4/27/2022

Arcturus's Payload

Autonomous Rover

Group 35

Team Members

Brett Parker	Computer Engineering
Adriano Gussoni	Electrical Engineering
Michael Schumacher	Computer Engineering

Senior Design - Advisors

Dr. Lei Wei	Primary Advisor	Ph.D. M.E., B.E.
Dr. Samuel Richie	Secondary Advisor	Ph.D. E.E.

Table of Contents

Table of Figures

Table of Tables

1 Executive Summary

In a race to explore and collect data from distant planets, a vast array of aerospace agencies has been working on projects that involve sending automated and / or man-controlled robotics into the depths of space to ultimately land on and explore our neighboring planets. All the while, communicating data and media back to mission control on Earth. One example of such a robot is the Mars 2020 Perseverance rover, which has scientific equipment to test for microbial life on Mars. [P2020] Another example of one of the more recent Mars robotics projects is NASA's project Ingenuity. Opposed to Perseverance 2020, which is used for ground-based studies, Ingenuity is the first aerial robot (drone) to have successfully arrive, land, and test powered and controlled flight on another planet. Surely, interplanetary exploration is still a young but quickly growing study. [ING]

Asking graduating, senior level UCF students to build a robotic rover; fully integrated, with the top-of-the-line components and technologies; within a two-semester period would be a heavy and costly task to burden. Thus, we are thankful that our sponsor, Aerojet Rocketdyne Coleman Aerospace, for providing funding for a scale high-powered rocket featuring an integrated, robotic payload. The general ideas that govern full scale rocket launches compared to amateur scale rocket launch, scale by cost, size, and complexity. These first principles include (but are not limited to): propulsions, control & navigation, and data/media transmission. Successful launching systems, recovery systems (parachutes), and payload operation are also crucial for any full-scale mission to be considered a success.

As graduating students studying engineering, the scope of the payload team was elevate the idea of "build-it yourself robotic kits" that are sold by various online vendors; by creating a custom payload design which will excel in performance compared to the afore mentioned kits. Our team intended on creating a one-of-a kind robotic payload which would meet or excel current technology in amateur robotic rocket payloads by condensing the theories and ideas that we have been studied throughout our educational career until graduating in 2022.

Many fields in technology are becoming more advanced with integration of smaller, more powerful computers. It is up to the graduating students to help progress these fields by learning how to solve problems in technical design that arise when building and testing these complex technologies. For inspiration in our payload design, our team explored the newest technologies in:

- Data acquisition (environmental sensing)
- Data transmission (media or parametric).
- Drive systems via control theory.
- Our team will strive to make the robot fully autonomous.[1]
- Reliable, live video data transmission.
- Low power electronics.

[1] The term autonomous, being very broad nowadays, is defined in "Requirements & Specifications", Autonomous actions.

2 Project Description

The basis of our team project was to make a payload that can be placed within a scale Aerojet rocket (Arcturus) to compete in an amateur rocket launch and rover competition [FAR]. This payload is a robotic rover which is housed inside a deployment canister and inserted into the payload section of Arcturus for later deployment. Given that the dimensions of the rockets hull diameter must be at least 3" from the project proposal documents; there is some flexibility in the size that the payload can have. Though, the size of the payload does affect the performance of the rocket. Because of this, our team strives to produce a compact payload.

This document will only cover the scope of the payload system (Figure 1) and is intended to be a subdocument to be appended to the main *Arcturus* Aerojet design document either by section or reference. For our project, the definition of payload includes the following items:

- Autonomous Rover (Main scope)
- Rover deployment sled (Holds rover in place)
- Payload canister (Affixed inside of rocket; holds sled)

The mechanical and aerospace engineering team has designed a "pill-shaped" canister where the rover is housed. Our rover team must meet sizing requirements brought forth from the rocket design team to be eligible for the FAR competition of the competing senior design payload teams.

The payload must be remote-controlled or autonomous to meet competition requirements. Thus, we have chosen the more challenging option; fully autonomous. Upon successful rocket ejection and after safely landing, the rover should be able to return to the origin of launch for maximum competition points.

Finally, the payload should have video transmission beginning from successful payload ground deployment to the end of its journey. The implementation of artificial intelligence will not take place in this project, making object identification and predestined pathfinding navigation unachievable.

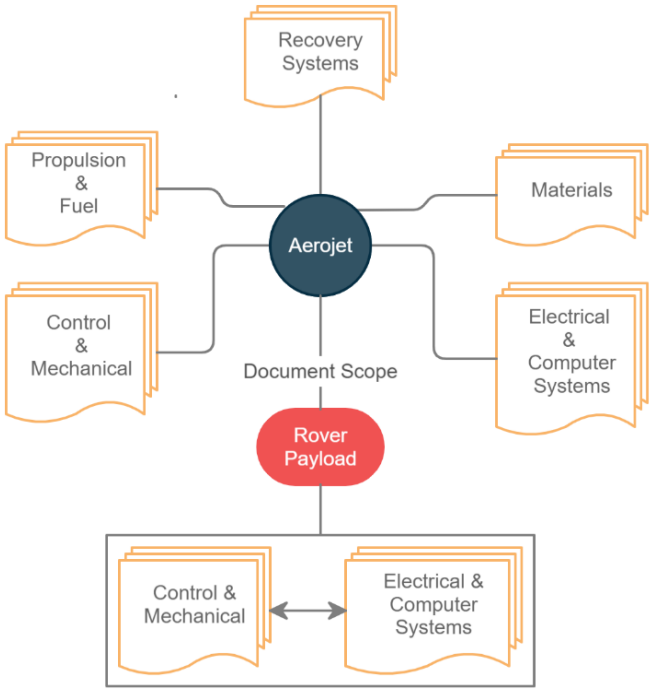


Figure 1 – High level overview of the project scope [F1].

2.1 – Motivation & Goals

Our main goal was to successfully implement as many components relevant to the FAR-1030 competition, listed in the given project proposal documents, as our part of the project allows. One of these project features is remote controlled (RC) versus autonomous rover which awards 1000 or 3000 points respectively with successful implementation. A 500-point award is given if live video data be stored on a memory card. The team hopes to incorporate all these potential aspects described in the competition document, into our design with the hopes of maximum achievement in competitive points and ultimately claiming first place.

A current problem that many companies making autonomous products face is guidance and control. Many of the systems that are being developed must be robust to handle the worlds upcoming challenges. As autonomous systems become more control savvy, power supply systems, real-time scheduling focusing on deadline success ("hard real-time system") faces new emerging challenges. Thus, these two problems in current robotics were heavily considered throughout the design of the robotic payload.

Another problem that robotics face is power consumption. Therefore, it is a goal of the team that this payload be equipped with the sufficient power for its mission. The scale distance of this mission is minor, with the main dependent variable being ground distance traveled from the launch site, as the rover will have to self-return to the launchpad. Given that the rocket will not go perfectly straight up and will have some horizontal velocity component; this distance can vary dramatically. Therefore, low power consumption would be ideal to achieve maximum operation time of the rover.

Besides successful completion of competition requirements, it would be fulfilling that the payload encompasses the minimal sensing technologies (temperature, time of day, humidity). These standard technologies should be incorporated into the final design. It should also be noted that special attention was given to the transmission of the video that is captured; to hopefully achieve uninterrupted video streaming of the rovers' journey.

2.2 – Objectives

To achieve the above goals, our team must be able to realize methods that will make these goals achievable. Please note that final design may include tools that might not be listed directly in this report that aid us in completing our goals. The team has highlighted some objectives below which will aid us in a successful and functional project and are considered top priority.

2.2.1 – Power Objectives

To reduce the power demand, a low power microcontroller and microprocessors will serve as the brain of the rover. Making decisions based on the received data from sensors. drive motors, and input/output (I/O) components used. Special attention was placed on the voltage and current regulation to the various systems of the rover. Voltage

regulation components were specifically matched to the components that they drive as this rover serves one purpose and has tight size requirements. There is minimal to no room for upgrades to the power system after the final design.

2.2.2 – Video Transmission Objectives

To complete the goal in the FAR 1030 competition requirement [FAR] of a reliable live video broadcast; the rover should feature a stand-alone video recording system. This video capture and transmission system should feature a powerful enough antenna and transmission module. This system should also feature a processor which can handle the resolution and capture rate of the camera.

2.2.3 – Payload Deployment Objectives

The objective to successfully return to the launch pad heavily depends on the time at which the rover is deployed. Deployment of the rover has been visualized in two methods:

Deployment Method 1:

To minimize the lateral distance that the payload canister will encounter during descent, the rover must evaluate the canister as soon as possible after main parachute deployment for the nose cone / motor (1,000 ft). This method involves the addition of a parachute for the rover mounted inside the canister. When the canister deploys the rover and parachute combo, the rover will pick up vertical velocity in attempt to minimize lateral drift. If this method is chosen, the team must determine the altitude at which to release the rover such that the rover does not exceed 150 feet per second free fall descent before the rover parachute deploys. This method may show up later in the document referenced at HALO (High Altitude Low Opening) deployment.

Deployment Method 2:

The deployment canister will descend with the nosecone of Arcturus. Both the nosecone and the payload canister will reach ground level where the canister will deploy the rover via an opening system on built into the canister. This method is high risk, such that the rover will land in an unfavorable area, featuring unpredictable terrain and must traverse a far greater distance, impacting power consumption heavily. Deployment method 2, while high risk, may be the team's only option if the HALO deployment method is unobtainable due to constraints from the rocket team. Deployment method 2 may show up later in the document referenced as HRLR (High Risk Low Reward) deployment.

2.2.4 – Motor Control Objectives

The objective of successfully returning to the launch pad also depends on the rover's ability to have concise control over its drive systems to eliminate potential error build up over time. Even a small amount of error introduced into the data system in terms

of viable control data (orientation, position, speed, etc.), has the potential to become unbounded under certain circumstances without error reduction using system controllers.

While our team has very minimal practice in real-world control theory experiments, The team at minimum should be prepared to implement feedback from the drive motors to the processing system. Having a feedback system attached to the motor (the data feedback is either the position of the drive shaft, or output speed of the shaft) allows for more precise updates to the speed controller, allowing for more control over the error signals that the rover produces from start to finish. To achieve precise control, a form of digital controller was implemented into the rover (more about these digital controllers in Section 3 - Research and Relevant Technologies.

2.3 – Engineering Requirements & Specifications

The requirements that our project must abide by comes mainly from limitations put on to the payload team from the rocket design team. The FAR 1030 competition tries to give freedom to the creator of the rocket, only placing requirements on the rocket itself which are related to, but not a part of this payload project scope. The below featured requirements are the most important to the success of our project and may be confirmed through testing.

2.3.1 – Autonomous actions (FAR 1030 Requirement option 2)

The rover should carry out its tasks without intervention or any human guidance. To further define "Fully autonomous" for this project: The rover shall be required to:

- Self-wake from sleep period. (Sleep period defined as: time from location parameter value acquisition and finish storing GPS location to memory on the launch pad → until ejection from rocket (ground 0').
- Boot microprocessor on descent, in effort to gather video data as soon as possible.
- Sense when on ground level, and that it is no longer descending. (Full wake of MCU and safe to leave deployment sled)
- Guide its way from the landing location to the launching location using RF signaling, GPS, or pathfinding[2].
- Periodically gather general, **non-control** related sensor data at a rate of one sample per minute until completion of the journey, stored to memory.
- Have the ability and agility to avoid obstacles of a certain size (definition of "obstacles" and requirements below).

[2] Due to the time constraints on this project, a pathfinding algorithm will only be implemented into the rover's software if time allows.

- Acknowledge successful return to home within 15 ft of launch pad.
- Avoid obstacles that are within size and range of two feet in diameter and 6 feet away.

It should be specified now, that A "backdoor" (user operated override) was installed as this method should be highly considered and practiced; considering results of the Path-finder mission; which almost resulted in mission failure, due to a conflict in scheduling hard real-time task sets (priority inversion) [PATH]. While our project will remain on earth, if at any point the rover shows sign of potential harm to property or people, it would be wise to have this method to prevent damage to surroundings. This feature shall be the only user-controlled feature after launch of the rocket and used only in emergencies.

2.3.2 – Power Supply / Systems

Some of the rover's functions are operational for the entire descent and not just after making ground contact. We need to ensure a base level of power delivery for live video transmission and GPS tracking.

- The power system should be able to supply enough power for the live video transmission subsystem.
- The rover should be able to supply enough current to the motors during its operation.
- The power system should be responsive enough to change directions or brake before the rover contacts any obstacles (as determined by the control logic).
- The power system should deliver power to all the main drive motor controllers simultaneously and equally or as needed to ensure proper operation
- The power system should consume as little energy as possible while the rover is idle (in flight) as to extend the battery life and duration of the rovers' travel back to the launchpad.
- Main power wires must be held firmly in place at all times including take-off.

2.3.3 – Live video transmission

Video shall be required to be transmitted from the rover, back to the team. There are no FAR competition requirements that place a value on the quality, or packet loss of the transmission. It is our personal team requirement that:

- Video transmission distance must be a minimum of 400 ft[3].

[3] 400 ft value is an estimate (from the Arcturus team) of minimum rocket lateral drift from the launch pad as the rocket ascends to 10,000 feet

- The data transmission from the video transmission circuit should be able to work on RF bands of 5.8 GHz (channel on this band was determined later) as to not interfere with the transmission from Arcturus' transmissions.

2.3.4 – Other Requirements

The requirements below, briefly describe some additional requirements. Some of which have been given from the rocket design team. Failure to meet these requirements will result in unpredictable actions during rocket launch and recovery. The values from the below listed requirements are numerically and compactly depicted in Table 1. While some of these requirements are not specifically geared toward computer and electrical engineering, failure to meet sizing requirements will ultimately lead to our team being ineligible to compete in the FAR 1030 competition.

Maximum mass

- Total weight of our payload (combined ejection sled, attachment system, and rover) may not exceed 9.5 lbs. (4.31 kg). While the rover itself must weigh less than 2.19 lbs. (997.9 g).

Maximum size

- The maximum size of our payload comes from the design of the Arcturus team (Appendix A1): (including sled and canister) is 40.64 cm length, 12.7 cm outer diameter; and should be smaller than the above values.

Power consumption

- Depends on what size drive motors are used with respect to the size of the rover. The goal was to be as energy efficient as possible, so the current predicted requirement is no more than 20 W/hrs.

Obstacle avoidance

- The definition of "obstacle" in terms of this project: We want the rover to be able to avoid collisions with obstacles in the direct 120° field of view in the forward direction of the rover. The size of the obstacles must be at least 2 feet across and 6 feet away. We want our rover to be able to avoid this object no later than 2 feet away from said obstacle.

Body / frame materials

- The frame of the rover should be rigid, and able to withstand the forces experienced during launch, apogee deployment, payload deployment, landing, as well as terrain traversal

2.3.5 – Requirement Specifications Table

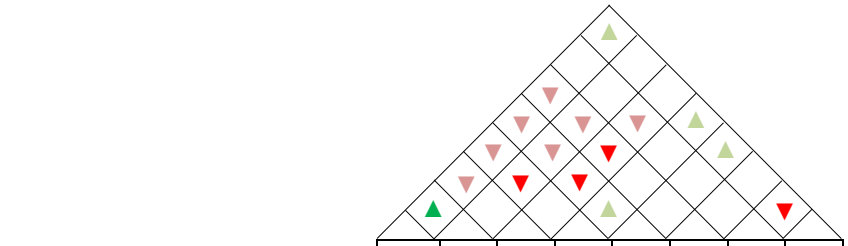
Featured EE./CpE. Rover Specifications Table	
Parameter	**Value**
Obstacle avoidance	- Yes ☑ No ❑
1. Object detection size 2. Detection radius 3. Detection range 4. Avoidance range	- 2 ft diameter - 360° - 2 ft - 1 ft minimum
Video capturing and transmission	
5. Distance	- 450 feet minimum
Able to move under load	
6. Weight to move on top of rover	- 500 grams minimum
Autonomous	- Yes ☑ No ❑
7. Features	- Self-maneuvering - Self-return to home (400 ft Distance) - Must traverse terrain reliably
Power Consumption	
8. Steady state conditions: 9. Transient turning and acceleration change	- Less than 12 W/hr. - Less than 20 W/hr.
Green = testable parameter	

Table 1 – Project Engineering Specifications / Requirements

2.4 – Quality Function Deployment

2.4.1 – House of Quality

The House of Quality is a graphical method used in Quality Function Deployment that shows the tradeoffs and their impact of the adjustment of engineering requirements to the final design. If at any point during the design and engineering requirement is needed to be changed, this diagram will show the overall impact to the project and the various tradeoffs encountered. Since the project has quite a few requirements that must be fulfilled, the engineering requirements depicted have the most impact on the overall aspect of the finished product and many of the overall systems fall into one of the nine categories depicted in Figure 2. Our most sensitive requirements are the power consumption, project cost, and the project physical size.



▲ = Strong Positive Correlation
△ = Positive Correlation
▼ = Strong Negative Correlation
▽ = Negative Correlation

"+" = Positive Influence
"–" = Negative influence

			Power Consumption	Total Payload Size	Project Cost	Return to Home	Terrain Traversal	Uninterrupted Stream	Secure Payload (Tiedown)	Project Weight
			Engineering Requirements							
			–	–	–	+	+	+	–	–
Power Efficient	Marketing Specifications	+	▲	▲	▽	▽	▽	▽		△
Compact		–	▲	▲		▼	▽	▽		
Cost		–	▽		▲		▼	▼	▽	
Reliable		+	▽	▼		▲	△			△
Autonomous Features		+	▽	▽	▼	△	▲			△
Video Quality		+	▽	▽	▼			▲		
Durable		+			▽				▲	▼
Agile		+	△			△	△		▼	▲
Engineering Requirement Targets:			≤19 W / hr	≤ 40.64 cm	≤ $2,000	400 ft Travel to home reliably	≤ 2 feet away (obstacle)	≤ 2 feet away	≤ 8G force	≤ 997 g

Figure 2 – House of Quality diagram. [F2]

2.5 – High Level Overview

This section is devoted to a high-level overview of the primary systems. Their block diagrams follow.

2.5.1 – Rover Hardware / Electrical / Software Systems Overview

The hardware and power supply can be divided into the following subsections (the below Figure 3 shows the general overview of hardware, power, and information flow:

Power supply and regulation.

- Power source(s).
- Computer and sensor voltage regulation.
- Drive motor current limiters and voltage regulation.

Drive systems

- Drive motors.
- Motor controller drivers.

Computer systems

- Microprocessor.
- Microcontroller.

Media and sensing components

- Environmental sensors.
- Range sensors.
- Orientation sensors.
- General purpose input and output.

2.5.2 – Rover Software System Flow Overview

The software overview depicts a timeline of the rocket launch (from prelaunch to rover return to home). It must be stated that the team will have no control over the rover once the rocket has undergone launch. Therefore, any data that needs to be stored prior to launch must take place in the prelaunch section. After which the rover's electronics are set to minimal power consumption mode. Figure 4 shows the general timeline / algorithmic flow of the following listed software components:

Prelaunch data acquisition

- Stores GPS location into memory.
- Deploy homing beacon (optional).

Launch & Apogee (Rocket)

- Sleep (low power mode).
- Await proximity sensor trigger

Rover canister deployment

- Boot microprocessor, start MCU clock and begin acquiring GPS location data.
- Start video transmission.

Touchdown of payload

- Full power deployment to entire system.

Traversal and parking

- Control loop for direction sensing (also includes sensor data acquisition).
- Once "home" → MCU self-shutdown.

2.5.3 – Original Hardware and Power Systems Block Level Overview

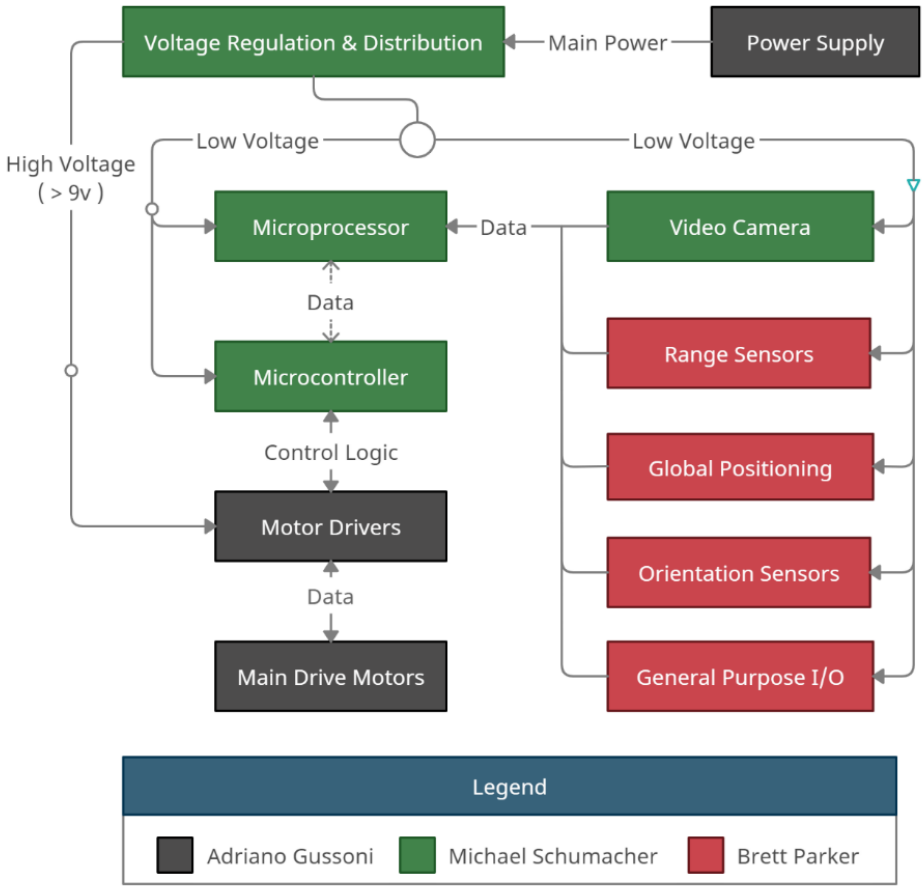


Figure 3 – Hardware Block Diagram. High level overview of the power and hardware systems
**Final overview may differ slightly* [F3]*

2.5.4 – Hardware and Power Systems Block Level Overview

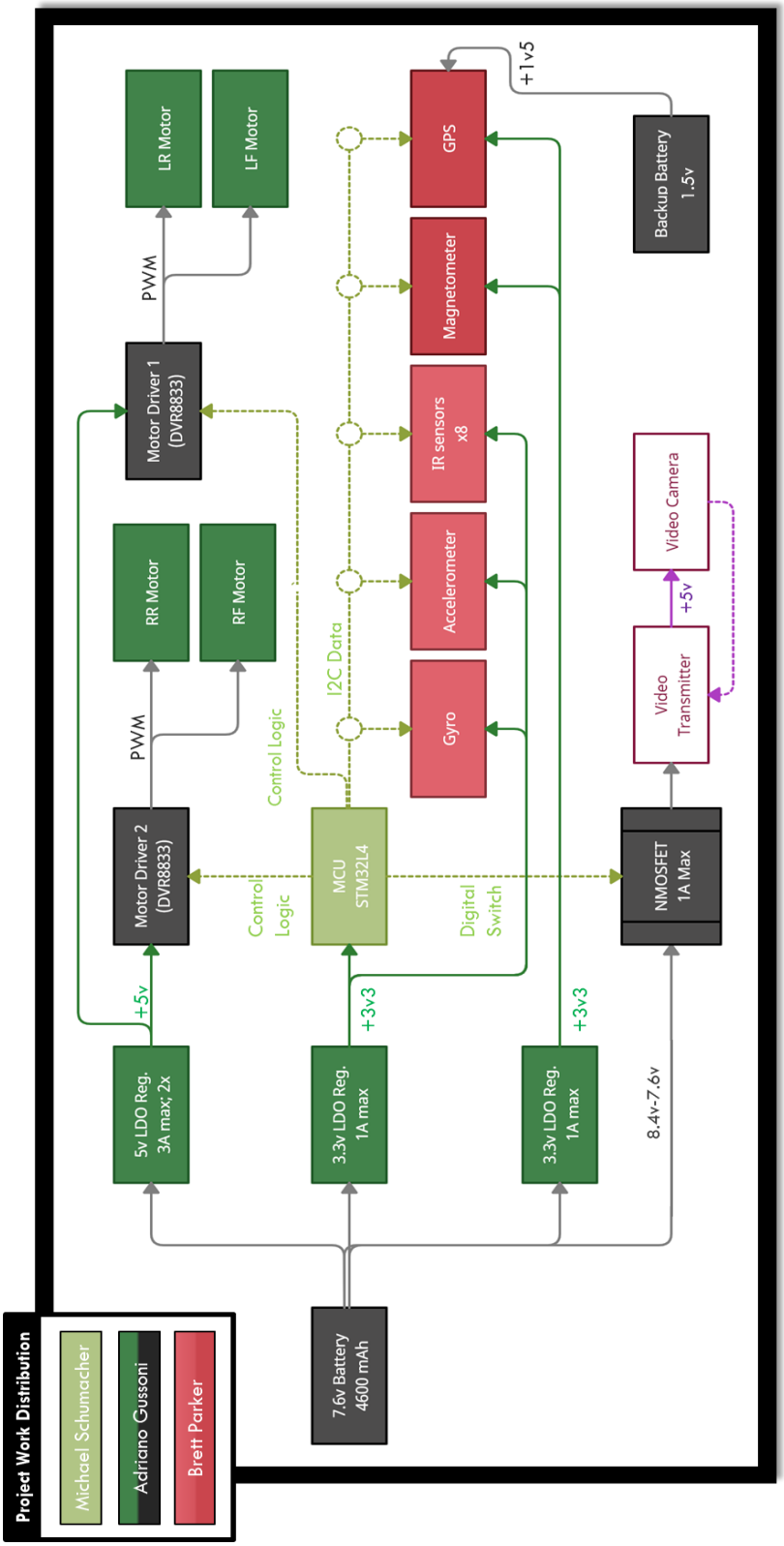


Figure 3.1 – Hardware Block Diagram. High level overview of the power and hardware systems Final Design

2.5.5 – Software and Control Systems Block Level Overview

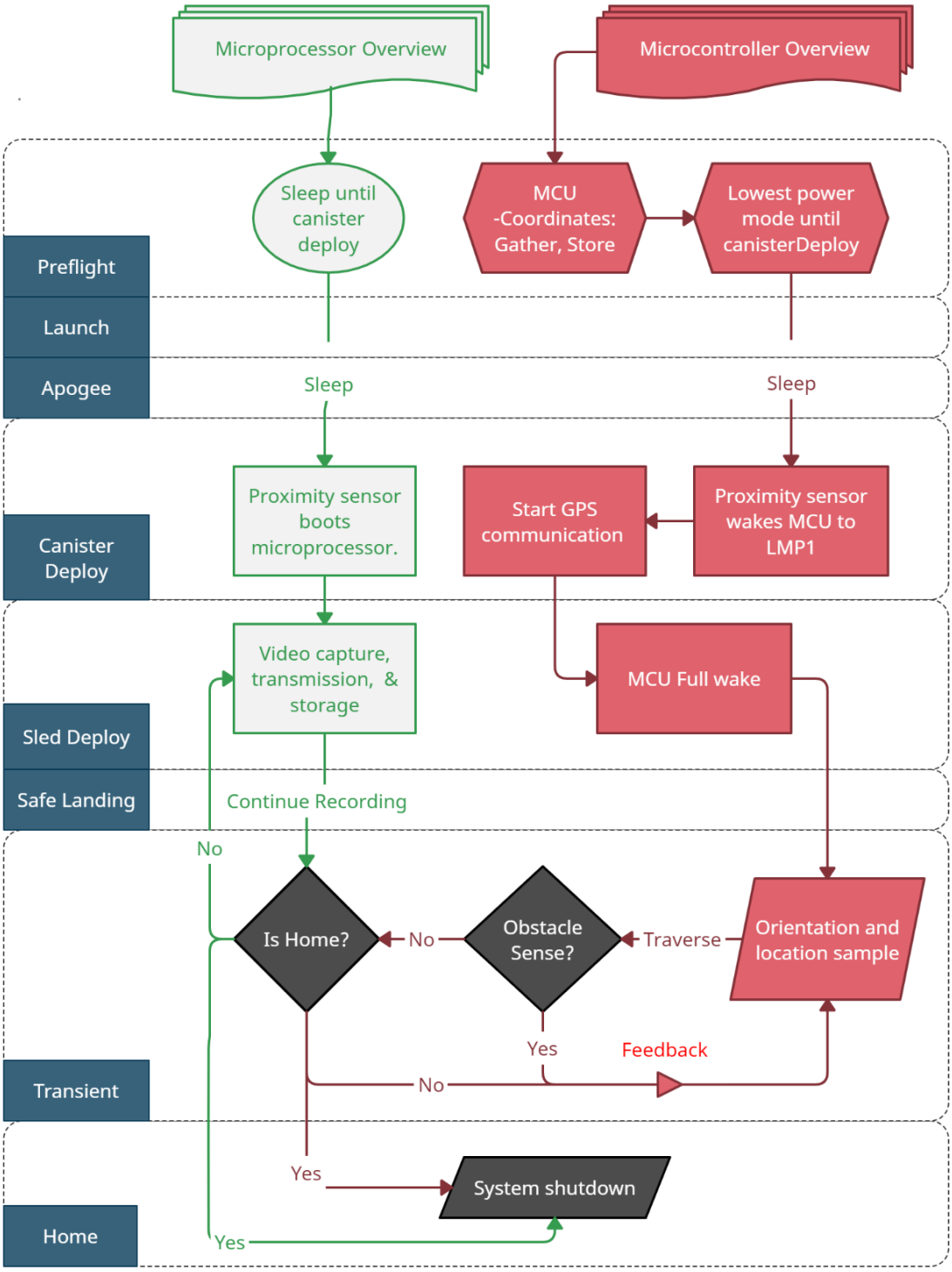


Figure 4 – Software Block Diagram. High level overview of the software flow from the prelaunch state to the "returned to home" state. [F4]

Final overview may differ slightly

3 Research Related to Project / Part Selection

This section is dedicated to the related projects and products that exist which share similar technologies to the rover project. Following related projects and products, is an overview of the technologies that exist that could potentially be useful for our team project. Finally, a detailed comparison between different parts is examined to determine the best component for the project.

3.1 - Existing Similar Projects & Products

User operated, autonomous, and autonomous hybrid products and projects come in a variety of sizes. Most of which have different functions, that help improve quality of life conditions and productivity in many of the leading work fields. As technology in these fields improve, many more products will emerge, all built to aid humans.

3.1.1 – Early Industry Project – Apollo Lunar Landers (1971)

Developed within a time frame of 17 months, the lunar lander (Figure 5) was the first vehicle to roam on the moon during the Apollo 15 mission. With a total weight of 210 kilograms, this rover was designed to carry a payload of 490 kilograms including the weight of two passengers. The rover played an important role in effort to reduce the amount of labor the crew had to put forth in the gathering of scientific data.

