board are powered by either 3.3 volts or 5 volts. This means other PCB will be required to regulate the power supply to these values. This is the most commonly used regulator on PCBs and is proven to be all that is needed to power all of the components.

Table 20: List of Components Power Requirements (Made with ChatGPT, 2025)

Component	Operating Voltage	Operating Current
ESP32/STM32F4	3.3V	50 – 250mA
Sensors (Altimeter, IMU, GPS)	3.3V(Sometimes 5V)	3 – 10uA, 3mA, 29mA
SD Card	3.3V	50 - 300mA
Display	3.3V or 5V	80 - 250mA
USB-C	5V Normal	500 mA
Aircraft Outlet	12V (Varies between 10 and 14.4)	5 - 10A

There are two different types of regulators that can be used to control the power regulation with the first being a linear regulator and the other being a switching regulator. Both come with their pros and cons to its use.

The linear regulator is a simple process of flowing the current through a circuit with a reference voltage that has a precise target.(citation 42) The feedback network will then divide the output voltage down to the reference level and the error amplifier compares the output to the reference. The circuit will then adjust by either opening the transistor to raise the current which will raise the voltage or by closing the translator to decrease the current and the voltage. The extra voltage is exposed through heat. This continuous voltage adjustment in the voltage will dampen the voltage to meet the required power supply.

The switching regulator works by turning the current on and off.(citation 42) The inductor will rapidly connect and disconnect from the 12V source to adjust the output voltage. The controller will monitor the output voltage through the feedback loop and will adjust the duty cycle accordingly which is just the amount of time the power is connected and disconnected. When the voltage is too high then the amount of time connected will decrease and if the voltage is too low then it will disconnect longer. Since

the source is constantly being connected and disconnected the current flow will not be smooth so the regulator uses capacitors to smooth the flow out. There will also be a diode used to allow current flow when the power is disconnected.

Table 21: Linear vs Switching Voltage Regulators (Made with ChatGPT, 2025)

Aspect	Linear Regulator	Switching Regulator
Main Element	Transistor	MOSFET (High-speed switch)
How it Works	Drops extra voltage as heat	Switches on and off rapidly
Control	Analog feedback loop	Pulse with modulation duty cycle
Key Parts	Transistor, diode, op-amp, resistor, capacitors	MOSFET, diode, inductor, controller, capacitor
Efficiency	40-60% (Heat loss)	85-95%
Noise	Very Low	High (needs filtering)
Heat	High	Low
Complexity	Simple	Moderate-Complex
Best Use	Low-power, noise-sensitive circuits	High-power, efficient systems

Taking all of these factors into effect with the type of circuits that will be being used is a combination of both. For the 5V a switching regulator is the more fitting circuit for this project. It is more efficient due to its disposing of extra energy through heat loss and high efficiency is a requirement for this project. The power regulator is a good place to start with higher efficiency. The switching regulator is more complex to build and makes it more prone to errors and is more expensive but it is also more efficient and overall just a better pick for the 5V. For the 3.3V a linear regulator will be used. It will provide a low noise and a clean output voltage. It is a more simple circuit to build and is the correct choice for the 3.3 regulator.

Now that the type of regulators is selected the buck converter and the low drop out regulator part can be selected. First looking at the 5V buck regulator, two good options are the TI TPS54331 and the TI LMR33630. (citation 43)The main difference between these two options comes down to efficiency. The TI TPS54331 is a non-sync rectifier. This means that at lighter loads it is less efficient and is beat out by the sync rectifier in the TI LMR33630. This makes the better of the two options the TI LMR33630 mostly

due to the sync rectifier but the remainder of the specs of the two are shown in the table below. It does not require a schottky diode and yields a lower noise.

Table 22 5V Regulators Comparison (Made with ChatGPT, 2025)

Feature	TPS54331	LMR33630
Type	Non-Synchronous buck	Synchronous buck
Input Voltage	3.5 V – 28 V	3.8 V – 36 V
Output Current	Up to 3A	Up to 3A
Switching Freq	570 kHz	1.4 MHz
Efficiency	Moderate	High
External Diode Required	Yes	No
EMI Performance	Average	Better
Thermal Performance	Average	Cooler
Space	Larger (More parts)	Smaller
Cost	Lower	Slightly Higher

Moving onto the 3.3V regulator there are once again two good options for the regulator with them being the TLV75733 and the TLV76733(citation 44). These are more similar than the two options for the 5V. They are both low noise LDO regulators with a 3.3V fixed output voltage. The TLV7577 utilizes 1uF capacitors to remain stable and the TLV76733 are stable with 10uF capacitors. The main difference for choosing the part comes down to the data sheets. The TLV76733 has an easier to follow data sheet with a good example regulator. This makes it the more suitable option opposed to the TLV7577 where more research or custom design would be required to create the regulator, which can lead to more design errors. Taking this into consideration, TLV76733 is the better choice mainly because of its data sheet.

Table 23: 3.3V Regulators Comparison (Made with ChatGPT, 2025)

Feature	TLV75733	TLV76733
Type	LDO Regulator	LDO Regulator
Input Voltage	2.0 V – 5.5 V	2.55 V – 16 V
Output Current	Up to 1A	Up to 1A
Output Noise	Low	Low
Quiescent Current	-25uA	-50uA
Stability	Yes (1uF capacitors)	Yes (10uf capacitors)

3.2.2 Microcontroller

There will be a total of two microcontrollers required in total for the HUD. There are two on the main board and one that is more for the processing end and other that is for display. The processing will need to be able to take in the data from all three of the peripherals that are giving information and display them accurately and quickly. There is no time for high latency on the display. The other microcontroller on the main board will be used for the display. A high-quality picture is necessary for the display since it is getting reflected multiple times through different lenses to be displayed properly on the HUD.

There are many good options for microcontrollers but the two main options for the peripheral computing are the ESP32 and the STM32F4. Both components require 3.3 V of power. The main features of the ESP32 is a Dual Core 32-Bit MCU, with built in Wi-Fi and Bluetooth (citation 45) (citation 46). It provides strong performance for graphics logging and communications. It also has very good libraries and development support. The STM32F4 has high reliability and great real time performance. It is industrial grade and used in avionics to deal with peripheral data.

The STM32F4 has great balance between processing power, reliability, and hardware flexibility that is required for the sensor management. It's built around the ARM Cortex-M4 architecture and also includes a floating point unit (FPU) which is important for real time sensor fusion tasks (citation 45)(citation 46). These tasks include combining the gyroscope, accelerometer, magnetometer and altimeter data all into a stable attitude and position output. This capability makes it more capable to handle these tasks rather than the 8 or the 16-bit MCUs which could struggle to handle the math intensive calculations that need to be done for flight stabilization. On top of this the STM32F4 has the

capability of supporting multiple high-speed communication buses. This allows for simultaneous and independent connections to each individual peripheral with no bottlenecks. This could make it so that if the processing power is enough there may not need to be three individual MCUs. It also has a robust Direct Memory Access (DMA) and different timer systems to confirm there is constant sensor polling and logging even while fusion is being run which is important for the consistent frame updates to the MCU display(citation 45)(citation 46).

Further beyond the technical abilities of the STM32F4 is a MCU that is designed for industrial and aerospace-grade reliability (citation 45). In environments where it is frequent to have noise temperature and vibration shifts frequently there needs to be a built-in facility to handle these changes. The STM32F4 has built in brownout protection, watchdog timers, and an EMI-tolerant I/O which will make sure that it operates at high reliability even in harsh air conditions. Since its operation voltage is at 3.3V logic it fits perfectly with other common day sensors which also mostly operate at this voltage. Its large capabilities with libraries and its industry support makes its development smooth and debugging faster.

There is one big drawback with the STM32F4 and its the amount of coding it requires to make it work. There is more background coding required to pull the data peripherals as opposed to other options. The other MCU option is the ESP32 and it's much simpler to code but does not provide as fast computing. It still has dual cores and 32(citation 45) (citation 47) bit architecture to be able to handle the heavy computing just at a slower rate. It has optimized math instructions and can complete the peripheral data at a high enough rate to update the HUD reliably. While it doesn't process as fast as the STM32F4 the simplicity of the coding is a tradeoff worth taking. Taking all these factors into account the STM32F4 is able to operate at high speed with excellent precision and robustness but it is a more difficult MCU to code while the ESP32 is slower but easier to use and for the purposes of this project will have sufficient computing power to complete the tasks at hand.

For the display portion of the MCU the ESP32 could be used or a Raspberry Pi 4. The ESP32 while it is good for the peripheral data it is not ideal for the display. (citation 47) The main issue is its inability to handle video feed and the use of the camera will require this to be displayed. While the Pi 4 is not the most ideal option due to slow boot time and some extra delay it can display a video file clearly which is a crucial aspect of the HUD. Two of the positives of the Pi 4 is the fast CPU with the quad cores and its easy to design UI(citation 48). It is capable of powering large projectors or light LCD panels making it perfect for this project.

Table 24: Hardware Comparison for MCU (Made with ChatGPT, 2025)

Feature	ESP32	STM32F4	Raspberry Pi 4
---------	-------	---------	----------------

Processor	Dual-core Xtensa LX6 (240 MHz)	ARM Cortex-M4 (up to 180 MHz)	Quad-core ARM Cortex-A72 (1.5 GHz)
Architecture	32-bit microcontroller	32-bit microcontroller	64-bit single-board computer
RAM	520 KB internal	192–256 KB internal	2 GB – 8 GB LPDDR4
Storage	Flash (built-in / external SPI)	Flash (external)	microSD card (up to 512 GB)
Operating System	FreeRTOS / bare-metal	Bare-metal / RTOS	Full Linux (Raspberry Pi OS)
Connectivity	Wi-Fi + Bluetooth BLE (built-in)	None (needs external modules)	Ethernet, Wi-Fi, Bluetooth, USB
Power Use	~80–250 mA active	~50–150 mA active	600–1000 mA idle (up to 3 A full load)
Voltage	3.3 V logic	3.3 V logic	3.3 V logic (5 V input via USB-C)
Target Use	Peripheral Computing	Peripheral Computing	GUI Systems

Taking all these factors into account, the two MCUs being picked for this project are the ESP32 which will be used for calculating the flight data by using the information collected by the sensors. To display all this data, a Raspberry Pi 4(citation 48) will be used due to its easily designable UI and its capability to show a video stream from a camera.

3.2.3 Altimeter

An altimeter is a device that has the capabilities to measure altitude by detecting the changes in air pressure and then uses calculation to convert the measured data into height above sea level. For this project, the most practical way to achieve this task is by using a barometric altimeter. This device uses miniature pressure sensors to take the readings.

When the pressure reading is decreasing, that means that the altitude is increasing and at a predictable rate. This means that using the sensor data readings the altitude can be mathematically converted into an altitude using the International Standard Atmosphere (ISA) model. Since the only reason needed to calculate this is pressure, the only sensor that is really needed is a small digital barometric pressure sensor. Two good options for this small barometric pressure sensor are the BMP390 and the MEAS/MSE5611. (citation 49) They are both similar in price and either would be a good option.

The BMP390 is a more modern option with an ultra-compact digital pressure sensor that is highly precise and has low power consumption. It has the capabilities to operate over I²C or SPI and has a good pressure noise of 0.5 Pa RMS which means it only translates to a few centimeters of error. It has built-in temperature compensation and has the ability to sample up to 200Hz. This makes it fast enough to operate for the HUD where high speed is required (citation 49).

The other option, the MS5611, is an older-generation sensor, but it is the more proven. It is a high accuracy barometric sensor that is frequently used in drones and other small avionics devices. (citation 51) It is capable of 24-bit resolution with very low noise as well as good temperature stability. Similar to the BMP390 it communicates digitally using the I²C or SPI and it outputs the compensated pressure as well as the temperature values. This reduces the need for extra calibration. This sensor does draw more power than the BMP390, but it is still very often used in flight systems due to its reliability and predictable behavior in varying environments (citation 50).

Using either of these two options will sample the pressure periodically through the ESP32 sensor and compute the altitude using a simplified hypsometric equation (1).

$$h = 44330 \times \left(1 - \left(\frac{p}{p_0} \right)^{0.1903} \right) \quad (1)$$

In this equation, the p is the pressure that gets measured, and the p_0 is the reference sea-level pressure. The 44330 is a constant derived from standard sea temperature divided by temperature lapse rate. The system will take the readings and filter them through a low-pass filter to help remove turbulence noise. This data could be combined with GPS data to help make some of the results smoother. Since the ESP32 has the proper capabilities with strong floating-point performance and built in timers it will be able to handle these calculations to be able to run real time sensor sampling and logging (citation 49).

Table 25: Hardware Comparison for Altimeter (Made with ChatGPT, 2025)

Feature	BMP390	MS5611
Type	Barometric pressure & temperature sensor	Barometric pressure & temperature sensor
Pressure Range	300 – 1250 hPa	10 – 1200 hPa
Resolution	≈ 0.03 m	≈ 0.1 m
Pressure Accuracy	±0.5 hPa typical	±1.5 hPa typical
Temp Accuracy	±0.5 °C	±0.8 °C
Supply Voltage	1.65 – 3.6 V	1.8 – 3.6 V
Current Consumption	~3 µA typical (low-power mode)	~1 µA standby / ~10 µA active

Since the cabins on most Cessnas are not pressurized, nothing needs to be mounted on the outside of the plane which means everything can be self-contained. As long as there is a vent on the enclosure then the pressure will change and the proper reading can be made. Taking all the aspects into consideration while the MS5611 is an older more proven pressure sensor, the BMP390 is a better fit for this project.

Table 26: List of Parts for the Altimeter (Made with ChatGPT, 2025)

Category	Part / Component	Value / Type	Notes / Purposes
Sensor	BMP390	Digital Barometric Pressure Sensor(I ² C/SPI)	Main Sensor
Power	Ferrite Bead(Optional)	600 Ω at 100 MHz	Filters noise on 3.3 V line
	Capacitor(Bulk)	4.7 µF – 10 µF	Stabilizes sensor supply

	Capacitor (Decoupling)	0.1 μ F (close to VDD)	Removes high-frequency noise
Bus / Interface	Pull-up resistors	2.2 k Ω – 4.7 k Ω (SCL/SDA)	Required for I ² C communication
	Address/Mode Resistor	10 k Ω (CSB or PS pin)	Sets I ² C or SPI mode
Connector	Pin Header	6-pin(3.3V, GND, SCL, SDA, INT, ADR)	Connection to ESP32 on the main PCB

3.2.5 Inertial Measurement Unit (IMU)

The IMU is arguably the most important sensor for the HUD. It provides the most data to the system real-time that will make it capable of determining the aircraft's pitch, roll, and yaw. This device will most commonly combine nine different angles from three different sensors. The three sensors are a 3-axis accelerometer, a 3 axis-gyroscope, and a 3-axis magnetometer. This gives its nine degrees of freedom(9-DOF) (citation 52). The accelerometer will measure the linear acceleration which includes gravity, and the gyroscope tracks rotational motion. The magnetometer is essentially a digital compass that helps correct the yaw and the drift. These three sensors in tandem provide information continuously about the plane's orientation in a 3D space that will allow for the ESP32 to calculate precise altitude data to be displayed on the HUD.

The ESP32 uses the IMU data along with a sensor-fusion algorithm to combine the quick response of the gyroscope with the long-term reliability of both the magnetometer and the accelerometer. This combination will result in smooth and accurate tracking of the attitude during a flight. This will remain true even in turbulence or sharp banks and turns. The IMU communicates with the ESP32 through the I²C or SPI which is operating at 3.3V logic (citation 52). The alignment of the IMU is very important because everything it measures has to do with orientation. It should be centered on the main PCB so that its axes align with the aircraft's pitch, roll, and yaw directions. Proper placement can also cut back on vibrations as well as orientation errors and should ensure stable readings for the HUD.

There are two different options for measuring the required data for the IMU. The first is to use a 9-DOF chip to get the data and the second is to use a 6-DOF accel/gyro with a separate magnetometer for the other three axes. The main difference is the complexity (citation 52).

The 6-DOF accel/gyro paired with the separate magnetometer is good because it gives much more accurate sensor data, but it's much more difficult to build. Since the magnetometer is separate, it means a higher quality sensor can be selected and allows for lower drift or higher ranges. This will reduce magnetic interference because they are not pre-built inside each other. While it is more accurate, it does require two chips which means more routing and power conditioning. The design needs to be larger, leaving a bigger footprint. On top of this, the sensor needs to be calibrated where the 9-DOF chip comes pre calibrated (citation 52). This makes more room for error and ultimately the less desirable option. For the purposes of this project, the complexity of the dual chips makes this option too complex.

The 9-DOF integrated chip, while it is less accurate due to magnetic interference, its design is much simpler. The loss in accuracy is due to the magnetometer sitting close to the gyro's oscillator and can cause noise and degradation of header accuracy. This noise is a tradeoff of a much less complex PCB with less components, easier integration and smaller design. The 9-DOF chip comes with pre-aligned axes which makes the sensor math easier as well (citation 52). While this option does have more noise and could yield less accurate data, it is a simplistic design and still provides good data.

Two good options for 9-DOF chips are the ICM-20948 and the LSM9DS1. Looking at the first option, the ICM-20948 can run with a voltage anywhere between 1.8V and 3.6V. It is a reliable and frequently used option for designing PCBs for embedded inertial work. It is a low power module that only requires about 3.1mA for all 9 axes. It is also very low noise. On top of all that, this has a Digital Motion Processor (DMP)(citation 53) that allows for some of the computing to be offloaded.

The second option for the IMU is LSM9DS1. It runs between 1.9V and 3.6V. It is designed primarily for 3.3V logic. It requires very lower current and like the ICM-20948 is a reliable option that is used in HUD design for embedded inertial work (citation 66). This option, unlike the previous hardware, does not have the built-in DMP to offload computing, making it require more of the ESP32s computing.

Table 27: Hardware Comparison for IMU (Made with ChatGPT, 2025)

Feature	ICM-20948	LSM9DS1
Axes	9-axis (3 accel + 3 gyro + 3 mag)	9-axis (3 accel + 3 gyro + 3 mag)
Voltage	1.71 - 3.6V	1.9 - 3.6V
Accel/Gyro Ranges	±2/4/8/16 g; ±250/500/1000/2000 dps	±2/4/8/16 g; ±245/500/2000 dps

Mag Range	$\pm 4900\mu\text{T}$	$\pm 2/4/6/8/$ gauss
Power Use	3mA	4-5mA
Extra	Built in DMP	Simpler

With the chip design picked out, the best option for the IMU sensor is the ICM-20948. It provides all the required needs for the HUD. It has low power consumption and an integrated Digital Motion Processor (DMP)(citation 53). This allows for some of the math and heavy computation from either the sensor fusion or the motion filtering to be offloaded from the ESP32. It is designed for low voltage operations and operates at 3.3 V. It also has the flexibility to add sensors later if needed, and its small compact design gives it a small footprint and makes it easier to incorporate in the design.

Table 28: List of Parts for the IMU(Made with ChatGPT, 2025)

Category	Part / Example	Purpose / Notes
Main Sensor IC	ICM-20948	Core sensors
Power Regulation	Low-noise 3.3 V LDO	Supplies clean, ripple-free voltage to the sensor
Decoupling Caps	0.1 μF + 4.7 μF near sensor	Reduces voltage noise
Bus Pull-ups	2.2 k Ω – 4.7 k Ω (SCL/SDA)	Needed for I ² C communication
Optional Filter	Ferrite bead on 3.3 V rail	Blocks high-frequency noise from the main board
Connector	JST-SH or 4-pin header (3.3 V, GND, SCL, SDA, INT, ADR)	Connection to the main PCB (ESP32)

3.2.6 Global Positioning System (GPS)(Strech Goal)

The GPS is an important peripheral because it does not only just give location but it is capable of providing other bits of information on the HUD such as airspeed over ground, heading and even a backup altitude estimate. A GPS works by receiving weak satellite radio signals from multiple different satellites that are orbiting the earth. When each of the signals are received they also come with a time stamp. From this data it is possible to

then determine the distance to the satellite based on how long it took to receive the signal. With the use of four different signals its is possible to calculate latitude, longitude, attitude and time using trilateration.

The two main pieces of the GPS are the GPS antenna which receives the data from the satellites and the GPS Receiver Module(GNSS) which decodes the satellite data(citation 54). Some of the other components are the Low-Noise Amplifier(LNA) which boosts the weak satellite signals received from on the antenna. The SAW filter will block out noises near the GPS to get a better signal. The UART/I²C connection will connect to the ESP32 to provide data and location. It uses 3.3V which goes well with the rest of the power requirements.

There are two good options for the antenna and receiver module, the first being the u-blox NEO-M8N paired with an active GPS antenna (citation 54). In this configuration it is a high-performance option that has many uses in UAVs and other experimental avionics. The NEO-M8N is a professional grade option that uses multiple different satellite constellations at the same time to make more accurate readings. It is fast and has the capability of the 10Hz updates and would work well with the HUD (citation 55). The hardware can signal lock which means even in turns and with vibration it should still maintain a strong connection with satellite signals making it very reliable. When this is combined with an active antenna that has a built in LNA it makes it very sensitive and low noise. Overall, this is a very good reliable option.

The second option is a Quectel L76K with a 25mm Ceramic patch antenna. This is a more compact power efficient system that is a little simpler (citation 54) (citation 56). It also can connect to multiple systems and provides a good signal. It has fixes but its recommended use is in open skies. The antenna used is a small ceramic antenna that is mounted directly to the PCB. This makes it easy to design but this is the reason for it being less reliable. Taking these factors into account, this option is much more compact but less reliable.