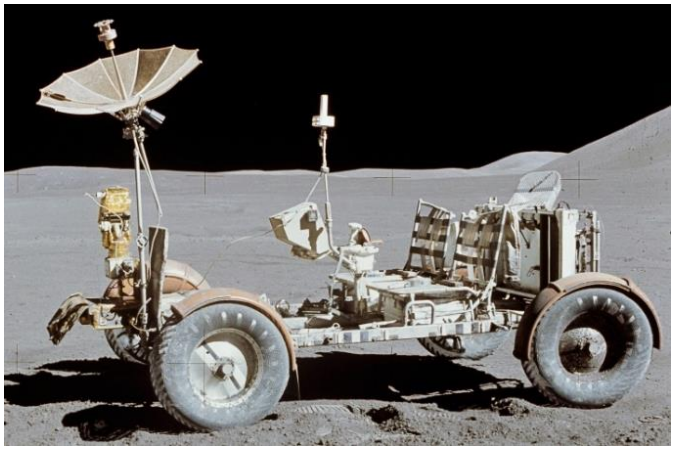


Figure 5 – Apollo 15 lunar lander
Reprinted with permission from [F5]

Equipped with 36 volt, 121-amp hour batteries, the Falcon could travel vast distances over the rough terrain. The rover used four individual drive motors, one for each wheel, with a total of 0.25 horsepower per wheel and capable of reaching 10,000 RPM. 0.1 horsepower steering motors were attached to all four control arms to allow maximum turning radius. The longest single traversal distance capable during the Apollo 15 mission was 12.5 kilometers with a maximum range of 5.0 kilometers from the lunar lander. The design of the rover frame allowed it to compactly fold and fit inside the lunar lander where, once landed, the rover was assembled by the two crew.

In these early years, the first rovers had been operated by the user, with minimal automated navigation. The rover was operated with the use of a simple control panel that allowed forward, back, left, and right movements. This panel also featured information about the power system and navigation of the rover relative to the lunar module. Despite being man operated, the 38-million-dollar lunar lander program managed to expedite the gathering of data on Apollo missions 15, 16, and 17. [APP]

3.1.2 – Latest Industry Project – Rover (2020)

Deployed February 18th, 2021, the Perseverance rover (Figure 6) was set out to search for signs of microbial life on Mars. This hybrid autonomous rover weighs roughly the same amount as a compact car at 1,025 kilograms and can be controlled back home on Earth, many miles from Mars. Perseverance is autonomous and can roam around the planet without user input. Which is beneficial considering it takes roughly 5-20 minutes for a radio signal to reach Mars. The rover is equipped with a mechanical arm which holds a drill to gather material samples of Mars' surface. Perseverance is fit with the latest environmental sensing technology and features 7 primary instruments:

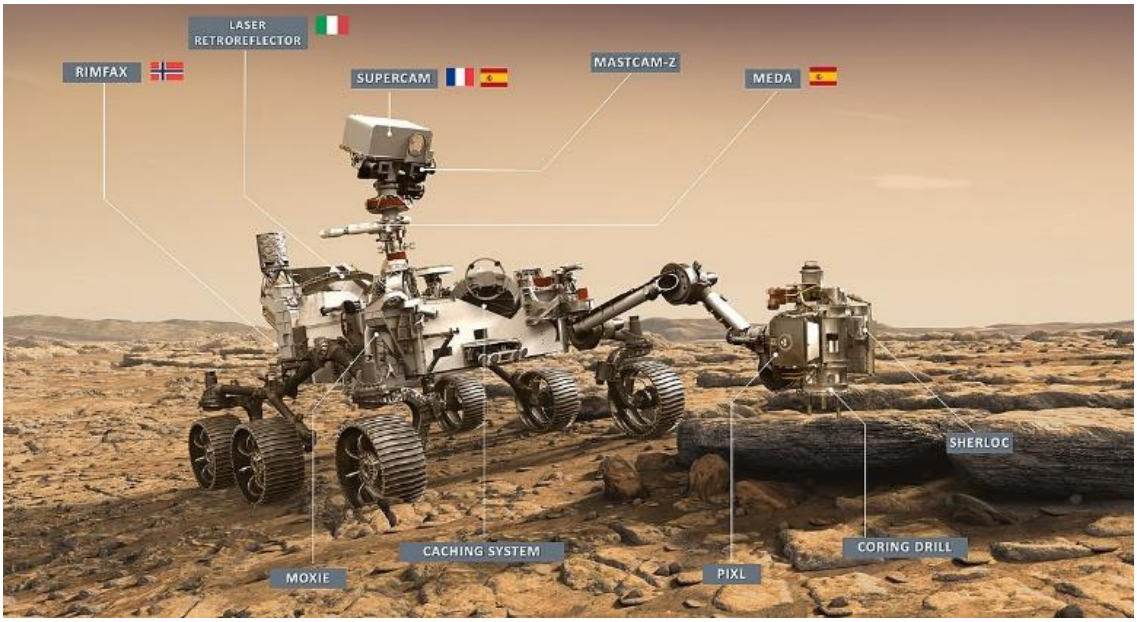


Figure 6 – Perseverance rover, 2020
Reprinted with permission from [F6]

"Mastercam-Z"	• Panoramic and stereoscopic imaging capability
"MOXIE"	• Carbon dioxide to oxygen converter
"MEDA"	• Set of sensors that measure temperature wind speed, pressure, humidity, and dust size
"PIXL"	• An X-ray spectrometer for fine scale elemental composition for Martian material
"SHERLOC"	• An ultraviolet imaging laser to determine fine-scale mineralogy and organic material
"RIMFAX"	• A radar to provide centimeter-scale resolution of geological structures
SuperCam	• Provides imaging and chemical composition of elements on Mars.

These sensing technologies combined with an on-board scientific material cache will allow scientist to progress the exploration of Mars; someday leading to its colonization without having a human ever step foot on the "Red Planet". [P2020]

3.1.3 – Industry Consumer Product – Autonomous Case Equipment

The latest lineup of industrial grade agricultural equipment has seen advancements in autonomous technology with driverless tractors produced by Cape. The pictured Cape tractor is one of the latest cape autonomous tractors (Figure 7). Larger autonomous tractors are being produced by case power for easy cultivation (see this Case industry sized platform) This farming equipment has been fitted with wireless technology, GPS, and powerful computers to allow automation of tasks around the farm to take place. Automated equipment such as harvesters, tractors, and fertilizer sprayers can operate around the field together within very close proximity of one another, accomplishing the trivial tasks and leaving employees to focus on more important tasks around the site (see John Deere Future of Farming).

Figure 7 – Case IH autonomous tractor
Reprinted with permissions from [F7]

Control algorithms and artificial intelligence have been placed into the framework of these machines; many of these driverless vehicles operate tandem on tasks. For example, a grain harvester must have a way to unload the harvested product requiring the use of not just the harvester, but also a tractor trailer. The owner may oversee the actions Productivity is increased with these machines by saving fuel and idle time with effective task scheduling. It is likely that many new lager autonomous platforms will be developed such as these tractors to maximize efficiency in complex systems.

3.1.4 – Defense Industry Project – Lockheed Martin's S.M.S.S.

The Squad Mission Support System (S.M.S.S.) was developed by Lockheed Martin for the U.S. Army (Figure 8). Designed for team-based support logistics, the S.M.S.S. was successful at demonstrating autonomous traversal via a satellite in 2013 since then has it has been in active use in many countries. This platform can aid soldiers by hauling a payload of 540 kg: taking the load off the soldiers' journey. With six independently driven wheels, this platform is powered by an 80-horsepower, turbo-diesel motor which shows an example of an autonomous system which does not rely on an electric drive.

Equipped with long-range antennas, laser range detector (LiDAR), and both infrared and color cameras, the S.M.S.S. can reconstruct its environment to self-traverse all sorts of terrain, even water. Using onboard computers, the S.M.S.S. can navigate via supervised[4], preset waypoints via satellite or may be user operated. The vehicle allows the user to operate controls either through voice or the control panel via a communication network.

The modularity of the S.M.S.S allows it to be deployed as a scout, mobile power system, or mobile communications platform allowing complete versatility which benefits the platoon not only by hauling heavy equipment, but also reducing the number of soldiers on the field. [AT]

Figure 8 – Squad Mission Support System
Reprinted under the Fair Use copyright act, Section 107, Purely educational.

https://t2conline.com/smss-squad-mission-support-system-for-army/

[4] All instances of these related projects have autonomous functions. Wither it be tracking and following or self-navigation. This does not imply that the commander of the device it unable to control the device. All autonomous action may be overridden at any time and if applicable, turned over to human decision and control via a remote network.

3.1.5 – Consumer Product – WORX Landroid, Robotic Lawnmower

The WORX Landroid autonomous lawnmower provides the everyday consumer the ability to mow their lawn hassle free. The top-of-the-line WORX Landroid (item WR155), features a 20-volt, 6.0 amp-hour batter which can provide a mow on yards up to half an acre. The "Artificial Intelligence Algorithm" (AIA) technology that the Android is equipped with allows it to take precise right angle turns which aids in the robot's ability to traverse narrow cutting paths. The AIA technology also ensures the rover will take the best possible path to avoid cutting the same area twice. To avoid obstacles, the robot features bump sensors.

Equipped with various environmental sensors, the Landlord can sense weather conditions; if it is raining, the robot will remain at the charging station until the rain passes. The Android is also can return to the charging station on its own, by sensing the battery levels and when the battery is low. Much like the industrial products, this autonomous robot can also be controlled with the wireless 2.8 GHz band (802.11 b/g/n) if the owner chooses so. See WORX Landriod for more info.

3.1.6 – Most Relevant Project - TEAM HŌKŪLELE FAR 1030 Rover

In the 2021 FAR 1030 Rocket/Payload competition, Team Hōkūlele presented this radio-controlled robotic payload which was housed inside of a high-powered rocket let for the 30,000-foot launch competition. Shown in Figure 9 is the radio-controlled rover inside of the payload deployment canister. The rover has a forward faced camera which transmitted and/ or stored video.

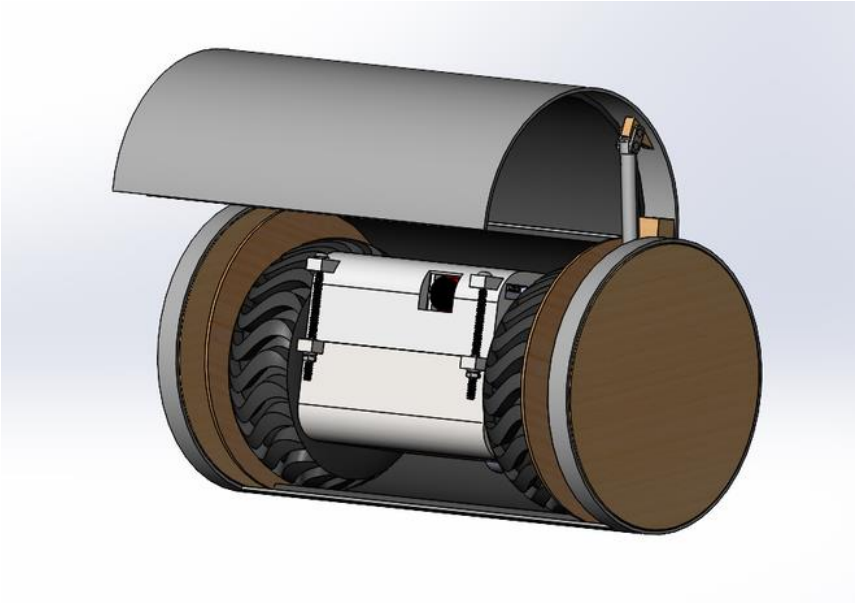


Figure 9 – Team Hōkūlele's RC rover inside deployment canister.
Reprinted with permissions from [F9]

The design is lightweight to reduce adding unexpected stability issues with the rocket during launch and the burn out stage of the motors, but also durable enough to survive impact from the canister landing. The rover competed in the "10 foot" competition, where, once the rover was successfully deployed, would travel a forward distance of 10 feet. Although the rover was only traveling 10 feet, it was designed with the intention of being operated a few miles away. [TH1030]

3.2 – Relevant Technologies

The recent development of autonomous, robotic technology includes a vast number of technologies. Most of these technologies are not solely for robotics; but have evolved from simpler systems and technologies. Our team has overviewed some of the more prevalent technologies found in autonomous robotics:

3.2.1 – Orientation Sensing

Magnetometer

Magnetometers measure electric field or magnetic dipole moments. Commonly used to measure the earth's magnetic field in the applications of a digital compass. Magnetometers will also be found in metal detectors.

GPS (Global Positioning System)

A satellite-based radionavigation system owned by the United States. Applications of GPS include determining location, navigation, tracking, and mapping. Applications vary from avionics systems to consumer based mobile products.

Accelerometer

Accelerometers are electromechanical devices that measure proper acceleration. Proper acceleration is the acceleration on an object from a rest frame of reference that does not have to include earth's gravity. In simpler terms, accelerometers measure forces acting on an object. These devices and measure forces in up to three dimensions. Applications of accelerometers are seen in unmanned drones for stability, airbag deployment systems, mobile consumer electronics, and self-balancing robots.

3.2.2 – Object Sensing

Infrared Range Finding:

The main technology that is used for object sensing is IR or infrared light sensing. The IR sensor technology sends out a very specific light wave frequency and has a receiver that retrieves the reflection of that light wave from whatever object was hit, as seen in Figure 10. From the time taken for the light to reach the receiver, the sensor can give an accurate distance reading of the object that reflected the light wave.

Since IR sensors use a beam of light to calculate the distance (which are very small in diameter), these sensors are very precise however have small coverage. There is a chance that if the sensor is not pointed directly at the object that the sensor may not be able to pick up the object since its light wave misses the object. However, IR sensors are very accurate up to around 0.3 meters for most small versions that would fit in our design.

For long range IR sensors there is a newer Lidar technology which is a combination of light waves and radar. These advanced IR sensors can reach much farther distances accurately and provide more data than regular IR sensors. With Lidar sensors it is common to see servos attached to the sensor to increase the angle at which the sensor can be used to pick up object distances.

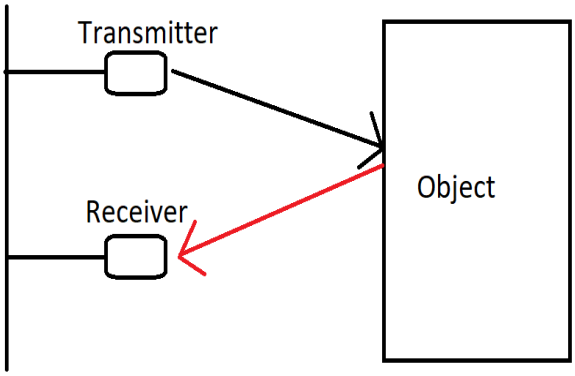


Figure 10 – Example of Infrared Laser Technology [F10]

Ultrasonic Range Finding:

Another useful technology in the same vein as IR sensors is ultrasonic sensors. These ultrasound sensors use the same method of sending out set frequencies above the human hearing spectrums and receiving them, however ultrasonic sensors use sound waves instead of light waves. This generally makes ultrasonic sensors have a farther reach than IR sensors. Since the ultrasonic sensors are using sound frequencies, this makes the environment affect them more so than IR sensors.

If the sensor is operated in a noisy environment the ultrasonic sound waves may end up being interfered with which will in turn, make the sensors transmit incorrect data to the processing system. Ultrasonic sensors could also be viewed as "the first line of defense" in obstacle detecting hardware. Since the sensor has a wide angle of projection, larger obstacles can be seen by the ultrasonic sensor in an earlier timeframe than IR technologies, which might be more useful in a more open environment (less foliage).

3.2.3 – Drive systems

Stepper Motors

Brushless DC motors where one full revolution of the shaft is divided into steps. The motors can be commanded to go to a position without the need for feedback, assuming the motor is rated for the torque needed. Used in precision applications such as 3D printers, flatbed scanners/printers, and camera lenses.

DC motors ungeared

Another brushless DC motors, usually very fast in rotation of the output shaft. Precision of the position of the shaft is usually not a concern for these applications but these

motors do have options for precise control through feedback wires. Some applications of DC motors include medical rotary tools, handheld power tools, robotics, and children's toys.

Geared DC motors

Geared DC motors are the same brushless DC motors as stated above, but the output shaft is governed with a gear set. The rotation of the motor through the gear set reduced the output RPM of the shaft but provides far greater torques. Applications where geared DC motors excel are larger operations of DC motors such as factory equipment, automotive systems, and industrial robotics. Miniature versions of these geared DC motors are also produced for smaller applications which require a high torque output.

3.2.4 – Digital Controllers (Control theory / Algorithms)

Precision, error correction, and stability are needed in automation. Controllers are used to help implement with these parameters. In autonomous control, closed-loop feedback is used. Figure 11 depicts a high-level overview of a closed loop feedback system. Output data is fed back into the system via sensors to a digital controller where mathematical operations can be used to processes and manipulate future output data and ultimately correct the new output signal to the desired level. Control systems exists everywhere, from automatic heating irons to the autopilot system on a commercial airliner.

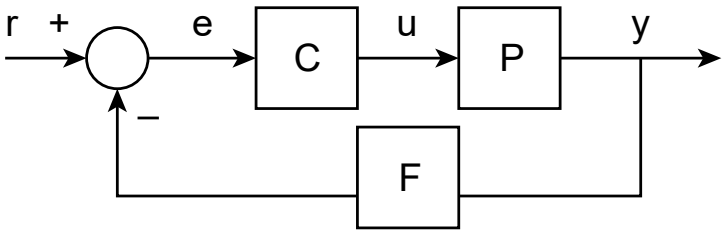


Figure 11 – Typical closed-loop feedback block diagram.
r = input, e = error, y = output
C = Controller, P = Plant, F = Feedback/Sensing
Reprinted with permissions from [F11]

3.2.5 – Live Video

The main technology used in live video transmission is the radio frequency transmitter (Figure 12). These transmitters can send radio frequency all the way from KHz to the GHz range. They normally take in some type of data and convert that data to a of a select frequency that then is sent out to any listening receivers on that frequency in some given distance.

When taking a closer at what the actual transmitter technology does, we see it is broken into three stages. First the transmitter chip has an oscillator that creates alternating current waves at what whatever frequency the transmitter is set to transmit at. This wave, named the carrier wave, and the information that is going to be transmitted is then moved to the modulator.

The modulator then adds information in one of two ways. Amplitude modulation (AM) will slightly increase or decrease the power of the carrier wave depending on the input information that was to be sent. Frequency modulation (FM) instead changes the frequency of the carrier wave. After this carrier wave has been packed with information it is then sent to the amplifier.

The job of the amplifier is increasing the power of the carrier wave which in-turn makes a more powerful and stable broadcasted wave. This wave is then sent to an antenna that converts the wave into a radio wave and broadcasts the wave to the surrounding area. The antenna sends out the data in a spherical shape which is important when deciding the position of the transmitter antenna and receiver antenna.

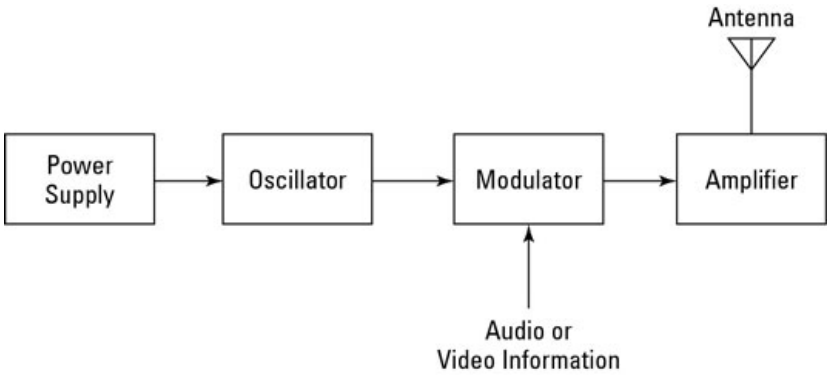


Figure 12 – Video transmission technology overview [F12]

3.3 – Strategic Components and Parts Selection

3.3.1 – Close Distance Sensors (0m – 0.3m)

We used sensors to aid in the obstacle avoidance of the rover. When looking at what type of sensor to use, it needed to be low cost, smaller than 3cm, and functional in a desert like environment. There are two common types of sensors that can be used for our needs which are ultrasonic and laser/LED sensors.

The big difference between an ultrasonic sensor and laser sensor is how they produce distances. The ultrasonic sensor sends out soundwaves of a specific frequency listens for the echo to determine distance, while the laser sensors use light in much the same way. This distinction is important if we choose the ultrasonic sensor, since we have constraints on what wavelengths we are allowed to use to not interfere with communication waves.

Table 2 compares the best sensors suited for our project, one ultrasonic and one laser. In the end we choose to use 8 of the laser sensors instead of the ultrasonic sensors. The reason we choose the laser senser over ultrasonic is because it beats out the ultrasonic sensor significantly in size. We need a sensor that is as small and light as possible which the VL53L0X is about 3x smaller than the CUSP-TRS0-18-2400-TH.

It's also important to point out the range in distances we need since the ultrasonic sensor does have much farther reach than the laser sensor. For our rover, we use the sensors to sense objects that are at max 0.3 meters away. This means the ultrasonic sensor would be significantly overkill for our project's needs.

Another factor in choosing the laser over ultrasonic is the total area the sensors pickup. The lasers beam is about 900nm wide while the ultrasonic sensors will roughly pick up any echo from 60 degrees both ways. Although this a significant amount of area covered by the ultrasonic sensor, we again believe the objects will be so close and consist of things like rocks and bushes that 2-3 laser sensors will be sufficient in picking up objects in front of the rover.

The last important deciding factor is how ultrasonic sensors act outside. We are not sure how much sound wave activity will be going on around the rover and this may severely harm the accuracy of the sensors. Ultrasonic sensors are known for picking up a lot of noise when used outside compared to laser sensors.

Close Distance Sensor Comparison

Part Name	**CUSP-TRS0-18-2400-TH**	**VL53L0X**
Part type	Ultrasonic Sensor	Laser Sensor
Distance range (m)	0 - 18	0 - 2
Dimensions (cm)	1,65 x 1.20	.44 x .24 x .1
Price ($)	5.0	5.04
Voltage	5 V	3.5
Weight (g)	2	0.15

Table 2 – Close distance sensors comparison

3.3.2 – Far Distance Sensors (0.3m – 1.8m +)

For long-range sensing it is common to use the same ultrasonic sensors from above. Another popular long range object detection plan is using a camera with some AI for object detection. We would like to use Lidar in this project as it is a relatively new technology and, in the last couple years, has become more affordable. Lidar is another kind of IR laser sensor like the small sensors mentioned above, however Lidar has a significant range and information boon that the previous IR sensors don't.

Unfortunately, with Lidar being a newer technology there is not many options to choose. In our case, we would like to spend as little money on this sensor as possible since it's not the most important part of our rover. The first big comparison between the two Lidar sensors is the actual components. The TFmini is just the sensor while the RPLIDAR is a 360-degree rotating sensor. In the case of our project, going a full 360 degrees around us is not necessary since we are going in a relatively straight line back to the launch pad.

On the other hand, the TFmini has no rotation to it at all, it points at a single specific point with 2 degrees of vision. For us to use the TFmini in our project and have it more useful than a normal IR sensor, we would have needed to attach it to a motor and have it rotated enough to scan in front of the rover. This would be a significant boon to our flexibility compared to the full 360-degree sensor since we can pick a motor that fits the size constraints of our project.

Long Distance Sensors comparison

Part Name	**CUSP-TRS0-18-2400-TH**	**TFmini S LiDAR module**	**RPLIDAR A1M8-R6**	**Raspberry Pi Camera Module V2**
Part type	Ultrasonic Sensor	Lidar	Lidar	Camera
Distance range (m)	0 - 18	0.1 – 12	.15 – 12	0.3 - ∞
Dimensions (cm)	1,65 x 1.20	4.2 x 1.5 x 1.6	9.8 x 7.0 x 6.0	2.3 x 2.5 x .9
Price ($)	5.0	40	99	32
Voltage	5 V	5 V	5 V	3.3 V
Weight (g)	2	12	170	3

Table 3 – Long Distance Sensors comparison

The final reason the 360 sensor is not reasonable for our project is its size. Although we may be able to find a way to fit a 9.8 cm component into our project, it is far out of our initial project requirements. When comparing the TFmini to an ultrasonic sensor the previous points against the sensor can still be used here. Ultrasonic sensors become unreliable when there are other soundwaves around and it makes long distance sensing too unreliable.

The last long range object detection we would like to compare is Lidar vs AI camera object detection. For this comparison we used the raspberry pi camera module V2 because of its compatibility with our PI board and its popularity in the topic.

The first thing to note with this camera is the distance it can scan varies wildly on the AI programming running behind the scenes. Assuming the camera stands still the field of view of the camera becomes larger the farther out you go from the camera. In our use the camera would be fully capable to capture the ranges we want (0.3m – 1.8m). There are 2 important differences to the AI camera method and Lidar sensors which is backend code and computation power.

The Lidar sensors send distances through I2C or UART to the computer or MCU hooked up while the camera sends frames of varying resolutions to the computer that then must be processed into data that can be fed into a neural network. Then the neural network will finally predict the distance of whatever objects it found to the MCU. This is significantly more processing power required than the Lidar sensor. The only reason to add AI capabilities to this project would be in the novelty and not much practical use.

3.3.3 – Camera

The camera for our rover is responsible for the gathering of video for our live stream requirement. Since we initially planned to use some type of Raspberry Pi for the live video transmission, we compared the general Raspberry Pi branded camera verse other camera models to see if the Raspberry Pi camera is really the best option. Although we moved away from the Raspberry Pi design, the camera comparison is still useful for our live video system.

After researching the best camera models to use for the Raspberry Pi and comparing them, we find there is not much difference between the camera models. The first relevant distinction is what model of the Raspberry Pi the camera can connect to. At the time of comparison, we are leaning toward using the Raspberry Pi zero which means the only camera we wouldn't be able to use from above in the Pi NoIR. If we do change our Pi board it is important to note that the ¼ inch mini camera will no longer connect to our board, however there is a version very similar to the ¼ inch mini camera that is applicable to other Raspberry Pi versions, part name B012QRGUCQ.

Although each camera has the same resolution capabilities, its crucial to note that the Pi NoIR camera blocks infrared light. This means the camera is more adapted to recording in areas with less light while images look washed out when in average daylight settings.

Camera Comparison

Part Name	**Raspberry Pi Camera Module**	**1/4 Inch 5MP OV5647 Mini Camera**	**Pi NoIR**	**WolfWhoop C5 micro**
Connectivity	All Pi models	PI Zero	A & B model	All Pi models
Video Output	Digital	Digital	Digital	Analog
Price ($)	33	25	27	10
Dimensions (cm)	2.4 x 2.5 x 0.9	1.2 x 0.9 x 0.5	2 x 2.5 x 1	1.2 x 1.2
Max Transmission Resolution (px)	1080p @ 30fps 720p @ 60fps	1080p @ 30fps 960 @ 45fps 720 @ 60fps	1080p @ 30fps 720p @ 60fps	420p @ 30fps
Voltage (V)	5	3	3.3	5
Weight (g)	3	5	3	2

Table 4 – Camera comparison

Since the resolutions are the same amongst the cameras, the biggest comparisons need is between the dimensions, voltage, and build. For the standard Raspberry Pi camera, the camera is a small circuit board with a long connection band. Both the NoIR and Mini are on significantly smaller boards with connection bands as well. This was considered

when we designed where the camera would go and how far it needed be to the Raspberry Pi board.

One last notable difference between the cameras is their price and voltage requirement. The Raspberry Pi camera is about 50% more expensive than the other 2 cameras and requires more voltage supplied by ~ 2V. Although the price is not too big of an increase the voltage requirements play an important part in what camera we choose.

For the last camera on the list, the Wolfwoop micro, the main reason this may be an option is for its analog input. It is significantly cheaper than the other cameras, but the quality suffers because of that. Since we decided to go without the Raspberry Pi module, this camera is the one we choose. With the analog nature of the camera, it can be hooked directly to the transmitter and send video that way.

3.3.4 – GPS

For GPS units we are looking at the specific chips with the intention to add them to our PCB design. So, the modules we are wanting to compare are just the chips to see which chipset would be the best fit for our design.

Before comparison it's important to note that all 3 of these GPS modules run in the range of 1.5 GHz to 1.6 GHz, which is in the range of what the avionics team says they may use for the rocket. However, we still want the GPS module for post launch operations. Each of these modules also have an internal battery with about a 30 second hot start and 1 second cold start time. As seen above, each chip also has access to GPS and GNSS though the YIC chip does not state the current input for GPS.

GPS Comparison

Part Name	**TAU1105-1010A00**	**MAX-M8C-0**	**YIC51009EBGG**
Cost ($)	9.75	11	10.65
Accuracy (m)	2.5	2.5	3
Channels	40	72	99
Dimensions (cm)	1 x 0.9 x 0.2	1 x 1	1 x 1 x 0.3
Voltage (V)	3.3	3.3	3.3
Current (mA)	GPS – 25 GNSS - 35	GPS – 18 GNSS - 17	GNSS - 45

Table 5 – GPS comparison

When it comes to the differences between these modules it seems the biggest one will be the current drain on our rover's grid. The MAX chip has the lowest current drain out of the three chips which may be significant if we find we need to limit how much power consumption our rover is using.

Another key difference is the accuracy of the units. Normally having more channels, like the YIC chip has, results in a greater accuracy since the chips can access more satellites. However, that seems to not be the case here as both the MAX and TAU chips have better accuracy by 0.5 meters. The last notable difference would be the price of the units. The MAX chip is $1.25 more expensive than the TAU chip, however we choose the MAX chip for its low voltage and large number of resources online.

3.3.5 – Microcontroller

The microcontrollers we are interested in are small and low power, but still able to carry out all the tasks the rover needs to do. The MCU will be the component that controls all the other components on the rover and processes the data that the sensors gather. These controls will range from turning subsystems on and off, gathering data, and sending data throughout the rover. For this reason, it's important we find a MCU that has a fast enough processor speed to keep up with these tasks, but not be overkill and waste battery power.

There is an important key factor when picking MCUs to compare, they must have a development board with some way to program the chip. Since we plan to do a significant amount of breadboard testing when building the rover, this will be invaluable. We must be able to not only quickly wire the chip up to other sensors, but also be able to program and reprogram the chip easily. For this reason, we may pass up a better suited chip if there is no development board that is affordable or available.

MCU comparison

Part Name	**MSP430**	**STM32L0**	**ATmega2560**	**STM32F3xZ**
Price Board ($)	20	13	16	20
Price Chip ($)	10	5	14	14
Architecture	16-bit	32-bit	8-bit	32-bit
Clock Frequency (MHz)	16	32	16	72
GPIO count	83	51	86	115
Power Consumption (µA / MHz) (max/min)	100 – active 0.4 - standby	88 – active 0.27 – standby	500 – active 0.1 – power-down	1033 – active 1.03 – standby
Operating Voltage (V)	1.8 to 3.6	1.65 to 3.6	4.5 to 5.5	1.65 to 3.6
Memory	128 KB FRAM	8 KB RAM 64 KB flash 2 KB EEPROM	8 KB SRAM 256 KB Flash 4 KB ROM	64 KB RAM 64 KB flash 2 KB EEPROM
Temperature Range (°C)	-40 to 85		-40 to 85	-40 to 125

Table 6 – MCU comparison

The first MCU and most obvious for us to consider is the msp430fr6989. This board is used in a couple classes at UCF and therefor is a comfortable choice. This MCU has a

16-bit architecture with up to a 16MHz clock. The power consumption for this chip is around 100 µA while active and 0.4 µA on standby. This makes the msp430 a relatively small chip in terms of architecture that may struggle with a high load of tasks going on at once, however there is support for a real-time operating system on the board so that could make it a valid choice. One of the main reasons we would like to use this board is the familiarity. Two of our group members have already programmed on the board and are used to programming and debugged the chip which would cut out the stress of learning a new chip.

The next chip we would like to consider is the ATmega2560-16AU. The ATmega chips are popular for being the chip on the Arduino uno and nano models. The Atmega2560 has an 8-bit architecture with a max clock speed of 16MHz. The chip has an active mode consuming 500 µA at 1.8V while only 0.1 µA in power down mode. The 16AU variant was chosen instead of the normal because of the access to a reasonably priced development board. Since this board is used in Arduino boards it also has 100% Arduino support if we want to use some of their libraries for our sensors later.

Another chip we are interested in using is the STM32L053R8. This is an ARM based MCU which is interesting to us since we would like to work with an ARM based processor in our project for future job applications. The ARM processor is partially known for its use in the Raspberry Pi models as a very fast processor. In that vein, this MCU has a 32-bit architecture with a 32MHz clock frequency. In its active mode the chip uses 88 µA while it uses .27 µA in standby mode. We choose this chip out of the extensive STM32 collection mainly by development board availability and relative size compared to the other chips we are using. As a side note this chip also has compatibility with Arduino like the ATmega chip.

The last chip we compared is the STM32F3 chip. This MCU is mainly included because of the timers and how they're set up. This chip is basically a much bigger version of the STM32L0, so therefor it is a step up in specs in all regards to the L0. Its draw backs are the significant power consumption because it is so much bigger than the other chips we have looked at. However, this chip has 14 timers with multiple channels which might be a factor depending on how many timers we need for motors, sensors, and miscellaneous components.

With most of the relevant information about each chip listed above there is a wide amount of difference between the chips. One of the important differences is in the architecture of each chip since we may find that we may need a bigger word size for communication with some component in the build. For now, however it's not too important of a factor since we don't believe we have any components or software that force us into a specific architecture.

Another important difference lies in the power consumption. The MCU will most likely be one of the more power-hungry components in the rover design. To this point the ATmega does use significantly more power than the other two boards and is the only board without a standby stage available. When we start designing battery solutions for our rover this may be reason enough to exclude the ATmega from our design for one of the other chips.

The next significant difference is in memory. Memory will be in relatively high demand for our project as we expect to have feedback motors, multiple sensors, and some control code to guide our rover to its end goal. In this case the ATmega does pull out ahead of the other two boards substantially even though it has a much higher power cost. With its 256KB of flash storage, we would be very comfortable with any program we expect to use. Memory of the STM32L0 is not that far behind the ATmega at a fraction of the power usage.

The last notable difference is in the operating temperature. Although we don't expect to be operating the chip at 85 °C the STM32L0 board extends that parameter to 125 which is quite a significant jump. If we find we have problems with heat dissipation with our rover or even while the rover is in the rocket bay, having a bit more temperature tolerance may be valuable.

Unfortunately, the STM model chosen did not have enough timers for our use. Furthermore, smaller chips of a similar caliber that did meet the timer requirement were all sold out due to the demand of electronics on the market currently. This led us to finally choosing and designing around the STM32L4S9ZIT6 chip.

3.3.6 – Motors

There are multiple types of motors that we could have implement into the design for the rover depending on our needs. We can choose between servo motors, DC motors or stepper motors. They each have their own uses and can be implemented in different functions. Electric motors all function under the principle of a magnetic rotor core being turned by the pulsing of an electromagnetic stator.

Servo Motor Comparison

Part Name	**FS90-FB Micro Servo**	**SER0049 180° Clutch Servo**	**FS90R Micro Continuous Rotation Servo**	**SER0037 360-degree Micro Servo**
System	Camera rotation	Camera rotation	Driving motor	Driving motor
Rotation Angle	180°	180±10°	360°	360°
Dimensions (mm)	23.2 x 12.5 x 22.0	23.2 x 12.0 x 16.3	23.2 x 12.5 x 22.0	20.0 x 8.2 x 16.3
Stall Torque (kg-cm)	1.5	1.8±0.2	1.5	0.75±0.2
Voltage (V)	4.8 to 6	4.8 to 6	4.8 to 6	4.8 to 6
Price ($)	7.95	5.00	4.95	7.90
Weight (g)	13	9	9	2.5

Table 7 – Servo motor comparison[5]

[5] Selected servo motors can be in the canister deployment system; not directly covered in this document.

Servo motors usually have restricted rotation, but they do have fine precision control and high torque up to their fixed angle. We could use a servo motor to control the rotation of the camera for the live video transmission. There are some continuous rotation servo motors which can give us the high precision and torque of a servo motor. However, they are also quite slow when comparing RPM to other types of motors.

Given our weight limitation for the rover, continuous rotation servo motors can be used. The FS90 series of servo motors seem to be more versatile and keeping the dimensions consistent means less tinkering with getting a proper housing for different motors. Depending on the weight of the rover, Servo motors might be used as they do provide a good amount of control and precision in a small package size and for a cheap price. The SER0037 Micro Servo is the lightest option but has the lowest torque rating. This could be a possibility as most of the rover's weight should be from the motors and the battery.

DC motors can be either brushed or brushless. DC motors are simple motors, they turn depending on voltage. Higher voltage means faster rotation and negative voltage means reverse rotation. The brushed variation of DC motors has higher torque at lower speeds, but they are more susceptible to heating problems from the friction of the brushes in the rotor and stator coupling. Heat becomes a greater problem which means that proper insulation becomes more important so that the motor can still function and move the rover. The brushes also become a point of failure as they need to be maintained consistently. For a single use application as this rover project, it's not a major issue as it should be maintained and in proper order before it is deployed.

Brushless motors are functionally like brushed DC motors but there is no point of contact between the rotor and stator. This reduces noise, points of failure for the motor and frequency of maintenance. It also increases the efficiency of the motor and the maximum torque of the motor. They are typically more expensive than the brushed DC motors.

Micro Metal Gear Motor Comparison

Part Name	**Micro Metal Gearmotor LP (Low Power)**	**Micro Metal Gearmotor MP (Medium Power)**	**Micro Metal Gearmotor HP (High Power)**	**Greartisan N20 High Torque Speed Reduction Motor**
Gear Ratio	298:1	250:1	150:1	150:1
Speed (RPM)	45	88	210	100
Dimensions (mm)	10 x 12 x 26	10 x 12 x 26	10 x 12 x 26	15.2 x 12 x 10
Max Efficiency (%)	27	31	37	-
Voltage (V)	4.8 to 6	4.8 to 6	4.8 to 6	4.8 to 6
Torque@ME (kg-cm)	0.44	0.45	0.39	0.4
Current@ME (mA)	90	170	310	160
Power@ME (mW)	150	310	680	-
Price ($)	16.95	16.95	16.95	9.95
Weight (g)	9.5	9.5	9.5	9

Table 8 – Micro metal gear motor comparison

Depending on the given gear ratio of the DC motor, we can prioritize torque over speed. The rover should be relatively slow moving as the sensors and MCU need time to process its surroundings before moving. We can achieve this by moving slowly or by stopping to scan before moving. Our choice of DC motors will greatly influence the overall specs of the rover.

Using low power motors means we need less battery life and can use a smaller battery saving on weight and work required from the motors. There is still a base amount of battery life required to make the other systems function, but most of the current will be going to the motor and driver systems. This means we can use a medium or high-power motor to make use of the available power.

All the Micro Metal Gearmotor (MMG) DC Motors have a rated torque of around 0.4 kg-cm. One of the constraints of our rover is that it must be under 1kg total with the deployment capsule. The stalling torque for each motor is around 4 times the load torque at max efficiency. With 4 motors, one driving each wheel, we can ensure that even if we only have 2 wheels on the ground, the rover should be able to move itself until it can stabilize itself and utilize the power from the other motors.

We also have the choice of getting a very low power motor with high RPM if the rover turns out being very light weight. Depending on the overall size and weight of the frame and battery, we might be able to use the N20 motor. The motors very low power requirement could mean a smaller battery choice which would weigh less.

The MMG DC Motors are all the same size, weight and connectors and have further variations to them in case we need a choice with more torque or less power consumption, the MMG DC motors were chosen to compare because they have similar torque ratings but have different speeds, efficiency, and power consumption. They are all interchangeable as they all use the same housing and have the same shaft diameter for the wheels. If we choose to use them then we could easily swap motors during testing or prototyping as we could run into burnout or motor damage during stress testing and trial runs. All the motors are also the same cost regardless of power consumption or gear ratio so we can test different configurations to find the most power efficient setup.

The MMG DC Motors have a second motor shaft on the opposite end of the main shaft where we can attach a magnetic encoder for more precise control. The magnetic encoder pair kit adds 1.5g of weight to the motors and extends the depth by 5mm. Using 2 encoders, we can synchronize the RPM for all the motors based on their voltage and current under load. We can then use 1 encoder on each side (left and right) to log the speed of the motors and show if we have any stalling (no rotation) or free turning (no load).

Stepper motors are the final type of motor that we could implement in our rover design. They characteristically use many electromagnetic poles where the rotor could rest between. This means that stepper motors have a very high level of control and torque but at the cost of size and power efficiency. Stepper motors draw power regardless of load unlike the other motor types so we can more easily predict the estimated battery life is a limited resource for the rover and the stepper motors and so inefficient with power, they are not the best choice for the main drive motors. We don't need the high precision degree

control that we could get from stepper motors so the constant power draw would be wasteful when we need to power the video transmission and control systems as well.

3.3.7 – Motor Drivers

To properly implement the motors, we need a motor driver that will deliver the correct voltage as well as enough current to properly utilize the motor. These motors need a PWM signal, and the drivers are chips that can deliver them from the control of a microprocessor.

Since both chips are dual channel, we can run two motors simultaneously from one driver system. The L293D has the advantage of accommodating a higher voltage power supply which is more common with higher capacity battery cells. The voltage for all the motors ranges from 4.8 to 6V, so either choice of chip should be sufficient for any motor.

Motor Driver Comparison

Part Name	**TI L293D IC**	**TI DRV8833**
Device Supply Voltage (Vcc) (V)	4.5 – 36	0 to 7
Input Voltage (V)	0 to 7	-0.5 to (Vcc + 0.5)
Output (Motor) Voltage (V)	-3 to (Vcc +3)	0 to 12
Channels (Outputs)	2	2
Peak Output Current (A)	±2	-
Continuous Output Current (A)	±0.6	±1.5

Table 9 – Motor driver comparison

The DRV8833 has better headroom with the continuous output current of ±1.5A per channel. None of the motors that we have chosen should come close to this limit, but it does give us the choice of upgrading our motors if weight and size allow. The peak current of the DRV8833 is the continuous output current as is the output is limited internally within the chip.

We have designed the PCB for the chosen chip and produced 2 two of them, one for each side (left and right). Since we have designed the drivers, they were made to interface easily specifically with whichever MCU we ended up utilizing. Implementation of either chip should be simple as they rely on the MCU for the PWM signal to drive the motors. The chips function on the principle of the H-bridge which is a very basic design. This motor driver design is one of the two that we produced for the rover and its specification and schematics will be detailed in a later section

3.3.8 – Battery

The single most important component of the rover is the power supply or battery as it determines our power delivery allowance for the entire rover and the functioning up-time. To implement the right kind of battery, there are 4 major factors we need to consider: voltage, discharge rate, capacity, and temperature limit. All the motor drivers can reach at least 12V, so we have a maximum voltage value. The discharge rate should be large enough it can provide enough current to any part of the entire rover that draws it. The battery should have enough storage capacity that it can run the rover for over an hour at full load. Finally, the temperature limit is important as any battery has a temperature where it will fail. If the battery starts to critically overheat, there's a very good chance that we lose the rover entirely.

The most common technologies used for rovers, drones and other RC vehicles are lithium-ion (Li-Ion) and lithium-polymers (LiPo). Li-Ion batteries are the older technology, so they are going to be the most affordable and well-developed option. LiPo batteries are use newer technologies and can vary from the battery to battery. Some LiPo batteries can provide the same power but with a higher capacity and a smaller footprint, but they use more expensive chemicals.

Main Battery Comparison

Part Name	**Tenergy Li-ion Rechargeable Battery Pack**	**EcoPower "Trail" Hard Case LiPo Battery**	**ProTek RC Low IR Si-Graphene LiPo Battery**
Battery Type	Li-ion (2S1P)	Li-Po	Li-Po (Si-Graphene)
Nominal Voltage (V)	7.4	7.4	7.6
Capacity (mAh)	2600	5000	4600
Max Continuous Discharge Rate (C [6])	1.92 (5A)	45	120
Operational Temp. (°C)	-10 – 60	0 – 50	0 – 50
Dimensions (mm)	38 x 72 x 19.5	46 x 138.5 x 25	47 x 95.5 x 19.5
Weight (g)	99	306	165
Price ($)	29.49	37.99	59.99

Table 10 – Main battery comparison

The Tenergy Li-Ion battery is a battery pack with 2 18650 batteries in series and a PCB for recharging using the two output cables. It's the most lightweight of our options and has the lowest discharge rate at 5A. That is enough to supply enough current to all the different systems in the rover but the main issue with it is its capacity. Without knowing

[6] C is a unit that refers to the capacity of the battery and is used as a ratio for active current charging or discharging. (e.g., Capacity = 3000 mAh, Discharge rate = 4 C = 12 A)

how much current the rover will be drawing over its entire deployment, the best bet was to have more capacity. This is especially true as the batteries tend to lose their peak voltage as they are used.

The EcoPower "Trail" LiPo battery is an affordable LiPo battery that has almost double the capacity of the Li-Ion battery. The major advantage of this battery is the very high maximum continuous discharge rate and capacity. We get almost twice the energy capacity with this LiPo battery at 5000 milliamp-hours. The major drawback is that is that it is very heavy for the rover at 306g, more than triple the weight of the Li-Ion battery. The size might also be a problem as the rover is very limited in size. It is considerably bigger than either of the other choices and could cause some conflicts. It also has a lower range for its temperature limit, but it shouldn't be a problem if the battery is properly covered when deployed in the desert.

The ProTek RC Low IR Si-Graphene LiPo battery uses a newer chemical technology which has a higher energy density. This means that we have an equivalent capacity to the previous battery but with a much smaller footprint. It is almost two-thirds the length of the EcoPower LiPo battery and weighs almost half of it. At 165g, it is a much more manageable weight and size for the rover over with the benefit of a high capacity. The issue with this battery is its price, as it costs twice as much as the Li-Ion battery at $60.00. The investment in the battery makes sense as it necessary to make everything else function. A poor power supply solution could jeopardize the functionality of the entire rover.

3.3.9 – Voltage Regulators

Regardless of our battery choice, we needed a chip that regulates our DC voltage output from the battery so that it can be used by the MCU, Raspberry Pi, Camera, and video transmitter. Each of these systems use a different input voltage but most electronic systems run on a standard of either 3.3V or 5V DC.

There are hundreds of choices for voltage regulators but all we need is Low Dropout Voltage regulator that can reduce our battery's nominal 7.4-7.6V to a useable DC voltage of 3.3V or 5V. We need a voltage regulator that can provide enough current for all different systems.

We needed a voltage regulator of 3.3V and the best choice for that would be the TLV2217 chip. There aren't many systems that use 3.3V and the ones that do, don't draw a lot of current; making the max of 0.5 amps output current acceptable. The TLV2217 is within the range of any of the battery's maximum and nominal voltages so even at full charge, we should get the proper output voltage with enough current. It is the cheapest chip to implement as well.

The 5V regulator part is under more stress as it is the higher power output regulator controlling all the motors and motor drivers. Almost every system uses 5 volts, so the max current output is the more important metric here. Both the LP38690 and the LM1085 have a higher rated max output current. The LM1085's higher current rating is appealing as the higher current rating gives us more headroom to make sure that the regulator isn't operating

near its maximum rating and heating up. The drawback to his chip is its fixed output settings which would deter us from being able to use it to regulate the motor drivers as well for a more consistent voltage input as the battery drains.

Voltage Regulator Comparison

Part Name	LP38690-ADJ	TLV2217	LM1085IS
Max Output Current (A)	1	0.5	3
Input Voltage (V)	2.7 to 10	3 to 12	2.6 to 27.5
Adjustable Output?	Yes	No	Yes
Output Voltage (V)	1.25 – 9	1.8, 2.5, 3.3	3.3, 5, 12, ADJ
Operating Temp. (°C)	-40 – 125	0 – 125	-40 – 125
Price ($)	1.89	1.42	3.01

Table 11 – Voltage regulator comparison

The LP38690 is a choice because it could be used for the motor drivers as well because we can adjust the regulated voltage. We would need one regulator for each dual channel motor driver as the they would also be limiting the current draw available for each of the drivers. The LM1085 is the best choice to ensure enough headroom for the current draw but it is the most expensive of the options.

The PCB for the voltage regulator system will include the motor driver chips as well. We have designed so we can have a compact design that will reduce the need for soldering and connections between the two systems as well as the number of PCBs which could help us reduce the size of the rover. The design and schematics of the voltage regulators and motor drivers will be detailed in a later section

3.3.10 – Boost Converters

As part of the voltage correction system, we assumed we needed regulated voltage source of 12V. The battery only supplies 7.6V nominal so we wanted a boost converter to step up that voltage for the video transmission system that utilizes it. The boost converter should be able to keep a constant 12V without much ripple and provide enough current as well for the dependent systems. The boost converter uses a switching mechanism to increase the output voltage while reducing the output current.

Our choices for boost converters can all take the battery's voltage in and provide the 12V that we need. The main differences between the three choices are the output current limit. The output current from the converter will always be smaller than the input current or the switching current so we need to make sure that boost converter can provide enough output current for its systems.

The LM2623 has the lowest maximum current output with a limit of 2A to the load. This choice would be easiest to implement in to the 12V rail design and gives a high duty

cycle output which increases its efficiency on the output. The LM5158 is like the LM2623 but it is qualified for automotive use which is why its voltage range is much larger. We did not come remotely close to utilizing it near its maximum output and that may have become a problem in terms of efficiency. The TPS61089 has a much larger switch current limit at 11.5A. The best choice overall would be the LM2587 as it has enough range in both input and output currents.

Boost Converter Comparison

Part Name	LM2623-Q1	LM5158-Q1	LM2587	TPS61089
Input Voltage (V)	1 to 14	1.5 to 60	4 to 40	2.7 to 12
Output Voltage (V)	1.24 to 14	2 to 83	1.23 to 60	4.5 to 12.6
Switch Current limit (A)	2.85	3.75	6.5	11.4
Max Output Current (A)	2	3	5	7
Duty Cycle (%)	90	93	98	90
Switching Freq. (kHz)	300 - 2000	100 - 2200	85-115	200 - 2200

Table 12 – Boost converter comparison

3.4 – Possible Architectures and Related Diagram

Some of the main structures, tools, frameworks, and architectures which will help us in completing our objectives for a successful project.

3.4.1 – Real Time Operating Systems (FreeRTOS)

Real time operating systems (RTOS) are a middleware, task scheduling kernel, that sits between the hardware and the application software. A RTOS is meant to aid the ability for the rovers' actions to have logical correctness under strict deadlines. A RTOS is not meant to speed up the execution of a task, but a RTOS should enforce predictable outcomes for a schedule involving a synchronous or asynchronous job set with a given priorities for tasks. RTOS can some of the tasks that our rover has are sensing and motor control to four drive motors.

It is highly possible that scheduling these tasks will require deadlines during certain states that the rover will encounter such as precision turning to avoid obstacles. Therefore, it was a possibility that we would be incorporating FreeRTOS into our project. FreeRTOS is a simple kernel which is composed of C coded source files. These source files contain functions for implementing a scheduling system for jobs based on threads, semaphores, mutexes, and software timers. Figure 13 shows the task state diagram and cycle of synchronous periodic tasks. The incoming arrow into the ready state can either be from a periodic task in queue or interrupt driven task. This scheduling system should prevent unwanted action during the rover's critical traversal moments such as avoiding obstacles.

Since an obstacle can be met at any time, this form of task (interrupt based) is an asynchronous task.

FreeRTOS keeps track of time based on the notion of 'ticks' where 1 tick is 10 milliseconds. If the team decides to implement a storage system for environmental data on a memory card, the gathering of data can have strict sampling deadlines. This would make the sampled data accurate to within 10 milliseconds in the period of the sampling; where the sampling time can be any discrete interval such as 0.1, 0.5, 1, 5… seconds. This from of sampling would be synchronous based and has a defined period.

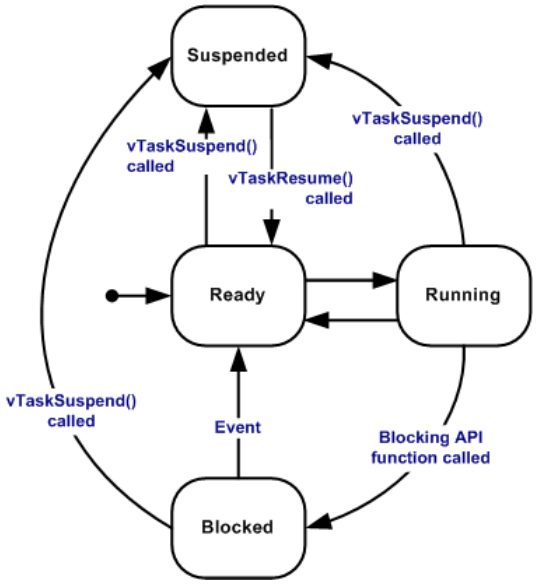


Figure 13 – FreeRTOS task cycle [F13].

It must be noted that FreeRTOS handles both asynchronous events such as interrupts and synchronous events which are predetermined, priority based, periodic tasks. FreeRTOS does not guarantee the completion of a task by a deadline; it is up to the team to make sure that the rover's processor is not overutilized and does not encounter priority inversion during these critical state tasks.

3.4.2 – Robot Operating System (ROS)

ROS is not an operating system in the traditional sense of the definition. ROS, despite the name, is a framework of libraries which hold the common I/O service capabilities that most robotic systems encompass. ROS is another middleware layer as it links hardware, to code with specific libraries. These libraries give the rover the ability to understand what its components are, which data they hold, and a way to communicate efficiently with other components. This in a sense, gives the robot an internal "sense of being" (Figure 14).

The framework allows messages and services to be created in the form of nodes (Figure 15), which may travel around the data system, informing other hardware components of relevant data and update actions. There are many user-created, open-source packages that are available for use with ROS which act as tools in the development of robotics. These packages help the designer with solutions to robotic navigation, perception, frame representation, and an environment to simulate the poses and actions of the robot.

Figure 14 – ROS features integrated into one easy to use platform
Reprinted with permissions from [F14].

Traditionally, embedded programming has been programmed through GPIO ports, with low level instructions. ROS being an open-source framework, has been updated by many contributors, libraries and functions which make programming robotics easier by taking away most of the low-level software implemented communications. With this said, the team may end up using ROS if a microprocessor is used for advanced pathfinding capabilities. ROS has a version that can be implemented on a microcontroller which is a much smaller framework. If the team does not choose to use a microprocessor, this smaller framework may still be added as a middleware component to simplify some tedious low-level tasks. It must be note that

Robot Operating System is in no way a Real Time Operating System. ROS has no way of providing scheduling capabilities dedicated to hard Realtime systems in its LTS version. ROS can run alongside a RTOS, in this case, FreeRTOS. ROS would provide the cognition for our rover and FreeRTOS would enforce periodic task scheduling and deadline of services and messages.

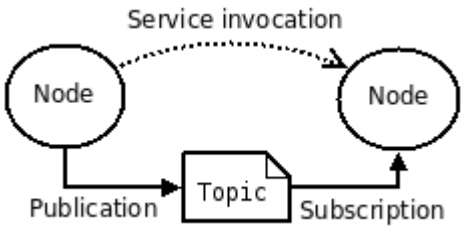


Figure 15 – ROS information exchange [F15]

3.4.3 – Reduced Instruction Set Computer Architecture (RISC)

A RISC architecture (Figure 16) usually involves a load store system which do direct operations on registers in the processor cache. This is opposite of a Complex Instruction Set Computing architecture which usually performs these operations directly in main memory (RAM) without the need of caching the data first. The RISC architecture involves simple instructions which help speed up the execution of instructions via pipelining (with diminishing returns in speedup as the pipeline becomes full).

It is evident that the rover will have to deal with an abundance of operations that involve direct manipulation of integer and floating-point data. Due to the nature of the frequency the rover uses this data, the team has chosen the RISC based ARM Processor to handle most of the raw data gathering and manipulation.

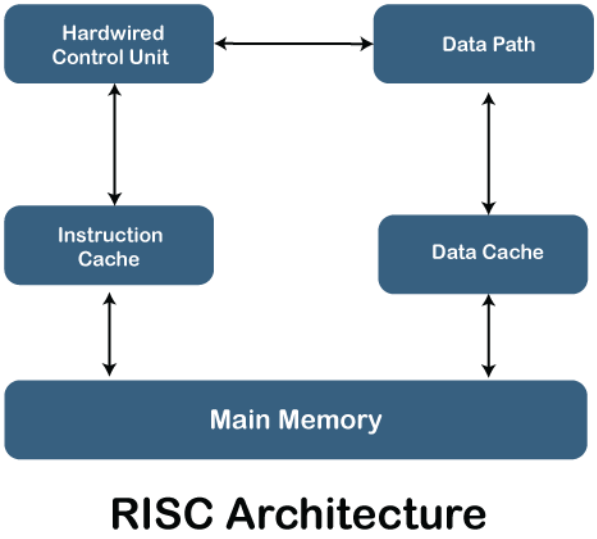


Figure 16 – RISC Architecture high level overview. [F16]

3.4.4 – Live Video

For live video we want something small, light, and easy to deploy. When thinking of architectures for this subsystem we must not only think of what is small enough to fit in our rover constraints, but also what is powerful enough to transmit our live video at distances possibly up to 1.6 km (although the range requirement listed is 400 m). Therefore, there are two initial architectures we considered:

Architecture 1:

The first architecture we thought of is shown in Figure 17. The key point of this architecture is the use of a Raspberry Pi to convert the digital input video to a DVB appropriate stream of data that outputs at 1.2 GHz frequency. Then the DVB modulator sends the RF wave data to an amplifier to be modified for whatever distance we want to cover with the RF communication.

The main benefit of this architecture with the Pi is that the Pi is very power efficient and easily programable with OOP using the Raspbian OS. The negative of this architecture is that its relatively expensive to find a DVB modulator and receiver at the size requirements we need.

ARCHICTECTURE 1

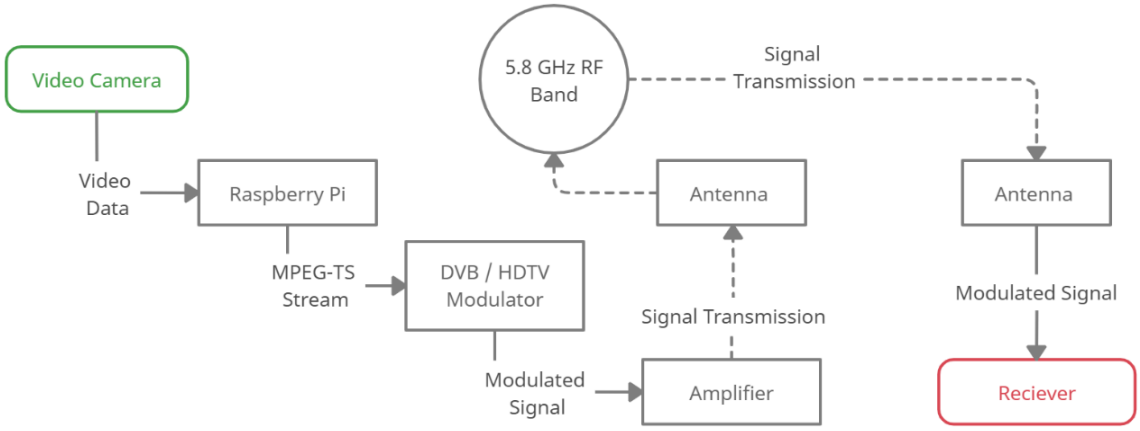


Figure 17 – Live Video Architecture 1 Flow [F17]

Architecture 2:

The second architecture which we are considering is shown in Figure 18. The important difference of this architecture is that we discard the Raspberry Pi and modulator for a single first-person video (FPV) transmitter that can directly take analog video as an input and outputs at 5.6 GHz frequency. This means we need to find an analog video camera instead of a digital camera; however, the transmitter converts the video data to the correct RF without any need to program the chip.

ARCHICTECTURE 2

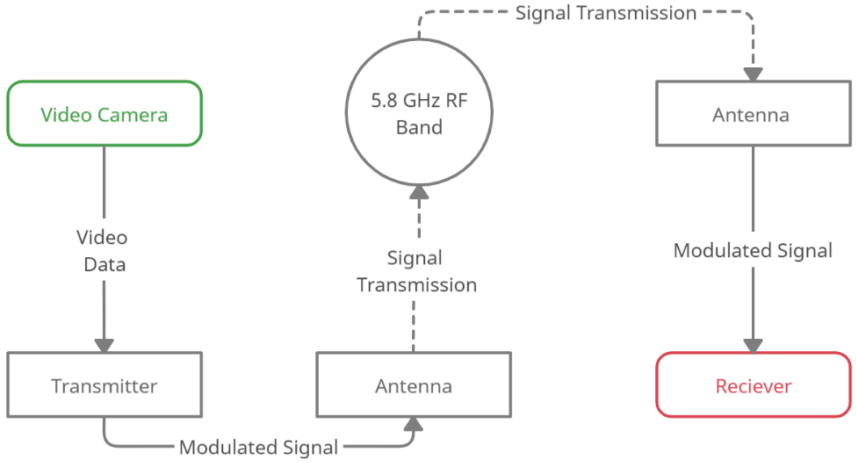


Figure 18 – Live Video Architecture 2 Flow [F18]

This type of system can be bought straight off the shelf as a complete unit, however if we source the separate parts, we are not only able choose the specific parts we want to hit our requirements, but also, we have complete freedom to place each component where we want in the rover and connect them to our PCB to control them.

In both architectures receiver in this instance will probably be attached to a personally computer instead of a standalone display. In which case the group will have to create or download a program that displays the live video to the team.

3.4.5 – Object Detection

For object detection there are many architectures available for small rovers like our project. The key deciding factors in which to pick are which sensors you want to use and how accurate and advanced you want the system to be. For our project we determined that advanced AI using camera vision was overkill, since our rover only needs to avoid rocks and varying amounts of foliage. This left us with two reasonable less advanced ideas which is using pathfinding algorithms or using simple distance sensors.

The first possible architecture we are considering is the use of pathfinding algorithms to detect objects and direct our rover in the best possible path back to the launch pad. This would require some sort of far distance sensors combined with short distance sensors as well as a significantly fast processor able to run the pathfinder code. There are a couple variants of sensors we could use for this architecture, but the most likely would be some type of lidar with a 180° capture angle. As for the pathfinder algorithm, there is also many different types we could implement. If we picked the lidar sensor, then using the Vector Field Histogram algorithm would be our go to algorithm since it is one of the most reliable 2d object detection methods.

The pros of the pathfinder architecture are that it's very reliable and there's a lot of online resources for coding help. The con of this architecture is mainly its complexity, but its greater expense. For this architecture to work not only would we need a lidar sensor that could keep up with the algorithm's input requirements, but we would also need a more powerful processor than anything a lower end MCU would have. This would mean we either must incorporate an energy and budget expensive processor to our PCB or delegate the task to some separate subsystem like a Raspberry Pi. Using a Raspberry Pi in this case would make the MCU irrelevant for our design and in turn ruins the PCB design requirement for our project.

The second possible architecture being considered is a pure IR sensor and ultrasonic sensor build. This method would use at least four IR sensors for the 4 sides of the rover and detect close range objects while there may be one ultrasonic sensor for forward-facing long-distance detection. The brains of the system would be on our MCU and work much like a Roomba vacuum cleaner. If we are about to hit an object, the MCU will rotate the rover until the sensors don't sense an object in front of us. We considered adding more control functions to make sure the rover was going the right way and to work against getting the rover stuck in between objects, but this is the general idea for this type of architecture.