Table 29: Hardware Comparison for GPS (Made with ChatGPT, 2025)

Feature	NEO-M8N paired with an active GPS antenna	Quectel L76K with a 25mm Ceramic patch antenna
GNSS constellations	Concurrent GPS, GLONASS, Galileo, BeiDou, QZSS, SBAS	GPS, BeiDou, GLONASS, QZSS (some boards also list Galileo)
Position accuracy	~2.5 m CEP typical	~2.5 m CEP typical, some modules quote ~1.5–2.5 m depending on config

Sensitivity	−167 dBm	−160 dBm
TTFF (cold start)	~ 26 s typical cold start	~15 s typical cold start
Max update rate	Max 10 Hz	Max 10 Hz
Supply & current	1.65–3.6 V supply, ~ 25 mA tracking	~2.55–3.6 V supply, 18–29 mA tracking
Antenna & extra current	~5–20 mA	None

Looking at these two different setups, the first is more reliable but much more complex to design. It can mount the antenna in a better place to receive a better signal, but that means it makes the system less compact. The second option provides a good alternative with it being more compact but less efficient. For this project, the second setup with the Quectel L76K and the 25mm Ceramic Patch Antenna provides enough reliability for this project.

3.2.7 Camera (Stretch Goal)

The camera will serve as an extension to the pilot's vision. A camera will be mounted on the dash and provide feed onto the HUD. To achieve this two good camera options are the Arducam IMX-219 Wide-FOV Pi Camera (IMX219, 160°) and the Raspberry Camera Module. Looking at the first option the IMX219 provides a high resolution with a very wide field of view of 160°(citation 67). With the wider camera lens does come a larger device.

The second option is the Raspberry Pi Camera Module v2 which is similar to the main difference being the narrower camera. This one only provides 62.2°, which is substantially smaller and really does not provide too much of a difference (citation 68). While this is much thinner, both cameras do utilize the same sensors. This means the resolution is also the same.

Table 30: Hardware Comparison for Camera (Made with ChatGPT, 2025)

Feature	Arducam IMX219 Wide-FOV (160°)	Raspberry Pi Camera Module
Sensor	Sony IMX219 , 8 MP	Sony IMX219 , 8 MP
Max still resolution	3280 × 2464	3280 × 2464

Video Modes	1080p30, 720p60, 640×480p90	1080p30, 720p60, 640×480p90
Field of view	160°	62.2°
Lens	3.15 mm, F/2.35 wide-angle, higher distortion	~3.0 mm fixed-focus standard lens, low distortion
Trade-off	Much wider FOV, but edges are stretched & softer because of wide lens	Narrower FOV, but less distortion and sharper detail across frame

Taking these factors into account the Arducam is the better fit for this project. While the camera is larger and has higher distortion, the main reason for adding the camera is to increase the field of view. With the use of the Arducam it provides more than double the field of view as opposed to that of the Raspberry Pi and makes it the clear choice of this application.

3.2.8 Data Communication Protocols

To properly communicate between MCUs and components, determining the protocol associated with said communication is important. This protocol will determine electrical and software integration of all system electrical components. A protocol must be selected that ensures synchronized transmission with minimal delay and error. Thus, the protocol must offer low latency, determinism, and noise resilience (citation 65). The top candidates for these protocols are UART, I²C, and SPI which will be compared below.

3.2.8a UART (Universal Asynchronous Receiver/Transmitter)

“UART is an asynchronous serial communication protocol that transmits data using two lines: TX (transmit) and RX (receive)” (citation 65). UART uses predefined baud rates for synchronization instead of relying on a shared clock signal like other protocols.

UART is very simple which makes it useful for low-pin-count or long-distance communication and requires minimal software overhead. UART has the disadvantage of only being a half-duplex which in the context of this system makes it more difficult to use for sensor interfacing due to the one-to-one communication that it has. Also due to a lack of clock synchronization, UART can have varied latency which will affect deterministic real-time performance (citation 65).

3.2.8b I²C (Inter-Integrated Circuit)

“I²C is a synchronous, multi-master serial communication bus that uses two lines (SDA and SCL) for data and clock” (citation 65). I²C is commonly used for low-speed peripherals as well as real-time clocks. It is primarily used and advantageous in this system due to its ease of use, support of multiple devices on the same bus, and interfacing with low-bandwidth sensors. Yet, due to its support of multiple devices on the same bus,

this can make I²C relatively slow which makes it unsuitable for streaming high-frequency peripherals like the camera. Overall though, I²C is a very strong choice for use in this system.

3.2.7c SPI (Serial Peripheral Interface)

“SPI is a synchronous full-duplex communication protocol that uses four lines (MOSI, MISO, SCLK, and CS) to exchange data between a master and multiple slave devices. SPI provides the highest throughput among the three, with speeds typically exceeding 10 MHz, making it well-suited for high-frequency sensor data such as the ICM-20948 IMU. Because SPI transmits data with dedicated clock and data lines, it achieves deterministic timing and minimal latency—key requirements for real-time HUD symbology updates. However, SPI requires additional chip-select lines per device, increasing wiring complexity. Despite this, its predictable timing, noise resilience, and high bandwidth make it ideal for the primary sensor bus in this system.” (citation 65)

3.2.7d Comparison and Selection

Now that the different communication protocols have been discussed, the table below can be used to compare them:

Table 31: Data Communication Protocol [65]

Criterion	UART	I ² C	SPI
Data Rate	Medium (~1 Mbps)	Low (≤ 1 MHz)	High (≥ 10 MHz)
Determinism (Timing)	Medium	Medium	High
Multi-Device Support	Low	High	Moderate
EMI Susceptibility	Low	High	Low
Wiring Complexity	Low	Low	High

Based on the table above and the project, I²C is the best choice for this project since it has the lowest wiring complexity with the highest determinism and support across all devices. This means that communication setup between devices will be relatively straightforward while still maintaining relatively low latency.

3.3 Software

3.3.1 MCU Selection from a Software Perspective

The main goal for this project is to develop a working heads-up display in the given timeline. This can be interpreted in many ways but for the purpose and scope of this project, minimal development time and lowest possible latency are the main objectives. This leads to the question of the selection of microcontrollers not only from a hardware side but from a software side. The usage of the MCUs does not change, one will still be a dedicated GUI, and one will still be a dedicated sensor reader. The ideal microcontroller combo will contain enough built-in functions and tutorials for usage that lead to low development time but are still able to be controlled manually enough for low latency.

The best microcontrollers to fit these constraints are the Raspberry Pi4, ESP32 and STM32. Below, the usage will be examined of the combination of each of these and ultimately what best suits this project's goal.

3.3.1.1 Raspberry Pi4

The Raspberry Pi4 is a quad-core, 64-bit Linux based mini-PC ideal for graphics, UIs, and high-level data fusion (citation 56). The main advantage of the Pi4 in this context is its exceptional graphics and computing power alongside how simple it is to develop with. Since the Pi4 is a self-contained computer, compiling and flashing is very straightforward. Alongside this is the massive community and library support which can make finding help easier than many other MCUs. Yet, the drawbacks to this MCU come from non-deterministic timing due to the Linux scheduling algorithm employed. Another drawback is the slower speed due to the use of a Linux OS which is a heavier OS than FreeRTOS. Overall though, the Pi4 has the most capabilities when it comes to graphics but the least when sensor reading is needed due to the Linux OS providing better graphics libraries despite being slower.

3.3.1.2 ESP32

The ESP32 is a dual-core Wi-Fi/Bluetooth-enabled microcontroller that is perfect for lightweight tasks (citation 56). The main advantage of this MCU in this project's context is the FreeRTOS by default. This allows for easy multi-tasking with minimal setup or driver code written when compared with other OS usage. This MCU also supports good SPI/I2C/UART with DMA and has a huge community online so finding tutorials is very easy. The main problem comes from a noisy Analog to Digital Converter (ADC) and timing jitter. This makes exact control more difficult since the device cannot be 100% accurate when reading analog signals and timing. There is also limited RAM for graphics on this MCU which makes it's ability to act as the dedicated graphics MCU difficult. Overall, this MCU is best suited for a sensor bridge designed to read, calculate, and send data to the graphics MCU.

3.3.1.3 STM32

The STM32 is an MCU optimized for real-time, low-latency tasks (citation 56). It is regularly used in industrial and aerospace embedded systems due to the exact control and accuracy one can attain over the timing and ADC. This excellent reading allows for high accuracy and precise system control which is crucial in real-time systems. This MCU also supports FreeRTOS, or the developer can choose a bare-metal approach and code all OS code from the ground up. The main drawback comes from the steep learning curve related to getting started using this MCU. The software used to develop on an STM32 is called STM32CubeIDE and CubeMX which is notoriously difficult to use with a smaller online community to get help from. This MCU also is not ideal for high-resolution graphics due to its limited graphics hardware. Overall, the STM32 offers the most precise control over sensor readings and calculations but comes with a steeper learning curve than the ESP32

3.3.1.4 System Combination Selection

The purpose of each MCU must be decided and which MCUs are best suited to work together that also seem the most feasible to use in the senior design project. The Raspberry Pi4 is only suited to act as a dedicated graphics MCU due to its slow sensor processing speed because of the Linux OS. The ESP32 could act as the graphics unit or the sensor processing unit but will be mediocre at both. The STM32 is ill-suited for graphics but would work very well as a sensor unit if it were not so difficult to develop. Given these conclusions, the ideal combo would be a good graphics MCU with a good sensor MCU. That means that the three combinations of MCUs would be: Raspberry Pi4 + ESP32, Raspberry Pi4 + STM32, and ESP32 + STM32. The following table compares the three MCU combinations and determines which combination is best suited for this project's purposes.

Table 32: MCU Combination Comparison [56]

Factor	Pi4 + ESP32	Pi4 + STM32	ESP32 + STM32
Graphics	Highest	Highest	Lowest
Clock Speed	Low	High	Highest
Latency	Medium	Low	Lowest
Learnability	Highest	Low	Lowest
Feasibility	Highest	Low	Lowest

When comparing these microcontrollers, not all desired factors are equal when weighing them against one another due to the timeline of this project. For this project, learnability and feasibility are some of the most crucial components when deciding on which technologies to choose. Due to the limited resources available for working with the STM32 and the amount of manual setup requirement needed for the STM32, this makes it impractical for use in this project. Therefore, the clear choice for the MCU combination is the Pi4 + ESP32 despite having weaker timing and speed when compared with the Pi4 + STM32. The Pi4 + ESP32 overall has the highest ranking when all factors are considered and are straightforward to implement with more minimal accuracy concerns.

3.3.2 Software Roles & System Context

Before discussing specifics of the software technologies this project uses, context surrounding the usage of the MCUs is needed. To start, sensor data will be read from the sensors by the ESP32 microcontroller. This MCU was chosen for reading the sensors due to it being fast and lightweight, making it perfect for low latency sensor reading and calculations. The ESP32 will read data from the IMU and altimeter and compute the desired orientation, compass heading, and altitude data. From there, the ESP32 will send this data to the Raspberry Pi4 for parsing and display on the GUI. The responsibility of the Pi4 is primarily for the formatting and display of the GUI and receives sensor data from the ESP32 telemetry line. The Pi4 will be responsible for the live-camera feed if

that goal is hit. This was chosen to allow for as little latency as possible when utilizing the live camera feed. From the Pi4, the GUI with all formatted data will be sent to the LCD for display on the HUD. The general flow of data is as follows:

Camera Feed ↘
IMU & Altimeter → ESP32 → Raspberry Pi4 → LCD

3.3.3 ESP32

3.3.3.1 Operating System

3.3.3.1a FreeRTOS

FreeRTOS is the real-time operating system kernel integrated into the ESP32-IDF Standard Development Kit (citation 57). This free real-time operating system controls process execution on the ESP32. FreeRTOS comes with task (thread) scheduling using priority and stack-based scheduling algorithms. This OS also has primitives to control thread execution such as queues, semaphores, mutexes, and event groups. Timers are also included in this OS to allow for interrupts. FreeRTOS is used on the ESP32 when a program is flashed on to the MCU the FreeRTOS kernel is linked into your firmware. Basically, whenever you build your program from the Arduino IDE, it also flashes FreeRTOS on to the ESP32.

3.3.3.1a “Bare-Metal”

The other option for an operating system would be the “bare-metal” approach. This approach requires manually programming all OS code including the startup code, setting clocks, initializing RAM/stack, setting up the interrupt vector table, writing all drivers for UART/I2C/SPI, timers, etc. (citation 57) Basically, any pre-programmed OS routines needed would need to be coded from the ground up. Yet, the advantage to this approach is precise system control and no overhead computation from an OS. Since the programmer must make all OS routines, they only need to make and run what is necessary. The general flow of using a “bare-metal” approach is to run the ROM bootloader, use startup code to set up things like the stack, then initiate peripherals such as the GPIO and UART, then begin the main superloop, and finally use Interrupt Service Routines (ISRs) to handle any data interrupts. The difference between using FreeRTOS and a “bare-metal” approach is given in the table below:

Table 33:OS Comparison [57]

Aspect	Bare-metal (superloop + ISRs)	FreeRTOS on ESP32
Concurrency model	Worse	Better
Timing & jitter	Better	Worse
Complexity	High	Low
Latency	Lowest	Low
Resource use	Lowest	Higher
Drivers & stacks	Worse	Better
Debugging	Lowest	Higher

Dual-core usage	High difficulty	Low difficulty
Power mgmt / idle	High difficulty	Low difficulty
Best fit	Worse	Better

By comparing the “bare-metal” and FreeRTOS, there are various tradeoffs and advantages to each. The main advantage of “bare-metal” is the low-overhead and low latency. If the system needs the lowest possible latency and lowest possible overhead, the “bare-metal” approach is the superior approach. Yet, the ESP32 and P.E.G.A.S.U.S. does not have these restrictive constraints and allow for some extra processing latency and storage overhead. The main constraint for this project is completion time, if there was more time a “bare-metal” approach may be viable due to the desire to achieve low latency but given the timeline of this project being rather short and the increased latency and overhead being minimal, FreeRTOS is the better approach for an OS for the ESP32.

3.3.3.2 Programming Language

There are three programming languages that will run on the ESP32 all with their own distinct advantages and disadvantages. These three are C, C++, and Python. The goal for this project is low latency but also the ability to complete it in a shortened time frame. Practicality and system latency requirements will be the main pieces discussed in relation to the selection of a programming language.

3.3.3.2a Python

Starting with the highest-level language Python, this language is by far the easiest to use. Even though using Python on the ESP32 is not actually Python but rather an interpreter in the firmware named MicroPython, it has its advantages (citation 57). Python is very fast for iterations and prototyping with very beginner friendly syntax. In turn, Python is a very good learning language and allows a new programmer to learn embedded systems easier than C or C++. The downside to Python though is that it is very slow when it comes to reading and processing sensor data. Python is an interpreted language and therefore spends a lot of clock cycles interpreting Python code into executable code. Not only is it slow, it also uses the interpreter and heap for memory which reduces room for buffers and logs. Python is great for rapid development, scripting, or even graphics but it pales in comparison to its counterparts when low latency is a primary desire.

3.3.3.2b C++

Going down one level brings in the usage of C++ which is an extension of the C language that adds object-oriented programming features like classes, objects, and inheritance. C++ grants the ability of zero-overhead abstractions which allow the developer to write code “using classes, template, and inline functions that, when compiled well, are just as fast as plain C” (citation 57). Unlike Python with heavy overhead for its use, C++ compiles to C which is the faster of all of our options. C++ utilizes Resource Acquisition is Initialization (RAII) which means that serial ports or buffers are automatically cleaned up when objects go out of scope. This means better memory handling than C due to no requirements to manually release memory. C++ is also perfectly interoperable with C by its ability to call C functions by wrapping them in lightweight C++ headers that allow the developer to mix new C++ code with existing C drivers. The downside to C++ though is

mostly related to its advantages. For example, by allocating memory at runtime with RAII, memory allocation is less predictable and can be slower and lead to a larger system overhead. Other bottlenecks can come from exceptions, which are also called try/catch blocks which allow for the developer to write code to handle specific errors in a graceful way instead of ending the program execution. However, exceptions can slow the code down by adding hidden stack usage. Overall, C++ is faster than Python and slightly slower than C with less memory usage than Python but more than C. The main advantage comes from C++ being easier to use due to advantages like RAII and its object-oriented capabilities.

3.3.3.2c C

The lowest level programming language is C which is by far the fastest with the lowest overhead. C compiles directly to efficient machine code with no interpreter or hidden layers like in Python and C++. This makes C extremely fast and occupy low memory and CPU usage due to the conversion directly into machine code. However, C requires manual memory management and if not handled correctly can lead to memory leaks or crashes. While manual memory management can lead to less memory usage if done correctly, it can also slow the system down if done incorrectly by occupying more of the heap. This can lead to increased clock cycles for a large heap size (ChatGPT, 2025). C also has no tools to organize large coding projects making C hard to scale for larger projects. There are also no built-in exceptions which means that if there is one bug in the whole code, the entire execution fails. Overall, C is the fastest and has the lowest overhead out of all three but requires the most manual usage as well which adds time to development. In turn, it is also not the easiest to use due to the manual usage requirements.

3.3.3.2d Comparison and Selection

Given all these trade-offs, it is best to organize them into one concise table which is given below:

Table 34: Programming Language Comparison [57]

Language	Speed	Stability	Ease of Use
Python (MicroPython)	Lowest	Lowest	Highest
C++	High	Highest	High
C	Highest	High	Lowest

In this project, the desire is to have the highest speed with the best stability and ease of use. By examining the table above, the clear choice for programming language is C++ due to it having high speed, the highest stability, and high ease of use. This makes it the only programming language that is not the lowest in one of the three desired categories.

3.3.3.3 Firmware Framework & Integrated Development Environment (IDE)

Another important selection is the IDE used for development. For the ESP32 there are a few different choices, these being the ESP-IDF (Espressif IoT Development Framework), the Arduino-ESP32 Core, and Platform IO. The ESP-IDF is the official SDK from Espressif built around FreeRTOS. Arduino-ESP32 Core wraps the ESP-IDF for ease of use. PlatformIO is a build system that supports IDF or Arduino (GPT). In the following sections, the various tradeoffs will be discussed and finally a selection will be made on which IDE is best suited for this project.

3.3.3.3a ESP-IDF

“ESP-IDF is the official SDK and build system from Espressif, written primarily in C and C++” (citation 57). ESP-IDF is designed to work seamlessly with the ESP32 and therefore makes it the only option for firmware frameworks for this MCU. This firmware framework builds directly on top of the chosen OS, FreeRTOS, and makes all hardware features of the ESP32 available for use. The framework uses CMake and Ninja to configure and compile firmware and provides access to all FreeRTOS APIs, task scheduling, and ISR (citation 57). By making all hardware feature available for use, this framework allows for maximum control and performance down to task timing and memory layout. This framework has access to low-jitter task loops and other advance features built in such as flash encryption. The most important part is that it is C and C++ compatible. The drawbacks to this framework are like almost any more manual control item, there is a steeper learning curve than the other frameworks due to the requirement of manual configuration of tasks, stack sizes, and pin maps. Due to the manual configuration requirements, this also means more verbose code. This framework is best used for final production firmware that requires predictable performance, multitasking, and hardware-level control (citation 57).

3.3.3.3b Arduino-ESP32 Core (IDE and Build System)

Arduino-ESP32 Core is an IDE that wraps around ESP-IDF as an abstraction layer for simpler development for students and hobbyists (citation 57). This IDE is Arduino system compatible and as such uses the Arduino build system for compilation and provides basic Arduino APIs that are wrapper functions for ESP-IDF drivers. Arduino-ESP32 Core also builds on top of FreeRTOS like ESP-IDF. The advantage of this IDE is that it is the easiest to set up and has the simplest syntax. There are also a large amount of external libraries built that integrate well with this IDE. The main drawback comes from it being a higher-level, simpler IDE which means less manual control of task priorities and memory allocation. This also means more tasks are abstracted and hidden which makes debugging latency or resource conflicts more difficult (citation 57). This IDE is best for rapid prototyping and simple applications that do not require intense resource management. Overall, the main advantage of this IDE is the simplicity of its use but its main drawback comes from this simplicity which does not allow for manual resource control like other IDEs do.

3.3.3.3c PlatformIO (IDE and Build System)

Platform IO is a cross-platform IDE extension that supports multiple frameworks, including the Arduino-ESP32 and ESP-IDF (citation 57). This IDE also acts as a wrapper for the chosen IDE such as Arduino or ESP-IDF. This IDE is used inside a

larger IDE such as VSCode which enables more modern code-navigation and debugging. This IDE is very easy to use and is completely cross-platform and works on Windows, macOS, and Linux which makes it ideal for working in a development team. The unified build system also allows switching between Arduino and ESP-IDF with a single config file. The downside though is the added abstraction layer which makes it more complicated when troubleshooting low-level build issues.

3.3.3.3d Comparison and Selection

The choice comes down to the use of PlatformIO and Arduino-ESP32 Core both at their base using the ESP-IDF framework. Both IDEs have their advantages and disadvantages summarized in this table:

Table 35: IDE Comparison [57]

IDE	Abstraction Level	Ease of Use	P.E.G.A.S.U.S. Use Case
Arduino-ESP32 Core	Highest	Highest	Low
PlatformIO	Lowest	Low	Highest

The main need for the IDE is to have the lowest level of access. This makes up for having a low ease of use therefore making the PlatformIO IDE extension the best use for this project.

3.3.4 Raspberry Pi4

3.3.4.1 Operating System

Just like for the ESP32, the Raspberry Pi4 also needs a specific operating system suited to the needs of this system. Primary concerns related to selection an operating system are boot time, resource usage, real-time responsiveness, and ease of use. The choices for an OS on the Raspberry Pi4 are slightly more numerous than on the ESP32 but not by much. These choices are the Raspberry Pi OS, Ubuntu Server 22.04 LTS ARM64, and the Yocto Project (custom build). All these operating systems have their own advantages and disadvantages that will be discussed in this section.

3.3.4.1a Raspberry Pi OS (Lite)

The Raspberry Pi OS is the native OS built by the Raspberry Pi Foundation based on the Debian Bookworm OS (citation 57). This OS comes with a desktop with slow boot time or an optional lite version with no desktop for projects that require a fast boot time and low resource usage. This OS is also primarily designed for the hobbyist or educator. The primary advantage to this OS is how easy it is to use and setup while still being relatively lightweight due to the optional lite version. The other primary advantage is that since this is the native Pi OS, it is tuned specifically for Pi hardware and communication protocols such as UART, SPI, and I2C as well as HDMI all work out of the box. Yet, like all high-level and easy to use technologies, this OS lacks control and is not strictly real-time. The

Linux scheduler introduces ~tens of milliseconds of latency fluctuations which could cause problems in the data update speed. This OS is not as fine-tuned as other OS options since it comes with more pre-programmed processes that run in the background. Overall, this OS is ideal for projects using the Pi hardware due to its fast setup, ease of use, and relatively lightweight resource usage for how high-level it is.

3.3.4.1b Ubuntu Server 22.04 LTS ARM64

The Ubuntu Server 22.04 LTS ARM64 OS is built and maintained by Canonical Ltd and is also Debian based with Ubuntu repos impacting the OS as well (citation 57). The primary usage for this OS is for servers or internet of things (IoT) devices and this OS does not have any kind of default desktop attached to it. The primary advantage of this OS is the familiarity most developers have with the Ubuntu OS making it very easy to begin developing with. There is also very good networking including SSH abilities and uses the same APT system as the Pi OS. However, there are not many more advantages to this OS as it is heavier than the other operating systems. It has a slower boot time and a larger memory footprint making it not ideal for real-time usage or systems that require a low boot time. There is also no drivers for Pi-specific peripherals and would need to be manually installed for use of peripherals. Another huge disadvantage is that there is no default GUI and would also need to be installed manually. Overall, this OS is best suited for networking and not ideal for usage with graphics despite the familiarity most developers have with it.

3.3.4.1c Yocto Project (Custom Build)

The Yocto Project OS is a community project that is customized by each user for their use with only buildroot and some layers being provided (citation 57). The main function for this OS is for highly custom embedded images that require the lowest possible boot time, lowest overhead, and fastest latency. The primary advantage comes from this OS being fully customizable meaning it has the lowest latency, lowest resource usage, and fastest boot time due to no external processes running. The only processes that run are the ones programmed by the developer to run. This OS can be used in mass-production as well since once it is made, a developer simply has to copy it to another MCU and it would work. However, like any low-level technology that allows for complete control of the system, this comes with the steep learning curve associated with that control. All drivers and layers must be coded from the ground up which requires a lot of time and knowledge to do. This OS also has poor debugging tools out of the box so fixing bugs is even more tedious. This combined with an already difficult system to work with makes the custom build ideal only for experts or those who need to minimize time. Overall, this OS is best suited for those who need the lowest possible time and resource usage, not the casual or entry-level developer on a shortened time window.

3.3.4.1d Comparison & Selection

The choices are once again a choice between lowest latency with highest difficulty of use or more latency with easier difficulty of use. The comparisons of the key factors to be considered are summarized in the following table:

Table 36: Pi OS Comparison [57]

Operating System	Boot Time	Resource Usage	GUI Compatibility	Ease of Use
Pi OS	High	Moderate	Highest	Highest
Ubuntu	Highest	High	Low	High
Yocto	Lowest	Lowest	Lowest	Lowest

Since this project wants to maximize ease of use with GUI compatibility and lower resource usage, the ideal choice for the operating system is the Pi OS. This OS has high to moderate boot time and resource usage with the highest GUI compatibility and highest ease of use. Making this the clear choice for the OS on the Raspberry Pi 4.

3.3.4.2 Programming Language & GUI Framework

Now that an OS is selected, similar to analyzing the ESP32 a programming language must be selected for use on the Pi4. The difference here is the Pi4 is used primarily for the GUI therefore it is unfair to consider only a programming language without also considering the available GUI frameworks as well. The main options are once again related to using C, C++, or Python. These being C + SLD2 + OpenGL ES, C++ + Qt, and Python + PyQt. The tradeoffs of using each of these will be examined then a decision will be made depending on which is best suited for this system.

3.3.4.2a C & SLD2 & OpenGL

SDL2 is a C framework that was originally developed for game development and is ultra-fast and has minimal overhead. OpenGL is a low-level GPU API that requires manual drawing of vertices, shaders, and textures (citation 57). The primary advantage of this combination is best performance with the lowest level of abstraction and direct hardware control. In turn, this makes the C based graphics libraries very fast and requires minimal overhead. The downside though is the limitations that are placed by having direct low-level access. Since there is no abstraction, the development is more difficult with these packages with all animations, lines, and curves needing to be hand drawn. This can make development more difficult and more time-consuming, especially for larger projects. This combination of packages and C is best suited for advanced game engines and aerospace companies due to its excellent performance and difficult use.

3.3.4.2b C++ & Qt Quick (Qt/QML)

Qt Quick is a high-performance embedded GUI framework that has its own scene graph and is designed for usage in HUDs (citation 57). This graphics framework has very fast performance by allowing for direct hardware access with minimal abstraction layers. This framework also supports direct-to-framebuffer support which allows it to launch full screen without a desktop manager leading to a faster boot time and lower latency (citation 57). There are many built animations and shapes in this graphics library which also significantly cuts down on development time. The downside to this framework is a slightly steeper learning curve than the Python based frameworks but less of a learning curve than the C based frameworks. This learning curve is mostly induced by the manual setup requirements for compiling and using the libraries. It also requires a solid

understanding of using threads in computing which could lead to development stall if those concepts are not familiar. Overall, this combination is very well suited for any heads-up display due to its speed combined with minimal abstraction and a lower learning curve than C based graphics frameworks.

3.3.4.2c Python & PyQt

PyQt is the same bindings as the previously examined Qt but uses Python instead of C++. Effectively, PyQt abstracts the C++ library in to Python code and simply converts it back to C++ on execution (citation 57). This library is very popular for prototyping due to its rapid development speed and ease of use. Making PyQt by far the easiest of the three graphics packages to get started with. Despite its ease of use, PyQt suffers from the main drawback of Python which is that it is slow and requires a lot of overhead. Since Python is an interpreted language, this means there are a lot more resources dedicated to interpreting this code down to a level the machine can use. Making Python and PyQt ill-suited in this use of this project and more suited to rapid prototyping.

3.3.4.2d Comparison & Selection

Now that each languages advantages and disadvantages have been examined and what their best suited use case is, it is time to summarize these ideas in to one table. This table is largely focused around the consistent goals of ease of use, resource usage, speed, and boot time.

Table 37: Pi Programming Language Comparison [57]

Language & Framework	Ease of Use	Resource Usage	Speed	Boot Time
C++ & Qt	High	Low	High	Low
Python & PyQt	Highest	Highest	Highest	Highest
C & SDL2	Lowest	Lowest	Highest	Lowest

By examining the table and weighing ease of use as a heavy requirement, it is clear that C++ & Qt is the best choice. This comes from this combination having the highest ease of use with the lowest resource usage and boot time with a high speed.

3.3.4.4 Data Logging

The selection of data logging in the HUD is primarily needed for debugging purposes. Without an ability to see data as the system is running or on the case of a crash, that would make debugging very difficult since there would be no indicator as to where the program failed or what exactly went wrong. Thus, making the selection of a data logging service vital to fixing problems in the project. The main choice for data logging is SQLite and CSV files due to them being relatively lightweight yet still very useful for debugging.

3.3.4.4a SQLite

SQLite is a lightweight, file-based SQL database that is frequently used anywhere lightweight systems are used such as phones and avionics (citation 57). Since SQLite is a

SQL based storage system, it integrates seamlessly with C/C++ through libraries. SQLite comes with all the advantages of the structured database system like enforcing schemas for structure in data storage as well as fast querying. SQLite integrates perfectly with C++ and Qt making it a strong choice for use in this project. However, despite these advantages, SQLite is heavier in storage requirements than CSV files and require more CPU usage.

3.3.4.4b Comma-Separated Values Flat File (CSV)

“CSV is a plain text format where each row is a line and each column is separated by commas” (citation 57). This makes for simple and readable files for all to use. The main advantage of CSV files comes from their simplicity, they have very minimal overhead and have very fast writes. Alongside the minimal resource usage comes the easy readability for humans. However, despite their ease of use, CSV files are more difficult for larger databases since they have no search mechanism like SQLite has. Finding data can be a challenge if that is the goal. There is also a data corruption risk if the system loses power mid-line, there could end up an incomplete row of data. Another downside is that CSV files require manual reading and writing instead of using libraries like SQLite allows.

3.3.4.4c Comparison & Selection

Now that there is some background on SQLite and CSV files, these advantages and disadvantages will be summarized in a table. The primary concerns are writing speed, scalability, query support, ease of use, human readability, and integration with Qt/C++.

Table 38: Data Logging Comparison [57]

Criterion	SQLite	CSV
Write Speed	High	Highest
Scalability	Highest	Low
Ease of Use	Moderate	Lowest
Human Readability	Low	High
Integration with Qt/C++	Highest	Low

Based on this table, SQLite is the clear choice for data storing. This is due to SQLite having a fast write speed with moderate ease of use. Combining these with easy integration and scalability, this makes SQLite the ideal choice for the P.E.G.A.S.U.S. project.

3.3.4.5 IDE

Since PlatformIO has already been chosen as the selected IDE extension, there is no need to compare other IDEs for the Pi4 since Qt can also be used as an extension through Qt Tools. The primary and only real option is VS Code for the IDE that these extensions will run in. VS Code is a universal standard IDE and is one the developer of this project is very familiar with. VS Code integrates seamlessly with PlatformIO and QT Tools

along with CMake extensions allowing it to become a full embedded IDE. While other IDEs such as Qt Creator or CLion could be used, these pale in comparison due to them either being slow, requiring a paid subscription, or one of the biggest reasons, requiring the usage of two different IDEs for development. Qt Creator and CLion cannot use both PlatformIO and Qt Tools which would mean the switching of IDEs. To avoid this and remain in the familiar IDE range, the developer has chosen VS Code to develop this project in.

3.3.4.6 Graphics Rendering

Now that the IDE and programming language have been chosen for the Pi4, it is important to discuss the rendering technology needed. A rendering technology is needed to ensure that the code can actually render the GUI and deliver deterministic, GPU-accelerated, low latency graphics performance (citation 64). Below, Qt Quick with EGLFS will be compared with Raw OpenGL ES for the selection of a graphics rendering package.

3.3.4.6a Qt Quick with EGLFS (DRM/KMS)

“Qt Quick is a high-level, GPU accelerated UI framework built on the Qt Scene Graph, which provides a declarative language (QML) for creating visual interfaces. EGLFS (EGL Full Screen) is Qt’s embedded platform backend that bypasses window systems and renders directly to the framebuffer using EGL + OpenGL ES” (citation 64). In this system, this package handles vector HUD symbology as well as animations and UI bindings. The Qt Scene Graph is then able to translate HUD elements into GPU draw calls. While EGLFS talks directly to DRM/KMS and the GPU so that no desktop environment is required. That way, the application is able to run in full-screen kiosk mode with vsync and page flipping (citation 64). This technology has high development speed, easy use, low latency, and easy integration with the rest of the system. Yet, the downside to this is that there is little fine-grained control meaning no true RTOS level determinism or customizations.

3.3.4.6b Raw OpenGL ES

“OpenGL ES (GL ES) is a low-level graphics API providing direct control of the GPU pipeline (shaders, buffers, textures). It is the foundation for many embedded displays, mobile systems, and avionics graphics engines” (citation 64). Raw OpenGL ES is used to manually define all rendering logic including vertex buffers, shaders, and draw cells. Any rendering of graphics is done entirely manually by the programmer. This allows for very fine-grained control with the lowest possible latency producing RTOS level determinism. Yet, this software takes a very long time to use in development due to the requirement of building everything from scratch. This software is also difficult to learn as there are more minimal resources available.

3.3.4.6c Comparison & Selection

Now that there is background information on the two GUI rendering technologies, a selection needs to be made to determine which is best suited for this system which will be shown in this table:

Table 39: Graphics Rendering Technology Selection [64]

Criterion	Qt Quick with EGLFS	Raw OpenGL ES
Render Latency	Low	Lowest
Feasibility	Highest	Lowest
Real-Time Determinism	High	Highest
GPU Efficiency	High	Highest

By examining the above table, it is clear that Qt Quick is the best solution due to it having low latency with the highest feasibility while still maintaining high determinism and efficiency.

Ch 4: Standards and Design Constraints

4.1 Optical Standards

4.1.1 ISO 10110 – Optical Drawings Standards

ISO 10110 is an international standardization for the dissemination of technical information as it pertains to optics and photonics, with a particular emphasis on drawings and schematics. In these standards, there are specifications on how to present certain information such as component dimensions, surface form, imperfections, and coatings. Since the scope of this aircraft heads-up display involves using an optical projection system comprised of various optical components as well as a photonic light source, this set of standards is highly relevant. Adhering to these standards is very important as the essential engineering technicalities will have to be communicated in a particular manner that can be understood by an optical engineer. For example, key design parameters, such as focal length, radii of curvature, and optical element diameter will have to be presented effectively.

When dealing with specific component aspects, such as these aforementioned parameters, it is important to cite them and refer to them according to the ISO 10110 standard. When presenting various elements such as lenses and mirrors, critical components of light manipulation in our polarized projection system, they must be visualized and illustrated according to the standard. In addition, key parameters for optical designs such as wavelength or range of wavelengths of projected light, source parameters, test fields, and materials must adhere to the ISO 10110 standard. It should be noted that according to the standard, the preferred layout of a drawing of optical elements or subassemblies is the tabular format. This is because of the fact the international standardization organization (or ISO) has determined that tabulating optical element data is the most effective way to present this information. This allows for consistencies across great numbers of drawings and large amounts of information to be digested in a manner that is manageable and consistent. It is also helpful to manufacturers, partners, and any suppliers / vendors that are trying to properly understand the system as well as its needs as they pertain to optical elements.

Some key components that must adhere to these standards are our specific optical manipulation components such as the system’s collimating / magnifying lenses, mirrors, and optical combiner. In order to effectively guide the light output from our polarized display to the user’s eyes, we must ensure the proper parameters of all of these components. Things like effective focal length, lens and mirror dimensions, and beam expansion are all design parameters that must be specified in accordance with ISO 10110. By incorporating the ISO 10110 standards into the design process, we can ensure that the optical components in our heads-up display for an aircraft meet the necessary internationally agreed-upon requirements for integration, reliability, performance, and compatibility. This will result in a more robust system that will help increase situational awareness in pilots.

4.2 Software-Related Standards

4.2.1 RTCA DO-178C

RTCA DO-178C is the “Software Considerations in Airborne Systems and Equipment Certification” and in turn is the primary global standard for developing safe and critical aviation software (citation 58). This standard ensures that all aviation software has the proper rigor associated with it to ensure that the software remains reliable, fault tolerant, and deterministic so that pilots or passengers are not endangered. As such, this is a very important standard for the P.E.G.A.S.U.S. project. This standard provides a framework for development, documentation, and testing software for avionics.

This standard was developed in the late 1970s and early 1980s as avionics transitioned from analog electronics to digital flight computers (citation 58). This standard has evolved slightly over time and is currently on the fourth edition with the following changes:

Table 40: DO-178 Version History [58]

Version	Release	Motivation
DO-178 (1982)	1 st Edition	Addressed the emerging role of airborne digital software but lacked structure and clarity.
DO-178A (1985)	2 nd Edition	Provided more guidance, but insufficient verification rigor.
DO-178B (1992)	3 rd Edition	Became the worldwide cornerstone for certification; introduced Software Levels (A-E) and structured verification objectives.
DO-178C (2011)	4 th Edition	Modernized the standard to address model-based development, formal

		methods, and object-oriented software, while improving clarity and reducing ambiguity.
--	--	--

This standard is primarily rooted in the philosophy that “Software used in aircraft must not contribute to failure conditions that could compromise safety” (citation 58). To ensure this philosophy is enforced, DO-178C emphasizes determinism, traceability, verification independence, no single point of software failure, and prevention of unintended functionality. These rules must exist since airborne software cannot “break” mid-flight, they are essential to the operation of the aircraft and if they break, that is a massive danger to all involved in the flight. As such, there are the following level of potential failure severity called the “Design Assurance Levels (DAL)”:

Table 41: Design Assurance Levels [58]

Level	Failure Effect	Required Rigor
A	Catastrophic (loss of aircraft)	Highest rigor (63 objectives)
B	Hazardous/Severe	Very high rigor
C	Major	Moderate rigor
D	Minor	Reduced objectives
E	No safety effect	No DO-178C involvement

“Software that displays attitude, heading, terrain or flight guidance cues typically falls between DAL-B and DAL-C because incorrect symbology or delayed updates could mislead a pilot and contribute to hazardous conditions” (citation 58). Despite this project not being commercially certified, this should not stop us from designing with real-time constraints, fault-tolerant software, human-factors correctness, and predictable sensor-fusion behavior in mind. The desired principles and their application to this project are summarized in the following table:

Table 42: DO-178C Principles Related to P.E.G.A.S.U.S. [58]

DO-178C Principle	HUD Application
Deterministic timing	Consistent IMU/GPS update rates and refresh latency
Rigorous verification	Ensure attitude and heading data is never misleading
Traceability	Each computed symbology element ties to a system requirement
Fail-safe behavior	HUD must degrade safely if inputs are lost or invalid
Independence of verification	Unit-testing and fault-injection separate from coding

DO-178C is a vital standard in the development of aviation software to ensure safe systems are implemented in aircrafts. While this standard was originally created to address the shift from analog to digital computers, it has become a vital standard for enforcing safe and deterministic software. This standard underlies all avionics software and is vital to implement in this project to ensure compliance with safety standards.

4.2.2 MISRA-C

MISRA-C is a general standard to produce safe code in embedded systems. Developed in the early 1990s by Motor Industry Software Reliability Association (MISRA), it “is one of the most widely adopted coding standard for safety-critical embedded C software” (citation 58). Despite its origins in the auto industry, MISRA-C is now used in all industries for systems that cannot risk undefined behavior, memory corruption, race conditions, or unexpected hardware states. The evolution of MISRA-C has allowed it to adapt as the industry has changed over time.

Table 43: MISRA-C Version History [58]

Version	Release	Purpose
MISRA-C:1998	1 st Edition	Prevent unsafe C features and undefined behavior in embedded systems
MISRA-C:2004	2 nd Edition	Improved clarity and adopted aerospace and defense industries
MISRA-C:2012 (Current)	3 rd Edition	Added stronger safety rules, modern C support, and better rule classification
2019/2023 Amendments	Updates to MISRA-C:2012	Enhanced compatibility with ISO C standards and clarified specific rule language

Since C is inherently an unsafe language to use with null pointers, unrestricted casts, dynamic memory pitfalls, etc. this standard was made to add some safety to the C code. The goal of this standard is to “eliminate sources of undefined or unpredictable behavior” (citation 58). To do this, certain rules are enforced such as:

- Restricting dangerous C constructs (ex: recursion, dynamic memory allocation, unbounded loops)
- Encourage analyzable and testable code
- Guarantee predictable timing and execution paths
- Support static analysis and traceability
- Improve long-term maintainability

The overall goal of MISRA-C is to ensure software is controlled, consistent, and verifiable, it is not about measuring the correctness of the output. MISRA-C has three

levels of rules relating to how seriously each rule should be taken. If a rule is mandatory, it must always be followed without exception. If a rule is required, it should always be followed unless a formal justification to not follow it is presented. If a rule is advisory, it should be followed but can be relaxed when appropriate to do so. In relation to this project, the key MISRA-C objectives are:

Table 44: Relevant MISRA-C Objectives [58]

MISRA-C Objective	HUD Software Relevance
Predictable timing	Ensures update loops for symbology and filtering execute deterministically
Safer pointer and memory usage	Prevents firmware crashed that could blank or freeze the HUD
Static analyzability	Allows verification that no undefined behavior exists in the flight loop
Maintainability and traceability	Supports the DO-178C aligned verification strategy

MISRA-C is a vital standard across all industries that use embedded software. To ensure safety across vital systems, certain standards related to C and C++ must be made hence the usage of MISRA-C. This standard aligns with DO-178 as well to ensure the delivery of testable and predictable airborne software.

4.2.3 MIL-STD-1472

MIL-STD-1472, also titled “Department of Defense Criteria Standard: Human Engineering” is a widely adopted standard that controls how information is presented visually to human operators. First created in 1968 and revised over the decades, this standard provides rules so that equipment interfaces (such as heads-up-displays) are intuitive, readable, and safe. Despite this standard being primarily developed for military aircrafts, it has influenced all aspects of software development that has human-machine interfaces. The table below shows how MIL-STD-1472 has evolved over time:

Table 45: MIL-STD-1472 Version History [59]

Revision	Approx. Era	Key Improvements
MIL-STD-1472A/B	Early avionics integration (1970s)	Initial cockpit HMI rules, display visibility criteria
1472 C/D	Glass cockpits emerge (1980s-1990s)	Emerging digital display guidance, workload limits
1472 E/F	Modern HUD/HMD era (2000s-2012+)	AR-based symbology, lightning, screening, usability, cognitive load control

This standard has evolved to focus on a specific premise: “A pilot must be able to perceive, interpret, and act on displayed information rapidly, accurately, and with minimal cognitive workload” (citation 58). To ensure this, this standard focuses on specific human factor areas such as readability, contrast, information hierarchy, standardizing symbols, workload reduction, and parallax and alignment. By focusing on these, the design of the P.E.G.A.S.U.S. GUI can be formatted in a way that is safe and helpful for the pilot. The following objectives will be focused on in relation to this project:

Table 46: Relevant MIL-STD-1472 Rules [59]

MIL-STD-1472 Rule	HUD Interpretation
DON'T clutter the pilot's focal area	Limit symbols to essential flight data (attitude, altitude, heading)
Critical data must be instantly readable	Stable fonts, no over-animation or jitter
Maintain consistent symbol geometry	Horizon line and pitch ladder spacing must be uniform and intuitive
Use proper luminance and contrast ratios	Software should provide adjustable brightness modes
Avoid misinterpretation under stress	No ambiguous symbols or overlapping visual effects

This standard is especially relevant to this system so that the pilot is not overwhelmed by information in the GUI and can read and interpret data easily. By ensuring that data is formatted simply and safely, the P.E.G.A.S.U.S. can display data helpfully and ensure no safety issues for the pilot.