The pro of this architecture is its simplicity. The whole system will run on a single MCU which would be able to handle the load of multiple simple sensor inputs and calculate what to do next. The cost of this architecture is also significantly less since we already need a MCU for our design requirement so the only added cost is the small IR sensors and possibly an ultrasonic sensor. Design wise this will also give us good experience hooking up multiple sensors to an MCU and integrating them into a PCB design.

3.5 – Parts Selection Summary

Close Distance Sensors

For object detection sensors we settled on the laser sensor VL53L0X and use multiple of them for different directions. The main reason we chose this over the ultrasonic sensors is size and accuracy. The laser sensors have every small beam and therefore, a very small transmitter and receiver. Close range laser sensors didn't seem to have many variables that would affect them unlike the ultrasonic sensors which had to consider the possible noise interference of the environment. The laser sensor is also cheaper than the single unit transmitter/receiver ultrasonic sensors while more expensive but smaller than the double unit ultrasonic sensors.

Far Distance Sensors

We ended up not using any far distance sensors at all since we have decided to drop any type of lidar or pathfinding algorithms. In the case we needed a long-distance sensor we would have used the ultrasonic sensor CUSP-TRS0-18-2400-TH. Although we didn't end up picking this sensor for close range, none of the cons of the ultrasonic sensor would be too big of a deal for long range distances. We would still have to be wary of noise interference, but we would only use one of these sensors for straight ahead of the rover which would nullify the size problem of having multiple of these sensors on the rover.

Camera

The camera we decided on is the Wolfwhoop C5 micro. The main reason we choose this camera was for the analog direct video output. This output is fed directly into the transmitter we chose, meaning the only thing we had to connect to the PCB / MCU was the power lines which act as control lines (on / off). Another reason we chose to buy a separate camera and transmitter instead of just buying one off the shelf, is the flexibility that separate parts give us. We planned to put the camera towards the front of the rover and hardwire it to either the transmitter or the PCB while the transmitter is near the PCB towards the top to the rover (for antenna extension).

GPS

For GPS chips we choose the MAX-M8C-0 mainly because of the lower power drain compared to the other chips in that price range. This chip also looks to have decent accuracy given its price and number of channels it has access too. If needed, we can also add an external antenna which will help improve this signal. The integration of an external antennal on the GPS chip is rather easy and has direct reference in the datasheet.

Microcontroller

The MCU we decided to go with is the STM32F3xZ. The defining factor behind this choice is the timers and how they were set up on the other chips. When trying to connect all our motors and sensors we found the timer pins on the other chips overlapped with other timer pins (PWM) that we needed to be separate. The F3xZ chip was the smallest chip we could find in stock that also had timer pins that satisfied our design needs.

Although we did choose STM32F3xZ chip, we were not able to obtain the chip given certain market problems. This led us to use the STM32L4S9ZIT6 instead, however these chips shouldn't be too different when it comes to our design requirements.

Motors

We choose to use the Micro Metal Gearmotor High Power motor as it gives us the best ratio of torque to RPM from any of the other choices. The price for all the MMG motors is the same, so being able to choose the one with the highest power rating would give us the best chance of getting the rover to move under its own weight. Our choice of motor directly impacts our choice of motor drivers and voltage regulators.

Motor Driver

The motor driver we chose is directly influenced by our choice of motor. We know that our chosen motor's stall current is 1.5A so we chose the DRV8833 that can provide that current per output channel. The other reason we chose this driver is that it has a current sensing feature that lets us limit the amount of current drawn by the motors per channel. This gives us more leniency when choosing our voltage regulators.

Voltage Regulators (Step-Down)

The voltage regulators we chose to use were the LM1085 regulators as it has variants for both 3.3V and 5V outputs and has a current limit of 3A. If we choose not to limit the current of the motors, it can still provide enough current if we connect one regulator per motor driver chip. They are also simple to implement as the fixed input variants only need an input and capacitor at the output to reduce the ripple voltage effect.

Battery

The battery we chose is the Protek Si-Graphene LiPo battery as it gives us the best capacity, weight, size and output current. Since the battery uses silicon-graphene, which has a higher energy density than the other LiPo technologies, it provides a higher energy capacity at a lower weight than the other choices. Due to the battery technology being more energy dense, the battery is also smaller which lets us fit it within the frame for the rover.

4 Related Standards & Realistic Design Constraints

This section is dedicated towards finding and applying standards to our design as well as noting specific design constraints that play a role in our planning and execution of the project.

4.1 – Standards and Their Design Impact

Standards can be everything from something we used to guide us in designing some subsystem to a piece of equipment that we buy that already uses the standard so we must adapt it.

4.1.1 – C Programming (ISO/IEC 9899)

The current C programming standard is ISO/IEC 9899. This standard specifies the form and establishes the interpretation of programs written in C. This standard first mentions the syntax and constraints of the language which shows how to write C code in a uniformed way to keep code consistent and easy to read. The next topic is semantic rules for C programs which focuses on things like when to capitalize variables and function names. This too will make the code more uniform and readable if each group member keeps to the standard. Thirdly, is the topic of input and output data and how they should be processed. This will impact us mainly during the construction phase of our project while we are coding and debugging our rover. The last topic covered in the standard is the restrictions and limits imposed by conforming to this implementation of the C standard. This last topic just underlines the reasons to conform to the standard which is mainly for the readability and uniformity of our C programs.

We used C and / or C++ programming to program our MCU for things such as motor control, sensor control, and sending signals to subsystems. The impact of implementing the C standard into our project for any C programs we write will be in the readability of the code, not just for our group but for anyone who may try to read and understand the code later.

4.1.2 – Measuring Antenna Distances (IEEE 149)

There is a standard for best practices for measuring different types of information using an antenna, IEEE 149. This standard first lays out the standard spherical coordinate system used for antenna measurements. Using this coordinate system, the authors mention some ways to position the antenna for the best results. The most relevant for this project is the fixed-line-of-sight which is used for all GPS receivers. This type of range configuration is used to have the test antenna operating in the receive mode so it can then get the signal from some located fixed source antenna.

The standard then goes into the many ways to calculate where you should place your antenna for any given desired results. It mentions how you should expect some error from the surroundings depending on what RF waves are being used. It also mentions the

error you should expect from the antenna from ground-reflection and how to calculate the distance you should move your antenna up to nullify this error.

Another helpful topic the standard covers is how to make sure the rate of incoming data isn't limited by things we can change. One of the components mentioned is the bandwidth of the output circuit of the antenna. Depending on how we decide to design the output of data we may see a need to increase the bandwidth which will in turn reduce the sensitivity and the dynamic range of the system.

We used an antenna to increase the range of the GPS module on our rover. Using this standard to decide where to place our antenna to be the most useful will help greatly in the design of the rover. This standard will also help us be prepared for some of the errors and inaccuracies of antenna communication and give us guidance in how to fix some of these errors.

4.1.3 – Software and Systems Engineering – Software Testing (IEEE 29119)

Since our project will be heavily based around software programs interacting with pivotal hardware components, we used the ISO/IEC/IEEE 29119 standard for software testing. This standard lays out the function of software testing was to "analyze a software item to detect the differences between existing and required conditions" [4.1.3.A].

The first part of the standard mainly covers key definitions and concepts of software testing. This is important to the understanding of software testing, so you know what it means for a piece of software to be tested and not perform as intended. This section also points out that having a perfect piece of software is not a reasonable, let alone possible, expectation.

The second part of the standard goes more in depth on how to implement system testing. The general idea is you should create a test policy to use in making a test strategy. From this you apply it to your project where you test each unit or system in the project and collect results to see if the unit is working as intended or not. The intended way to test these units is by setup, monitoring, control, and then reporting on each unit. If variance from acceptable values is found, then some plan to correct the problem should be executed.

The third part highlights the importance of documentation. For this standard it labels documentation of testing as an output of said testing and should be used to thoroughly explain the results of the test. There are many types of test documents you can use, the ones this project will most likely use are the test status report and test completion report.

The fourth part talks about testing techniques. Some of mentioned tests that may be useful for us are specification base testing, structure-based testing, and experience-based testing. For specification testing, we looked at our project requirements and the requirements of the piece of software and checked if they matched intended results. Structure testing will be used to check the structure of the code to make sure it follows any standards or uniformity rules we require as a group. Experience testing will be used at each group

members discretion to test some factor they believe may be important to the success of the software.

We used this software testing standard to help us in testing our code during development and post development. We used code for object detection, motor control, live video transmission, and any system management that our MCU may do. This standard gives us a method to step by step test and revise our code, so it functions the way we desire it to.

4.1.4 – NTSC / PAL Video Transmission

For our live video transmission, we are using a camera and video transmitter that follow the NTSC / PAL standards. Both standards are ways to transmit analog color television. NTSC was developed in North America and is used to broadcast 525 lines of resolution at 30 fps while PAL was developed in Europe and broadcasts 625 lines of resolution at 25 fps.

The main impact of these standards on our project will be choosing the best one for our rover to use for live video transmission. We wanted to test each transmitting standard to see which provides the best resolution and is also the most reliable. Using these standards are as simple as switching the mode on the transmitter, since we have a camera that is directly feeding analog data into it.

4.1.5 – Generic Standard on PCB Design (IPC-2221)

The IPC-2221 standard lists many of the pivotal topics of PCB design. This standard talks about topics such as materials to use, electrical properties, component and assembly tactics, thermal management, and holes and interconnections. Overall, these topics are important to guarantee quality assurance and to make sure the PCB meets the feature requirements required.

One relevant topic covered in the standard is on clearance in board design. Clearance is defined to be the distance between two conductors or nodes measured by the air in between them. Some constraints which may be important for our project is component leads having 0.13 mm clearance and test probe sites having 0.6 mm minimum and 5 mm maximum. It's also stated mounting hardware should not protrude more than 6.4 mm below PCB surface. Like clearance there is also creepage which is the distance between two conductors or nodes based on surface measurements. According to the standard the space between two conductors should always be maximized for the best-case scenario.

Another important subject mentioned in the standard is on how thick traces should be for carrying specific amounts of current. The standard lays out relevant formulas to calculate how thick copper traces should be to account for distance traveled and current needed. It also describes how these thicknesses will have to be different for different parts of the board depending on where components are and what different traces can be placed.

The next topic covered that is applicable to our project is thermal management. This is first addressed with cooling mechanisms that could be used to lower the temperature of

the PCB. This is also where heatsinks are mentioned and the standard talks about where the best places to put them are. Most of the time heat sinks are mounted on the bottom copper pad of the PCB. This gives them sufficient coverage to allow for optimum heat transfer. The last topic of the thermal management is how thermal design effects the reliability of the PCB.

4.1.6 – IEEE Standard for Floating-Point Arithmetic (IEEE 754)

The floating-point standard (IEEE 754 Est 1985) overviews how computers format the real number system for arithmetic operations. Real decimal numbers and very large real numbers require a representation for which programmable code may be distributed without the worry that the code may fail due to a misrepresentation of these real numbers. Figure 19 depicts the numbering format for floating point numbers.

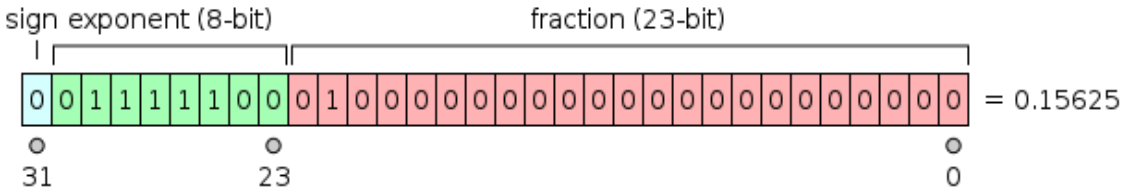


Figure 19 – IEEE 754 floating point standard layout. 32 bits wide.
Reprinted with permission from [F19]

This layout and representation of bits allows for the floating-point number system to represent numbers as large as 3.4028235×10^{38}. IEEE 754 also covers the rounding rules, operations, and exception handling techniques that allow for a more precise floating-point representation.

While we are not using the floating-point standard directly, we are using operations that involve the use of floating-point numbers relevant to analog sensing and GPS coordinates. At which point the IEEE 754- [1985-2019] section 9 may be referenced (Recommended [floating point] operations).

4.1.7 – American Standard Code for Information Interchange (ASCII)

American Standard Code for Information Interchange (ASCII) is a standard used for data transmission that can represent textual data and noninput-device commands. Using ASCII is important when addressing the problem of having two different computers communicate with one another. If one computer represents the letter "A" differently than the other computer, then there is no way for the two computers to be able to receive or compare data with each other. To solve this problem, we use the standard ASCII for both computers to appropriately represent letters, numbers, and punctuation marks through our software and stored within our hardware.

ASCII uses seven binary digits sequencing of 0s and 1s to classify up to 128 different characters. This means, for example, that 1010000 which stands for the upper-case letter "P" would be recognized in both systems that we intend to use the data in. The ASCII code is typically embedded in an eight-bit field for digital computers. This eighth bit was added to conform to the 8 bits in a byte design theory. This makes it much easier to use ASCII for 32-bit operations which is a very common size for personal computers today.

The ASCII standard will be used in our project by default. Since C++ programming languages use it as their default method of classifying characters, we did not have to implement it ourselves. However, we are directly using it when we want to send data throughout the system. For example, if we wanted to add an LCD to display some output data from a system for testing, we would need to send the data in ASCII format to the code section that is sending data to the LCD. The code section would then select the correct configuration for the LCD depending on what ASCII value we sent it. ASCII representation will be used to decode serial messages in the testing phase of the project. Most Integrated Development Environments have built in tools to decode registers and memory with the change of a setting.

4.1.8 – Requirements Specification Development (IEEE 1233-1988)

A design process standard that focuses on the proper process for designing requirements and specifications is IEEE 1233-1988. The model that the standard offers involve three main nodes: the customer, the environment, and the technical community. The process of coming up with requirements and specifications start with the team looking at how the in-question requirement is affected by constraints. The design team must spend time to cycle the requirement ideas and details to the customer of the project. Ultimately, it is the customer that must see be satisfied with the outcome of the project. The design team must consistently check with the customer and the technical community to see if the design process is feasible for their needs.

This is not a "one cycle complete process". This process must undergo many times throughout the project to make sure that the customer is up to date with any constraints and limitations brought forth.

For Senior Design 1, our main customers are the rocket design team. Our rocket design team is ultimately responsible for acquiring most of the bulk the points in the FAR 1030 competition. Thus, the requirements have been discussed with the rocket team and followed the customer There is a fine balance of obtainability of a requirement specification to the practical implementation of the requirement itself, requirements must be able to be realistic for the design team and the customers. Asking the technical community to see if an idea, processes, or design has already solved or will help to fulfill the requirements and needs set by the customers.

Ultimately our we must present our project to Aerojet. In Senior Design 2, our team should have a clear understanding of the rocket design team needs. At which point, our customers now become Aerojet. We may only be able to meet with the Aerojet sponsors once before the Far 1030 competition. So, we must prepare our presentation of our rover

and bring final questions and other concerns Aerojet may see with our rovers engineering requirements and specifications.

4.1.9 – Ontologies for Robotics and Automation (IEEE 1872-2015)

IEEE 1872-2015 is a standard for common robot definitions and axioms which help promote clear and common language for robotics Terms such as "fully autonomous", "pose", "orientation measure", help define a common understanding of robotics in the community and when using robotic terminology. Since many terms in robots are used loosely as umbrella terms for these concepts it would be wise to view this standard in depth if one wishes to pursue robotic development; In this document, we have been using IEEE 1872-2015 to help promote the correctness and use of these definitions.

4.1.10 - RoHS Compliance

Restriction of Hazardous Substances (RoHS) is an EU standard whose goal was to reduce the number of hazardous substances within electronics. The standard targets 10 specific substances listed below.

Cadmium (Cd): < 100 ppm
Lead (Pb): < 1000 ppm
Mercury (Hg): < 1000 ppm
Hexavalent Chromium: (Cr VI) < 1000 ppm
Polybrominated Biphenyls (PBB): < 1000 ppm
Polybrominated Diphenyl Ethers (PBDE): < 1000 ppm
Bis(2-Ethylhexyl) phthalate (DEHP): < 1000 ppm
Benzyl butyl phthalate (BBP): < 1000 ppm
Dibutyl phthalate (DBP): < 1000 ppm
Diisobutyl phthalate (DIBP): < 1000 ppm

Figure 20 – List of hazardous substances that are limited

Beyond just what these electronics are made of, the RoHS compliance is intertwined with the Waste from Electrical and Electronic Equipment (WEEE). Where RoHS only focuses on the substances used to create the electronics, WEEE focuses on how the electronics are treated and recovered for recycling. The WEEE compliance mentions things like dismantling and recovery instructions, take back operations, and end user info and marketing.

Although these standards are not required in America, many of our electronic parts will follow this RoHS and we tried to favor components that do follow this standard. Since this is a standard for the companies producing the components, we won't be specifically applying anything directly to our project, but we did look for components that follow these RoHS rules as they are important for the environment. As for the WEEE compliance, one of the rules of WEEE compliance is that they are sold in 100% EU markets, so we did not select any parts with WEEE compliance as well as RoHS. We did, however, keep in mind the WEEE compliance guidelines and try to find suppliers that take the extra time to make components that our recyclable and not harmful to the environment.

4.1.11 - Reflow Oven Process Control Standard (IPC-7801)

IPC-7801, is a standard about reflow process control. In our project, it will be likely that the team chooses to use Surface-Mount Device (SMD) components on our printed circuit boards. SMD technologies require a different form of soldering opposed to components with leads and legs. For this purpose of mounting a variety of small SMD components to PCB, a reflow oven will be used. Some bigger integrated circuits can be soldered with a heat gun; only if our board needs SMD components on both sides of the PCB will we use a hand air iron. Using a reflow oven with a correct temperature profile will reduce risk of damaging components from over exposure to heat.

This reflow process control standard lays out various methods and profiles for using different soldering materials. Most of these profiles include the same overall control processes for reflowing PCBs. The circuit board is placed inside the oven where the temperature is ramped to a determined value for "presoaking". The PCB can reach a temperature which does not harm most integrated circuits. This presoaking step is necessary; component pads differ in size, some chips have large grounding planes, and overall, the thermal absorption of the materials in components can vary. Presoaking (usually) allows a distributed and even temperature before the ramp to reflow. This will allow high thermal absorbing parts to stick to the board assuming the proper profile has been used.

The temperature then ramps upwards for a short period of time, reaching the melting point of the solder and flowing solder (with the help of flux). Immediately after the peak of the ramp a cool down phase begins, where the temperature is a declining ramp to ambient temperature IPC-7801 helps prevents mass produced PCB component from being improperly soldered via reflow, and in our case, we do not wish to reflow more than we must per PCB made in this project due to improper profiling. Figure 21 depicts the reflow process as described.

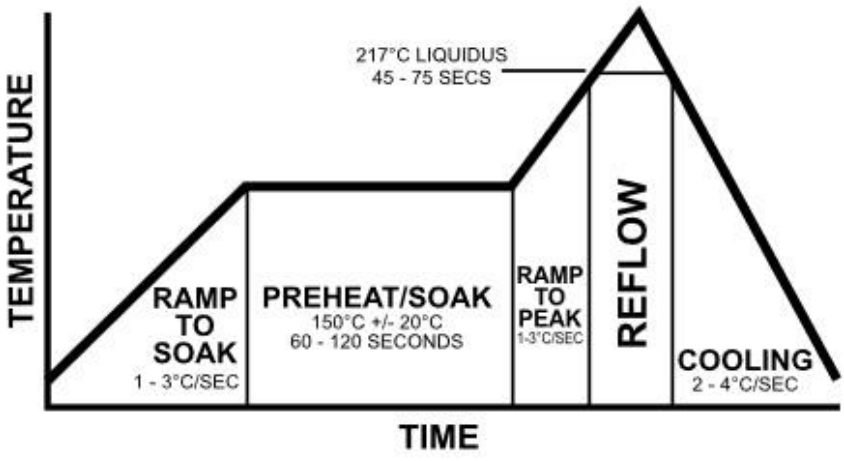


Figure 21 – Reflow oven example temperature profile and timeline.
Reprinted with permission from [F21]

4.1.12 – Serial Peripheral Interface (SPI) [De Facto Standard]

Serial Peripheral Interface (SPI) is a synchronous serial form of communication between two or more electronic components. It has become a de facto standard since now most components ship with SPI being one of or the primary way of intercommunication between the component and other components.

In embedded systems, which is how we would be using the standard, SPI runs off a serial clock provided by the master of the system. For our rover this will be provided by our MCU as it will be the master in all cases. The master out slave in (MOSI) and master in slave out (MISO) are the lines in which the data will flow. For our application of SPI, we would most likely use the independent slave configuration over the daisy chain configuration for the MOSI and MISO lines. This is because the independent slave configuration is faster as it takes less clock cycles to complete and we have many digital wires available on our MCU. The "slave select" (SS) wire, which is in essence the enable wire, will be connected to a digital pin on the MCU and be raised high to select that component for data transmission.

As SPI is something decided on by the manufacturers, we did not specifically work SPI into our design. However, we did use SPI in our design for components. These standard effects our design as we must keep it in mind while picking components as not all components have I2C available.

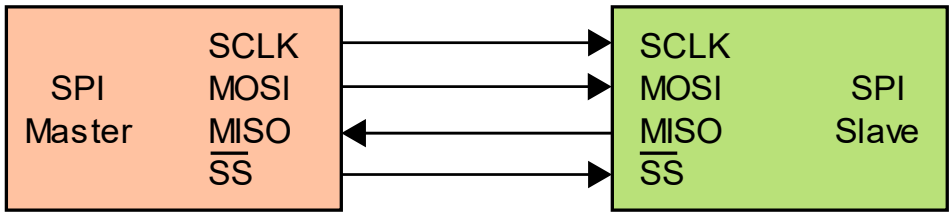


Figure 22 – SPI Single master single slave interconnection.
Reprinted with permission from [F22]

4.1.13 – Inter-Integrated Circuit (I2C) [De Facto Standard]

Inter-Integrated Circuit (I2C) is another synchronous communication bus protocol like SPI. I2C has also become a de facto standard for communication between embedded systems components. I2C differs from SPI in many ways, but the most obvious is that it uses two wires for communications across all components instead of SPI's best case of 3-4 (depending on single component).

I2C's two wires, as seen below, are the serial data line (SDA) and serial clock line (SCL). Both wires will be connected to our MCU as our MCU will be the master just like how it is in SPI. The SDA line will carry data to and from the MCU and whichever component is sending / receiving data. The SCL will be provided by the MCU as it is the master in the system. This protocol allows for up to 127 devices to be connected at once which is very attractive compared to SPI's requirement to add more wires for each component. This allows for more flexibility in the project since we don't need to worry about having enough enable pins coming from the MCU like in the SPI protocol.

Much like the SPI standard, we did not directly incorporate this into our design in the sense that we must create and I2C bus. We did need to keep an eye out to make sure the parts we want can use I2C as we plan to use it for most of our sensors. This standard impacts our design by allowing us to be lenient on initial designs for connecting sensors up to the MCU. Since I2C only uses two wires no matter how many components you have (up to 127), we can add or remove components from the design as we see fit and not have to worry about redesigning the connections to the sensors or the MCU.

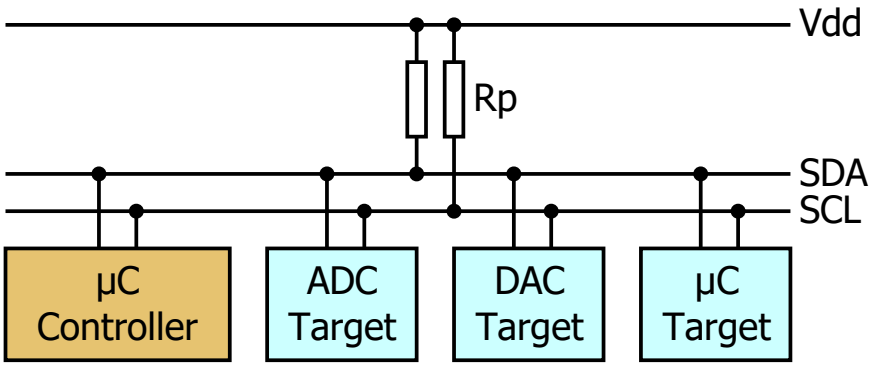


Figure 23 – I2C interconnection featuring one master and three targets
Reprinted with permission from [F23]

4.1.14 – Standard for Rechargeable Batteries for Mobile Computing Devices (IEEE-1625)

Our design will be implementing a multicell Lithium-Ion Polymer rechargeable battery pack solution which is subject to this standard. This standard does not cover safety, security, health, or environmental protections for the implementation of the battery. It is instead on us, as the designers, to ensure that we follow the battery manufacturer's specified use conditions and ranges.

This standard covers how these batteries are meant to be designed and manufactured for multi-cell rechargeable battery packs. This information is used to design the approaches with which manufacturers and end-users can evaluate the performance and specs of the battery pack. This is done so that the end-user (which would be us as our rover design

is autonomous) can utilize the battery pack safely within specifications and operating conditions which would maximize the lifespan and reduce potential use errors.

The most important part of this standard is the design for the rechargeable function of the battery pack. The rechargeability of the battery is determined from battery to battery but the charge control for the batteries is the only direct interaction the end-user has with the battery. The charging rate and temperature specifications are given by the manufacturer in the data sheet based on this standard to reduce misuse by the end user.

4.1.15 – IEEE Standard for Sensor Performance Parameter Definitions (IEEE 2700-2017)

The IEEE 2700 standard provides common definitions for many sensors that we plan to use in our project. The standard sets up a common framework that is used among a significant portion of the industry in terms of defining sensors. The standard also defines what certain performance terms and tests tell the user and how the user can use this performance data to use and design their projects. The standard covers a wide breadth of sensors including accelerometers, magnetometers, gyro meters, barometers, hygrometers, temperature sensors, ambient light sensors, and proximity sensors.

The purpose of this standard was to ease the system integration burden on developers and accelerate the time it takes to development embedded systems to completion. Using the terminology to define sensor performance data that is stated in the standard will hopefully reduce the non-scalable integration problems met by embedded engineers.

Specifically in our project, this standard helped us research and choose sensors for our rover. Understanding the performance measurements, stated in datasheets, and listed among websites and blogs, helped us come to a better conclusion on specific parts. We specifically used this standard to help further our knowledge on accelerometers, magnetometers, gyroscopes, and proximity sensors since these sensors are used directly in our rover project. We hoped that including this standard in our project would also help us test the performance of the sensor when we receive them. Since this was the first time most of us have used some of these sensors, this standard helped us determine if the performance stated in the datasheet is truly what we received.

4.1.16 – Efficient infrared sensor and camera-based monitoring system (ICECCPCE-2013-6998764)

This standard talks about efficient control algorithms for IR sensors and for camera monitoring systems. They standard goes into detail on how to detect and record objects when they cross the area that is covered by a monitoring scheme. The goal of this standard was to lower the consumed power and storage capacity required for systems like these monitoring systems. The standard goes into a method of sensing that uses IR sensors for their low power consumption and allows them to independently turn on cameras when something crosses their area. This allows the cameras to stay turned off when nothing is happening but then be turned on when the IR sensors pick up some activity.

This standard goes into an in depth look at a system that we may use on the rover if we implement some camera object detection system. At the very minimum this standard will help expand our knowledge about IR sensors and using cameras to capture detected objects. This standard also points out important power consumption issues that we may run into the with use of a camera for object detection.

4.1.17 – Information Technology-Ubiquitous green community control network-Control and management (ISO/IEC/IEEE 18881:2016)

The ISO 18881 standard talks about the connection between sensors and how to efficiently transfer data between sensors and an MCU. This standard starts by defining all the sensors and components they mention in the standard, which helps the team understand some of the common abbreviations used for components. It then goes through setting up a framework to build your sensor's connection scheme into. Then it mentions how to make sure each sensor can connect to another specific sensor or MCU. Next a Gateway (GW), which is the path that data travels to go from component to component/storage, is defined and explored.

Next the standard brings up different protocol standards, some that we have covered with other standards so won't go into here. After this it dives into configuration and how to get and set up the data required to configure these modes of transferring data. The standard finishes on mentioning some ways to implement structures using these data protocols for efficient data transfer between components.

This standard impacted our design significantly when deciding how to connect the multiple sensors we needed. Using the knowledge we gain from the standard, in terms of different ways to set up GWs, helped use design a more efficient system that won't have to be completely reworked if we were to replace a component. This standard also has some naming conventions that we used in our software development when naming data pulled from sensors.

4.2 – Realistic Design Constraints

The list of design constraints for our project will be based on ten types of general design constraints and how they specifically effect our project and design decisions.

4.2.1 – Economic and Time Constraints

The economic constraint of this project is outlined by our sponsors, Colman Aerospace, and by the lead project advisors appointed by UCF. Each payload group is limited to a budget of $2,000.00. This budget is also constrained by necessity defined by the UCF advisors. For example, there are a few small, lightweight, and accurate 360-degree lidar sensers that are in the couple hundred to couple thousands of dollars range, however there are many other less effective and less expensive ways to solve the same problem as this lidar sensor would solve. In this case, the advisors urge us to find a different solution that uses the budget in a reasonable way.

The economic constraint forces us to consider how much money each of our piece's cost compared to how important they are to the project. If we spend all our budget on sensors, then we won't have enough money to power the sensors and move the rover. This economic constraint is also important to our group as we must keep our rover very small compared to normal rovers. So, although we may be able to find many common electronic chips and sensors that are cheap and widely available, we must settle for smaller more expensive chips that may be less efficient than its larger and cheaper counterparts.

Our project has two different time constraints. One time constraint is given by the FAR 1030 competition while the other is given by UCF senior design one and two classes required for graduation. The time constraint of the FAR 1030 competition have a completed payload design by Friday June 4th, 2022, ready for rocket inspection. The payload must be ready for launch by Saturday June 5th, 2022, to compete in the competition.

The time constraints given by the UCF courses are much more precise and detailed. We need to have a design document ready to turn in on November 19th. By the time our design document is ready we are expected to be purchasing parts and starting the prototyping phase. By early/mid spring 2022, we are expected to have a working prototype. By the end of spring 2022 we are expected to have a working project that passes the requirements laid out by the group and by the FAR 1030 competition rules. The time constraints for our project are important in defining what problems we want to solve with our rover. Since this project is very open ended when it comes to what extra problems, we want our rover to accomplish we must pick reasonable problems to overcome in a timely manner.

4.2.2 – Environmental, Social, and Political Constraints

The biggest environmental constraint we have on our project is the reusability of the rover and its energy supply. Using nonrenewable batteries like common single use alkaline batteries is unacceptable for our design because of the environmental harm it causes long term. Therefor this constraint will make us focus on what renewable battery options are available and compatible with our rover design.

Another environmental constraint we have is not leaving any stray electronics or rover parts at the site our mission takes place. The only parts of our design that should be left behind is anything used to secure our rover during launch and deployment, like the canister and / or sled. Other than these parts, we had to focus on not leaving any rover parts on our return trip to the launch pad, as any discarded parts would be littering the mission area and harming the environment.

Our main social constraint would be the cost of our payload. Although we have a budget of $500.00, there is a social expectation that we kept the design as cheap as possible while still hitting the requirements of the project. This constraint forces us to strategically choose parts not solely based on performance, but also on what specific functions they add to our rover. There may be a great solution that costs $250.00 and a good solution that costs $100.00 and because of this constraint it's up to us to pick the best option at the best price.

The only political constraints placed on our project would be directed towards the aerospace team creating the rocket. They would have to make sure we do not break any

relevant treaties the U.S.A may have with nearby countries. For our personal part in creating a rover, there is no relevant political constraint we face.