4.2.4 I²C / SPI Communication Protocol Standards

Despite both I²C and SPI being basic communication buses, these standards are “governed by long-standing electrical and timing standards that establish how digital devices must exchange data reliably in embedded systems” (citation 58). These standards are some of the basic ones learned in most introduction to embedded systems courses because they are so widely used. Many sensors rely on these standards to communicate their data through.

4.2.4a I²C

I²C was created by Phillips Semiconductors in the early 1980s with the idea to provide a simple and efficient method for communication between ICs on PCBs (citation 59). This standard uses a two-wire interface utilizing a clock line called the SCL and a data line called the SDA. It relies on open drain signaling with pull-up resistors operating between 100 kHz and 3.4 MHz. I²C also defines requirements for “setup time, hold time, rise/fall time, voltage thresholds, and bus arbitration behavior” (citation 58). The primary advantage of I²C comes from it only needing two wires. This allows for the reduction of wire complexity in constrained systems. The deterministic nature of communication due

to the usage of a clock line is also crucial as it allows for reliable and predictable data transfer.

4.2.4b SPI

SPI was originally developed by Motorola in the mid-1980s to enable communication between MCUs and peripheral devices (citation 59). Today, it is governed by manufacturer-maintained timing standards and has become the standard industry specification for high-speed embedded communication (citation 58). SPI uses four lines that control SCLK, MOSI, MISO, and a chip-select that allows full-duplex communication with very little latency. By defining clock polarity (CPOL), clock phase (CPHA), and minimum timing margins, SPI ensures valid sampling. SPI operates typically at 200-1000 Hz for measurements of components such as the IMU. SPI is advantageous for low jitter, low overhead, and deterministic bit-clocking sensors.

4.3 Constraints

4.3.1 Software Real-Time Performance & Latency

Ensuring real-time performance is one of the most crucial constraints of this system as it feeds into many different aspects of this system. Ensuring continuous updates primarily means no disorientation for the pilot and no overcorrecting or under correcting due to stale data. Typically, humans perceive visual lag when displays update at slow than 50-100ms. Therefore, ensuring that the HUD updates below this threshold or as close to it as possible is essential for continuous and smooth display movement.

Ensuring the HUD updates below the given threshold can prevent “latency sickness” which is a kind of motion sickness caused by delay between physical movements and VR technology, typically seen in VR headsets (citation 63). If the HUD data is lagging behind movement in the aircraft, the pilot may become disoriented which could prove dangerous while flying. The other problem besides human physiology is related to correction of flight path based on stale data. Pilots frequently make updates based on the most recent data, therefore if data is out of date on the HUD by even a few seconds, this could lead to dangerous conditions for the pilot.

To ensure that data is current across all sensors, it is important to design with the slowest loop in mind. For example, if an IMU updates at 100-1000 Hz but a GPS only updates at 5-10 Hz, it is important for the HUD to be designed with this knowledge in mind so that the pilot does not see mismatches between GPS heading and IMU heading. This alongside ensuring determinism in the system to ensure consistent updates are crucial to the success of this system. To accomplish this, “lightweight algorithms, efficient data structures, predictable floating-point operations, and protocol configurations that guarantee bounded transmission time” are essential to ensure minimal latency.

4.3.2 Time

Time has been one of the most influential constraints on this project. Having only two semesters to design, develop, and test this project leads to many choices related to fastest

development vs best use. As shown in the technologies section, time to learn and use has been of the most crucial factors to consider. This has led to choosing faster and easier options instead of longer and more time consuming options.

The first semester of the senior design class is 16 weeks with the first few weeks being reserved for group formation and project proposal. This means that the first three weeks are not dedicated to working on the project but instead getting some ground work laid and an idea selected before any work begins. The final report is due one week before the end of the semester, meaning that there is only 12 weeks to complete the design and analysis at a prototype level. This leads in to the second semester where work begins right away with the expectation of having a fully working and tested product before the end of the semester. Overall, this means there are between 24 and 28 weeks to design, document, develop, and test this system.

To abide by these time constraints, many technologies were selected due to them being easier to use but still suitable for this use case. For example, not using a “bare-metal” OS as opposed to using any preprogrammed OS was chosen due to the “bare-metal” approach requiring much more time and work that does not fit in the scope of the senior design project. Ensuring technologies are selected with feasibility in mind is an essential aspect of the development of this system.

On top of selecting appropriate technologies, ensuring communication is efficient between group members, faculty, and vendors is essential. By minimizing delays here, the project has a higher chance of completion since there is no delays in development from communication issues. This also contributes to faster development and testing by ensuring if one group member is stuck, they receive the appropriate aid from group members or faculty if needed. The vendor communication delays are slightly more out of the scope of the groups control but can still be minimized by responding quickly to vendors and following up with vendors if needed. By staying on top of deadlines, ensuring the selection of feasible technologies, and efficient communication, the senior design project has a very high probability of achieving all goals and delivering a working product.

4.3.3 Economic

Economic constraints have luckily not been too much of an issue, but it is definitely important to note. While our project is not sponsored and entirely student funded, we are responsible for paying for everything. This means that most of our concern has come during the hardware selection phase due to this being where most of the highly priced devices need to be selected. These are unavoidable charges and were accounted for in the budget. It is just important during the technology comparison phase to weigh cost in as an important factor. Some of the pricier parts so far are the two different MCUs required, one being the Pi4 and the other the ESP32. While both parts are essential and need to be purchased, trying to find them from a reliable seller and for the lowest price is crucial to staying under budget. The different sensors required are BMP390 and the ICM-20948. While once again neither of these are super expensive parts, each piece of hardware will add up quickly and trying to find a good one from a reliable seller is important.

Even more expensive than the hardware components are all of the different lenses required to make the HUD. The display needs to be high resolution and needs to be capable of displaying a clear image. Each one of the lenses is very expensive due to their complex design and use of expensive material that can drive up the price tag of the project quickly. If it is possible to borrow some of these lenses, this could be a real help in saving money for some other things.

Arguably, the biggest economic hit can come from any miscalculation or mistakes in powering hardware. If not careful, a board could be fried easily if not given the proper power supply or current and would require replacement parts or be ordered. These replacements could add up quick especially if entire PCBs needed to be replaced, which makes it even more important to double check circuits and to take the correct precautions when supplying power to lower the chances of causing any damage.

To negate all these economic issues building up to create budget issues, it is very important to take the correct steps in trying to limit expenses. Trying to find cheap parts from reliable buyers and trying to borrow some of the more expensive lenses are two good ways to limit spending when buying parts. It is also important to triple check everything in the implementation and design of the circuit to make sure you do not have to buy too many replacement parts. Taking this precaution can help keep spending below the budget and lower the effect of economic limitations.

4.3.4 Health and Safety

Arguably, the most important constraint is health and safety. While all the prior constraints also have a large effect on the project, nothing is more important than providing a safe and reliable product for the pilot. When picking devices and lenses while price is important, the reliability of the part is even more important. While the cheapest option is always tempting, it could also mean the part is not as reliable. Reading product reviews and other users' experiences is important to confirm before purchasing.

On top of buying reliable parts, it is important to do extensive testing before manufacturing a product. While our project is only required to be proof of concept, it is essential that it ever to be brought to market so that it is confirmed to work properly. Some of the measurements it provides such as altitude are not bit of data that can be wrong. If the conditions are foggy and pilots' visibility is low, the readings need to be accurate. The difference in feet can make all the difference in a safe landing.

On top of being accurate, it also needs to be fast. On the display, while they do not need to be exact in real time, they need to be fast enough for the information to mean something. The altitude and IMU readings for pitch cannot be updated seconds after the reading, or the data is useless.

While the health and safety factor really is not tested until the implementation parts of the project where the sensors are being tested and the code is checked to work properly the idea of this is always a looming thought through testing and part selection. It is crucial

that all the sensors work properly, and the HUD updates quickly if it is to be used in flight. The pilot's health and safety is ultimately always the biggest concern

Ch 5: Application of ChatGPT and Other Similar Platforms

5.1 Case Study 1

Please refer to Appendix E [1a] and [1b] for the relevant prompt as well as output to and from ChatGPT for this section. ChatGPT did a good job of providing a compact list of components for the given prompt of designing an example HUD.

There are two sides of the coin when thinking about the issue of the use of AI in engineering design work. As far as the pros are concerned, AI provides a useful foundation and context when it comes to designing certain aspects of a project as it does a good job of filling in holes and unknowns in the process. For example, this interface (ChatGPT 5) provides a clean and concise interface and uses proper visual aides to display the relevant component information for this prompt in a clear and digestible manner. In addition, the answer to this question of designing a product is well thought out and considerate. Included are all of the elements necessary for a successful heads-up display in a vehicle. Included are the first principles design requirements in the form of virtual image sizes, field of view, and eye box dimensions. These values are then extrapolated and combined with discovered data regarding various products to come up with a list of elements including a laser-illuminated microdisplay that is collimated, reflected, optically combined, and projected to present a clear and cohesive image to the end user, in this case a vehicle operator. The use of AI in this case is advantageous because it is clear that the model identified the first principles requirements of the system, came up with values of elements to accommodate these end values, and then added descriptions regarding some justifications for these values. In addition, the model provides component types, specifications, and different variants of components to successfully achieve the same end goals for this project. This is incredibly helpful when trying to understand what criteria to design for and what criteria are relevant in the end construction of a HUD system.

All of this being said, AI does not come without its drawbacks. If you look closely at the AI output for this prompt, you can notice some flaws and inconsistencies in the answer provided. One of the flaws of employing the use of AI in engineering work or any professional work in general is that the model can be subject to incorrect interpretations of what the user wants or prompts it to do as well as giving blatantly incorrect answers. This doesn't happen every time, but it is still a drawback that the user has to consider the fact that the model is not perfect and thus has to fact-check the model when possible. When discussing this specific prompt, it is important to note that what was asked for

specifically was a set of components that would be suitable for a heads-up-display. That being said, the answer provided favored one type of HUD and did not account for alternatives, cost of the project, or desired specifications / requirements. Because of this, the model provided a list of components for a laser-based illumination HUD with a set of mirrors and lenses that were not tailored to any specific package. One major limitation of AI is that when you prompt it to design these types of systems, it may pick one design path for a particular reason and not necessarily be transparent as to why the decisions that were made were, in fact, made. Because of this, the user has to be vigilant and scrutinize in order to verify the output and its feasibility. In addition, the user must pay strict attention to the output as it they must agree with the design choices made, should they choose to be inspired by them or even adopt them in their own design. In addition, these design criteria, especially for this answer, are being made under some pretty heavy assumptions. These are assumptions like the operating conditions of the system, the power available, the package size or lake thereof that the system will operate in, the external temperatures of the system operating environment, and more. It is important as a user to understand what assumptions are made when a model returns a particular output. This is not necessarily a drastic drawback of using this technology in engineering design, but it is something to be vigilant about and be mindful of.

5.2 Software Case Studies

The usage of AI has become more prevalent in recent years as the abilities of generative AI has improved. For example, when ChatGPT became able to search the internet, that brought a strong increase in usage for me. Before this, ChatGPT only had a limited dataset but once it became able to search the internet and make analyses from that data, it became much more useful for everyday searches. In this project, I used ChatGPT for almost all of my information and felt I was able to learn more and faster than if I simply used Google. This is due to the limited experience I have with embedded systems and especially related to real-time systems. Without the use of AI, development of the software system would have taken much more time and would have been potentially missing key components. Yet, ChatGPT is not without its limits and should not be trusted blindly. Each answer it gave me I verified independently that the information existed and ensured that it was not making data up through an occurrence of an AI hallucinations (citation 60). ChatGPT was also not good at making a UI spec despite being able to make Figma flowcharts with no issues. Below, I will discuss the main advantages and wins of ChatGPT as well as the failings of it. I will also discuss the use of the Figma AI to create the UI specification.

5.2.1 AI In the Software Research Section

In section 3.3, this section was all done through one ChatGPT chat at citation 58. Due to the length of this chat and prompts given and received, it is easier to go to that citation to examine them, but specific prompts will be referenced. For example, the first prompt was the desired requirements given from the coursework document rubric and an

additional request at the end to include all needed software technologies related to the software development of the HUD project referenced in other chats cited in the appendix. From there, ChatGPT was able to give me a full layout on how to write the software research section. From there, subsequent chats were used to pinpoint specific technologies such as “bare-metal” vs an OS, or the selection of C or C++ or Python. Overall, AI has been very helpful in learning and comparing technologies in the software selections, yet it can be overly verbose.

For example, some prompts going back and forth with the software design gave me too much filler information and not enough relevant information. Sifting through what is helpful and what is not was done by common sense and some Google searches to ensure that the content was accurate. I found no issues with content accuracy at all and found all information helpful even if a little verbose. I did end up telling ChatGPT to shorten the lengths of its responses and that helped a little. Getting good responses from ChatGPT is all dependent on how well you structure your prompts. Hence why I was able to get a very good first response that just required a few more tweaks and then follow up questions. By providing the AI model with as much context as I can, the prompts and outcomes improve dramatically as shown by the information first given and the information given later in the chat.

5.2.2 AI in UI Specification Development Using Figma

Another key point of the usage of AI is the use of it with Figma. I did not know Figma had an AI tool to generate UI specs with and after testing it out found it worked very well. Referring to the Figma AI citation in the appendix, one can see that I only gave it a few sentences at a time but it was able to create exactly what I wanted. The usage of Figma AI was crucial in the development of the UI spec since my computers do not have the best graphics rendering capabilities which would make development of the spec slow. Yet, with just a few queries to the Figma AI, it was able to create exactly what I was looking for and in a fraction of the time it would have taken for me to create.

5.2.3 AI in Software Design

Despite the overall success in the assistance of the creation of the software design that ChatGPT was able to provide, it was not able to create a full UI spec despite saying it could. Referring to citation 62's chat, one can see that despite me giving the Figma link with edit access to ChatGPT and it supposedly being able to create a UI, it was unable to do so. I did not spend more than about 15 minutes on this before moving on to Figma itself but this proved the only limitation I found in ChatGPT for this entire project. Other than that, ChatGPT was able to create me a full software design template that I was able to go through, verify, and ensure that the idea made sense. Overall, the usage of ChatGPT streamlined development across all software sections and ensured I did not miss any key points that I would have been otherwise unaware of.

5.3 Hardware Case Study

When it comes to circuit designs, AI has some very helpful capabilities but also has its struggles. It can be seen that when asked to compare the two different devices for both the IMU and for the altimeter it is capable of searching the web and finding the exact specs. It's capable of comparing and contrasting them against each other in either table form or in just through text.

With this in mind it is capable of taking the circuit's requirements and matching the required parts but where it does start to run into issues is with the designing of an actual circuit. It is mostly difficult to properly present the information. If a request was made for ChatGPT to provide a detailed schematic for reference, it would not be able to achieve this task. While it has an understanding of circuits and has the knowledge to build a circuit better than a person, it does not know all of the requirements and cannot know unless given every bit of information for the requested board. It's also not able to present technical images very well so even if it could come up with a circuit, it would not be able to display.

While it does struggle to generate a whole circuit on its own, it does have a high understanding of pins and the functionalities of each. There are 40 pins on a Pi4 and keeping track of all these could be difficult, but ChatGPT acts almost as a discussional data sheet. If you're unsure what a certain pin does, AI is able to give an explanation as well as make suggestions and describe what the pros and cons could be of connecting to certain pins. For example, the Pi4 has both a pin label as 3.3V and two labels as 5V. Without looking at a data sheet, someone would not be able to determine what these pins really do and if they are meant to be inputs or outputs. While going to check the data sheet is definitely a good way of doing it, simply asking ChatGPT a question such as "What power should be connected to the Pi4?" will yield a quick but very informational quick passage that explains 5V is required and that the 3.3V is just an output voltage. While the datasheets have this information it will take longer to find and could even be less informational compared to the answer AI gives. The AI prompt will also include the side effects of improper hookups and will even give suggestions to improve connections. It is also mostly accurate when asking when to make individual connections. For example if you had asked "What pins should I hook up the altimeter to on the ESP32?" Once again ChatGPT will be able to come up with a very good and mostly reliable answer explaining the required connection and some of the circumstances if there are multiple things connecting to the same pins which there are in this project. While it is important to check the information given by AI because it can make mistakes, it is almost always a good starting point for any research or technical questions.

Ch 6: Hardware Design

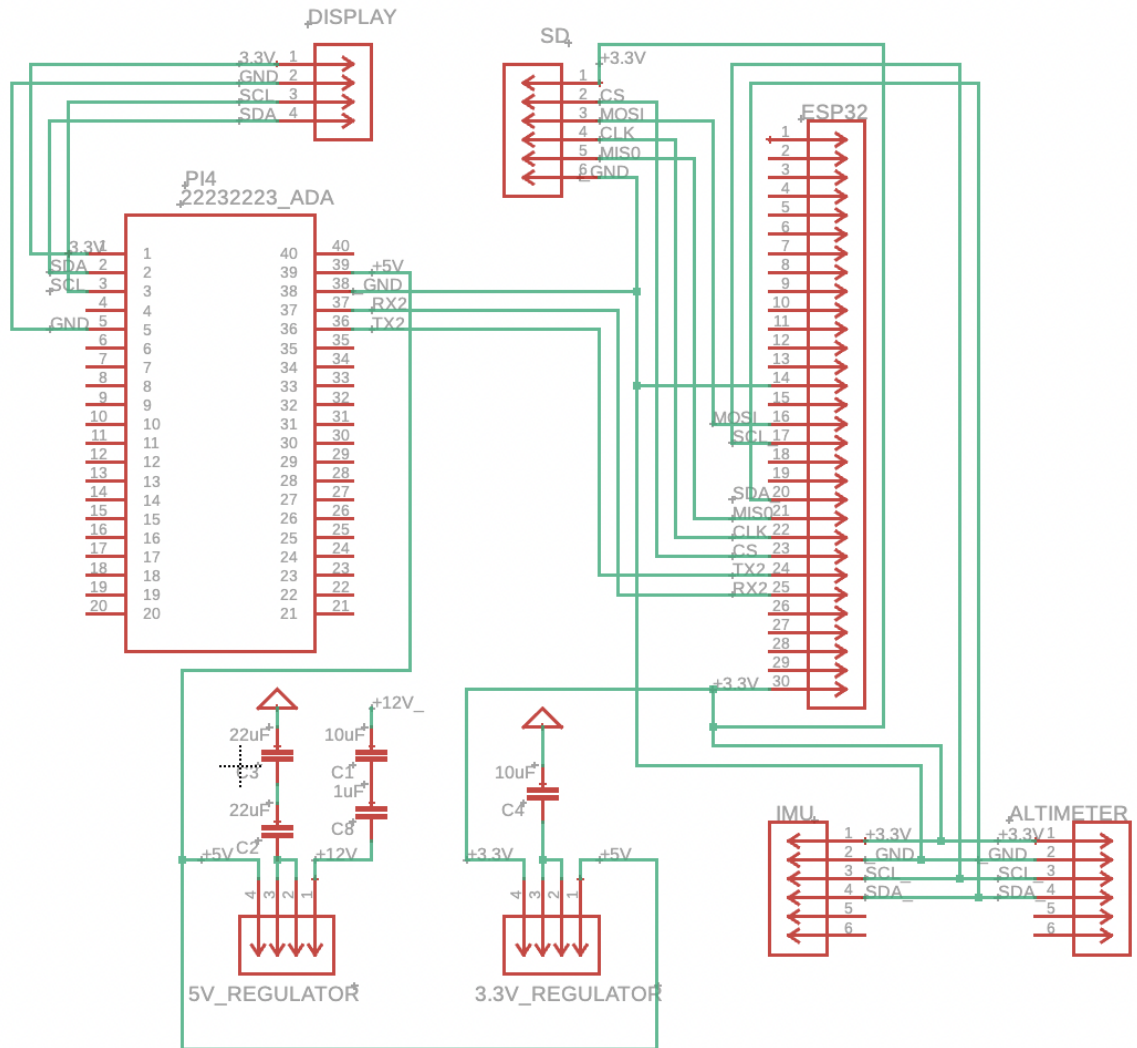


Figure 8: Main PCB

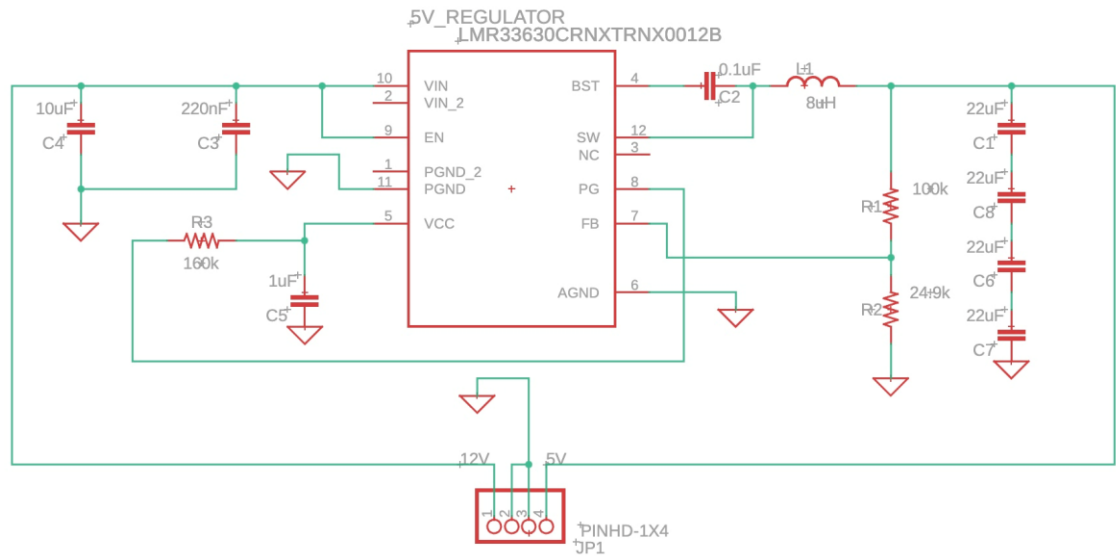


Figure 9: 12V - 5V Regulator

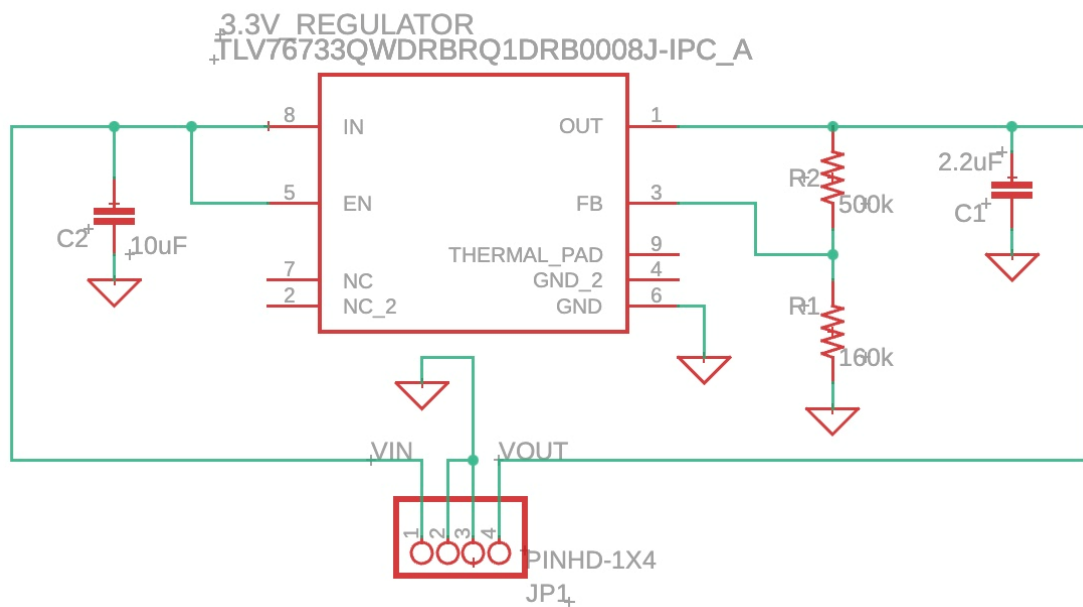


Figure 10: 5V - 3.3V Regulator

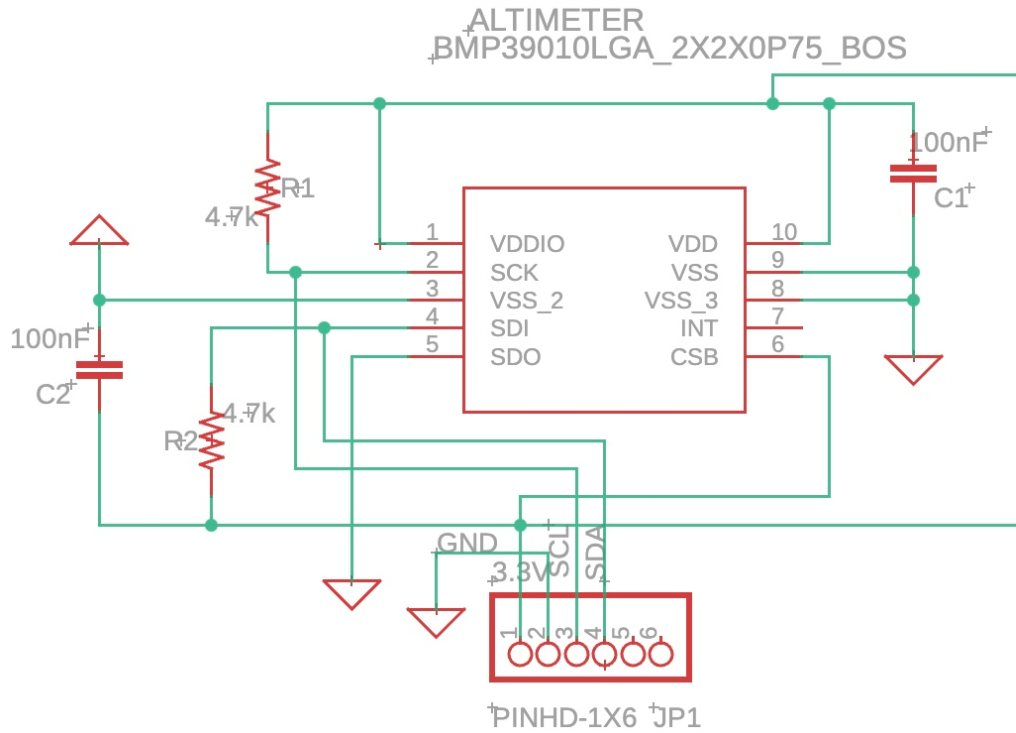


Figure 11: Altimeter Schematic

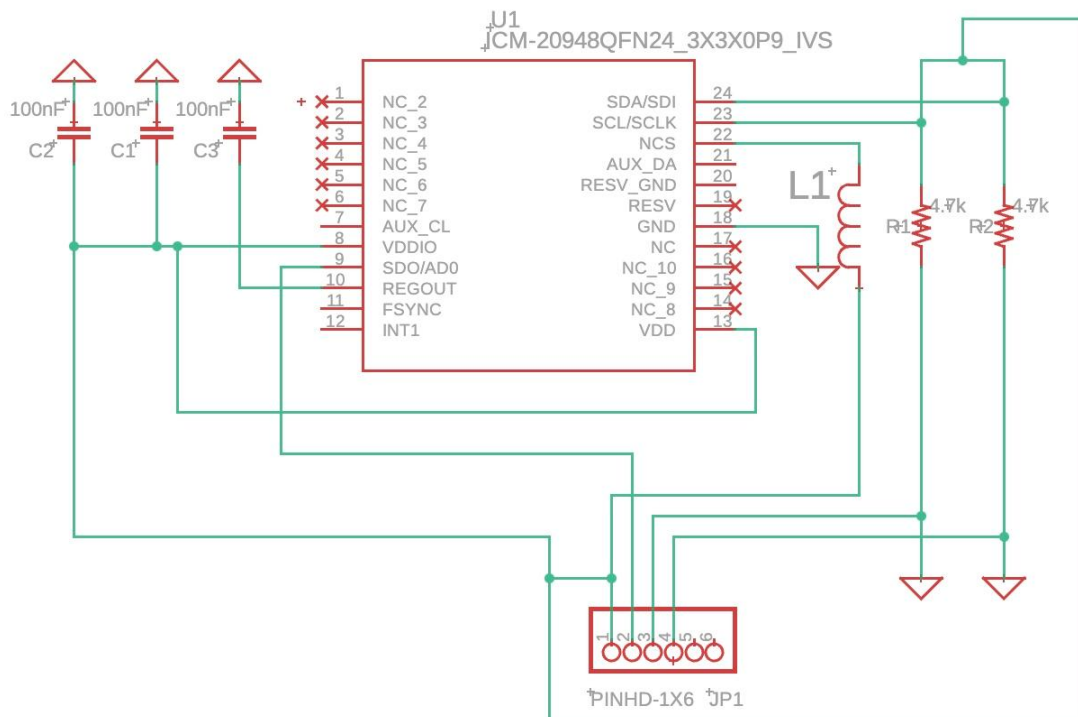


Figure 12: Inertial Measurement Unit(IMU) Schematic

Table 47: Optical Component Calculations

Component	Relevant Calculation
Collimating Lens	$f = \frac{8.634mm}{2\tan(19.8)} = 12.0mm$
Collimating Lens	$FOV_H = 2\tan^{-1}\left(\frac{W}{2f}\right) = 39.6 \text{ deg}$
Collimating Lens	$FOV_V = 2\tan^{-1}\left(\frac{H}{2f}\right) = 22.9 \text{ deg}$
Flat Mirror	$f_{mirror} = f_{collimator} = 12mm$
Flat Mirror	$W_{Mirror} = 2d \tan\left(\frac{FOV}{2}\right) + D_{eyebox} = 36.8mm$
Aspheric Mirror Diameter	$D = \left(2f \tan\left(\frac{FOV_h}{2}\right) + D_{eyebox}\right) \frac{1}{\cos(\theta)} = 17.7mm$

Ch 7: Software Design

7.1 System Overview

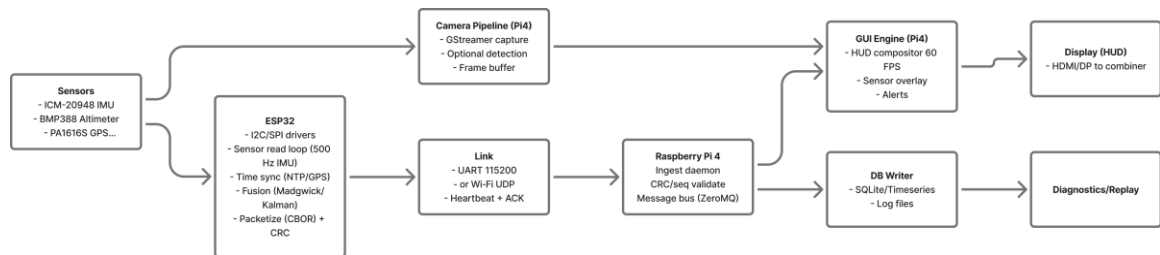


Figure 13: System Flow [61]

The P.E.G.A.S.U.S. project is designed using an ESP32 as a sensor reading and data calculation MCU and a Raspberry Pi4 for GUI development, data storage, and handling of a live camera feed. To understand how peripherals integrate with the given MCUs at a software level, figure 13 has been made to give a visual aide to the software flow concept. In this section, the overall system flow will be addressed with individual logical

flows being addressed in their respective sections. The user flow will be addressed in the Pi4 software flow section as that pertains more to the Pi4 than the software system overview.

Starting off with the sensors, the IMU and altimeter are read and processed inside the ESP32. Before these sensors are read, they are calibrated and then a larger loop begins where each of these sensors will have their own dedicated functions to ensure their calculations are done correctly. The sensors will be read inside the larger loop running on the ESP32 that is designed to read, process, and send data from the sensors to the Pi4. So, as the flowchart says, the ESP32 will be responsible for reading the sensors, syncing the time on its end with the Pi4, data filtering and fusion, and finally packetization and outbound transmission to the Pi4. Another piece being initialized before we discuss the Pi4 is the camera. This advanced objective is to be initialized and fed into the Pi4 on a separate data line from the ESP32 datagram.

The Pi4 receives the data and processes it for display on the HUD. To achieve this goal, the Pi4 will perform data sequence validation to ensure data is not received out of order before processing it and displaying it. The Pi4 will also be responsible for periodically logging data or logging data if there are any crashes. This data will be stored in a local database that is accessible by the developer. From here, the data is formatted in a GUI and sent to the LCD driver for view by the pilot.

So, to summarize, this system reads sensors and performs filtering and data processing on the ESP32. This processed data is then sent to the Pi4 for logging and display where the GUI is then sent to an LCD to be read by a pilot. Summarizing the components, we have a IMU for attitude data, an altimeter for altitude data, a camera to improve the pilots FOV, and an LCD for display on the HUD. By combining all of these components in the discussed order, we arrive at the desired project outcome.

7.2 ESP32 Software Flow

In this section, the specific flow will be discussed of how data is read, processed, and sent from the ESP32. The ESP32 fits in to the system as the data reading and processing MCU. It is assigned this task since it is very fast and better suited to read sensors than the Pi4 is because of its speed. Below is the flowchart for how the ESP32 reads and processes data.

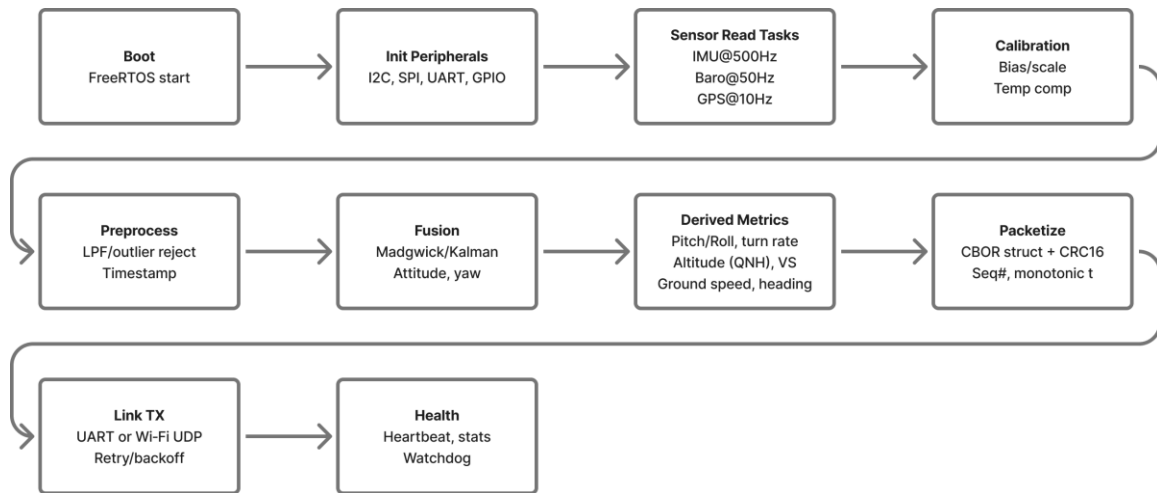


Figure 14: ESP32 Software Flow [61]

The ESP32 begins by booting up FreeRTOS, recalling that FreeRTOS was the chosen OS for the ESP32 because it allows the fastest development with the lowest overhead. From here, peripherals such as I2C, SPI, UART, and GPIO are initialized to allow for the sensors to be read by the system. After, the sensor read tasks are initialized at the frequencies shown in the chart. These frequencies were given based on the selected hardware components and the desired frames per second of the HUD. When the sensors are being read/booted up, the data must be “cleaned” and outliers in sensor readings must be purged hence the calibration and preprocessing steps. To best calibrate this system, it is best to place the system on a desired surface and let it calibrate for 5-10s. After, the ESP32 performs data filtering and fusion using madgwick filters to allow for minimal overhead and fast data fusion (citation 61).

After sensor data has been calibrated, processed, and fused, the desired metrics are derived using functions based on these formulas:

Formula / Expression	Description	Variable Definitions
$y_t = \alpha x_t + (1 - \alpha)y_{t-1}$	Low-Pass Filter (LPF) — smooths noisy sensor readings	y_t : filtered value at time t x_t : current sensor value y_{t-1} : previous filtered value $\alpha = \frac{\Delta t}{\tau + \Delta t}$: smoothing factor τ : time constant (ms) Δt : time step between samples
$b_g \leftarrow (1 - \beta)b_g + \beta g_{\text{est}}$	Gyro Bias Update — slowly corrects gyro drift	b_g : gyro bias estimate β : learning rate (small constant, e.g., 0.01) g_{est} : estimated gravity vector from accelerometer
$\mathbf{a}_n = \frac{\text{vec}(\mathbf{a})}{\ \text{vec}(\mathbf{a})\ }$		$\text{vec}(\mathbf{a})$
$\dot{q} = \frac{1}{2}q \otimes \omega - \beta \nabla f(q, a_n, m_n)$	Madgwick Quaternion Update	q : current orientation quaternion ω : gyroscope angular velocity (rad/s) $\otimes$: quaternion multiplication β : Madgwick filter gain $\nabla f(q, a_n, m_n)$: gradient of the error function using accel + mag a_n, m_n : normalized accel and mag vectors
$q_{t+1} = q_t + \dot{q}\Delta t$, then normalize	Quaternion Integration Step	q_t : previous quaternion $\dot{q}$: quaternion derivative Δt : time step
$\phi = \tan^{-1} \left(\frac{2(q_0q_3 + q_1q_2)}{1 - 2(q_1^2 + q_2^2)} \right)$	Roll (Φ) from quaternion	$q_0..q_3$: quaternion elements
$\theta = \sin^{-1} (2(q_0q_2 - q_3q_1))$	Pitch (Θ) from quaternion	same as above
$\psi = \tan^{-1} \left(\frac{2(q_0q_1 + q_3q_2)}{1 - 2(q_1^2 + q_2^2)} \right)$	Yaw / Heading (Ψ) from quaternion	same as above
$\psi = \tan^{-1} \left(\frac{m_x}{m_y} \right)$ after tilt-comp	Magnetometer Heading (tilt compensated)	m_x, m_y : horizontal magnetometer components after rotation Derived from rotating magnetometer into level frame using quaternion
$h = \frac{T_0}{L} \left[\left(\frac{p}{p_0} \right)^{\frac{L}{R}} - 1 \right]$	Barometric Altitude (from pressure)	T_0 : sea level temperature (≈ 288.15 K) L : lapse rate (0.0065 K/m) p_0 : reference pressure (QNH, ~ 1013.25 hPa) p : measured pressure R : specific gas constant for air (287.05 J/kg·K) g : gravity (9.80665 m/s ²)
$VS = \frac{h_t - h_{t-1}}{\Delta t}$	Vertical Speed	h_t : current altitude h_{t-1} : previous altitude Δt : time step
$v = \sqrt{v_x^2 + v_y^2}$	Ground Speed (from GPS)	v_x, v_y : GPS velocity components
$r = \omega_z$	Turn Rate (yaw rate)	ω_z : gyro angular velocity around Z-axis
$\text{CRC}_{16} = \text{CRC}(0x1021, 0xFFFF, data)$	CRC-16/CCITT Checksum	Polynomial: 0x1021 Init value: 0xFFFF Used for link error checking
$t_{\text{exp}} = t_{\text{pi}} + \delta$	Time Synchronization	t_{exp} : ESP32 local time t_{pi} : Pi time δ : estimated offset from round-trip

Figure 15: Relevant Equations [61]

After the data is calculated, it is packeted in a CBOR struct with the below structure:

```
seq:u32, ts_us:u64,  
quat:[f32;4], euler:[f32;3],  
imu:{ax,ay,az,gx,gy,gz}, mag:{mx,my,mz},  
baro:{p,T}, alt:f32, vs:f32,  
gps:{lat,lon,alt,gs,track,hdop,fix},  
health:{temp,vbat,loop_ms,drops}
```

Figure 16: CBOR Datagram [61]

This data is sent over to the Pi4 with retries to ensure the data arrives. From there, the system performs health checks and repeats. This section and flowchart give a closer view to the flow of code inside the ESP32.

7.3 Raspberry Pi4 Software Flow

Now that data has been sent from the ESP32, it is time to examine how it is processed on the dedicated GUI MCU. The Raspberry Pi4 creates the GUI, reads the camera feed, and logs data in a database for debugging by the engineering team. Figure 17 is the flowchart for the Pi4 to show how the system runs internally.

On boot, the system opens a listener to wait for datagrams from the ESP32. When a datagram is received, the Pi4 validates the data and ensures it is in sequence and time sync with the ESP32. From here, the Message Bus sends the data to the GUI Engine by publishing the data struct to the respective ZeroMQ topic. ZeroMQ is simply a messaging library designed for asynchronous programs by providing a message queue to send data along to destinations (citation 62).

As the listener port is being set up the camera stream starts. This stream is read through the GStreamer package and processed by the Pi4. Combining this feed with the data sent from ZeroMQ in the GUI engine creates the desired GUI. From here, the GUI is sent to the LCD where it is displayed to the pilot and here ends the GUI rendering feed.

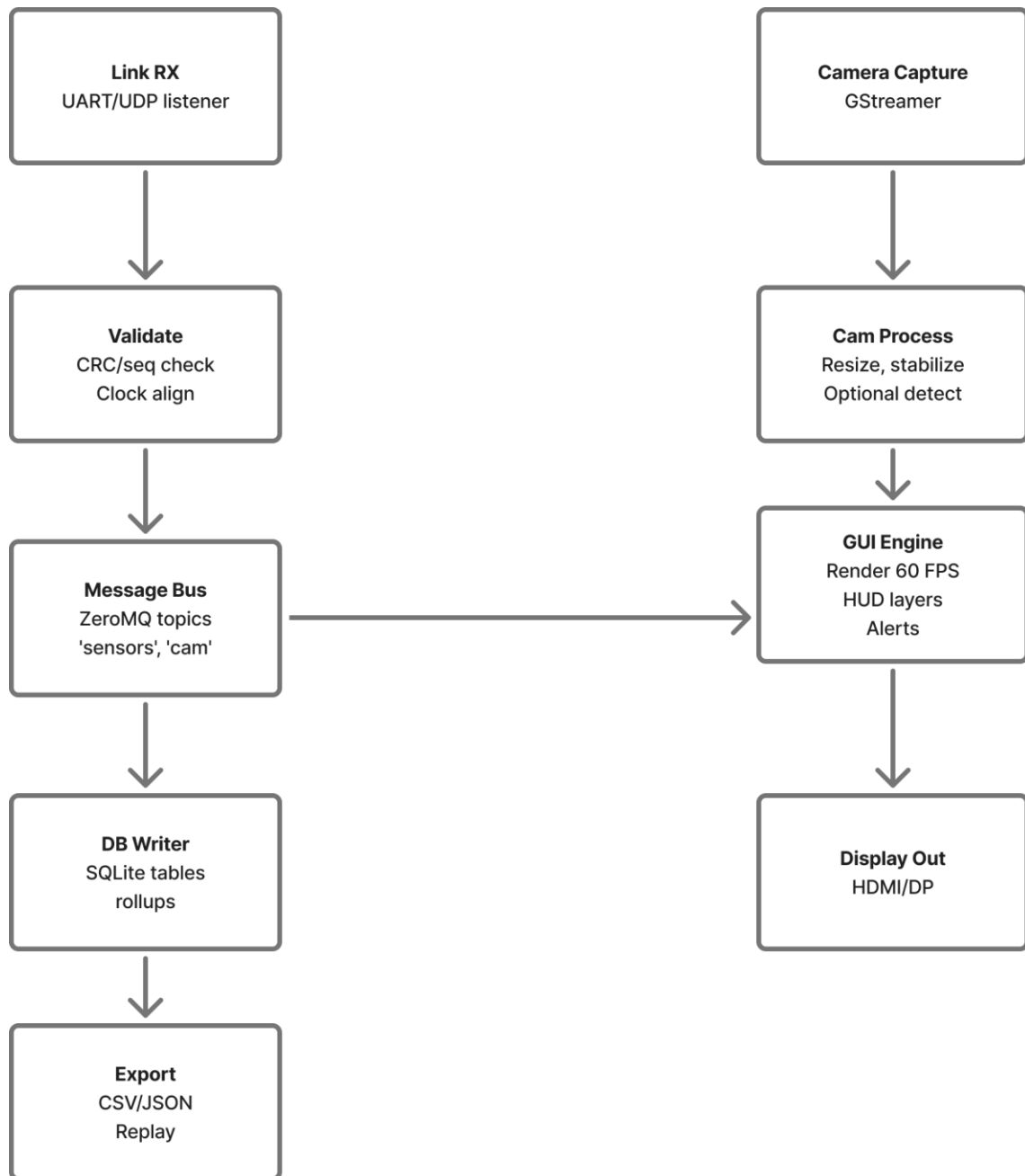


Figure 17: Pi4 Software Flow [61]

Another process happening is the database storage process. Data is also occasionally written to the database bus to the database writer. Here, the data is put in the SQLite tables for access by the development team. This data can either be accessed in the SQL database or exported as a CSV for analysis.