4.2.3 – Ethical, Health, and Safety Constraints

One important safety constraint we have for the rover is making sure we can keep it under control. Since we are making a system that is autonomous, this leads to the creation of many systems that can work without any human interaction. We must take this into account when testing and deploying the system, as we want to make sure the rover does not get stuck in some logic loop that denies us access to the control of the rover. If we were to lose control of the rover, it may harm people nearby by running into them or harm itself by running into some surface that would crush it.

Another safety constraint is the placement of various sensors on the rover. For example, the Lidar sensor is a device that shoots out laser beams of infrared light to sense the distance of object. This laser is also dangerous enough to harm someone's eyes if pointed directly at them. Therefor it is important that we are aware of this positioning constraint for these types of sensors, so they do not harm others around the rover.

4.2.4 – Manufacturability and Sustainability Constraints

One of the biggest manufacturability constraints we have is the current shortage in electronic chips. For the last year or two now various computer chips have gone out of stock with huge lead times, some as high as 1 year. This constrains us to choose some parts because they're in stock instead of the best fit for our needs. This also indirectly makes our project more expensive. Since most of the inexpensive choices were sold out, we had to choose some parts that would not normally be chosen over cheaper alternatives.

Our sustainability constraints mainly stem from our environment constraints. Although the launch location for our project is designated for rocket launches, we still don't want to damage the environment any more than is necessary. For example, we won't be able to avoid every plant and organism while our rover is pathing home. However, we want to make sure we don't destroy plants that we can avoid like bushes and cactus.

Another sustainability constraint would be making sure we choose parts that can be reused and not discarded. This constraint is mainly a variable for choosing our battery and the method of charging the battery. This constraint also forces us to make sure our deployment method and any internal actions do not litter parts such as springs, pieces of metal, or any electronic components on the ground as this would degrade the launch site for future launches and the sustainability of the rover for future uses.

5 Initial Design, Architectures and Related Diagrams

5.1 – Video Capture & Transmission Subsystem

The first subsystem (Figure 24) in our project is the live video transmission and all its components. The design we decided to go with is an analog camera that sends its information straight to a 5.8 GHz RF transmitter. The transmitter is turned on or off based on our PCB logic. When on, the transmitter sends out 5.8 GHz RF waves via the antenna connected to it. We then have a receiver, separate from the rover, on a matching frequency and channel to receive the video.

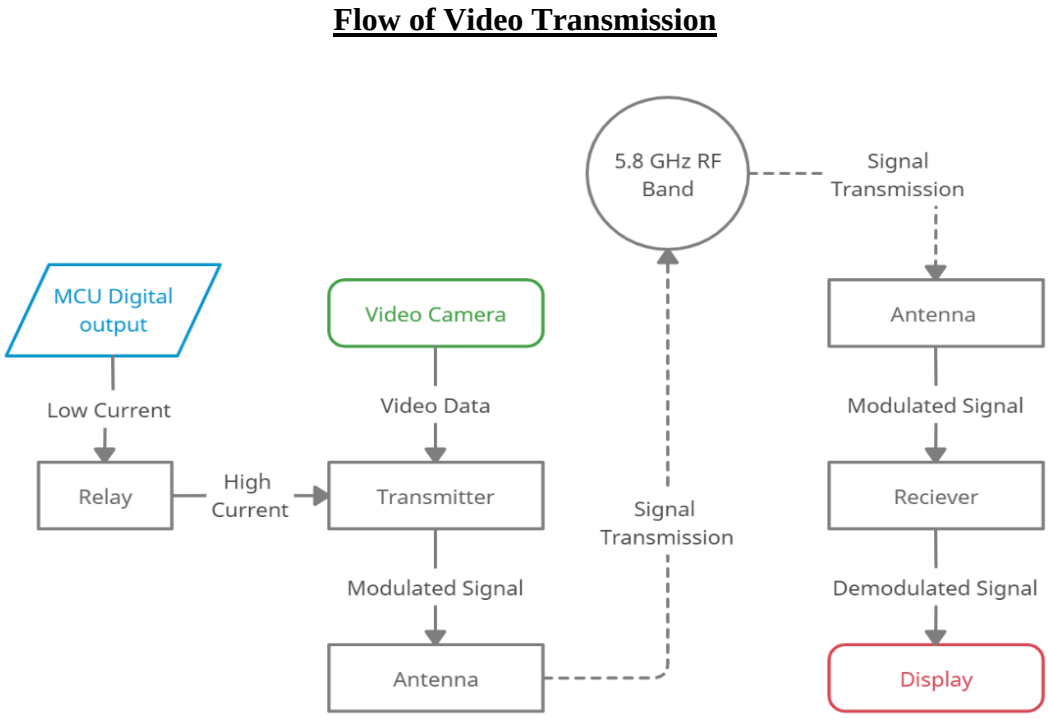


Figure 24 – Signal path of the video transmission system. Starting at the green oval (Video Camera) and ending at the red oval (Display). MCU will trigger current to the video system. [F24]

Starting with the camera, we chose the Wolfwhoop micro camera mainly for its size and output type. After looking at Raspberry Pi cameras we noticed that although the Raspberry Pi modules gave very good video quality and frame rate, all this data was digital instead of analog. Most transmitters we found wanted the data in analog form. This can be done by the Raspberry Pi, but it made more sense for us to remove the Pi altogether and use a camera with the correct output for our chosen transmitter.

The purpose of the PCB in this system is simply to supply power. When the MCU that is controlling our rover wants to start the live video transmission, it will supply power to the transmitter and camera to turn them on. The transmitter is hard set to the frequency and channel of our choosing so there is no need for the PCB / MCU to do anything more.

The transmitter ended up being the deciding factor for most of our design. We looked at many different RF ranges for price, quality, and simplicity. We ended up choosing a 5.8 GHz FPV (First Person View) video transmitter for not only its range (in distance) but also its bandwidth. The transmitter comes with an antenna that should exceed our range requirements and a matching receiver.

When choosing which frequency we would use, the limiting factor seemed to be bandwidth. For our initial design that used the Raspberry Pi, we were finding transmitters that couldn't even transmit 420p at 1 frame a second, or about 300kbps. Around the 1 GHz range we found some transmitters that would be able to send 1 or 2 frames a second at 420p, but this is the range our GPS will be working in. Finally, we found these 5.8 GHz transmitters which have the bandwidth able to match our cameras output.

As visualized in Figure 25, frequency is not equal to bandwidth, they are directly linked. The higher the frequency the shorter the wavelength gets which in turn increases how quick your radio frequency changes from peak to trough. The faster these waves osculate the more information can be sent in a smaller amount of time. From the figure below (Figure 25); the lower radio frequencies have wavelengths that are very long which is why the bandwidth is smaller. This is the reason we can use the smaller frequency transmitters to send live video without severely impeding video quality.

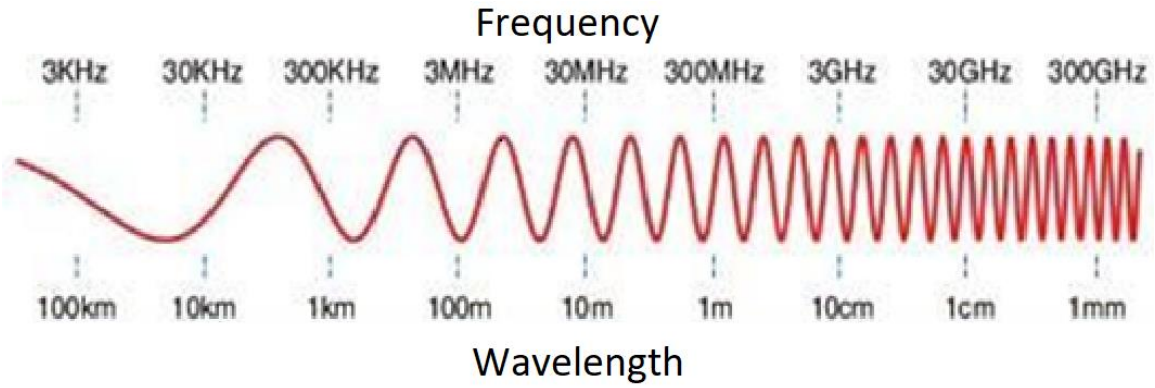


Figure 25 – Visual of how wavelength effects bandwidth and frequency [F25]

Another hurdle the transmitter produced was its range as transmitters on their own only transmit 10 – 100ft away. To solve this, we have an antenna attached to the transmitter which should extend the signal to about 2km in range. This would be highly dependent on the field of view of our rover from our receiver. Since there is no constraint on how our receiver must be set up, we have set the antenna on a servo motor to prop it up as high as we can for the best results. In worst case scenarios, we can find a better antenna, however it would most likely be much more expensive.

The last design choices made for the transmitter was based on Federal Communications Commission licensing. There are a few bands which are labeled license free, and the 5.8 GHz frequency is one of them. Many of the lower frequencies we were looking at

would have required a license to use or we would have been fined for using them. Initially we were looking for transmitters in the 800-900MHz range since that is also license free bands, however in combination with the video quality increase we decided 5.8 GHz would be the better choice for the project.

The schematic for the live video transmission is very simplistic as seen in Figure 26 below. Since we chose a camera that can not only feed directly to the transmitter but is also can be powered from the transmitter, there was minimal work to do to get the set up to work. The 12V in the schematic comes directly from our PCB however this voltage is a variable voltage that can range from 7V – 30V based on quality of video sent, antenna used, and how far the transmitter must send the radio waves.

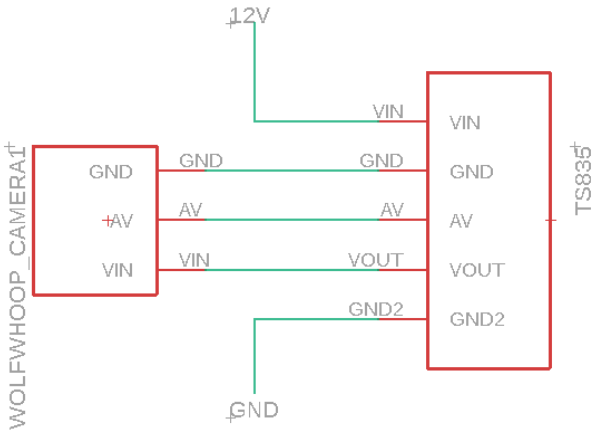


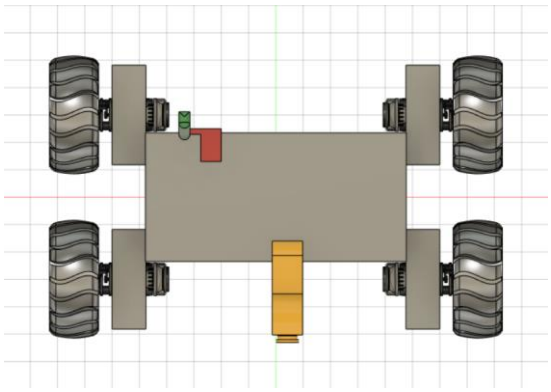
Figure 26 – Schematic for live video transmission [F26]

The antenna and receiver are left out of the schematic above since generally they are not a part of our rover design. The antenna we used did come directly with the transmitter but is rated to transmit the distance we need. If we do need to change out the antenna it most likely because we ran into range issues during testing. The receiver also comes with the transmitter and the receiver will not be on the rover, it will be connected to our laptop wherever our group is located during the rover's mission.

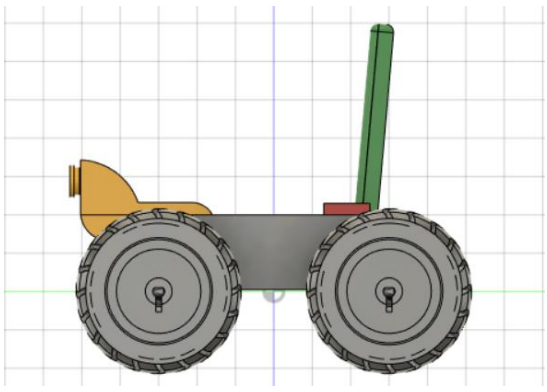
The positioning of the live video system is as seen in Figure 27. The yellow box is the camera which is placed towards the top and front of the rover, directly under the IR sensor. The transmitter is depicted in Figure 27 as a red box which would be mounted to the top of the rover near the center as to avoid any interference that the motor encoders and other sensors may cause, but also to have access to the top of the rover for the antenna. The green box is the antenna which is horizontal with the rover pre-deployment and will be propped up vertically post deployment.

Initial placement of live video components

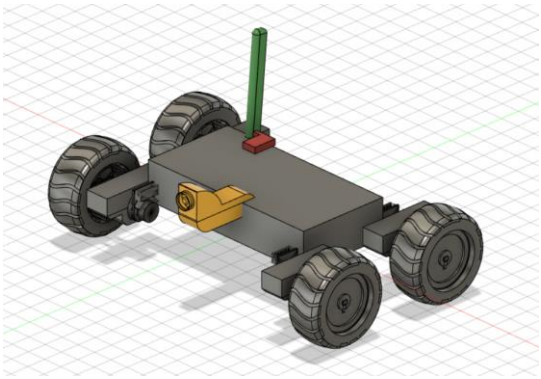
TOP VIEW



SIDE VIEW



PERSPECTIVE



Green component: Antenna, Red component: Transmitter. Yellow component: Video camera.

Figure 27 – Placement of live video components [F27]
Camera components sized as rough estimates, final prototype may differ

5.2 – Sensors Subsystem

The MCU is a part of each of the subsystems in some way, so for the MCU subsystem we decided to focus mainly on the GPIO connections that are not shown in other subsystems. This was to say we show the distance sensors, IMU sensors, and GPS.

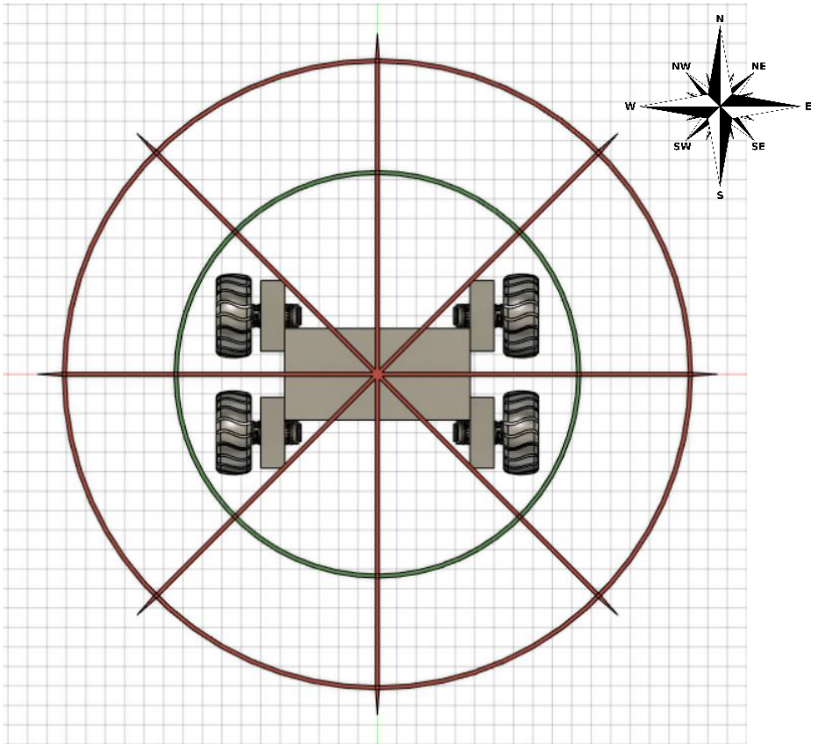
Distance Sensors

We decided to implement an eight-direction detection algorithm which means there will be eight IR sensors 45° apart from one another. This setup will allow us to detect objects a full 360° around the rover. A key design feature here is that each "corner" sensor is at 45° angle from the face of the rover instead of being at the actual corners of the rover.

Since our rover is a rectangle, this keeps the sensors equally spaced on the collision circle. This means whichever way the rover turns we were able to line up a sensor in that direction.

In Figure 28, the rover is depicted gray, the green circles are the inner and outer collision boundaries, the red lines are where the sensors are pointed, and the red circle is the max distance of the sensor. For mounting the sensors, we put them in the eight locations where the redlines meet the blue box and then point them at the correct angle away from the center of the rover. As for the collision boundaries, we had to calculate the distances at which each red line crosses a green circle and those are the two distances we be check for when we read from the sensors

SENSOR LAYOUT



Red circle: Maximum sensor distance. Red lines: Sensor line of sight. Green circle: "Close avoidance ring"

Figure 28 – Sensor layout. [F28]

As for the wiring of each IR sensor, they are connected to an I2C line as a slave while the MCU is the master. The sensors are read one after another to have a valid sense of the distances from objects in each of the directions. We have initially decided the frequency of checking the sensors will be around one second, however the final value was determined to be closer to 200ms during the testing phase of the rover. This is because we wanted to scan distances more often than we expected to during the initial conceptual stage. Something that would only become apparent as we tested the functionality of the rover.

IR SENSOR SCHEMATIC

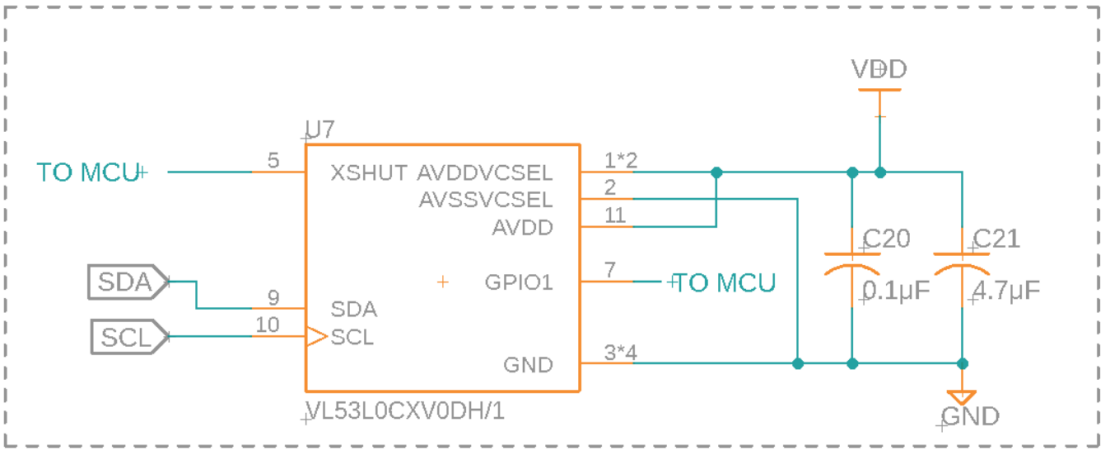


Figure 29 – IR Sensor Schematic [F29]

IMU Sensors

For keeping track of our local position, we decided we would include an IMU design into our PCB. The IMUs purpose was to tell us how fast the rover is moving and help us relate where we are to where we were. The IMU is a combination of three sensors which is an accelerometer, gyroscope, and magnetometer. The accelerometer will give us the change in 3d space. The gyroscope tells us the angler change in 3d space. The magnetometer will be compared to the accelerometer data to relate it to the earth's magnetic field which will provide the yaw measurement which is another type of rotation in 3d space.

The IMU sensors will be connected to the same I2C lines as the distance sensors above. This means they will also be slaves while the MCU is the master so it can control when the data is read from the IMU. It is important that the MCU reads the data from all three sensors at the same time, much like the group of distance sensors, since the calculations need to all be made in the same location at roughly the same time for the data to be significant.

For the IMU system we had the option between an off the shelf unit or building it into our PCB. We decided to go with building it ourselves so we would understand the technology more than if we bought an off the shelf version. This however had some drawbacks as the off the shelf units tended to have their own MCU to gather and produce the roll, pitch, and yaw data through I2C. This meant instead of getting understandable data straight from the sensors we must process it in our rover's MCU to produce the values. This means more work for our MCU to have to process. However, we should have enough processing power to make it work.

It is important where we put these sensors as well since the magnetometer uses the hall effect which might be interfered with by the motor encoders and the GPS. Our current solution to this problem was to build the sensors into the PCB which will be located towards the center of the rover. The motor encoders are at the bottom while the GPS is at the

top, so the center of the rover is our best bet for the electronics not to interfere with each other. If this placement isn't good enough, we'll just have to make sure to only use one of the three electronics at a time to minimize the interference with each other. Placing the IMU system close to the center of the rover is also beneficial to the calculations of the roll, pitch, and yaw since the values are approximating a point and not the whole rover itself.

MAGNETOMETER SCHEMATIC

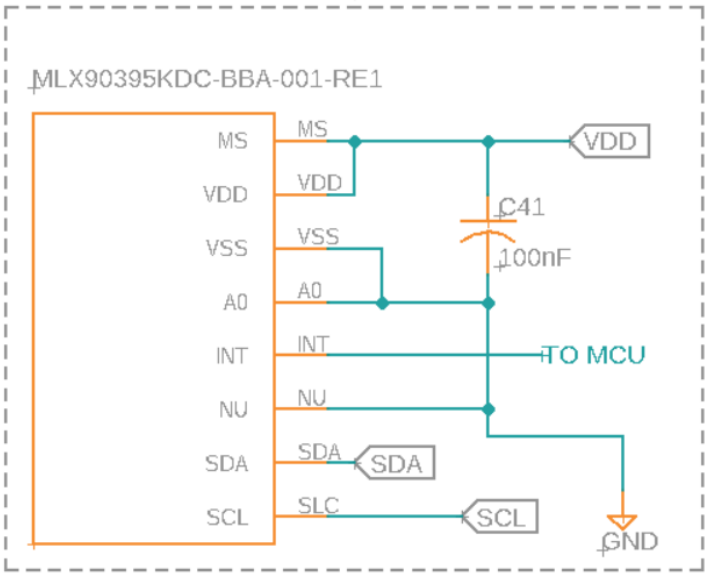


Figure 30 – Magnetometer Schematic [F30]

GYROSCOPE SCHEMATIC

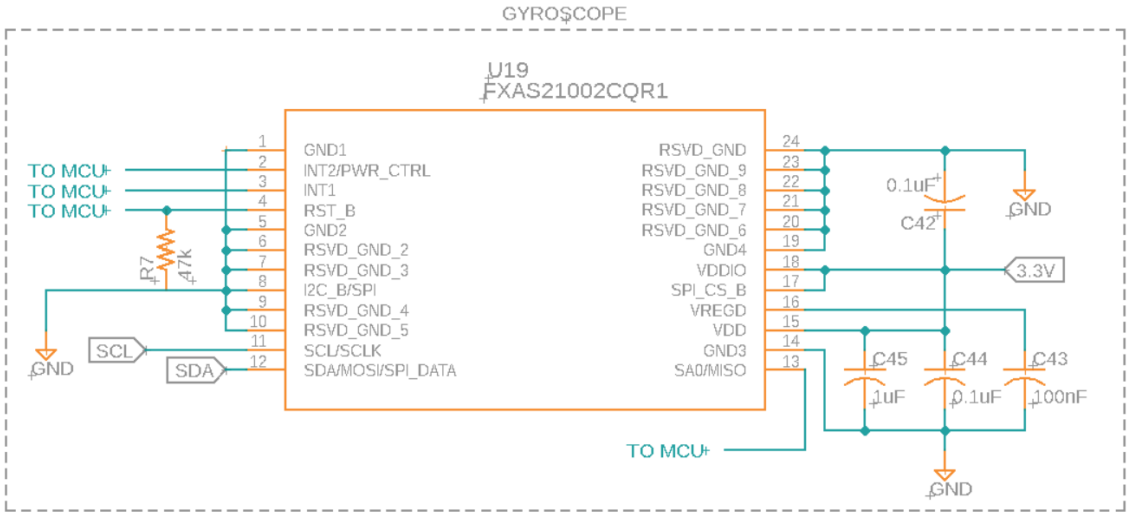


Figure 31 – Gyroscope Schematic [F31]

ACCELEROMETER SCHEMATIC

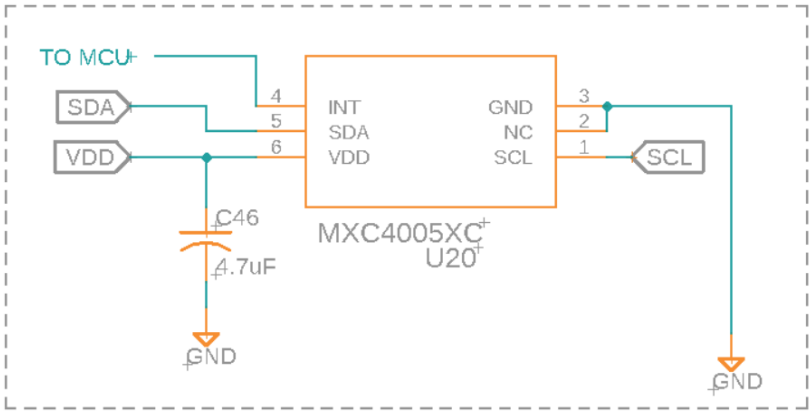


Figure 32 – Accelerometer Schematic. [F32]

GPS Sensor

Like the IMU sensors, we use a GPS sensor to keep track of not only where our rover is but also where our destination is. The big difference between the IMU system and the GPS is that GPS gives us a global position instead of local position. The information we expected to get from the GPS is the latitude and longitude of our rover and the destination. This is our primary source of information when it comes to directing the rover back to the launch pad.

GPS (BARE MINIMUM) SCHEMATIC

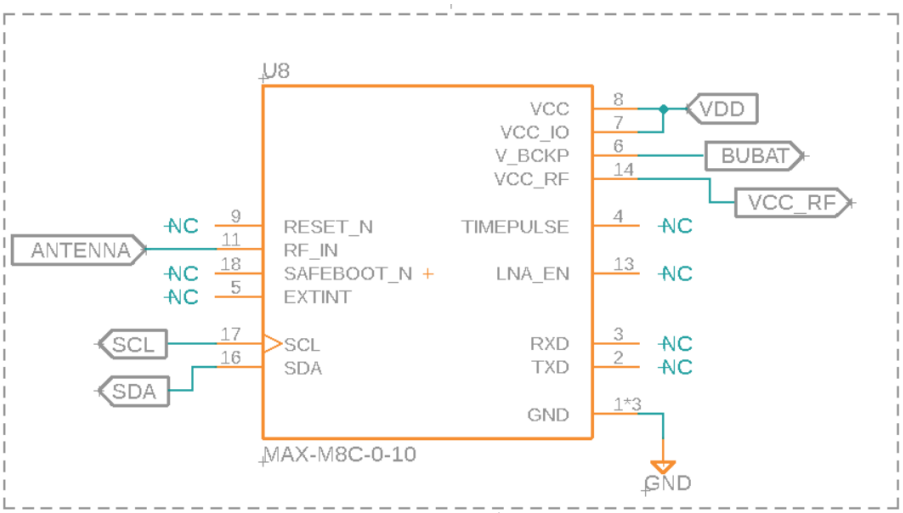


Figure 33 – GPS (Bare Minimum) Schematic [F33]

Placement of the GPS is also an important topic when trying to use a GPS in a project. We want to place the GPS in a position where it has access to the sky and is far away from other electrical components to avoid any interference they may present. For these reasons we wanted to place the GPS towards the top of the rover much like the video

transmitter, which will keep it away from the PCB board and the lower down motors. However, on removing the daughter board this became unviable. However, if we could remake this system again, we would try to get the GPS off the main board and away from any magnetic fields.

The GPS can be connected through I2C like the other sensors. This means we were able to connect it to the same I2C line and have all the sensors be on the same control line coming from the MCU. We weren't decided on if we would hard code the finishing position, or if we would need to pull it while our rover in within the rocket waiting for launch. We ultimately didn't make a concrete decision as we didn't get to test with rocket integration. However, we planned to pull data from the GPS when we landed and then every time, we made a rotation with the rover. Keeping the GPS data up to date would allow for us to always pick the best, or shortest, path back to the launch pad.

GPS (AMPLIFIED RF IN) SCHEMATIC

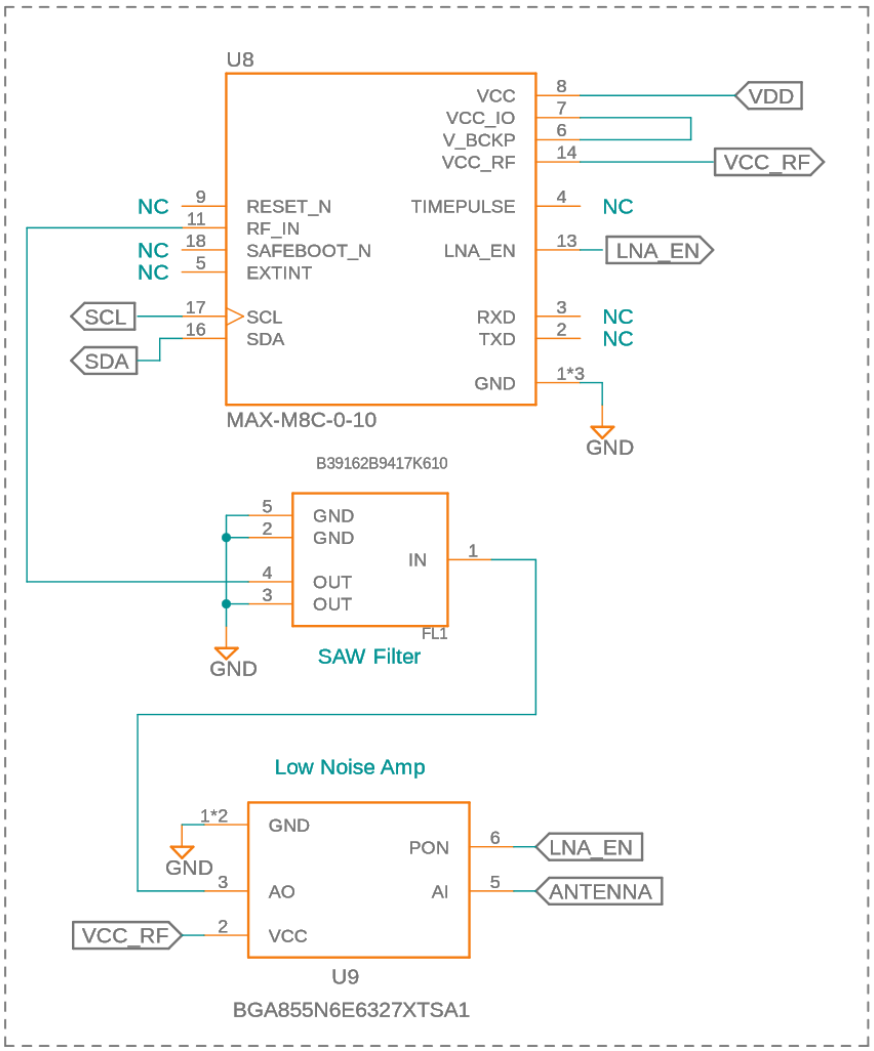


Figure 34 – Schematic for GPS Passive amplified antenna schematic [F34]

5.3 – Power Subsystems

The power delivery subsystem is the combination of two systems: the voltage correction system and the motor driver system. The goal of the voltage correction system was to take the nonregulated voltage from the battery and limit or boost its output for use by all the other subsystems. For all our subsystems, we need voltages of either 3.3V, 5V or 12V. The battery's nominal voltage is 7.6V which is the minimum that it will always push regardless of the charge on the battery. The 3.3V and 5V regulators are stepdown voltage regulators which are simple to implement but may be inefficient in terms of power consumption. The 12V regulator would have required a boost converter to increase the battery's nominal voltage to a steady 12V for the video transmission system.

The motor driver system is a secondary system to the voltage correction system which provides power for the motors to run. Each motor driver can take up to 2 PWM enable inputs from the control system and use them to power the motors. The DRV8833 chips that we are implementing for the motor drivers are a dual H-bridge design, meaning we have two outputs with the ability of running forward or backwards.

Voltage Regulators

The main design considerations for the voltage regulation subsystem are battery connections to each regulator and their outputs to each different subsystem. Due to the maximum current draw from the motors, we implemented a 5V regulator per motor driver as the regulators have an output current limit of 1.5A. One of the DRV8833 motor drivers has a rated output of 1.5 A per channel with 2 channels per driver. We don't expect the motors to pull more than 0.8A even if they were to stall. However, it is better to have that headroom if the motors were to have a stalling failure so that we don't burnout the voltage regulators and have to replace our entire custom board.

We have a total of three regulators with a 5V output: two for the motors and one for all the other low power electronics such as the camera and sensors. We also need two 3.3V regulator system as it is supplying power for over dozen low power electronics like the GPS, MCU, IMU and all the IR sensors. We planned to use the LM1085 regulators for both designs as that would be more than enough current headroom for the low power electronics in the design.

Our 12V boost converter would have been our single greatest current draw system as its purpose was turn a low-voltage, high-current unregulated input into a high-voltage, low-current regulated output. This regulator only supplied power for the video transmission system and that has a low current draw of around 0.25 mA under load. The boost converter has an output with a ripple voltage due to the high frequency switching necessary to produce the higher voltages. We could have used a capacitor to help reduce the impact of the ripple voltage as capacitors can't change voltage instantly. This would give us an output voltage that has a more consistent 12V with less variability.

The schematic for the 3.3V regulator is simple and all we really need to implement is a couple of capacitors to reduce the noise from the input and ripple rejection on the output. We can see the schematic for the 3.3V regulator in Figure 35.

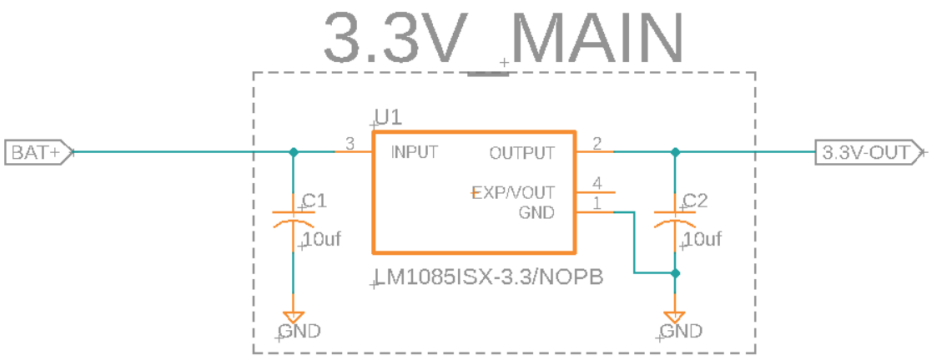


Figure 35 – 3.3V Voltage Regulator Schematic [F35]

The schematic for the 5V regulator is identical to the 3.3V regulator but it uses the fixed 5V regulator chip instead. We still implement the capacitors on the input and output for noise reductions and ripple rejection for a cleaner signal output. We are implementing this design three times: two for the motor drivers (one per driver) and one for low power electronic systems. The overall power schematic will illustrate how we arrange the 5V regulators for the motor drivers. The basic schematic for the 5V is shown in Figure 36.

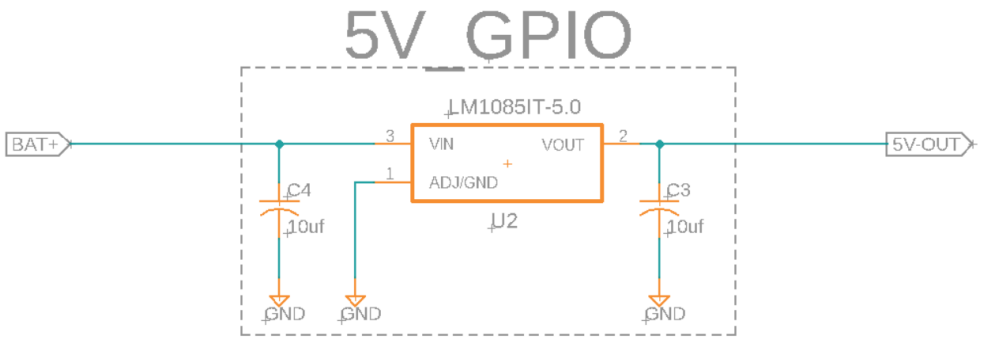


Figure 36 – 5V Voltage Regulator Schematic [F36]

The schematic for the 12V step-up regulator is the most involved and required the most design consideration. The compensation (COMP) pin is where we connect the branches that dictate the frequency of the switching for the regulator. We use an inductor between the input and switch pins to store the energy from the close switch circuit in the magnetic field. This inductor needs to hold enough energy to step up the voltage to 12V across the capacitor. We use capacitors in the input and output to clean up the signal as we did in the other regulators. The diode used in the design is a Schottky diode with the lowest possible forward voltage we can find. This diode is simply used to ensure the capacitor doesn't discharge when the switch closes. This design wasn't implemented in the final

version of the rover, but we want to show the consideration that was put into this 12V regulator. The schematic for the 12V boost converter is shown in Figure 37.

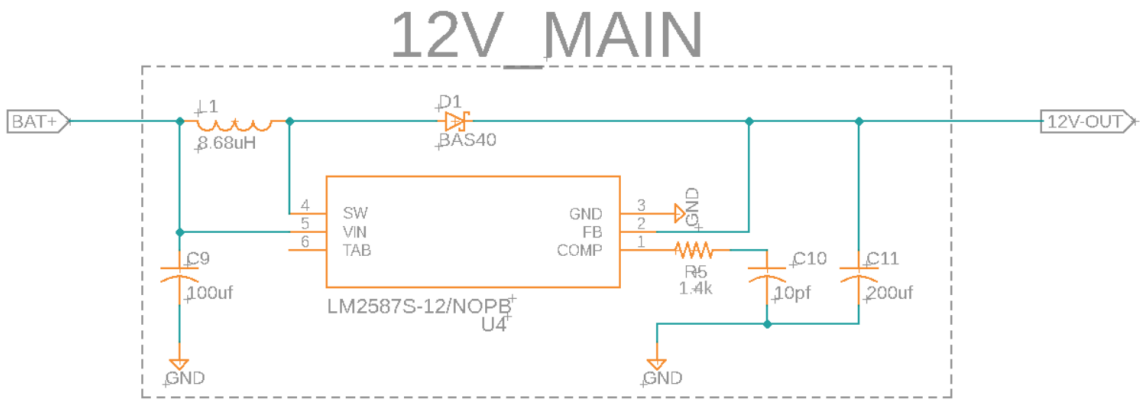


Figure 37 – 12V Boost Converter Schematic [F37]

Motor Drivers

The motor drivers we have decided on, the DRV8833, can be used in a direct fashion with only forward, reverse, and brake control or using a PWM signal to control the speed of the motor. Given that we wanted to use encoders with the motors to allow for control, we decided to use the PWM signal method. This let us choose different speeds for the motor depending on the PWM duty cycle from the MCU. Using PWM inputs also let us choose whether we let the motor slow decay (brake) or fast decay (coast) while going forward or backward .

The logic for the PWM inputs can be seen in Figure 38. The basic idea for controlling the motors is that when holding one of the input pins low and having the other be our PWM input, we can turn the motor forward or back, depending on which pin is constant, and have it fast decay. If we hold the constant pin high and control the other with PWM, we can have the motor turn with slow decay.

xIN1	xIN2	FUNCTION
PWM	0	Forward PWM, fast decay
1	PWM	Forward PWM, slow decay
0	PWM	Reverse PWM, fast decay
PWM	1	Reverse PWM, slow decay

Figure 38 – Motor Driver PWM Logic Table

The DRV8833 chip has a built-in current regulation feature for each motor. These current sensing pins are denoted as the xISEN (A or B) pins. We can use these xISEN pins to control the current draw limit of each motor. Since the motors we are using are identical, we use resistors of the same value each of the 4 motors. The datasheet for our motor shows us that our current for stall current is 1.5A but we can limit that further with the current

control. The reference voltage is set to 200 mV so we can calculate our resistor value by deciding on a current limit. We limit the current to 1A to keep the start-up time of the motor low but still able to draw enough current in the case we need to run the motor at max power (~0.81A). The current being limited at 1A ensures that the motor drivers are never draw more current than the voltage regulators can supply as well. With a current limit of 1A, our current sense resistors would be around 0.2 Ω (or 200 mΩ).

There are some capacitors used in the implementation of the motor drivers, but they are specified by the manufacturer in the data sheet. The capacitors at the motor input VM, VINT and across VM and VCP are given by the datasheet and are meant to be implemented as specified. The rest of the pins are our digital logic input pins used to control motors and the voltage outputs used to drive the motors. The basic schematic for the motor driver is shown in Figure 39.

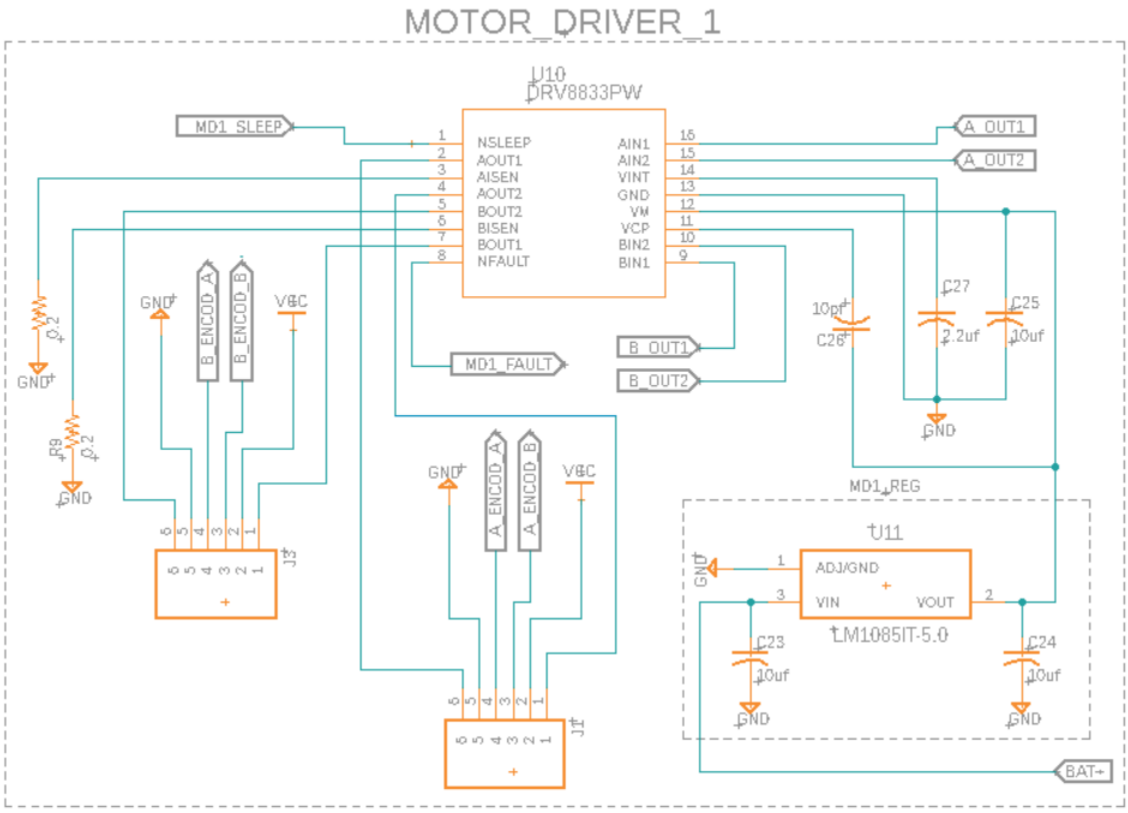


Figure 39 – Motor Driver Schematic [F39]

The way we chose to turn our rover was by having the wheels on a given side turn while the others are stationary. Since the DRV8833 has 2 channels, that means we can drive both wheels on one side (left or right) together from the same driver. This lets us move the motor driver's outputs to far sides of the power delivery PCB where they can be closer to the motors, they are powering. We know that to turn, we fully utilize one side of the rover. This means that we could tie the xIN1 pins to each other and the xIN2 pins to each other. Having both channels controlled by one pair of outputs from the MCU ensures

that the motor are always moving in sync and reduces the required connections to the power delivery PCB.

5.4 – Processor Pin Assignment Design

Below is the final layout of our pin assignment. With the removal of the motor encoders, we no longer have PWM timers set, however these pins were redesignated to XSHUT pins for the IR sensors.

Pin Name	Signal Name	User Label
PE3	GPIO_Output	XSHUT3
PE4	GPIO_Output	XSHUT2
PC14	RCC_OSC32_IN	
PC15	RCC_OSC32_OUT	
PF6	GPIO_Output	XSHUT1
PF7	GPIO_Output	XSHUT8
PA0	UART4_TX	
PA1	UART4_RX	
PA2	GPIO_Output	Motor1+
PA3	GPIO_Output	Motor1-
PA4	GPIO_Output	GPIO1
PA5	GPIO_Output	Motor2-
PA6	TIM16_CH1	Servo
PA7	GPIO_Output	GPIO2
PC4	GPIO_Output	GPIO3
PB0	GPIO_Output	GPIO4
PB1	GPIO_Output	Relay
PB3	GPIO_Output	Motor2+
PF14	I2C4_SCL	
PF15	I2C4_SDA	
PE9	GPIO_Output	Motor4+
PE11	GPIO_Output	Motor4-
PE13	GPIO_Output	Motor3+
PE14	GPIO_Output	Motor3-
PB15	GPIO_Output	GPS_INT
PD12	GPIO_Output	XSHUT7
PD13	GPIO_Output	XSHUT6
PG6	GPIO_Output	MAG_INT
PC6	GPIO_Output	XSHUT5
PC7	GPIO_Output	XSHUT4
PA11	USB_OTG_FS_DM	
PA12	USB_OTG_FS_DP	
PA13	SYS_JTMS_SWDIO	
PA14	SYS_JTCK_SWCLK	
PB6	I2C1_SCL	
PB7	I2C1_SDA	

Table 13 – MCU Pin Assignment

5.5 – Overall Schematics of Systems

5.5.1 – Power System Overall Schematics

We can see an overall schematic for all the implemented voltage regulators and motors drivers in Figure 40 and Figure 41 respectively. The basic design for the power subsystem is the uses 5 regulators and 2 motors drivers. It interfaces with the battery, MCU and the motors and will need the proper plugs for each of them to ensure connection that can withstand the expected forces form the rocket.

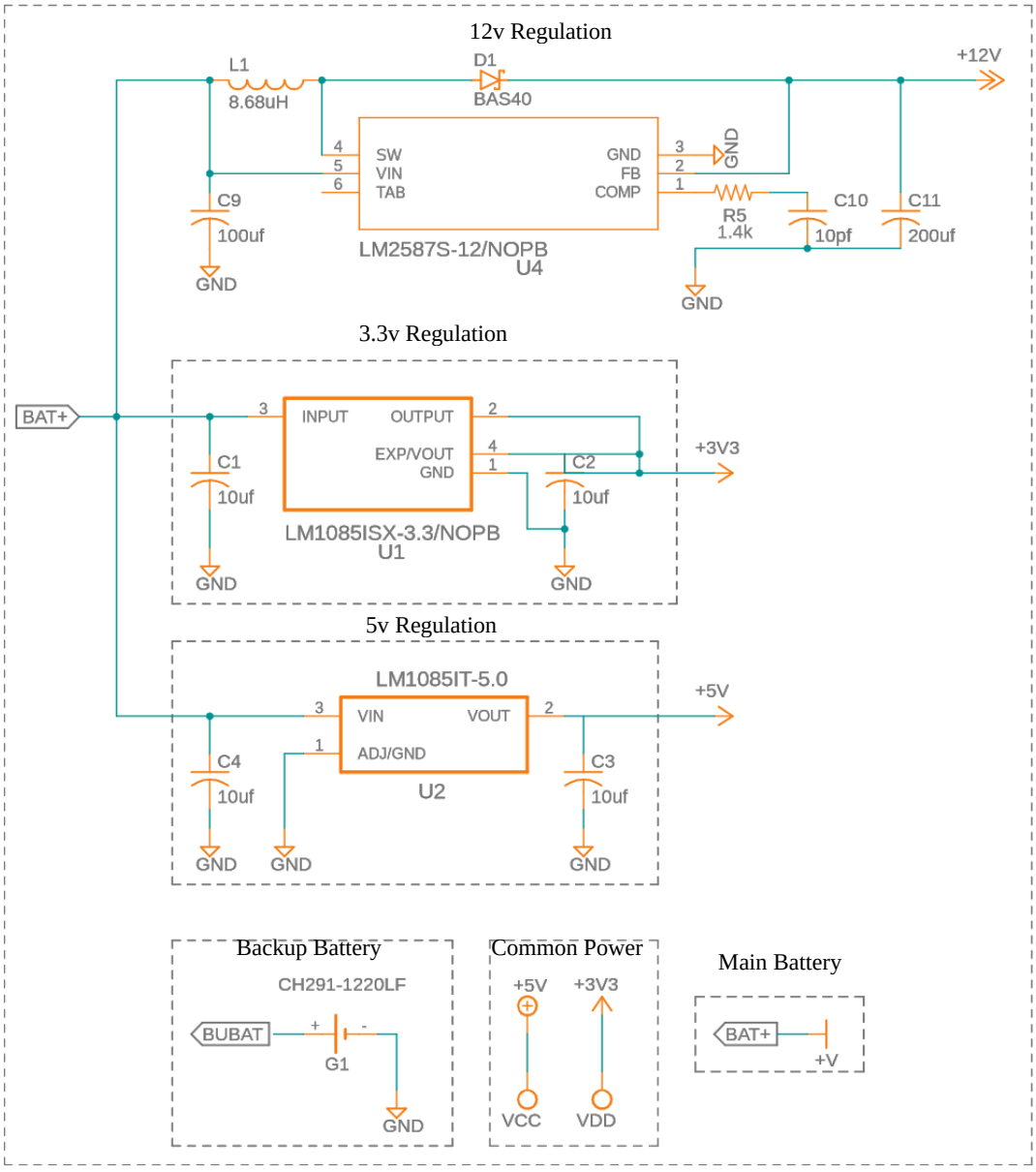


Figure 40 – Power Delivery Subsystem Schematic [F40]

5.5.2 – Drive System Overall Schematics

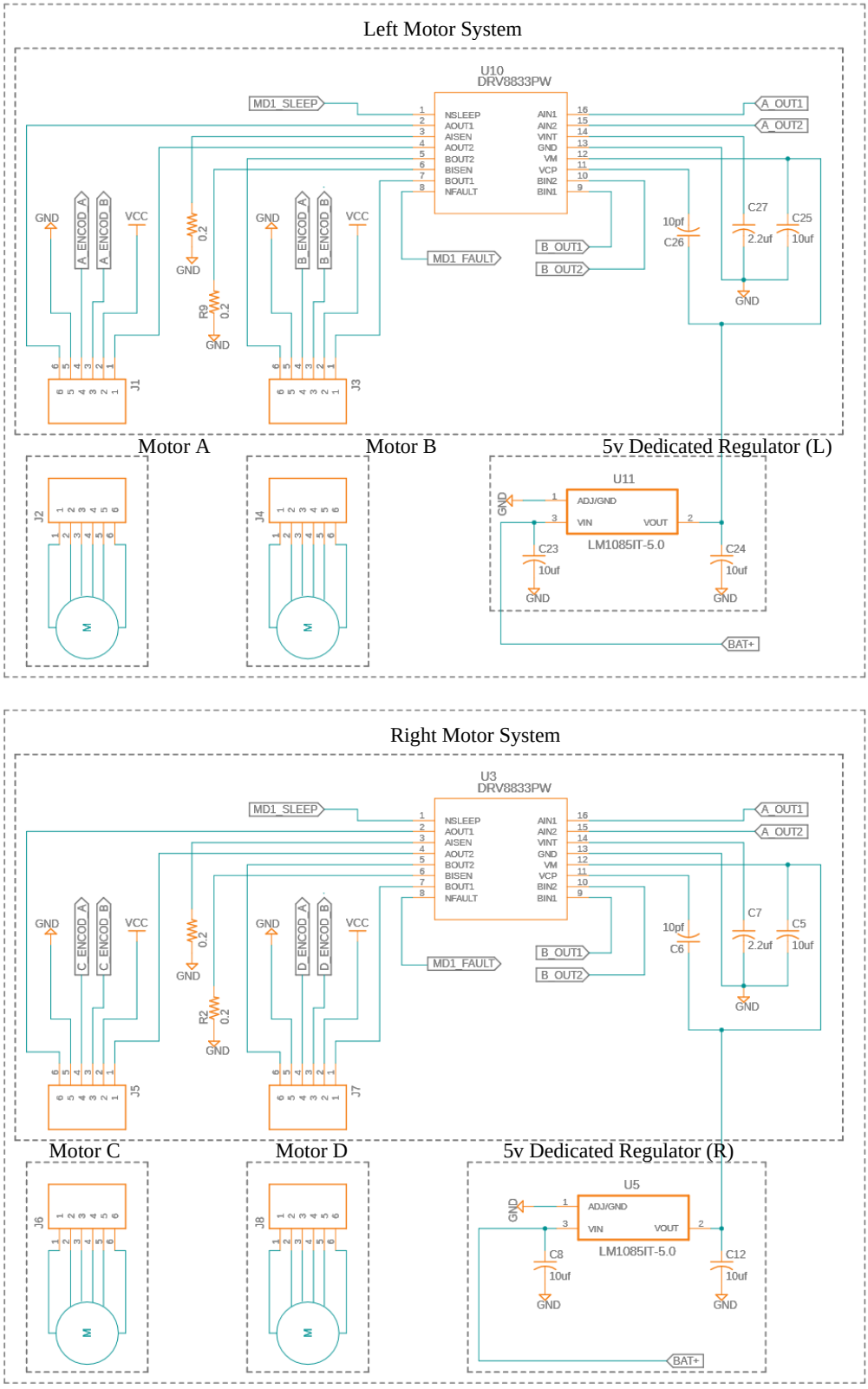


Figure 41 – Drive Subsystem Schematic [F41]

5.5.3 – Sensing and Object Detection System Schematics

Figure 42 shows that each sensor will be connected to the I2C (SDA/SCL) line running from the MCU. Important details to note is for the gyroscope/accelerometer; the VDD needs to be dropped by 0.3V according to their datasheet because the interface voltage needs to be VDD + 0.3V. For the rest of the components, we used mostly what each datasheet recommended for I2C. It should be noted that there will be seven other laser sensors wired the same way as the single sensor however we showed one here since the rest was redundant. The connections to the MCU will be fully shown in the integrated circuits section of the paper [Section 6.1 – Integrated Schematics].

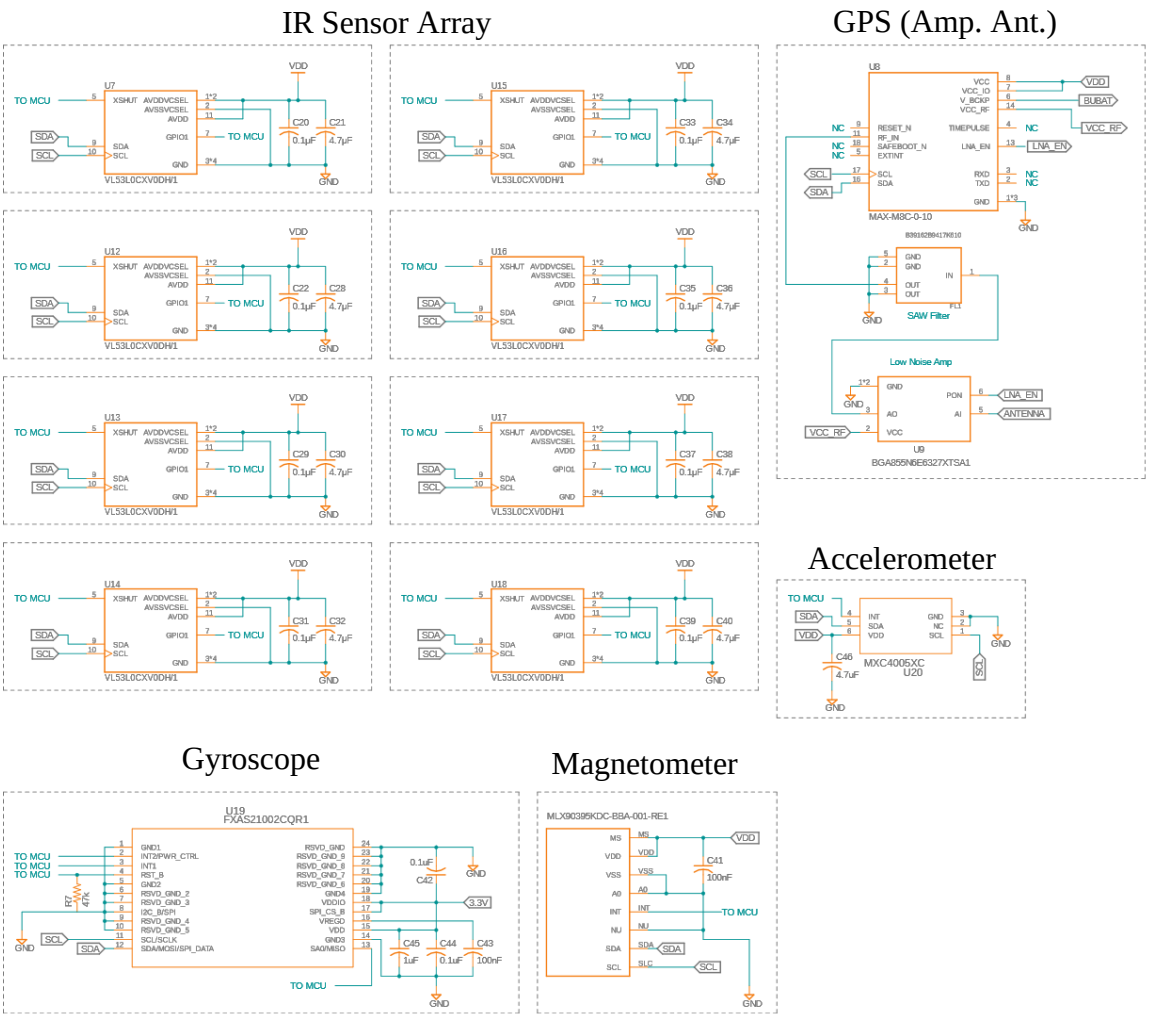


Figure 42 – Object detection overall schematics. [F42]

Featuring 8 IR sensors, the 3 sensors which make up the IMU (Accelerometer, Gyroscope, Magnetometer, and the GPS connections.

5.5.4 – Computing System Overall Schematic

The processor overall schematic is shown below. The notable features are the 32.768 KHz oscillator, decoupling capacitor array, I2C communication lines, programming jumper, reset function, and GPIO lines attached to the respective pin functions stated in (Figure 43). Please note that the IC package used is LFQP-144 pins and the resulting symbol for the STM32F303ZET7 is split into three symbols:

COMPUTING SYSTEM OVERALL SCHEMATIC

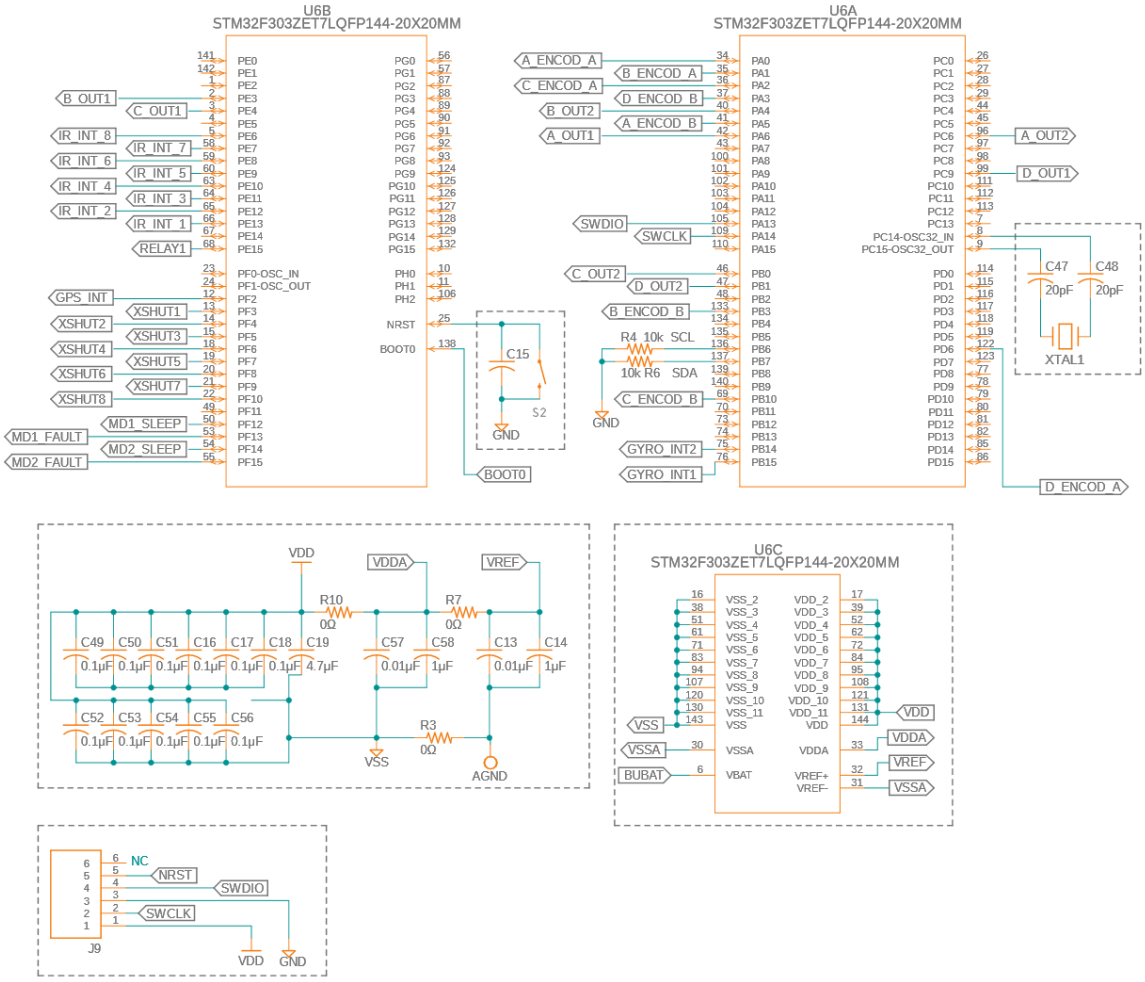


Figure 43 – Computing system overall schematics [F43]

***Top Right** – GPIO Pin set 1, which hold ports A, B, C, and D*

***Top Left** – GPIO Pin set 2, which hold ports E, F, G, and H, **Left Bottom** – External programming jumper*

***Left Middle** – Decoupling capacitor array, **Right Bottom** – MCU input power*

5.6 – Software Design

The software for our project runs on the MCU on our PCB. This means that our single program will oversee controlling our movement system, retrieving sensor data, miscellaneous controls around the rover, and data processing. Although this is a lot of tasks for a single chip to handle, after rough estimates on processor needs, we believe that the software should have plenty of time and storage for all tasks listed.

Movement Software Design

After researching the multiple methods, we could implement to move and turn the rover, we decided on a 4-wheel drive system that will be able to turn our rover without use of any steering devices in the frame.