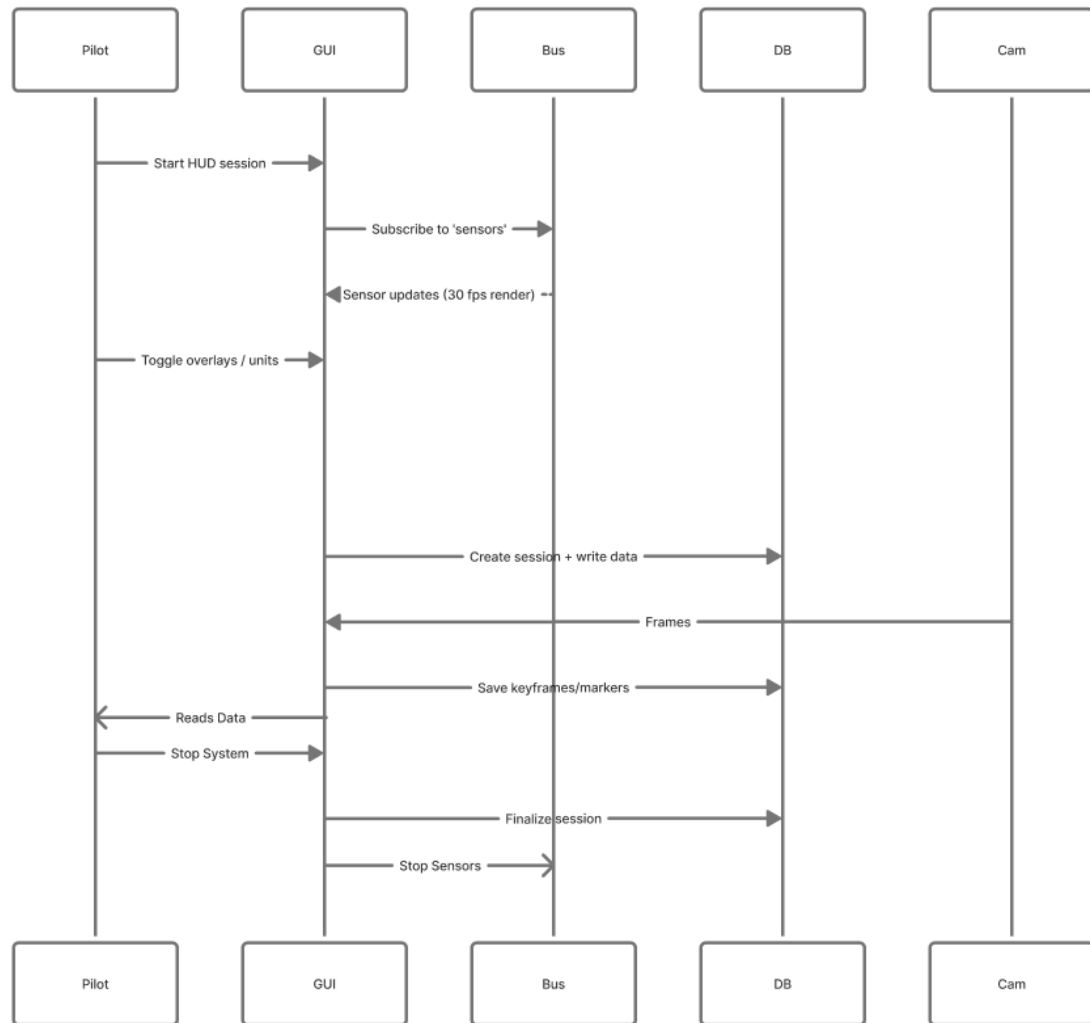


Figure 18: Pi4 Use Case Diagram [61]

Above is the use case diagram for this project. The use case diagram shows what happens from end-to-end, as in from the pilot to the sensors and back. In this use case diagram, the bus represents the ESP32 and the sensors since that is where the Pi4 receives data from which is the MCU the pilot interacts with. So, the pilot starts the HUD session which is turning the system on which immediately subscribes to the 'sensors' data queue which is where the data is received from the ESP32. Then, the sensor data is received, and the GUI is updated.

If the pilot wants to change the view, they select the toggle and the view changes from data to camera or vice versa. Occasionally, the GUI MCU writes data to the DB as well as shown by the GUI MCU to DB arrow. When the pilot wants to end the session, the system shuts down and ends data collection.

7.4 ESP32 & Pi4 Integration Flow

Now that the overview, ESP32, and Pi4 flows have been discussed, the next flow needed is the connection between the ESP32 and the Pi4 or the MCU integration flow as shown in the chart below. This chart shows the flow from power on to normal operation and focuses more on the connection between the ESP32 and Pi4 than any previous charts discussed.

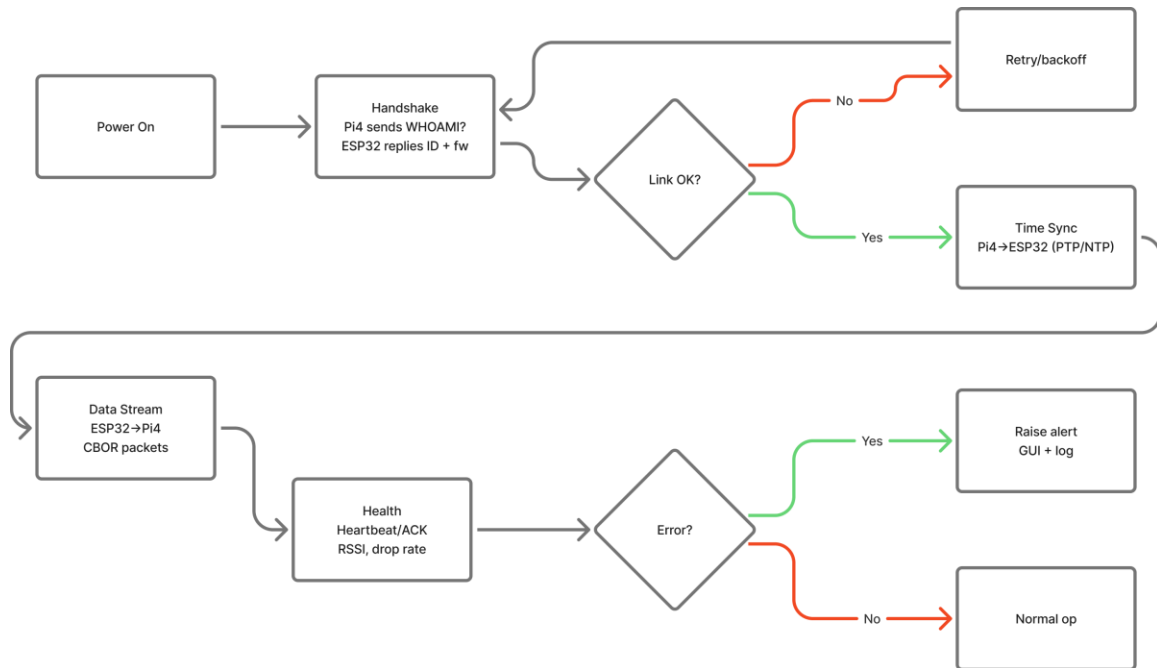


Figure 19: MCU Integration Flow [61]

The integration begins by powering the system on and performing a handshake. Here, the Pi4 sends a WHOAMI message and the ESP32 replies with an ID and other message. Then, we check if the link was ok and performed successfully. If the link was not successful, the system backs off and retries the handshake by performing the same operations. If the handshake was successful, the time sync begins between the Pi4 and ESP32. It is important to perform this handshake so that real-time calculations are synced across devices to ensure minimal display latency. If the ESP32 has a processing delay, the Pi4 needs to be aware of that delay to ensure data is not output incorrectly for the pilot. So, after the time sync, the data stream between the ESP32 and Pi4 begins and the CBOR packets are sent. The data being sent in these packets was shown earlier in Figure 16.

To ensure the continuous data stream is not broken and the system is unaware, periodic health-checks must occur. These are done by sending small ACK signals back and forth between the MCUs and ensuring that enough of them are received to properly communicate the data. Finally, the system checks if there was an error in either the health-check or other parts of the system. If there was an error, the system is aware of it and logs it. If there was not, the normal operation of the system continues.

7.5 Software Components

7.5.1 Database

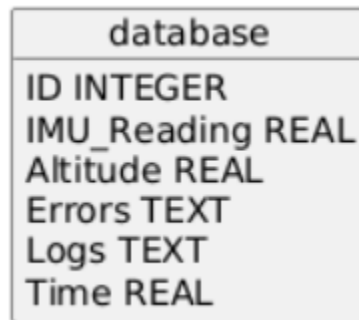


Figure 20: Database Schema

The database schema is very simple, it just contains an ID, IMU reading, altitude reading, errors, and logs. The ID is necessary for good SQL structure and identifying certain rows by ID. IMU_Reading is stored as a REAL (SQLite float type) to allow for debugging of IMU values in case of transmission error, data calculation error, etc. Altitude is also stored as a float for the same reasons as IMU_Reading. Errors are stored as text which are effectively long blocks of text in SQLite. Logs are also stored as text to ensure that if there are any log statements at a given measurement, the development team can pinpoint them. Finally, the time field is taken in case the time of an error or log is needed.

7.5.2 UI Spec

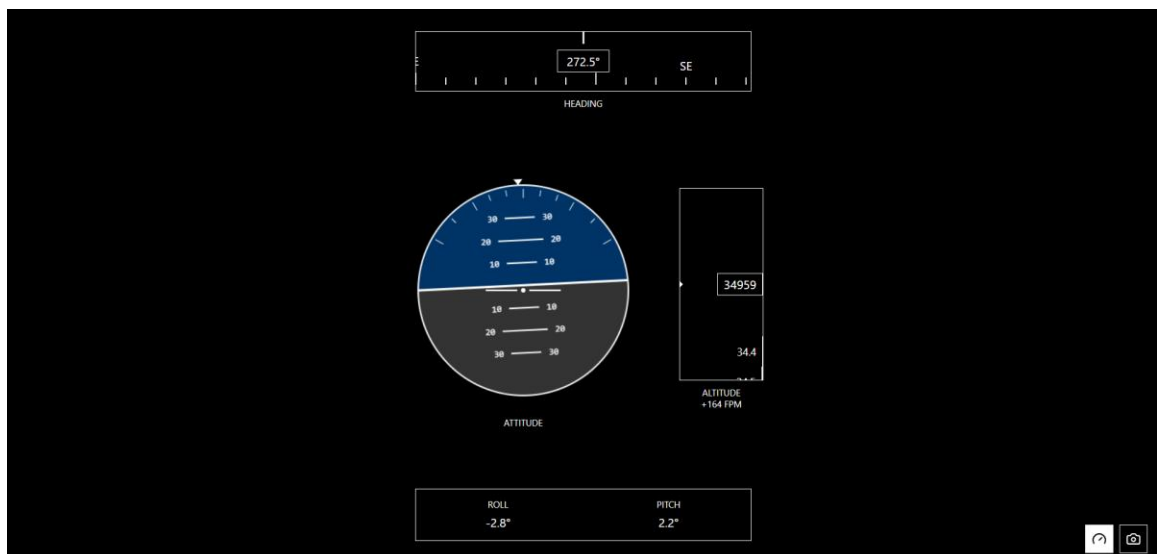


Figure 21: Data UI Spec [Figma AI]

Figure 21 is the UI specification for the data view of the HUD. This view is responsible for showing attitude, altitude, and compass heading data. This is a mock up of what it

will look like and the only differences between this design and the final product will be due to any unforeseen system shortcomings. For example, say the Qt library does not have the animations to produce stable orientation changes, this will be compensated for by the next best animation. Otherwise, this design will be very close to the final product. Note the bottom right which indicates a button to change between the data and camera view.

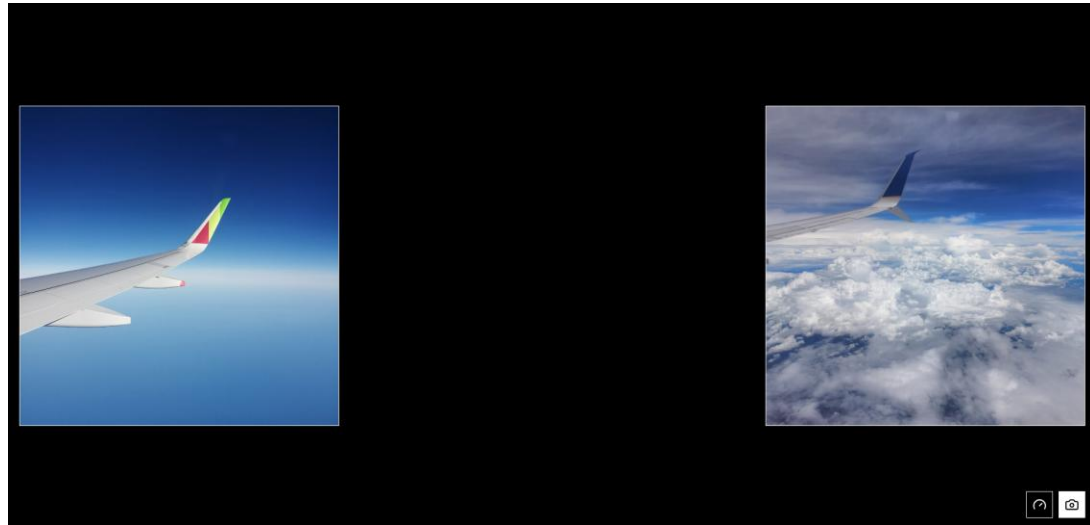


Figure 22: Camera UI Spec [Figma AI]

Above is the camera UI specification which shows a mock of the camera view of the HUD. The two camera feeds will be on both sides of the display. Here, two sample images are used taken out of the side of an airplane to show two different feeds. The main field of view will remain visible for the pilot hence the middle being blank. Also note the buttons in the bottom right denoting that the view is in camera mode and not data mode anymore.

7.6 Conclusion

The software design is simple overall. The overall flow was mentioned in chapter 3 and to reiterate that, the design flow is:

Camera Feed ↘
IMU & Altimeter → ESP32 → Raspberry Pi4 → LCD

As discussed, this would be the overall flow with each MCU having its own flow denoted above. To summarize the ESP32, the sensors are initialized on boot, then the ESP32 begins to process the data and packages it to be sent to the Raspberry Pi4. From here, the summary of the Pi4 flow is it simply receives this data, checks it, then sends it to the GUI engine for rendering. Before this is done though, the Pi4 must make a connection with the ESP32 through a handshake then data can be processed. While data is being processed, the Pi4 also handles the live camera feed and occasionally logs data with the schema given in section 7.5. The GUI will render the UI given in section 7.5 as well.

Ch 8: System Fabrication/Prototype Construction

For all the boards, Autodesk Fusion was used for both the design of the board and the tracing of the wires. Each of the boards are grounded on both the tops and bottom planes to ensure proper functionality along with adjustments to the trace size to allow for adjustments after delivery if needed. All the regulators and the sensor boards will be connected to the main PCB by using male and female pin headers.

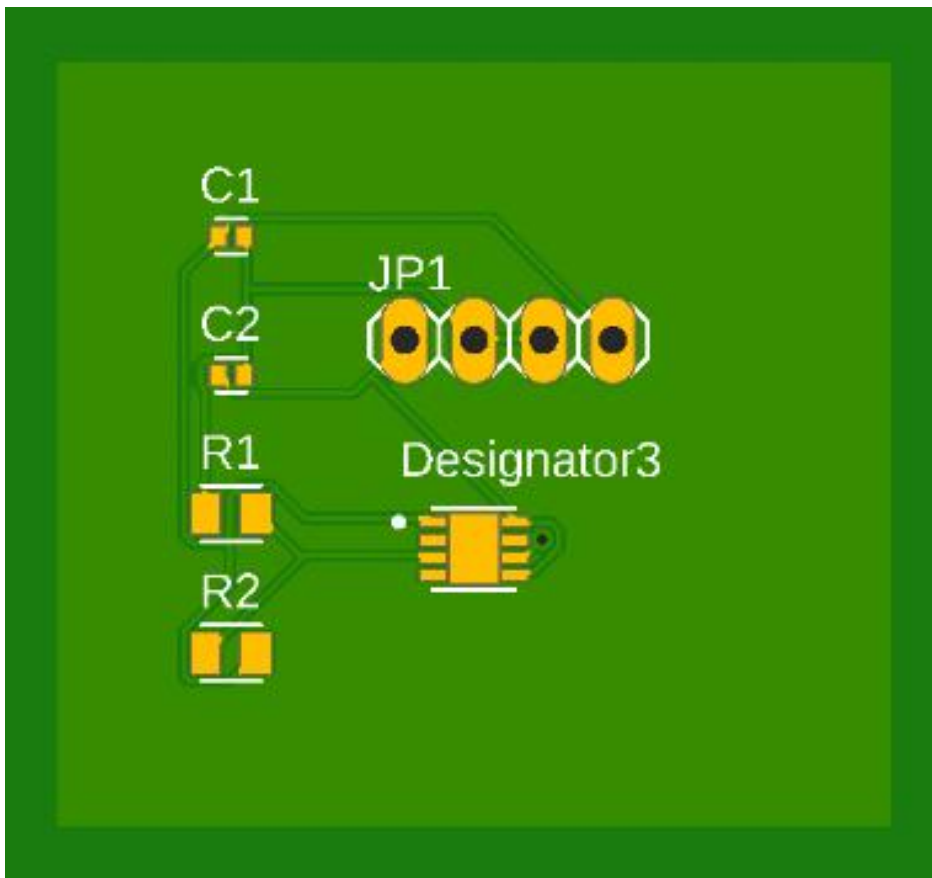


Figure 23: 3.3V Regulator

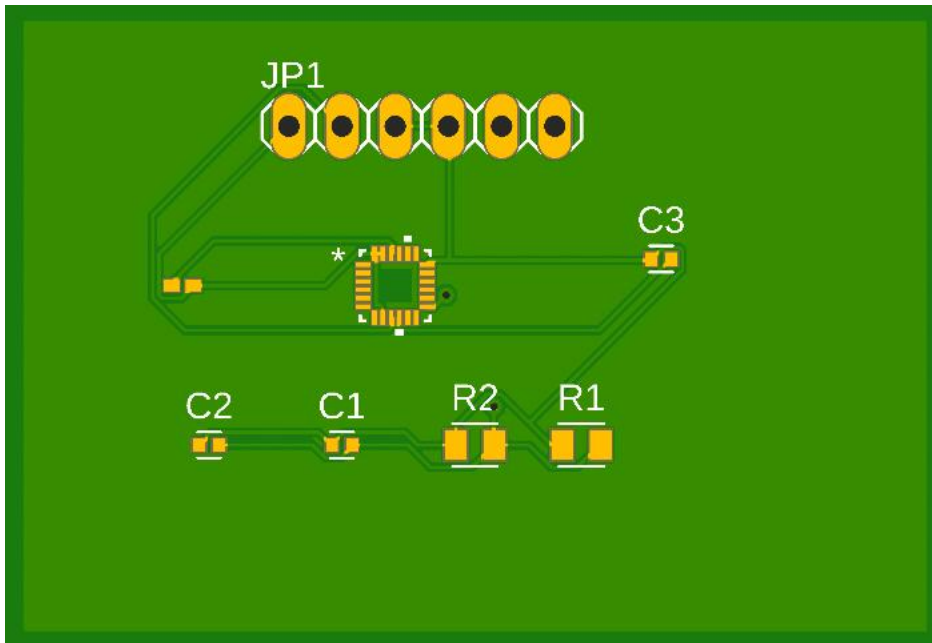


Figure 24: 5V Regulator

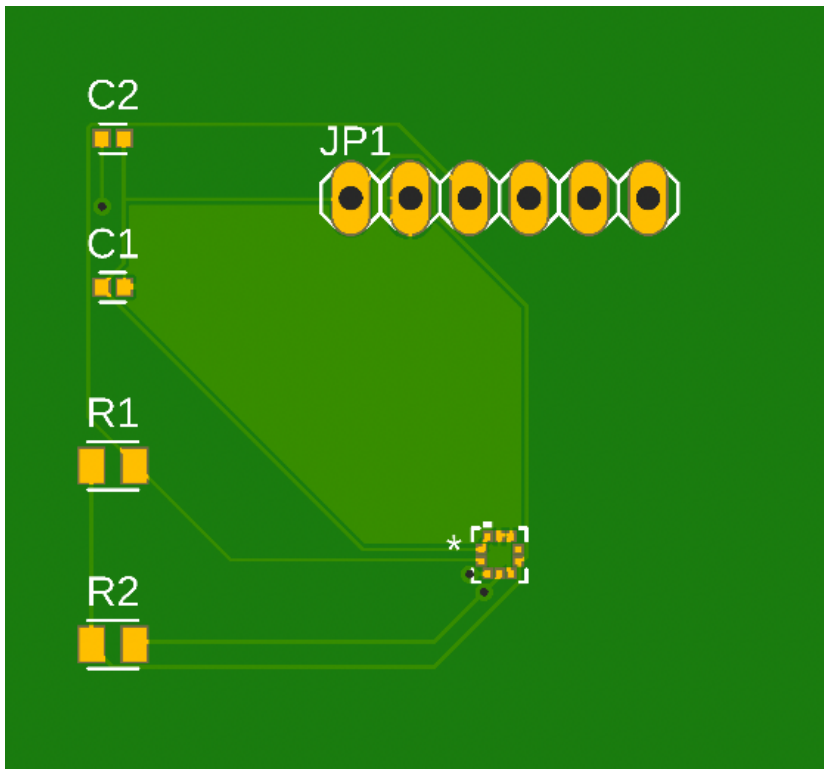


Figure 25: Altimeter

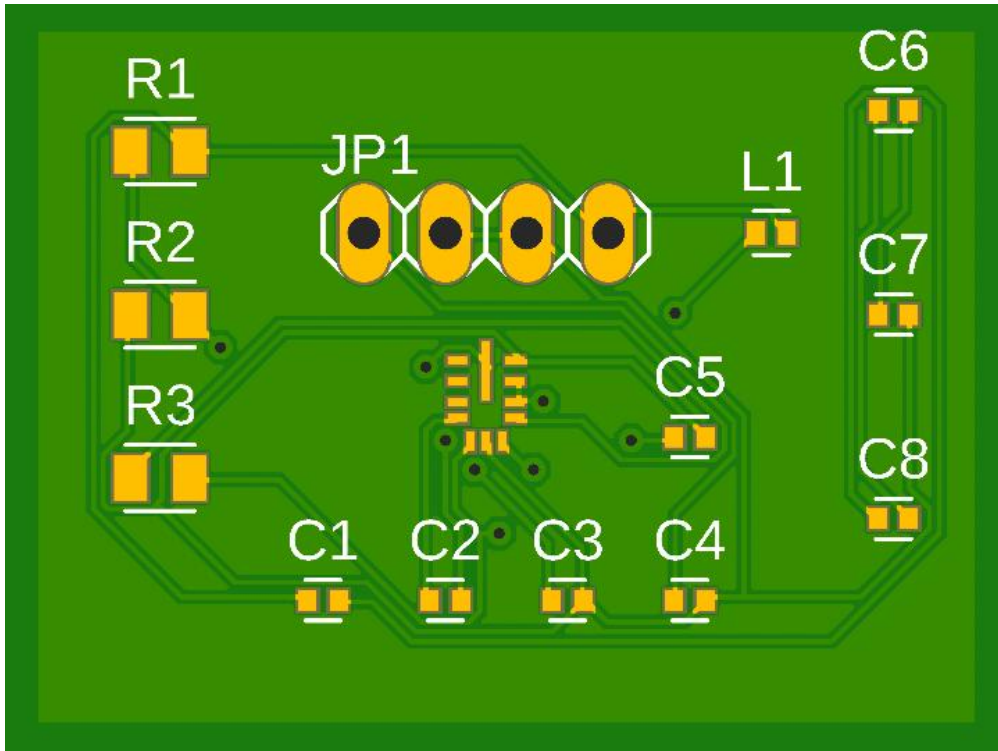


Figure 26: IMU

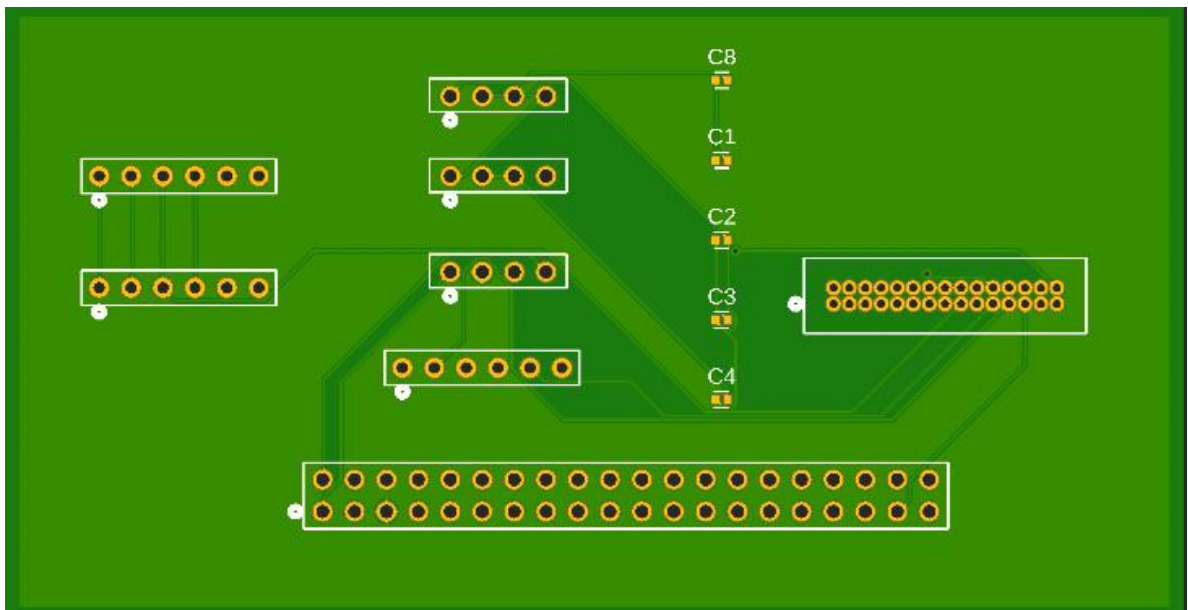


Figure 27: Main PCB

Ch 9: System Testing

9.1 Hardware Testing

9.1.1 OLED / Pi4 Test

The first test was to confirm that the OLED screen and Pi4 were working as desired. To accomplish this, the Pi4 was booted up and normal operation was tested by connecting to the internet, working in the terminal, etc. Once there were no issues with the Pi4, the next step was to test writing to the OLED screen from the Pi4 to ensure that the I2C pins worked as expected. The word “PEGASUS” was written to the screen as shown below to confirm the system was able to output data properly.

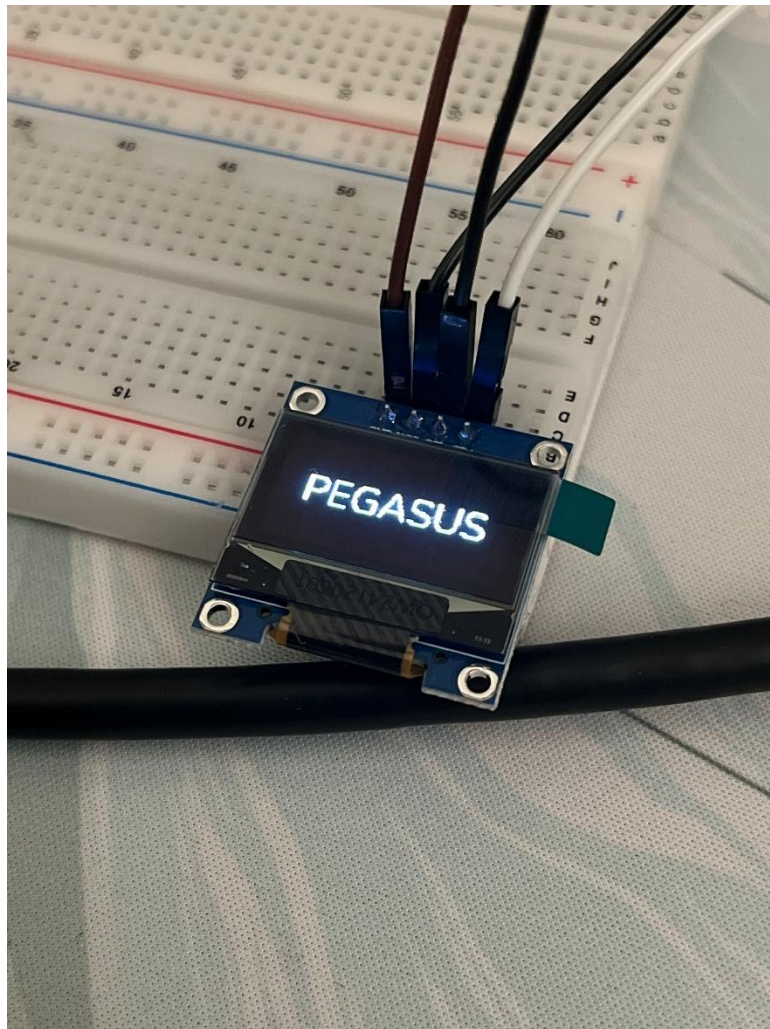
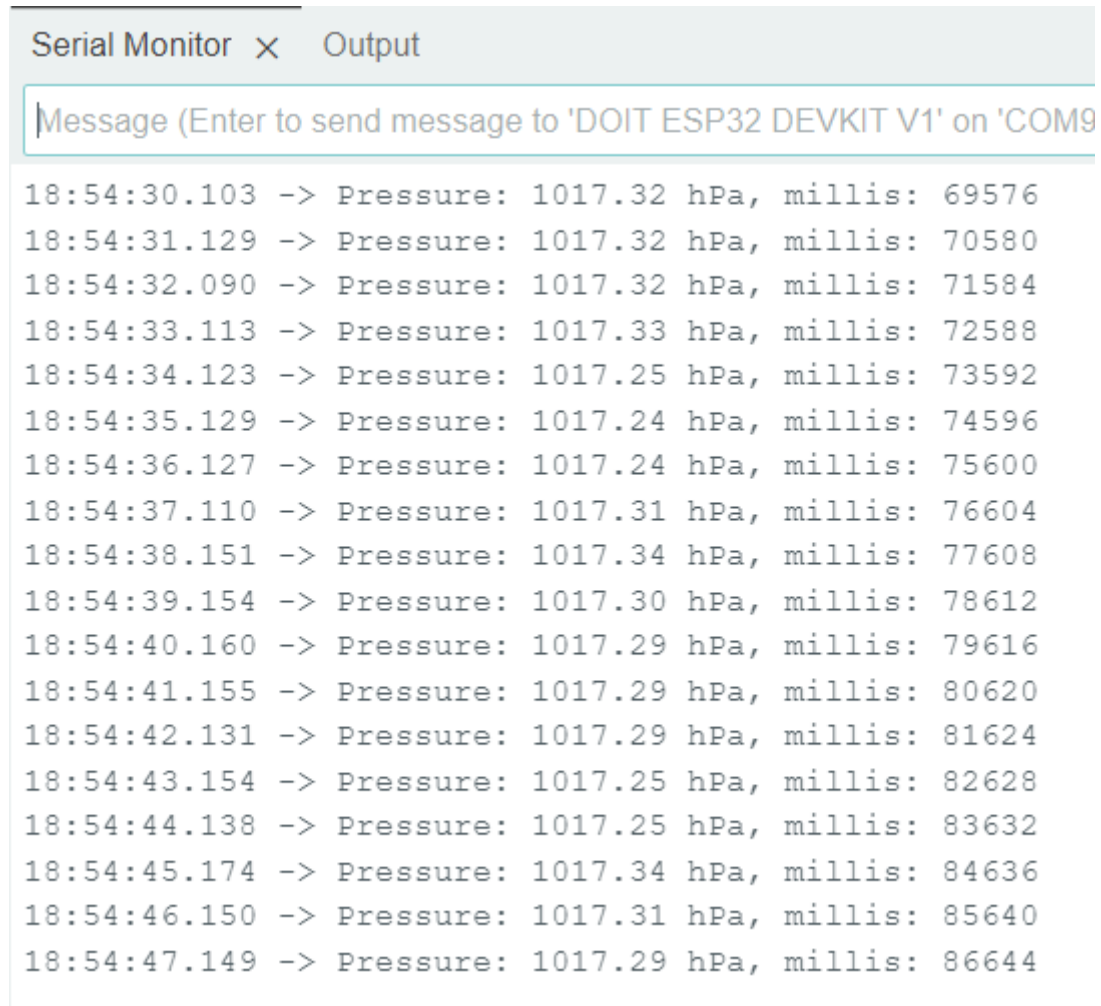


Figure 28: OLED / Pi4 Test

9.1.2 Altimeter / ESP32 Test

Next, to ensure that the ESP32 and altimeter worked, the ESP32 was connected to a laptop by USB, and the altimeter was connected over I2C to the ESP32. Once both boards had lights showing they were getting power, simple code was written to read and display the altimeter value in the serial monitor as shown below.



The screenshot shows a Serial Monitor window with a title bar that says "Serial Monitor" and a close button. The window has a text input field at the top with the placeholder text "Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM9)". Below the input field, there is a list of 17 lines of data. Each line represents a timestamp followed by an arrow and a string containing pressure and time values. The pressure values are in hPa and the time values are in milliseconds.

Timestamp	Pressure (hPa)	Time (ms)
18:54:30.103	1017.32	69576
18:54:31.129	1017.32	70580
18:54:32.090	1017.32	71584
18:54:33.113	1017.33	72588
18:54:34.123	1017.25	73592
18:54:35.129	1017.24	74596
18:54:36.127	1017.24	75600
18:54:37.110	1017.31	76604
18:54:38.151	1017.34	77608
18:54:39.154	1017.30	78612
18:54:40.160	1017.29	79616
18:54:41.155	1017.29	80620
18:54:42.131	1017.29	81624
18:54:43.154	1017.25	82628
18:54:44.138	1017.25	83632
18:54:45.174	1017.34	84636
18:54:46.150	1017.31	85640
18:54:47.149	1017.29	86644

Figure 29: Altimeter / ESP32 Test Serial Monitor

9.2 Software Testing

9.2.1 Dynamic OLED Test

To ensure that the OLED was able to display a dynamic value, a clock was chosen to be displayed on the screen. This code was straightforward and proved that the OLED and Pi4 were capable of running a dynamically changing value instead of a static value like the word “PEGASUS” as previously demonstrated. Below is the image of the clock that

was displayed, the line running through the middle is the clearing and returning on the pixels of the screen for the new time value.



Figure 30: Clock OLED Test

9.2.2 System Integration Test

After the dynamic OLED test, the next test was to ensure that an altimeter value could be read from the altimeter by the ESP32 and then sent to the Pi4 for display on the OLED. This was accomplished by assigning a value of sea level as the value at boot of the system and formatted in a GUI similar to the one shown in Figure 21. This resulting test is shown below.

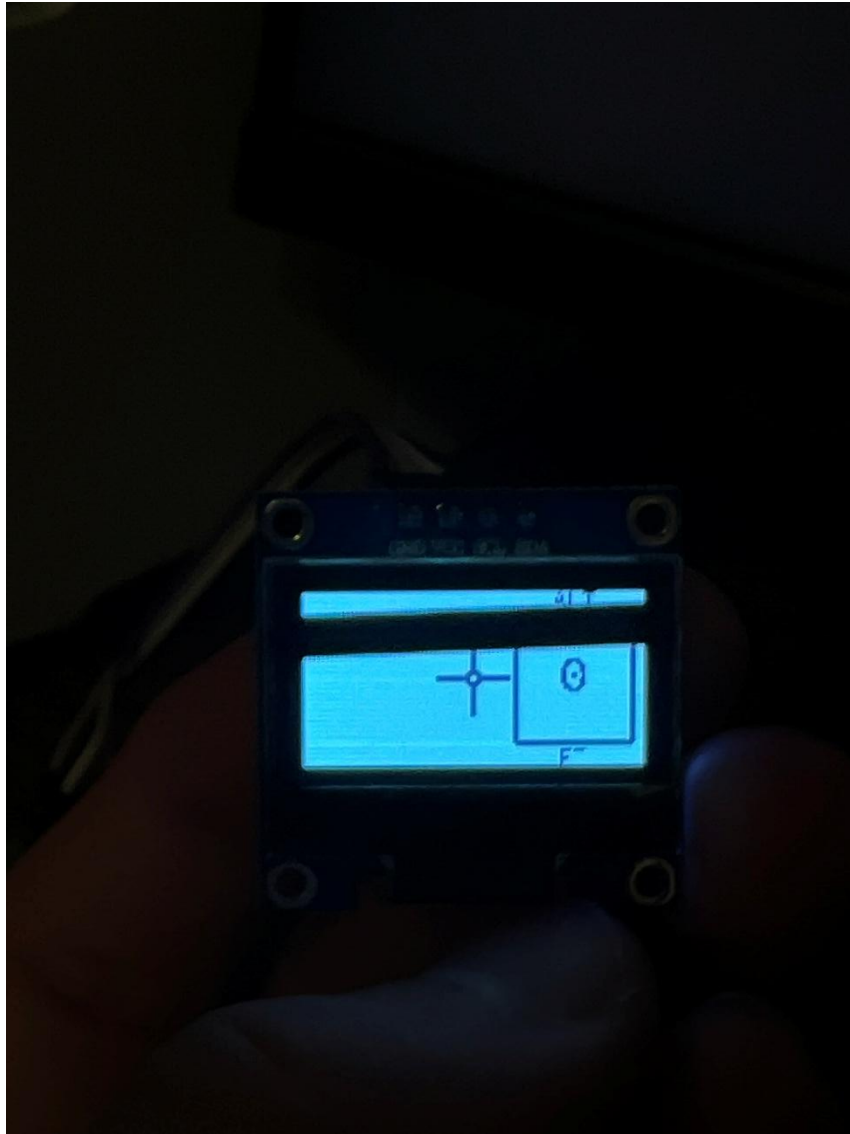


Figure 31: Initial System Integration Test

After some minor adjustments, the device was able to accurately read the current altitude above sea level in a given location if the pressure of that area was given. The factor of differences in pressure at different locations will need to be taken into consideration as further designs are made to the project but for testing, the system integrates as designed and displays the accurate height above sea level value as shown below.



Figure 32: Final System Integration Test

Ch 10: Administrative Content

10.1 Project Milestones

Due to the large scale of this project, it is important to stay organized and have a clear table of all due dates across the timeframe of this project. Our team is utilizing Jira to stay organized by planning sprints and epics around assignment due dates to allow everyone to stay on task. Jira allows for full team visibility of who is working on what part of the project and what their progress is. Below is the table of assignment due dates that help us organize our Jira board.

Task	Duration	Status
------	----------	--------

Senior Design 1 and Documentation		
Finalize Project Idea	8/18/25 - 8/26/25	Complete
Finalize Design Objectives and Goals	8/26/25 - 9/2/25	Complete
Committee Formation	8/26/25-9/5/25	In Progress
Divide and Conquer 10 Page Document	8/26/25 - 9/5/25	In Progress
Divide and Conquer 10 Page Document - Revision	9/5/25 - 9/12/25	Not Started
Midterm Report	9/12/25 - 10/31/25	In Progress
Midterm Report - Revision	11/3/25 - 11/7/25	Not Started
Final Report	11/7/25 - 11/25/25	In Progress
Prototype	10/1/25 - 11/25/25	Not Started
Senior Design 2		
PCB Assembly	TBD SD2	NA
PCB Testing	TBD SD2	NA
Optical Design and Casing Assembly	TBD SD2	NA
Optical Testing	TBD SD2	NA
Final Presentation	TBD SD2	NA

Figure 33: Project Milestones [13]

10.2 Budget and Funding

The budget for this project was set at \$750 in total. Material use and the project have worked around this constraint to build the following bill of materials:

#	Item / Part No.	Qty	Unit Price (USD)	Line Total (USD)
1	Raspberry Pi 4 Model B (4 GB)	1	\$55.00	\$55.00
2	Adafruit Feather ESP32-S3 (STEMMA QT)	1	\$19.95	\$19.95
3	BMP390 Altimeter breakout (STEMMA QT)	1	\$13.95	\$13.95
4	ICM-20948 9-DoF IMU (STEMMA QT/Qwiic)	1	\$14.95	\$14.95
5	Quectel L76K	1	\$9.90	\$9.90
6	25mm Ceramic Patch Antenna	1	\$4.19	\$4.19

7	32GB SDHC Flash MEM CARD	1	\$6.98	\$6.98
8	Arducam IMX-219 Wide-FOV Pi Camera (IMX219, 160°)	1	\$12.09	\$12.09
9	TLV76733	1	\$0.65	\$0.65
10	LMR33630	1	\$2.55	\$2.55
11	6 Pin Male Header	3	\$0.36	\$1.08
12	6 Pin Female Header	3	\$0.31	\$0.93
13	4 Pin Male Header	1	\$0.34	\$0.34
14	4 Pin Female Header	1	\$0.17	\$0.17
15	0805 100k ohm Resistor	2	\$0.10	\$0.20
16	0805 24.9k ohm Resistor	1	\$0.10	\$0.10
17	0805 500k ohm Resistor	1	\$0.10	\$0.10
18	0805 160k ohm Resistor	1	\$0.10	\$0.10
19	0805 4.7k ohm Resistor	4	\$0.10	\$0.40
20	Ceramic X7R 100nF Capacitors	4	\$0.10	\$0.40
21	Ceramic X7R 220nF Capacitors	1	\$0.10	\$0.10
22	Ceramic X7R 1uF Capacitors	4	\$0.08	\$0.32
23	Ceramic X7R 10uF Capacitors	2	\$0.08	\$0.16
24	Ceramic X7R 22F Capacitors	6	\$0.08	\$0.48
25	Ceramic X7R 0.1uF Capacitors	3	\$0.08	\$0.24
26	Ceramic X7R 2.2uF Capacitors	1	\$0.08	\$0.08
27	Ferrite Bead 600ohm 100mHz	1	\$0.10	\$0.10
28	8uH Inductor	1	\$1.23	\$1.23
29	OLED Display	1	\$139	\$139
30	Multi-Order Quarter Waveplate	1	\$17	\$17
31	Collimating Lens	1	\$113	\$113
32	Reflective Mirror	1	\$25	\$25
33	OAP Mirror	1	\$185	\$185
34	Magnifying Lens	1	\$56	\$56
35	Photochromic Film	1	\$40.50	\$40.50
36	Optical Combiner	1	\$100	\$100
37			Total:	862.04

Figure 34: Bill of Materials [13]

10.3 Work Distribution

Below is how the project workload will be split up by area. The four areas of the project are Optical Engineering, Electrical Engineering, Computer Engineering, and Project Management.