This is relevant to the software design as our software will instruct each motor which way to move and how fast. In this case, each wheel will have a motor attached to which means our software will drive four motors. For the example seen in Figure 44, our program will tell motor 1 and 3 to move back while motor 2 and 4 move forward. This will result in a counterclockwise motion and will act as our rover's ability to turn left.

VISUALIZATION OF ROVER TURNING

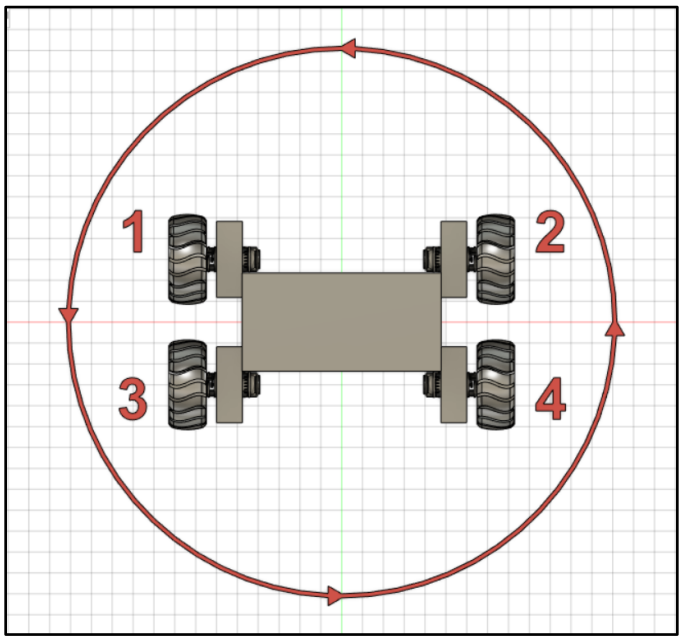


Figure 44 – Visual of what parts the software will control for turning [F44]

Since we initially had our rover as being longer than it is wide, we expected there to be a little deviation to the circular path because of some wheels getting dragged. We were able to account for this with trial and error as we did not use the encoders in our final build. In the best-case scenario, the wheels would have turned together without dragging, but they do drag and skid while turning. We were able to vary the speed of the rotating

motors, via motor feedback to the software, to make up the difference between expected and actual results.

With the turning of the rover covered the movement of the rover is significantly simpler. There is some time to travel decided on by the software or hardcoded in by us. The speed of the rover can be decided by the software depending on factors outside of the motor control. While moving, the rover can be stopped by the software at any time if it senses that the rover may collide with an object or if it's calculated the rover is unable to turn without collision.

Sensors Software Design

The software oversees collecting sensor data and processing and/or executing on that data if needed. The sensors will be in the same three groups as listed above in the GPIO section, that is distance, inertial measurement, and GPS. This section of the sensors will be primarily to explain how the data will be collected and then prepared for the logic section of our program.

For distance data, our software will collect from these multiple times a second. This will be done using I2C and will collect the distance of any object in eight specific directions in respect to the rover. This data will be converted to an integer for our program and be used not only in respect to their value, but by what sensor they came from as this will dictate the direction of the object in respect to our rover.

The sensors used for inertial measurement will not be used as often as the distance sensers since we only need to know where the rover is when we go to make decisions on where to move next. These three sensors will also be connected though I2C to the MCU which means our code will also involve fetching these measurement values. The accelerometer, gyroscope, and magnetometer will each give the program three float values through I2C which will be in respect to the three axes of a graph, that is XYZ.

This data alone was not very valuable to us, so the goal was then to convert the data into values such as roll, pitch, and yaw. The way we did this in code is by following the steps laid out in Figure 45.

After these calculations are completed, we are then finally able to use the roll, pitch, and yaw values as integers in our code. These values help us figure out where we are compared to where we were in a local sense.

The GPS system is connected to the I2C line as well. The data that is collected from the GPS chip is the latitude and longitude of our rover. This data is converted to integers for the program. The purpose of this data was to figure out our location in a global sense while the IMU system is gaining the local location. We use this data mainly to compare where we are currently to the destination of our rover.

There are a few electronics that our software must control through either turning them on or off. Most of this work is done during start up or during shut down.

The biggest thing our software turns on and off is the live video transmission subsystem. The rover was supposed to have live video after rover deployment until the end of the rover's mission. This means when our rover gets out of sleep mode after deployment, our code will send a signal to turn on the power line to the live video. In this section of code, it also must extend the transmitter antenna which is done by the powering on of a servo. Once power is driven to the live video subsystem it will power the camera by itself and send video without our program's interference.

The rest of this control design is for miscellaneous servos and motors. For example, if there was any need for a boom arm extension for our GPS, it would have been a simple extend or collapse type of control that would be controlled by the control part of the software.

CONTROL DATA DESIGN

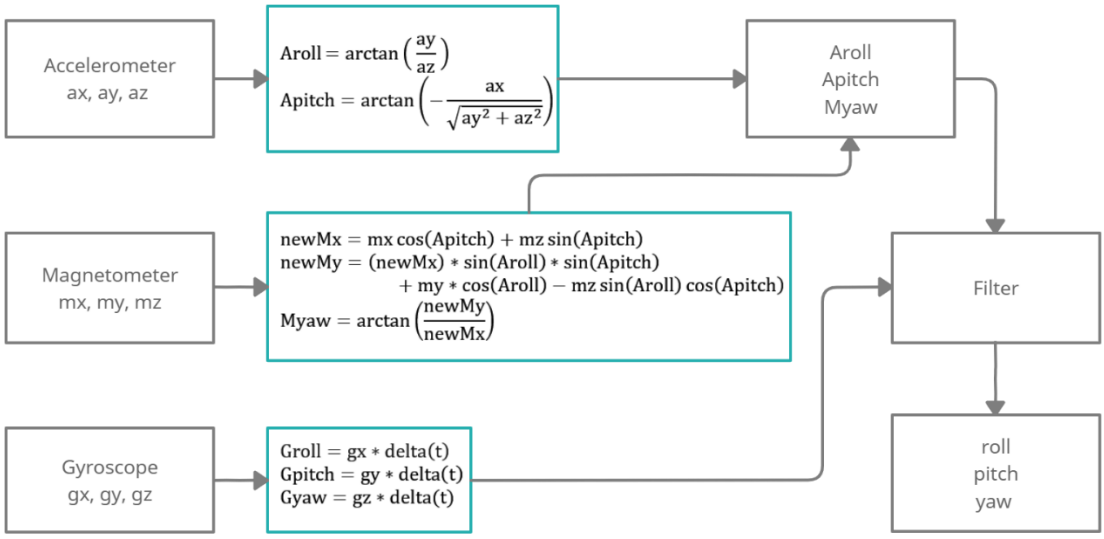


Figure 45 – Flow from input to used data from IMU [F45]

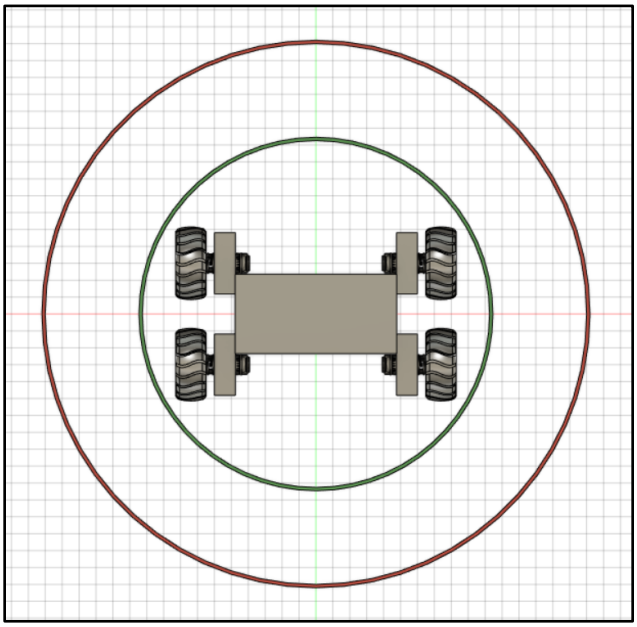
Object Avoidance Software Design

For object avoidance software we have two main algorithms, one for functionally working our rover around the object and another for close range actions that requires immediate attention. These two algorithms will also use two different distance ranges from the eight directional sensors.

The main object detection algorithm will check the sensors for distances correlating to the red circle (outer ring) shown in Figure 46. Although it's important to note that we are only checking eight points on that circle and not the full 360°, the circle is just there as helpful representation. The green circle (inner ring) will be the mission critical distance where the rover will not be able to completely turn around if any object passes that circle.

Another important detail that can be shown from Figure 46, is that no matter which way the rover turns, the same directions will be covered just by different sensors. In code this means if we have sensor 1 pointing north and we turn 45° east, the sensor that was pointing northwest will now be pointing north. However, our rover will be pointing towards the northeast so the forward-facing sensor will now be pointing northeast instead of north.

VISUALIZATION OF SENSOR DISTANCE



Red: Outer Ring, Green: Inner Ring

*Figure 46 – Visualization of Sensor Distance *Sensor distance is not to scale* [F46]*

This is significant so we don't have to wonder which directions our sensors are pointing in the code. The only thing we need to do is update which sensors are pointing in each direction whenever we rotate the rover. This will reduce the strain of calculating angles in the code and help keep everything symmetrical.

Outer Ring Algorithm

The outer ring algorithm is based on the same idea as following a wall inside a maze to find the exit. If we sense an object in front of us, we can follow the edge of the object and go around it to get back on track to our destination. A good example of this is if we hit a circular rock on our straight path then we can go to the left or right of the rock and follow its edge until we eventually end up back on the same line but on the other side of the rock. The image in Figure 47, shows the flow of the algorithm.

The starting of the outer ring algorithm is whenever we are moving in a direction and hit an object for the first time. Detecting an object will move us from the general control loop that is going straight to the destination and place us in the outer ring algorithm until the algorithm is finished.

The first thing we do when we enter the algorithm in code is save our current position and save the direction that we initially sensed the object. The direction we initially sensed the object in will be the start of our looping algorithm to move to the first sensor that does not have an object sensed in the outer ring region. This direction will be noted in the code as one of the eight directions we have sensors pointing.

OUTER RING ALGORITHIM

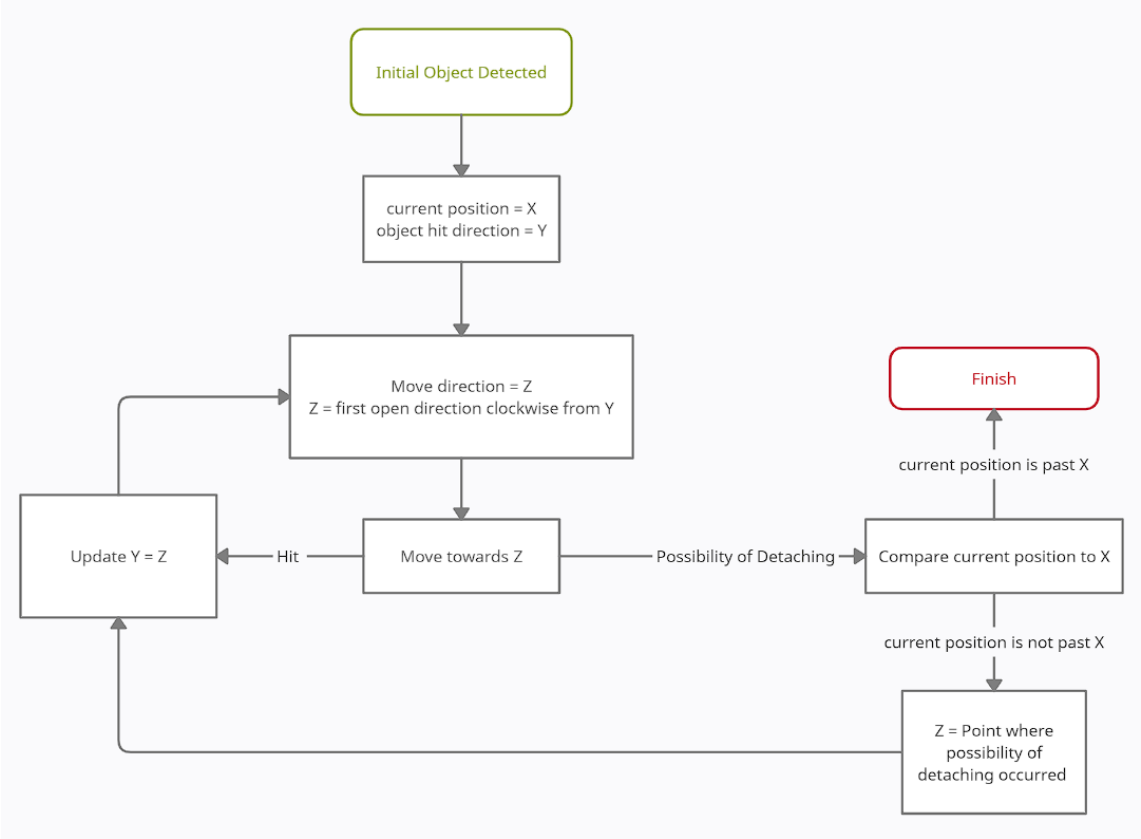


Figure 47 – Flow of Outer Ring Algorithm [F47]

The position data that is saved should be in terms of some vector <x, y> and should be orientated in the direction of the object. This vector will be calculated using either GPS or IMU data whichever one we find to be more accurate during testing. This position data will be used to tell if our rover has gone past the initial point of contact, which would mean we are ready to detach from the object and start going towards our destination again. This idea can be seen in Figure 48.

The first encounter is shown with the red circle, encountering the object; this is the initial location data (position) it saves. It is clearly seen that when the final rover position goes to detach from the object it is farther ahead of the initial rover position.

After the initial data is captured, the program picks a direction clockwise from the initial point of contact. Always choosing a clockwise position from the point of contact will insure we are hugging the object as closely to the left of our rover as possible. We then move forward until we receive one of two interrupts. The first option is that we sense an object in the direction we are moving. The second option is we are going to detach our last contact point on the left side of our current direction.

VISUALIZATION FOR EXIT CONDITION

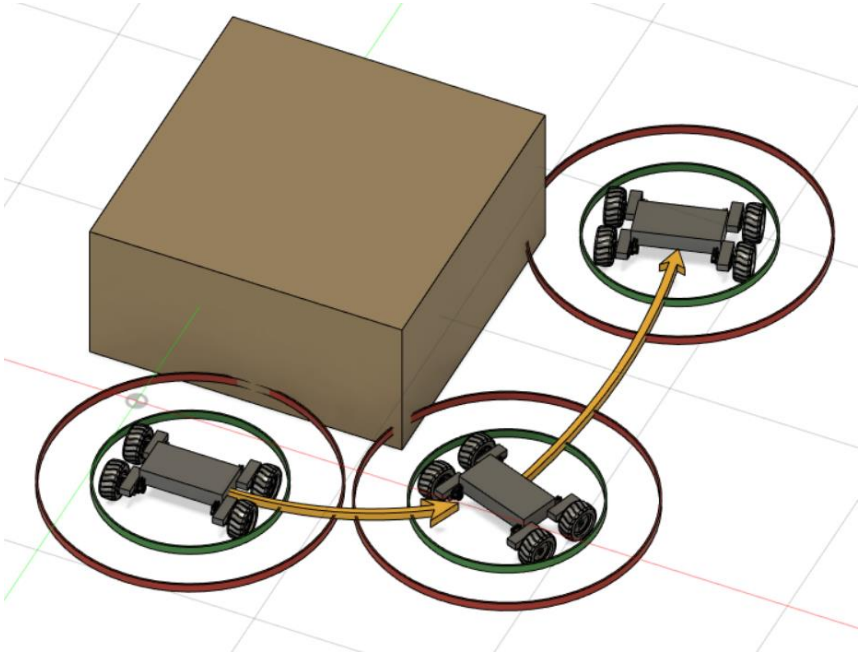


*Figure 48 – Rover starting position in the bottom left; encountering an object and traversing around. *Diagram is not to scale* [F48]*

If the first option occurs, then we update the direction of the object hit and again pick another direction with no objects in the way currently and clockwise from the direction we just hit the object at.

When the second option occurs, we know we are about to become detached from the object on the left side of the rover. Left in this sense is defined as the first three directions counterclockwise from the direction we were moving. When this happens, we compare our current position to the initial position recorded. From this comparison there is again two situations that can occur, either we are past the initial position, or we are not past the initial position.

If we are past the initial position, then this is where the algorithm stops, we return to the main control loop. However, if we see that we are not past the initial point of contact then stop moving in the direction we current are and recalculate where to go. To do this,

we update the value of last direction hit to the last point of contact on the left side of the rover and pick a new direction clockwise and open from that point.

This situation can be imagined from Figure 48, at the bottom right corner of the object. The rover would be traveling east then notice the northwest point is about to detach from the object. It then turns north to continue following the object.

This point where the object is about to detach from the object is calculated and captured by the clock cycle that we use to check all eight sensors. So, when we are checking the sensors in the software, we see if there is an object that is X distance away, which may be right on the outer ring what we are sensing, so in that case it would be expected that X+1 would be outside of the outer ring since we are traveling in a direction that affects these three sensors.

Inner Ring Algorithm

The inner ring algorithm is much more simplistic than the outer ring. The purpose of this inner ring was to avoid allowing any object to enter this ring unlike the outer ring which allows objects to enter. The reasoning behind having this ring was to allow our rover to turn at any time. As stated in the movement section of software design, we have designed the rover with the intent that it turns in a circular motion about the center of the rover. The for the inner ring algorithm is shown in Figure 49.

This algorithm is triggered by one of the sensors showing a distance within the inner circle, most likely while in the outer ring algorithm code. When this happens, we calculate a new direction to move that is opposite the direction of the collision. In this case, we don't care about what is touching on the outer ring since we know that at bare minimum this sensor that sent the interrupt is also connected at the outer ring. We then turn and move in the direction we choose for some distance of our choosing for some X distance in between the outer and inner circle.

If we have another object enter the inner circle during this time, we restart the flow back at the initial block and move away from that object. Eventually, we no longer have any objects X distance from any sensors and that is when we return to the object avoidance loop to continue moving around the object.

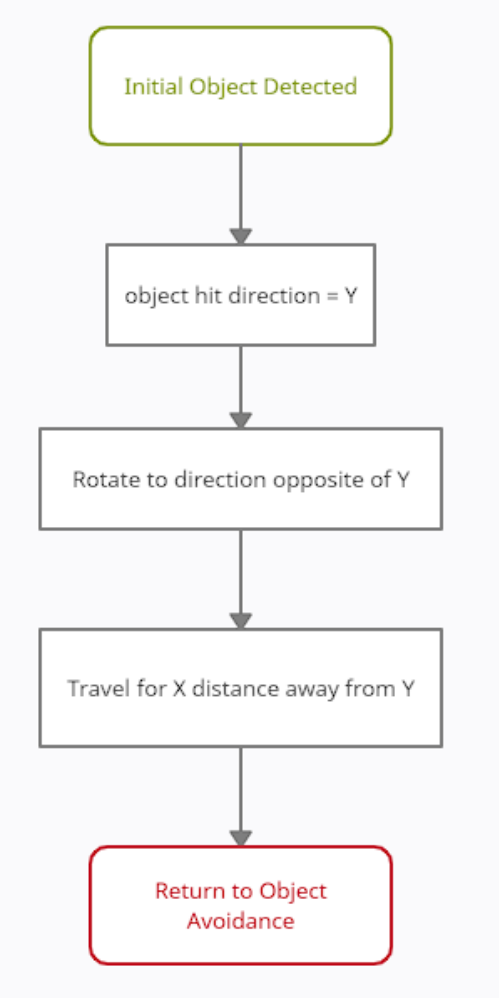


Figure 49 – Flow of Inner Ring Algorithm [F49]

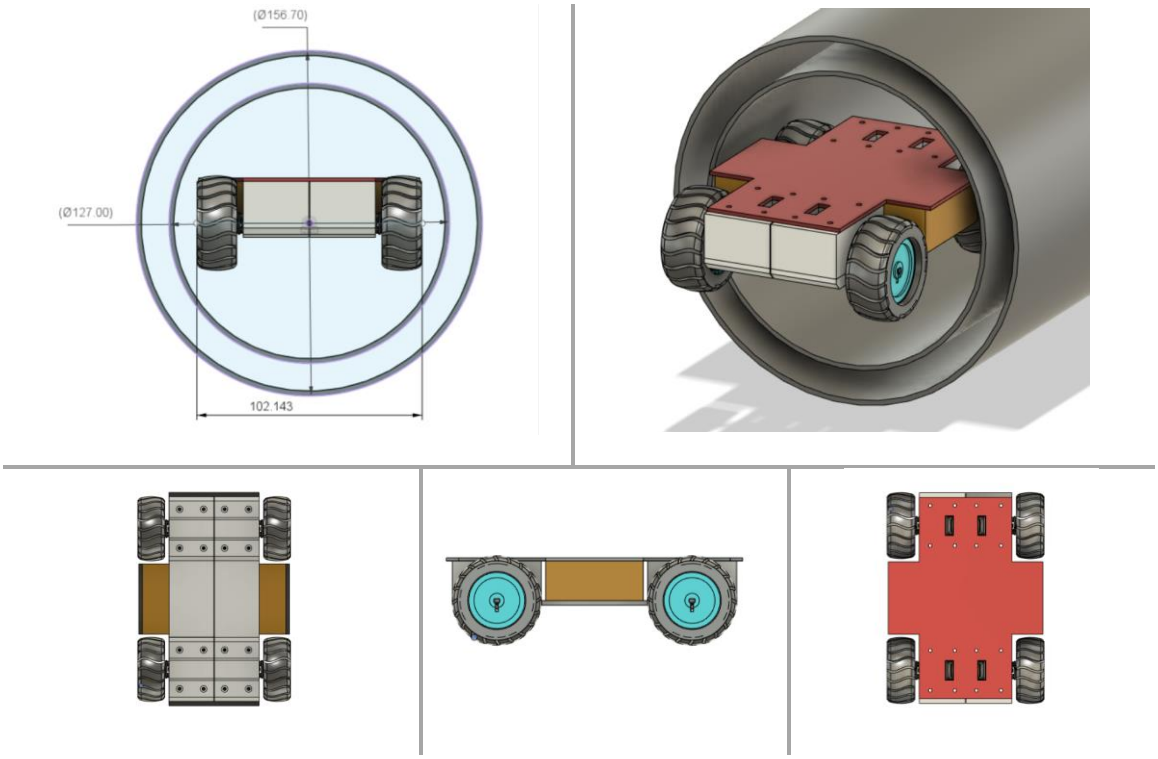
5.7 – Rover Structural Design

This section is dedicated to the initial framework for the dimensions of the rover. Although not directly related to electrical and computer engineering, the team lacks a dedicated mechanical engineer. Thus, the design of the rover and canister must be designed solely by our team of 3. An overview of the rover design will follow.

Given that the rocket hull restricts the overall sizing specification of our rover; it must be very compact. It is given by the main rocket design team that our rover is less than 12.7 centimeters wide and less than 40 centimeters long. If we wish to deploy or rover via the HALO method. It would require the rover to "roll out" of the canister as it falls vertically, this method entails a forward-facing design. The rover must also be held inside a canister. The geometry of putting a rectangle in a circle involves sacrificing either width or height to achieve maximum of either parameter.

The minimum possible size of the rover without the battery came out to a square with dimensions of 88 x 88 mm (motors and 43 mm wheels only). With the battery in place to rover has a width of 102 mm Since the motors have exposed micro gears and an exposed hall effect encoder a custom backet has been designed to hold the motors snug, braced to the main PCB plate holder, and ultimately protecting the exposed parts from the elements.

ORIGIINAL ROVER SIZING



Red: Plate to hold PCB, Yellow: Battery, White: motor covers

Figure 50 – Original Rover Chassis Design
*[F50] *This is just one design option. Deployment sled not shown. Final design may differ**

6 Project Prototype Construction and Coding

6.1 – Integrated Schematics

Figure 51 shows all the subsystems and their interconnections.

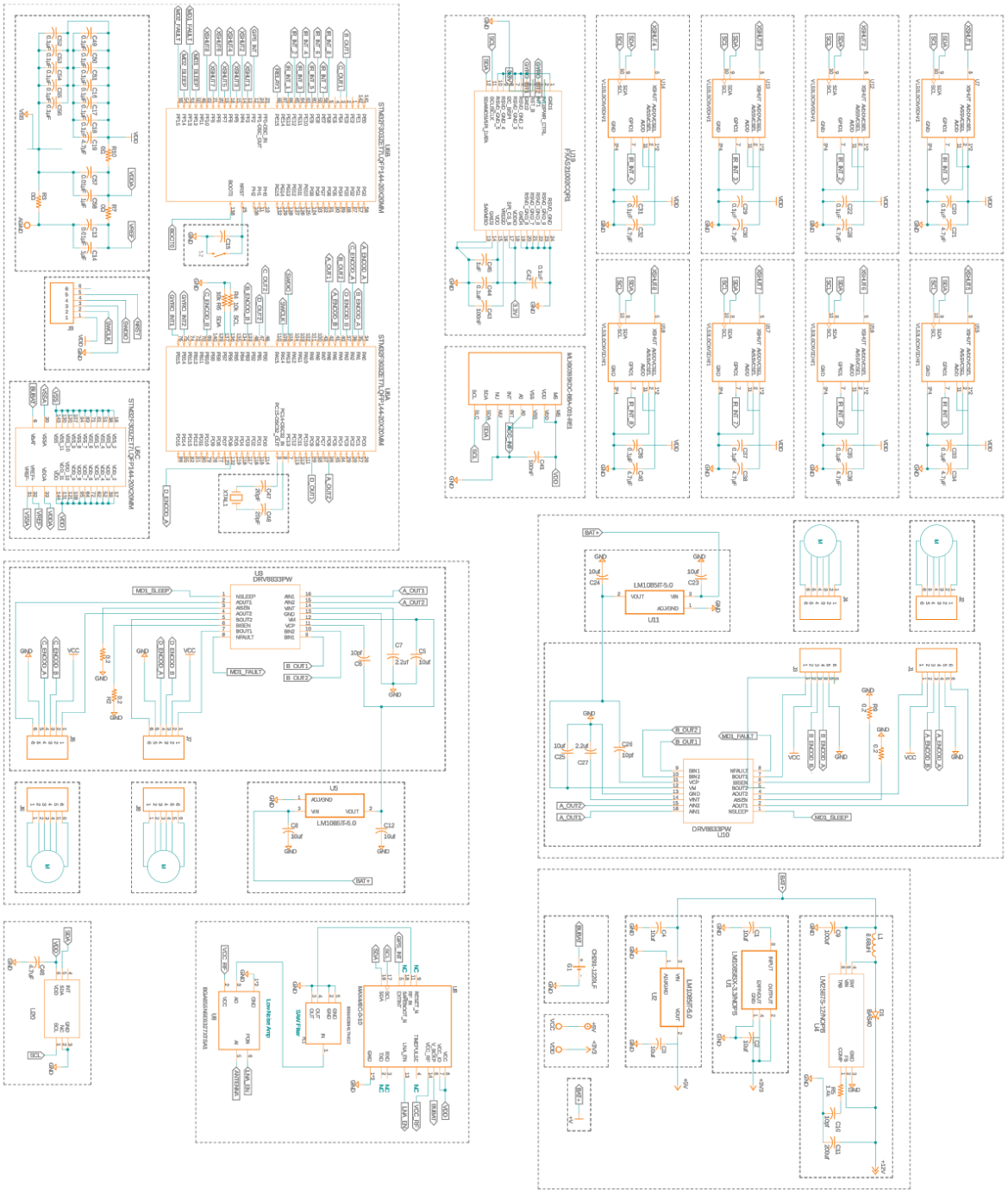


Figure 51 – All rover systems and their respective connections. [F51]

6.2 – PCB Vendor and Assembly Plan

The PCB vendor that we chose to print our designs was JCL PCB as they can provide the board specifics were looking for as well as providing us a surface mount technologies stencil (SMT stencil) for our design. They are fully based in the United States and can deliver our printed designs and stencils within less than a week after ordering. The price for a two trace-layer PCB is the same as the one trace layer which lets us take route our traces in a cleaner fashion.

Our board designs are not complex enough to require anything more than 2 layers of traces in the PCB. Using a 2 trace-layer PCB lets us reduce the size and cost for the PCB. We can only order a minimum of 5 copies of the PCB designs which isn't a problem considering that they cost 2 dollars for the minimum size of 100mm by 100mm. Given the conservative estimate of 50mm by 50mm for our PCB designs, we would have more than enough space to mount all our components.

The SMT stencil was our main method for mounting our components as the majority of our implemented passive circuit components were surface mount parts as well as our IC chips. This let us reduce the footprint as well as the profile of the PCB design. The more compact our PCB designs are, the smaller we could make the rover overall which let us fit into our frame easier. The frame we wanted to use would have been made of metal so we would have needed standoffs for mounting the PCBs on the frame. The standoffs already increase the profile of the PCB, so the shorter we could make everything, the better it would have been to fit the rover in our size constraints.

The assembly procedure for our PCB designs relies on the SMT stencil as the main mounting step. The connections between PCBs were left to soldering as we were not able to implement all our designs in a single PCB. We needed to ensure that the rover wouldn't have any broken or severed connections, so soldering was our best bet to keep a solid connection through the launch, apogee, and detachment stages. The rover would have been under multiple different forces during those stages and even a single severed connection could have meant failure for a subsystem or even the whole system.

JCL PCB includes assembly for free with their design printing services but it requires us to purchase the parts we wanted through them. We instead utilized the SMT stencil design and flowed the board ourselves as we could not ensure that we would get the components we required through them. We could acquire the ICs we needed through the other vendors and mount them ourselves.

6.3 – Final Coding Plan

Our final coding plan is as follows: For the overall layout of our code, we planned to have multiple generalized functions that can be called to perform a specific task then return to the function that called it. Besides this modular approach, we have separated the software into general rover control and object avoidance algorithms. General rover control encompasses movement, positional sensor updates, and logic of moving towards the end destination. Object avoidance consists of the detection of objects and the algorithms used to avoid the object.

Next, we list the functions we were sure to have and what they accomplish. Note we left the software flexible during development purposefully so we could make changes on the fly. However, there are core functions such as "MoveForward()" that must have been implemented in one way or another.

First are movement functions. This includes moving forward, turning right, and turning left. These functions take in parameters such as movement time, direction (if turning), and speed. The function then sends information to the motor controllers based on the input. The function returns once the rover has completed its movement.

Next would be the positional update function. This function collects the GPS and IMU data. For the GPS data it calculates longitude and latitude from the data given from the GPS. For IMU output it calculate roll, pitch, and yaw and store those values and update run any update functions that require up-to-date GPS and IMU data.

The object detection is split into two separate functions, that is inner and outer ring functions. The outer ring function is a control loop algorithm that starts when an object is detected at the outer ring distance. This function then runs through the algorithm laid out in the software design portion of this paper. Completion of this function comes when the rover has gone around the object detected. The inner ring function is called whenever an object enters the inner ring distance. Again, the inner ring algorithm runs as listed in software design and then exits when it has successfully moved away from the object.

Another function that is used is a function to update the direction we want to move. For example, when we leave the object detection algorithm we need to decide where to go next. This function uses GPS information to calculate which of the eight directions, that our rover has access to or is the best (points closest to our destination) to travel. The function then returns for the control loop to turn and start moving in that direction.

Distance sensor data collection is a function used consistently in our program (approximately every 200ms). The purpose of this function was to quickly check each IR distance sensor and record the distances they provide as usable variables in our program. This function was created with the intent to be as fast as possible while keeping the accuracy that we need to confidently avoid objects.