Computer Engineering	Responsibilities
Luke Anderson	Software Design and Implementation

	MCU Selection and Implementation
	Website Design and Management
	Project Lead
Electrical Engineer	Responsibilities
Frank Murphy	PCB/PSU Design
	Hardware Component Selection
Optics Engineer	Responsibilities
Ethan Dill	Optical Design
	Component Alignment
	Optical Component Integration

Figure 35: Distribution of Workload

Ch 11: Conclusion

In conclusion, the PEGASUS Heads Up Display (HUD) will represent a more affordable and portable option for one of the highest selling non-commercial airplanes, the Cessna 172. Going back to the executive summary where it is mentioned that there are a shortage of aftermarket HUD options and this project looks to fill that gap. To achieve this multiple different peripherals like the altimeter and the Inertial Measurement Unit (IMU) will be used along with the coding for the MCUs to be able to handle and process incoming data. This data is processed quickly and displays on a glass using aligned lenses to create a visual flying aid for pilots.

The two main sensor components, the altimeter and the IMU, work in tandem to be able to provide the required bits of flight data. The altimeter will work by measuring the pressure and using an equation to be able to translate the pressure reading into an altitude. The IMU utilizes 9 different axes to be able to measure how the plane is moving. It will be able to read whatever direction the pitch of the plane is in in real time. Each of these peripherals will be integrated into a PCB to be controlled by the two different Microcontroller Units (MCU).

The ESP32 will handle all the heavy computing and make all the sensor readings actual displayable bits of information so that numbers make sense and are helpful for the pilots. Most of the code will be needed for the ESP32 but for the display the Raspberry Pi4 works better. The Pi4 is better at making the User Interface (UI) and is the main reason it is beneficial to use more than one MCU. The Pi4 will hook up to the display to give a high-resolution starting point for the lenses that will display the UI on the HUD.

To get the HUD required to keep the pilot's field of view clear while also putting the information in a convenient viewing location, a series of different lenses and mirrors are required. In addition, a self-dimming function is added to adjust the screen brightness automatically, so the flight data will always remain visible no matter the different weather conditions.

With the use of all of these different features, the PEGASUS HUD will work to bridge the gap in the lack of aftermarket HUDs for airplanes. This document is meant to show the steps taken to make the decisions and struggles dealt with on the hardware, software, and optical sides of this device. These ranged from technology research, part selection, hardware and software design, and even limitations. The extensive research and design that has gone into this project shows promise for a good final product that will be able to help pilots fly safely. With the continuing development of this project, new information will be learned and added to improve this device to be able to deliver the best possible final product.

Appendix A – References
Appendix B – copyright permission
Appendix C – data sheet (if necessary)
Appendix D – software code (if necessary)
Appendix E – ChatGPT prompts and outcomes (if necessary)
Appendix F – etc.

Appendix A References

1. Canal Geomatics. (2024, December 18). *Aviation navigation: The role of GNSS in Flight Management Systems*.
2. *Aircraft Performance Monitoring*. Federal Aviation Administration.
3. Skybrary. “Head up Display (HUD) | SKYbrary Aviation Safety.” *Skybrary.aero*, 2025, skybrary.aero/articles/head-display-hud.
4. Glowth, Evan , et al. “Senior Design 1 Divide and Conquer 1.” *Ucf.edu*, 2025,
5. MYGOFLIGHT. “SKYDISPLAY HUD.” *MYGOFLIGHT*, 2020,
6. Thurber, Matt. “Hands On: Flying Garmin’s New HUD in the Longitude.” *Aviation International News*, AINonline, 20 Aug. 2024
7. Garmin, and Ltd or. “GHD™ 2100 Head-up Display.” *Garmin*, 2025
8. “Heads-up Display to Stay Focused on the Road | HUDWAY Drive.” *Hudway.co*, 2020
9. Miuzei Raspberry Pi 4 Touchscreen with Case. Amazon, Miuzei, 2024, <https://www.amazon.com/dp/B0DP4S7FMK>.
10. Moman HGPS Speedometer Projection Reflector. Amazon, Moman, 2023, <https://www.amazon.com/dp/B0FN7QJFMH>.
11. OpenAI. Response to “A good name for this project.” 2025, <https://chatgpt.com/share/68b073c4-5ab8-800b-a79a-f5517ae46004>
12. OpenAI. Response to GUI image request. 2025, <https://chatgpt.com/share/68c22e12-1e94-800b-ae97-67b7ae68e07a>

13. OpenAI. Response to BOM request. 2025, <https://chatgpt.com/share/68b9c79e-5c7c-800b-a418-07ba72db9283>
14. Gemini Response F-18 HUD Manufacturers 2025, <https://share.google/aimode/fLnMEi9hST1fsBUYD>
15. Gemini Response to What is displayed on the F-35 HUD <https://share.google/aimode/D6YK7CE33Dp4gTO6v>
16. OpenAI. Response to Components Questions. 2025, <https://chatgpt.com/share/68c37f97-b250-800b-867e-6839088636d3>
17. Electro-Matic Visual Blog. “LCD vs LED vs OLED: Lighting Types Explained.” *Electro-Matic Visual Blog*, n.d., <https://blog.visual.electro-matic.com/lcd-vs-led-vs-oled-lighting-types-explained#:~:text=Budget%2DFriendly%3A%20If%20cost%2Fis%2C%20smartphones%2C%20and%20gaming%20monitors.>
18. Riverdi. “Understanding LCD: How Do LCD Screens Work.” *Riverdi Blog*, n.d., <https://riverdi.com/blog/understanding-lcd-how-do-lcd-screens-work>.
19. New Vision Display. “Types of LCD Technology.” *New Vision Display Blog*, 25 Mar. 2022, <https://www.newvisiondisplay.com/types-of-lcd-technology/>. *New Vision Display*
20. New Haven Display. “Types of LCD.” *New Haven Display Blog*, n.d., <https://newhavendisplay.com/blog/types-of-lcd/#:~:text=LCD%20screens%20can%20be%20grouped,across%20the%20various%20screen%20types.> *Newhaven Display*
21. U.S. Department of Energy. “OLED Basics.” *Office of Energy Efficiency & Renewable Energy*, n.d., <https://www.energy.gov/eere/ssl/oled-basics>.
22. OLED-Info. “OLED Introduction.” *OLED-Info*, n.d., <https://www.oled-info.com/oled-introduction>.
23. DisplayModule. “How Are Micro-OLED Displays Made?” *DisplayModule Blog*, n.d., <https://www.displaymodule.com/fr/blogs/knowledge/how-are-micro-oled-displays-made?srsltid=AfmBOorzg9yZiq1Tlp5aZHtRQMiiyrcmsHCZgH5ON-eV90gEsFg6z6Xs>.
24. (Repeat) OLED-Info. “OLED Introduction.” *OLED-Info*, n.d., <https://www.oled-info.com/oled-introduction>.
25. Ansys. “What Is a MicroLED.” *Ansys Simulation Topics*, n.d., <https://www.ansys.com/simulation-topics/what-is-a-microled>.

26. MicroLED-Info. “MicroLED vs OLED.” *MicroLED-Info*, n.d., <https://www.microled-info.com/microled-vs-oled>.
27. Edmund Optics. “Introduction to Polarization.” *Knowledge Center*, n.d., <https://www.edmundoptics.com/knowledge-center/application-notes/optics/introduction-to-polarization/?srsltid=AfmBOopsFWDY57AYwM3uMw80fRJlO5BaCgf4mOhcG0q1YDxbnBSobZlr>.
28. API Optics. “Visible-Light Circular Polarizer.” *API Optics*, n.d., <https://www.apioptics.com/visible-light-circular-polarizer/>.
29. Tower Optical. “Zero-Order vs Multi-Order Waveplates: What’s the Difference?” *Tower Optical Blog*, n.d., <https://toweroptical.com/zero-order-vs-multi-order-waveplates-whats-the-difference/>.
30. Aviation Today. “Quality Considerations — Aviation Head-Displays & HUDs.” *Aviation Today*, 13 Jan. 2021, <https://www.aviationtoday.com/2021/01/13/quality-considerations-aviation-head-displays-huds/>.
31. MeetOptics. “Fiber-Optic Collimators — Fiber-Optic Connector Types.” *MeetOptics Academy*, n.d., <https://www.meetoptics.com/academy/fiber-optic-collimators#fiber-optic-connector-types>.
32. GlassDoctor. “All About Mirror Glass: Types of Mirrors.” *GlassDoctor Expert Tips*, n.d., <https://www.glassdoctor.com/expert-tips/all-about-mirror-glass/types-of-mirrors>.
33. Edmund Optics. “Magnifying Lenses — How to Choose a Magnifier?” *Knowledge Center*, n.d., <https://www.edmundoptics.com/knowledge-center/application-notes/microscopy/magnifying-lenses-how-to-choose-a-magnifier/?srsltid=AfmBOor4Dv2sYPcOTyBgzvGZJqPovbR4KP9mynJVTpsRbNPsfSIIdSaZf>.
34. RP-Photonics. “What Are Deformable Mirrors?” *RP Photonics Encyclopedia*, n.d., https://www.rp-photonics.com/deformable_mirrors.html#:~:text=Deformable%20Phase%20Plate%20s,What%20are%20Deformable%20Mirrors?,for%20autofocus%20functions%20of%20cameras.
35. MeetOptics. “Parabolic Mirrors: Properties and Applications.” *MeetOptics Academy*, n.d., <https://www.meetoptics.com/academy/parabolic-mirrors#properties-and-applications>.
36. Fabmac Glass. “What Is a Smart PDLC Switchable Film and How Does It Work?” *Fabmac Glass Blog*, n.d., <https://www.fabmacglass.com/post/what-is-a->

[smart-pdlc-switchable-film-and-how-does-it-work#:~:text=A%20PDLc%20film%20is%20made,n.d.\).](#)

37. Ansys. “What Is a Heads-Up Display.” *Ansys Simulation Topics*, n.d., <https://www.ansys.com/simulation-topics/what-is-heads-up-display>.
38. ResearchGate. “Issue of Ghost Images in an Optical Combiner.” *ResearchGate Fig.*, n.d., https://www.researchgate.net/figure/Issue-of-ghost-images-in-an-optical-combiner-a-Causes-of-ghost-images-b-A-diagram_fig5_379327843.
39. Photonics.com. “Holographic Combiners Improve Head-Up Displays.” *Photonics.com*, n.d., <https://www.photonics.com/Articles/Holographic-Combiners-Improve-Head-Up-Displays/a64487>.
40. ChatGPT. “Response to the Variation of the Voltage Source on a Cessna.” *ChatGPT*, n.d., https://chatgpt.com/s/t_6901ad2c195c8191a3018a9a71c29e0b.
41. ChatGPT. “Response to What Type of Regulator Would Work Well with My HUD.” *ChatGPT*, n.d., <https://chatgpt.com/share/6901af17-c468-8001-a0b0-732a6a268f44>.
42. Texas Instruments. *LMR33630 SIMPLE SWITCHER® 3.8-V to 36-V, 3-A Synchronous Step-Down Voltage Converter*. Texas Instruments, n.d., <https://www.ti.com/lit/ds/symlink/lmr33630.pdf>.
43. Texas Instruments. *TLV767 1-A, 16-V Precision Linear Voltage Regulator*. Texas Instruments, n.d., <https://www.ti.com/lit/ds/symlink/tlv767.pdf>.
44. ChatGPT. “Response to Compare the STM32F4 and the ESP32.” *ChatGPT*, n.d., <https://chatgpt.com/share/6901b0e2-33b0-8001-9340-2f8664558608>.
45. STMicroelectronics. *STM32F427xx/STM32F429xx Datasheet*. STMicroelectronics, n.d., <https://www.st.com/resource/en/datasheet/stm32f427vg.pdf>.
46. Espressif Systems. *ESP32 Datasheet*. Espressif Systems, n.d., https://cdn.sparkfun.com/datasheets/IoT/esp32_datasheet_en.pdf.
47. Raspberry Pi Ltd. *Raspberry Pi 4 Datasheet*. Raspberry Pi Ltd., n.d., <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf>.
48. ChatGPT. “Response to Compare the BMP390 and the MEAS/MS5611.” *ChatGPT*, n.d., <https://chatgpt.com/share/6901b4ff-41d4-8001-9605-6776de826253>.

49. Bosch Sensortec. *BMP390 Datasheet*. Bosch Sensortec, n.d., <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp390-ds002.pdf>.
50. TE Connectivity. *MS5611-01BA03 Barometric Pressure Sensor Datasheet*. TE Connectivity, n.d., https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showDoc&DocId=Data+Sheet%7FMS5611-01BA03%7FB3%7Fpdf%7FEnglish%7FENG_DS_MS5611-01BA03_B3.pdf%7FCAT-BLPS0036.
51. ChatGPT. “Response to Compare Different IMU Sensors.” *ChatGPT*, n.d., <https://chatgpt.com/share/6901b661-c61c-8001-9d78-2137beb7346a>.
52. TDK InvenSense. *ICM-20948 Datasheet*. TDK InvenSense, Mar. 2024, <https://invensense.tdk.com/wp-content/uploads/2024/03/DS-000189-ICM-20948-v1.6.pdf>.
53. ChatGPT. “Response to Compare Two Different GPS Modules.” *ChatGPT*, n.d., https://chatgpt.com/s/t_6901b7b5518081919124b5a1e7d1ae74.
54. u-blox. *NEO-M8 Concurrent GNSS Modules Data Sheet*. u-blox, n.d., https://content.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_UBX-15031086.pdf.
55. Quectel. *L76K Hardware Design*. Quectel, n.d., https://files.waveshare.com/upload/d/db/Quectel_L76K_Hardware_Design_V1.0.pdf.
56. ChatGPT. “MCU Query.” *ChatGPT*, n.d., <https://chatgpt.com/share/68f7079a-33b0-800b-9205-72792f1cefc2>.
57. ChatGPT. “Software Research.” *ChatGPT*, n.d., <https://chatgpt.com/share/68f7076b-b5d0-800b-bdaf-9f16f2e005d8>.
58. ChatGPT. “Standards Prompt.” *ChatGPT*, n.d., <https://chatgpt.com/share/68fd3c5a-f898-800b-945c-718378b2675b>.
59. Wevolver. “I2C vs SPI Protocols: Differences, Pros, Cons, and Use Cases.” *Wevolver*, n.d., <https://www.wevolver.com/article/i2c-vs-spi-protocols-differences-pros-cons-use-cases>.
60. Laubheimer, Page. “AI Hallucinations.” *Nielsen Norman Group*, n.d., <https://www.nngroup.com/articles/ai->

[hallucinations/#:~:text=Page%20Laubheimer,to%20help%20users%20identify%20fabrications.](#)

61. ChatGPT. “Software Design Prompt.” *ChatGPT*, n.d., <https://chatgpt.com/share/69017c17-0898-800b-bbc5-bfffaad4d44b>.
62. Wikipedia. “ZeroMQ.” *Wikipedia*, n.d., <https://en.wikipedia.org/wiki/ZeroMQ>.
63. Google. “Latency + sickness + VR.” *Google Search*, n.d., https://www.google.com/search?q=latency+sickness+VR&oq=latency+sickness+VR&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQABjvBTIKCAIQABiABBiBDIKCAMQABiABBiiBDIKCAQQABiABBiiBNIBBzc5M2owajeoAgCwAgA&sourceid=chrome&ie=UTF-8.
64. ChatGPT. (n.d.). *Pi4 graphics rendering* [Online conversation]. Retrieved November 5, 2025, from <https://chatgpt.com/share/690bbb9f-3d88-800b-bc39-bff59fa66d2a>
65. ChatGPT. “Data communication protocols.” Shared link. <https://chatgpt.com/share/690d27e2-74a0-800b-91a6-38b175d65709>. Accessed 6 Nov 2025.
66. Adafruit LSM9DS1 Accelerometer + Gyro + Magnetometer 9-DOF Breakout <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-lsm9ds1-accelerometer-plus-gyro-plus-magnetometer-9-dof-breakout.pdf>
67. Sony IMX219 Datasheet https://blog.arducam.com/downloads/modules/RaspberryPi_camera/IMX219DS.PDF
68. Raspberry Pi Camer Module 3 Datasheet <https://datasheets.raspberrypi.com/camera/camera-module-3-product-brief.pdf>

Appendix E: ChatGPT Prompts and Outcomes:

[1a] Case Study 1 Prompt:

Please come up with optical components for a vehicle-based heads up display and make it succinct

[1b] Case Study 1 ChatGPT Outcome:

Here’s a tight, vendor-agnostic component list for a windshield-combiner automotive HUD (virtual image ~2.5 m, FOV ~12°×5°, eye box ~120×80 mm):

- **PGU (microdisplay):** 0.55” DLP (960×540–1280×720) or 0.45” LCOS.

- **Illumination:** RGB laser (635/520/450 nm) + moving diffuser (speckle control) or high-power LED (white + color wheel or RGB).
- **Integrator:** Light pipe or fly's-eye, $\sim 10 \times 10$ mm exit, $NA \approx 0.25$, AR both ends.
- **Projection/Relay Lens:** ~ 35 mm EFL, $f/2.8$, $\varnothing 20\text{--}24$ mm CA, 3-element (N-BK7 / N-SF10 / N-BK7), BBAR 420–680 nm ($R < 0.5\%$ /surf).
- **Aperture/Field Stops:** Metal masks at stop and image plane; knife-edge baffles, flocked.
- **Freeform Fold Mirror:** Ni-P plated Al, $\sim 90 \times 70$ mm CA, enhanced Al ($R \geq 94\%$ 430–680 nm); low-order freeform terms for keystone/field tilt.
- **Combiner (windshield inner surface):** Use OEM HUD zone; specify wedge control ($\approx 6\text{--}12$ arcmin) and **outer-surface AR** patch ($R \leq 1\%$).
- **Polarization Mgmt:** Linear polarizer (if LED) to favor s-pol at combiner; optional for lasers.
- **Dust/Protect Window:** 0.5–0.7 mm aluminosilicate, BBAR both sides.
- **Sensors:** Ambient photodiode (3–4 decades), temperature sensor on PGU.
- **Mechanics (optics-critical):** 3-point mount with yaw/pitch trims ($\pm 1.5^\circ$) and focus cam (± 0.5 mm).

Variant (dedicated laminated combiner, if windshield control is limited): 130×80 mm glass/PVB/glass with 30–40% partial reflector, zero wedge, hard AR ($\leq 0.8\%$); lets you shrink relay EFL to ~ 30 mm and fold mirror to $\sim 70 \times 55$ mm. **Typical performance from this stack:** ≥ 15 k cd/m² at eye (with $\sim 15\%$ system throughput), MTF ≥ 0.3 @ 3 lp/mm over $\sim 80\%$ FOV, distortion $\leq 2\%$ after optical + pre-warp.

Figma AI Citation:

Create me a heads up display for an aircraft that shows compass heading, altitude, attitude (orientation in space), and has a toggle on the bottom to switch between a camera and data view

Also create a camera view that shows two wings of the plane taken from the side windows with no data for the camera view



Reasoning >

I'll create a heads-up display (HUD) for an aircraft with compass heading, altitude, attitude indicator, and a toggle to switch between data and camera views. Let me build the necessary components.

I've created a complete aircraft heads-up display system with a data view showing compass heading, altitude, attitude indicator (artificial horizon), and airspeed, along with a camera view displaying both wing perspectives. The HUD features realistic flight instruments with animated values that simulate actual flight dynamics, and you can toggle between the data and camera views using the buttons at the bottom.

bottom.

Suggested next steps:

- Add additional camera angles (nose, tail, or ground view)
- Include warning indicators for altitude, speed, or attitude limits
- Add a GPS map overlay or flight path visualization

Aircraft Heads Up Display

Version 1



That looks great! But I have a few changes to request: remove the airspeed and just have the altitude. Remove the Ground Speed and Status. Make the whole design more minimalistic. Make the data view and camera view a smaller toggle in the bottom right. This needs to be programmable by C++ with the Qt graphics library so ensure that is possible and that the design is in general minimalistic.



Perfect! I've transformed the HUD into a minimalistic design that's ideal for C++ and Qt implementation. The design now uses simple geometric primitives (rectangles, lines, circles, and text) that map directly to Qt's QPainter functions. I removed the airspeed indicator, simplified the bottom info bar to show only roll and pitch, eliminated all gradients and complex effects, and moved the view toggle to a compact icon-only control in the bottom right corner. The camera view is now clean with just the two wing images and simple borders, making the entire interface straightforward to port to Qt using QWidget, QGraphicsView, or custom QPainter rendering.

Suggested next steps:

- Add grid lines or crosshairs to the camera view for alignment
- Include numerical display overlays for specific measurements
- Add warning indicators or limit markers on the instruments

Simplify HUD Design

Version 2



That looks practically perfect. My only thing is can you remove the box around attitude and just leave the rotating circle? DO NOT CHANGE ANYTHING ELSE AT ALL



Reasoning >

Done! I've removed the border around the attitude indicator, leaving just the rotating circle visible.

Remove box around attitude

Version 3



The data view looks perfect thanks! Now, one final modification. Can you make the Camera view show the photos of the plane more spread out and perfect squares? Accomplish this by shrinking the view to ensure 60% of the middle is visible and make the photos opaque as well



Reasoning >

Perfect! I've updated the camera view to show the photos as perfect squares that are more spread out. Each photo now takes 30% of the width (leaving 40% gap in the middle, so 60% of the middle is visible between them), they maintain a square aspect ratio, and are fully opaque.

Adjust Camera View Layout

Version 4



Close, now put the photos of the plane on the edges of the image and make them opaque. DO NOT MODIFY THE DATA VIEW



Reasoning >

Done! I've updated the camera view to position the square photos at the edges of the screen using `justify-between`, keeping them fully opaque. The data view remains unchanged.