6.4 – Board layout

Our main PCB ended up being a four-layer, compact PCB. The top layer featured most of the high-speed communication lines (I2C, USB 48 MHz), most of the power system, and the main MCU. The Top Middle layer held the 3v3 power traces that needed to run around the board to various systems. The bottom middle layer was used as a layer for the motor driver signals. Finally, the bottom layer held the position and motion sensors.

The IMU was placed on the bottom middle of the PCB. To try to isolate the sensors for possible interference from the other integrated circuits.

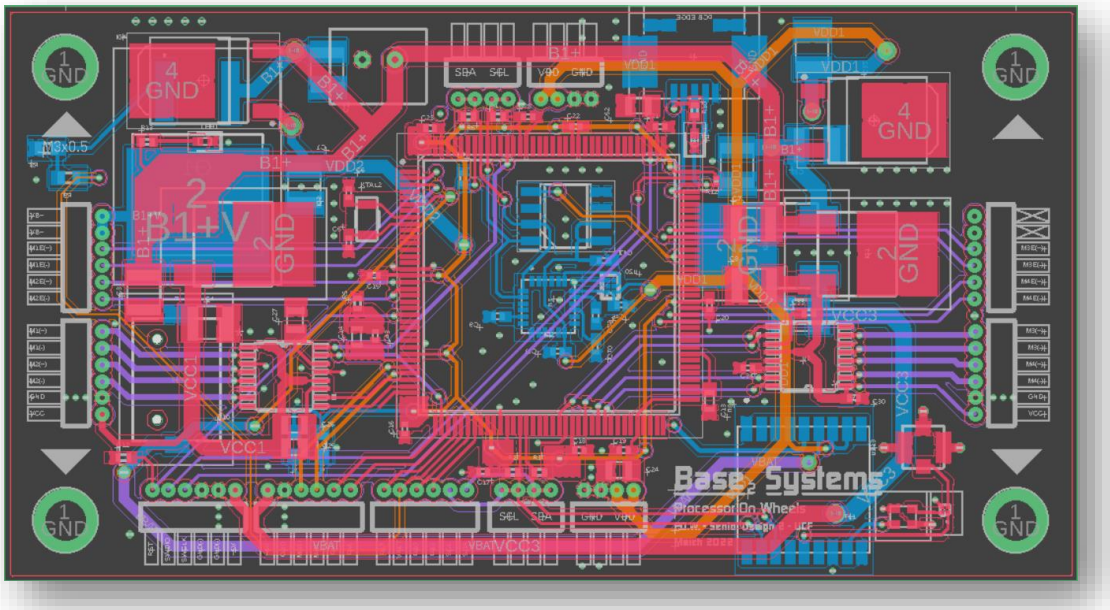


Figure 52 – Main PCB Layout. 4-layer PCB.

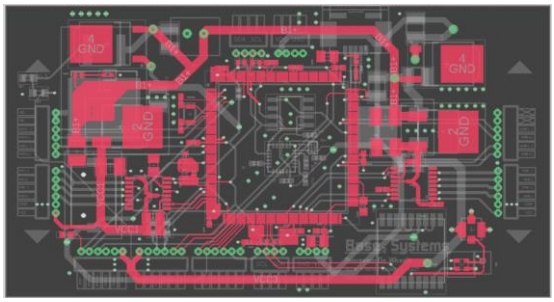


Figure 53 – Mainboard, Top Layer

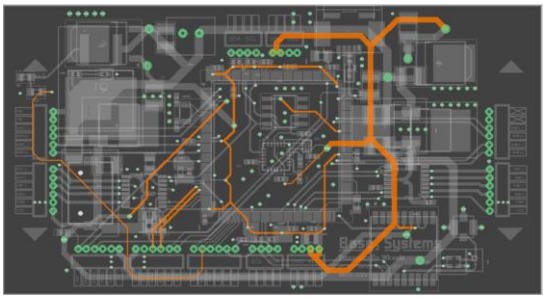


Figure 54 – Mainboard Top Middle Layer

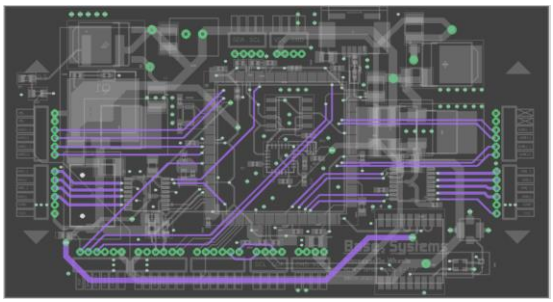


Figure 55 – Mainboard Bottom Middle Layer

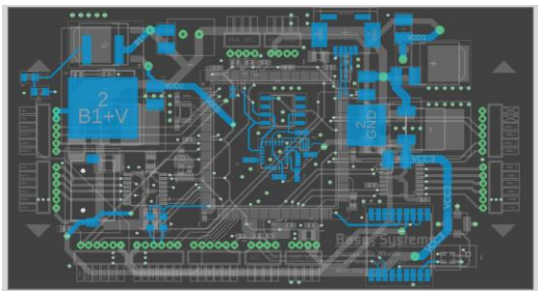


Figure 56 – Mainboard Bottom Layer

6.5 – Custom PCB Build

The main PCB and the IR sub-boards were made by JCL PCB. The assembly of these boards involved various small components such has 0402, 0201, and BGA (Ball Grid Array) package components. A digital microscope was purchased to aid in the placement of these components.

JLC PCB did have the option to purchase the stencils for the PCBs made. We decided to purchase both the mainboard stencil and IR board stencils. It turned out that purchasing these solder stencils was a huge timesaver in terms of component placement preparation. The figure below shows the IR sub board solder paste and an 0402 resistor under the microscope, ready to reflow.

Figure 57 – Custom IR PCB with solder paste and 0402 capacitor placed.

After the solder was reflowed for the main PCB and the IR Boards. The only thing left was to wire the IR sensors to the main PCB via solder cups. The pictures below and on the following page show the PCBs after components have been placed and reflowed.

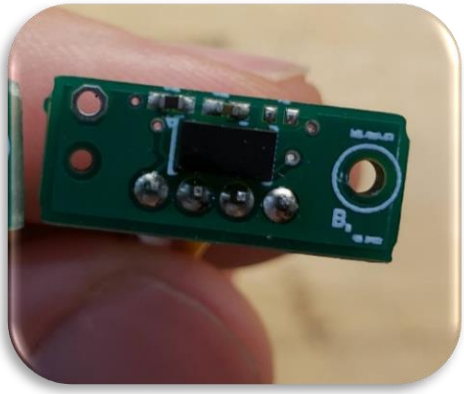


Figure 58 – Front of the custom IR sensor.

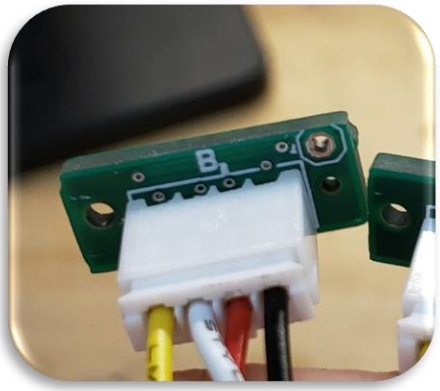


Figure 59 – Back of the custom IR sensor

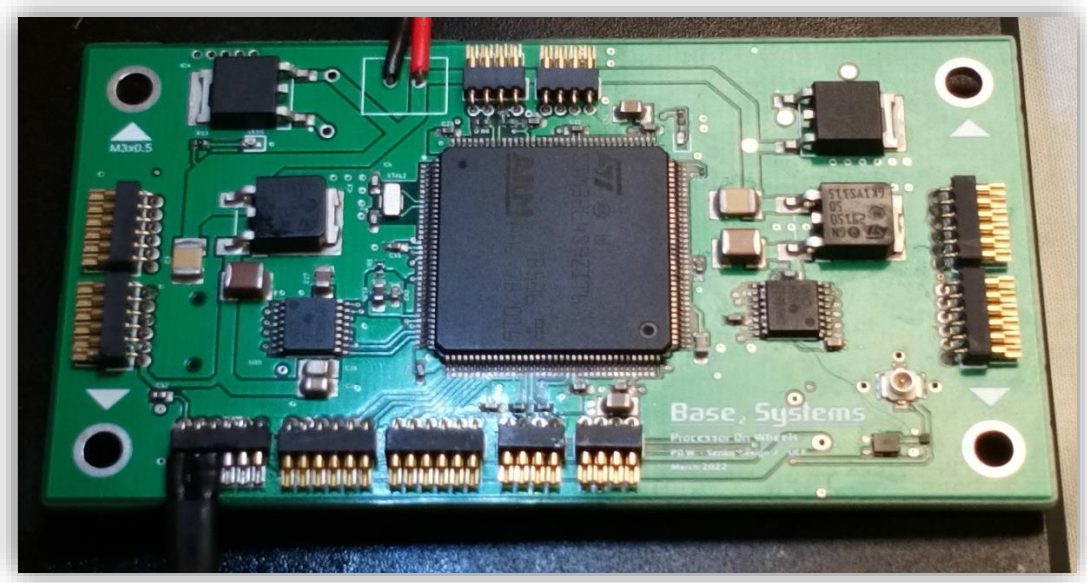


Figure 60 – Main PCB Top layer after reflowing.

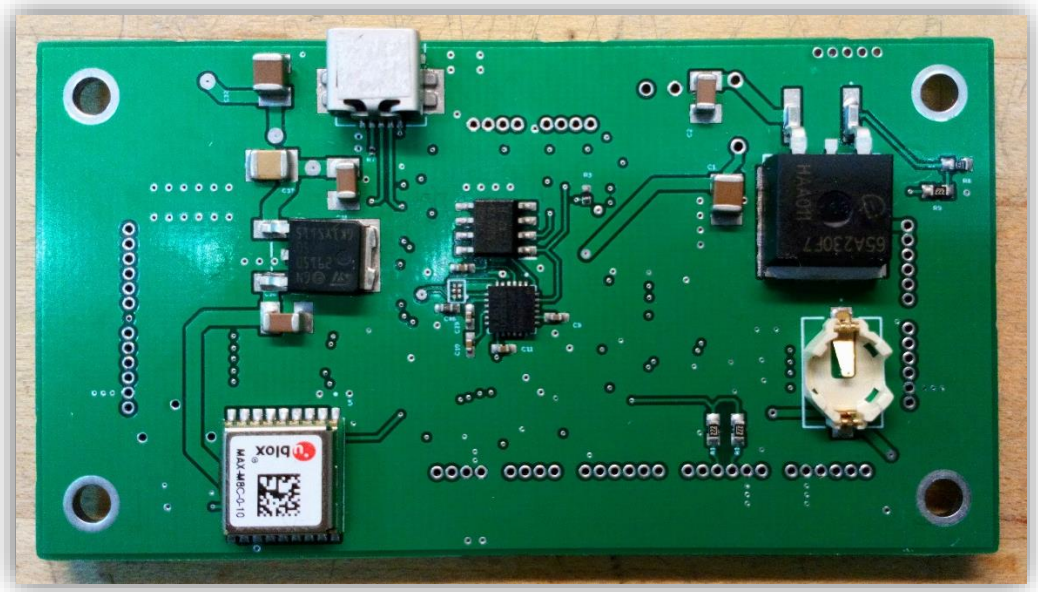


Figure 61 – Main PCB bottom layer after reflowing.

End Section 6.

7 Project Prototype Testing Plan

7.1 – Hardware Test Environment

Senior Design Laboratory:

For large scale part testing including minor component calibration, the main environment will be the Senior Design Lab at the University of Central Florida. The SD lab provides:

- Tektronix Oscilloscopes
- Tektronix Dual Arbitrary Function Generators
- Tektronix DMM 4050 Digital Multimeters
- Keithley 2230-30-1 Triple-Channel Power Supplies
- Dell Precision 3420 Computers
- SMD Rework Station
- Soldering and De-soldering Stations
- Digital Microscope Inspection Stations

At Home Testing:

Provided to the SD teams is also a handheld digital oscilloscope and digital signal analyzer which will be available for take home use if testing the smaller aspects of our project when the senior design lab is closed. Alongside these tools, a SD card reader will be placed into the design on the SPI lines for the MCU where the team can store testing data to be evaluated and analyzed in a program such as Excel, PyPolt, or MATLAB.

Structurally integrated hardware will also be tested at home. While the focus of this report has been on the computer and electrical side of our rover; we still must be able to integrate our hardware onto a rover, that is meant for a HPR launch. Some structurally integrated hardware / components will be scale and non-scaled variations of the rover subsystems that will be subject to destructive testing to obtain true limits of the requirements and specifications put on the payload team, from the main rocket design team.

7.2 – Hardware Specific Testing

7.2.1 – Drop Survivability Test

Test Objective: To test the impulse forces that wire connections and components will face during the landing event.

Test Method: Applies to the HALO deployment method. A parachute will be attached to the rover. Calculated height of drop test is about 50 - 100 feet for parachute opening distance to ground. Parachute and rover will be released from this point (potential drop sites still under investigation). Parachute deploys as normal. The rover obtains a landing velocity of less than 8 miles per hour (estimate for safe

landing). A calibrated shock sensor will be used to measure the impact force of landing.

Test Expected Result: Rover is expected to remain attached to the deployment sled. All PCB components including IC's, connected wires, as well as drive motors will still be attached and fully functional after this deployment.

7.3 – Software Testing Environment

Since most if not all our code will be on the MCU for our rover, our test environment will consist mostly of a computer connected to a development board with our MCU chip installed. For more advanced tests, like object avoidance, they will first be tested on the development board and then moved to our own PCB to be tested.

For software development we ended up using mainly C++. For testing we used an IDE like code composer or something similar, to be able to connect and run the program directly from our computer to the chip. Programming the chip like this initially provides an easy way to program and change the program whenever we want. This method also provided us with any specific MCU information that the IDE can show us. This normally means we have the ability to see the information stored in the MCU's registers. Beyond this we are also able to step through the code since our computers be in control of the MCU. Stepping through the code, line by line, was an important feature when trying to debug problems that occur throughout development.

The development board for our MCU is another important part of software testing environment. The development board allowed for quick programing of the board via the ST-Link the board provides. This also provided the debugging, step through, as mentioned above. The development board also has all the pins to the MCU easily accessible which was important for software development when we wanted to just test a specific piece of equipment and how it interacted with the program. The board also holds many input connections such as different types of USB connections as well as a 32.768 kHz crystal if we need an extra clock to help test the software. The inbuilt LEDs also provided an easy way to make sure certain parts of the code is functioning as we expect by flashing the LEDs.

After we finished testing the software using the board, we programmed a separate MCU chip using a stand-alone ST-Link device and copy the code to the new board. Using this multi-chip process allowed us to make tweaks to the code using the development board and then when we are satisfied with the changes, we can unsolder the MCU from our PCB and re-program the chip using the ST-Link device. Software-wise this was helpful as we wouldn't have to unsolder the chip every time we wanted to test or slightly change something in the code. We planned to do as much testing with the development board in a breadboard like environment before switching to the stand-alone chip environment.

7.4 – Software Specific Testing

7.4.1 – Software Communication with Hardware

This part of testing was aimed towards making sure each component that our software uses, shows up in the software correctly. This is different from hardware testing, for example, because were not checking if the IR sensor is sending correct data values we were instead checking if the values were documented in our software and accessible to be used in the code. Or in the sense of motors, we wanted to know that the software can send a command to the motor controller, not necessarily if the motor controller works or not.

Test Objective: Check that the software receives expected results from hardware

Test Method: For both input and output devices like sensors and motors respectively, have the MCU connected to the computer with an IDE and connect the device to the MCU. For input devices print their input to the IDE. For output devices send information from MCU to the device.

Test Expected Result: For input devices we expect the IDE console to provide accurate values at expected times. For output devices we expect to see the command executed on said device.

7.4.2 – Error Testing with IMU and GPS

We wanted to calculate errors within our software in respect to how much error is in the IMU and GPS values that we get from the sensors. This was checked to make sure we were calculating the correct error in real time that our software has calculated.

Test Objective: Check that error being used in software is correct.

Test Method: Start rover and let it travel in a straight line. Document initial error and final error of any sensors that we expect to hold error. After a set amount of distance stop the rover and again document calculated error of any sensors values that have it.

Test Expected Result: It is expected that whatever error our rover is calculating at X time after mission start is equal to how much error we expected to occur over that same time. This is important as over or under correcting values may lead to mission failure.

7.4.3 – Object avoidance

There were meant to be multiple tests run for object avoidance and they would have either been specifically inner ring or outer ring. Inner and outer ring tests would both be geared toward testing the algorithm and any interrupts during the algorithm.

Test 1

Test Objective: Check software correctly detects and enters object avoidance when objects appear in the outer ring range using the IR sensor data.

Test Method: Place an object in front of the rover. Through debug console check to see if the rover enters the object avoidance section of code when it enters the correct range.

Test Expected Result: Rover will enter outer ring portion of the object avoidance algorithm.

Test 2

Test Objective: Check if the object avoidance algorithm runs to completion.

Test Method: Place a square object in front of rover and start the rover. Observe the rover to see if it moves around the object and finally reverts to moving toward the destination again.

Test Expected Result: Rover should follow the square object around, never entering the inner ring part of the algorithm since we are using a square object. Once rover is past the initial point of contact with the object, the rover should visibly change back to facing toward the end destination. This will be conformation that we have left the object avoidance algorithm.

Test 3

Test Objective: Check that we enter inner ring object avoidance when object is within range.

Test Method: Place and object for the rover to detect. Once object detection algorithm has begun, place another object within distance for the rover's inner circle to detect.

Test Expected Result: The rover should stop moving immediately when the object enters the inner circle, via one of the IR lasers lines. The rover should move the opposite direction from the object and then return to the outer ring algorithm exactly where it left off.

7.4.4 – Test motor control

The software should hold simple commands like move forward, turn right, and turn left. It is important that each of these commands function as expected and for the time expected.

Test Objective: Check that movement functions act as expected.

Test Method: Place rover on flat and open ground. Using IDE debugger connection, have the program enter the three types of movement functions.

Test Expected Result: Move forward is expected to move the rover forward for some determined amount of time. Turn left should rotate the rover in and counter-clockwise motion. Turn right should rotate the rover in a clockwise motion.

7.4.5 – Test initial wake up

When our rover is deployed, it is expected we receive a signal from the rocket team, and this is used to wake up our rover. From a software perspective we want to make sure the rover wakes up correctly and engages its control loop to get home.

Test Objective: Check that the wake-up function will do all control actions required such as turn on live video, record initial data from sensors, and enter the algorithm to take us home.

Test Method: Supply signal that we expect to get from the rocket team. Observe how the rover responds via debugger and in real time.

Test Expected Result: The rover is expected to start broadcasting live video to our receiver, hold initial values such as GPS, IMU, and distance-to-objects values.

7.4.6 – Test shut down

Shut down is less important than start up, however still needs to be tested. Once the rover reaches its end destination, it is important that the rover shuts down all software algorithms as we don't want the rover to wonder away from the destination.

Test Objective: Check that the rover stops moving once it reaches the destination or close enough to the destination.

Test Method: Set end destination position and allowed error. Place rover some 5-10 feet away from the destination and start the rover.

Test Expected Result: The rover is expected to move in a relatively straight line to the destination. Once its within our determined error distance, the rover should turn off.

7.4.7 – Test scheduling

We plan to code the program in all-encompassing functions, such as move forward where the software sends many commands and runs many calculations within this single function. This allows us the opportunity to add an RTOS if we feel some system of our rover is not getting the time required to function. So first, we needed to test the rover from start to finish to see if the rover requires some form of RTOS.

Test Objective: Check rover can start and maneuver through some obstacle course and end at a given destination.

Test Method: Place rover at beginning of obstacle course. Start rover and observe if any tasks are not executed at the proper/expected time frame.

Test Expected Result: We expect the rover to go from start to finish and not miss execution on any parts of our control loop. If any functions are being ignored this will tell us that we may want to try RTOS to schedule deadlines and priorities for our functions.

End Section 7

8 Final Build Components

This section of the paper will go over the final build specifications of our rover. Specifically, this section will mention any parts or components that changed due to build/ functionality/availability problems. Systems not included here can be assumed to have followed the initial design specified earlier in the paper.

8.1 – IR Sensors

We were not able to use our own custom built IR sensors as we accidentally pulled the XSHUT pin high in the design. This led to the group attempting to solder a wire directly to the board, where we initially expected a resistor to VCC to be, however this wire was unreliable and broke off with very little pressure.

For this reason, we switch to use eight premade modules that had open VIAs for each pin on the board as seen below.

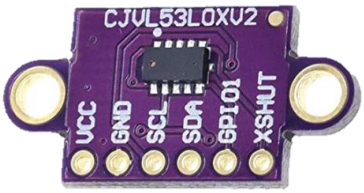


Figure 62 – Prebuilt VL53L0 module

We then soldered 30-gauge wire to each pin except for GPIO1 as we had no use for interrupt feedback. These boards were about two times bigger than our custom-made boards which meant we also had to slightly modify the rover's IR frame to accommodate the new size.

8.2 – Magnetometer

Due to our group's inexperience, we found out that the magnetometer on the bottom of the board was far too weak to pick up the earths gravitational field. Unfortunately, we had already sent this deign off to be manufactured so we had to use another premade module to make up for this. The module we ended up using is the HMC5883L as seen below.

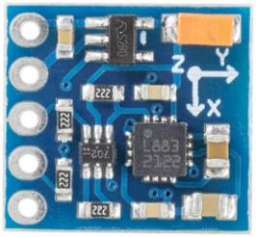


Figure 63 – Prebuilt HMC5883L Digital Compass (Magnetometer)

This module worked very well for us and was very small and easy to use. The accuracy for us ended up being close to plus or minus 2 degrees.

8.3 – Motors

Due to stock availability, we chose some N20 motors that already had encoders pre-installed and could be directly integrated with our board design. The only issue with these motors is that we couldn't find the rated power for them. We didn't know if they were low, medium, or high-power motors but our choice was limited in stock and these as an alternative were cheaper, we bought them to be used in testing. We attempted to use these N20 motors with encoders but ran into power output issues and had to ultimately replace them with stronger motors. This meant removing the encoder functionality from the rover design and relying solely on sensor data to track the rover's movement.

8.4 – 12V Boost Converter

The 12V boost converter system was an initially part of the power system but was removed as its inclusion would greatly increase the size of our PCB design. There were also concerns of thermal conditions inside the chassis with a boost converter and the only system being powered from this was the video transmission subsystem. In the end, the 12V boost converter and its design was omitted from the design of the rover as the video transmission system was functional at the battery's nominal voltage. The downside of this was the possibility of losing range if the input voltage for the video transmission was too close to its lower limit.

8.5 – Video Transmission System MOSFET

As the 12V boost converter was removed from the design, we lost some control over being able to switch the video system on or off from the MCU. The solution to this was to use a MOSFET as a basic relay for the video system. Instead of connecting the battery directly to the video system, we used the MOSFET as a switch that could be controlled by the MCU's logic outputs. This means that we can turn off the video system while the rover is asleep and docked within the payload capsule. After the rover is woken back up and deployed, we can turn the video system back on to start transmitting the video data. We decided to use a power MOSFET with a gate-source threshold voltage of 4V and a continuous current rating of over 1A. We ended up choosing the IPB65R230CFD7 power MOSFET which is overkill for our design, but it was the cheapest FET that fit into our requirements.

8.6 – Final Rover Frame

The rover frame, in terms of the upper portion, had changed many times throughout Senior Design 2. In the end, our rover was comprised of 8 main pieces. The image below shows these various components. The red component holds our video transmission system and the antenna servo motor. The yellow component is the video antenna holder. The black component is the middle plate which holds the upper and lower rover frame together and supports the battery. The remaining white pieces make up the rover body which hold the motors, PCBs, and video transmission plate.

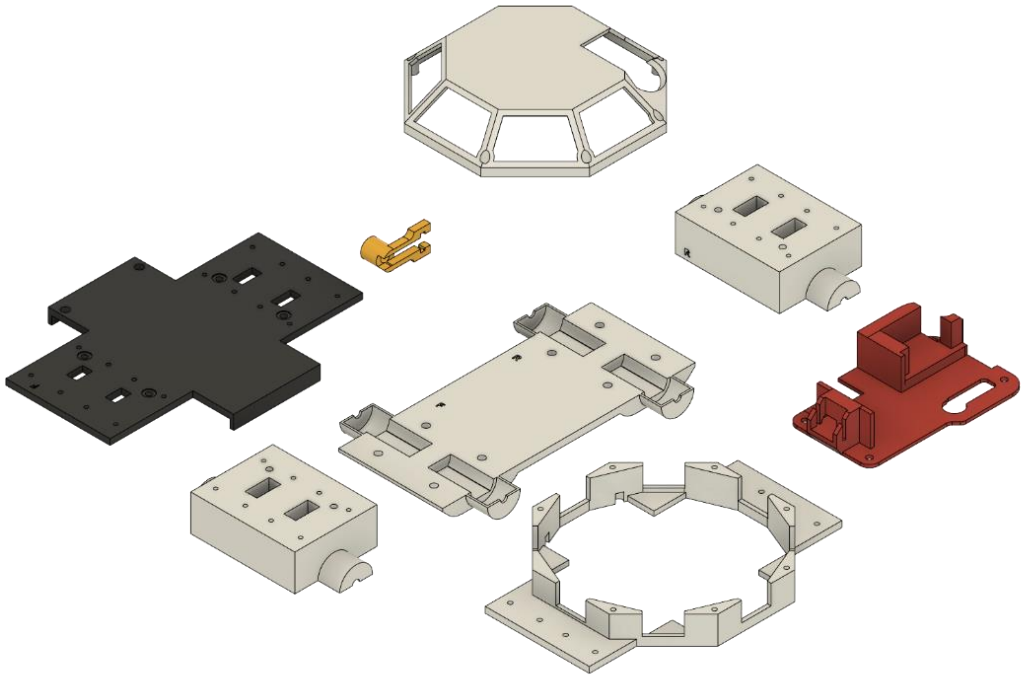


Figure 64 – Rover frame components

The frame was assembled with M2 and M4 screws. The PCB is supported by M3 standoffs which separate the video transmission plate from the main PCB. The tolerances in the rover ended up being 0.1 mm to hold components tightly. For wiring clearance, 3mm was given around the PCB to route and solder 30-gauge wiring. The pictures below show the frame in its concept stage and final prototype stage.

Figure 65 – Final Frame Concept

Shown below is the finished prototype chassis for the autonomous rover

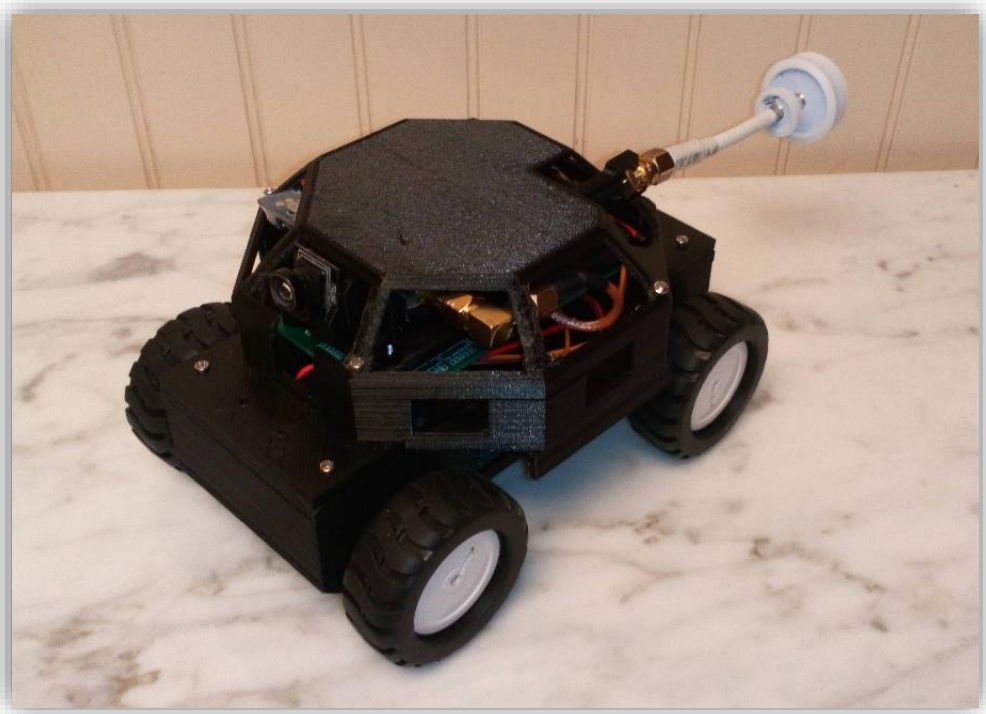


Figure 66 – Final Frame Design

9 Results and Conclusions. Changes for the Future

This section of the paper will address the results of our project and the lessons we would employ if directed to attempt this project again.

9.1 – Object Detection Results

The IR sensors we picked for this project ended up being very good and accurate sensors. If given the full 200ms the datasheet asked for, the sensors gave measurements with an error of plus or minus 4, as seen in the graph below.

Test #	Average Distance (mm)	Shortest Distance (mm)	Longest Distance (mm)
1	611	608	616
2	611	607	616
3	612	608	617
4	612	607	616
5	611	608	616
6	611	607	615
7	611	606	617
8	612	606	617
9	612	607	617
10	612	609	617

Table 14 – Average IR distance from object 2-ft away, @100 samples a test

Although the IR sensors were very accurate, they brought many problems for our team to overcome. First is the driver and API that the sensor uses. The manufacturers of the sensor do not disclose the registers that the chip uses which means you must use their API to access the chip. Unfortunately, this API was not already written with the STM environment in mind which meant we had incorporated a mixture of library searching and creating our own STM code for their API to use.

Secondly, the IR sensors were very unreliable in the breadboard phases of our project which in turn made them very hard to develop around. One of the big problems was the sensors seemed to not work in dark environments. In the initialization phase the IR boards would fail to communicate with I2C if the lights in the room were not turned on. Another problem was the connection with the I2C HAL functions. There were many times

the I2C would return a failure to communicate unless each sensor was power cycled fully before reuse.

However, after all these challenges, the boards performed very well after wired to our main PCB and built into our rover frame. We were able to reach our goal of detecting an object within 2 feet consistently.

9.2 – Live Video Results

The live video system was one of the more reliable systems in our design. We ended up taking out the 12V regulator from our design and only ended up giving the transmitter ~8.4V. Even with this drop in voltage we were easily able to achieve our 400 feet goal and even push it to 650 feet.

The system did present multiple challenges when we had to incorporate it into the build. Although we expected the transmitters size, it ended up getting harder and harder to fit it as we had to add more wires and more components to the inside of the rover. In the end we settled on a servo hoisted antenna that would initialize horizontal while the rover was not in use and then extend upward once live video was needed.

9.3 – Rover Load Results

The motors used in our design changed three times through development either because of a mistake of the team, damage to the motors, or shipping mistakes. In the end, our rover was able to move at a base load of 500g, which passed our requirement. Furthermore, we were also able to achieve our "want" goal of 1kg.

Not wanting to cause unnecessary damage to the motors, we stopped our test at about 1.3kg, at while our rover could still move and overcome the torque required for start and stop movements. This requirement was created to give the rocket team flexibility when it came to how heavy our rover was and we believe we have shown that our rover has achieved the desired flexibility.

9.4 – Conclusions

In conclusion, we were able to get the rover designed and built enough for basic rover functions. These functions include movement, object detection, antenna deployment, and live video transmission. Unfortunately, we did not have the time to finish the object detection and test the on-board GPS or IMU systems.

If asked to take on the challenge of doing this project again, there are a couple changes we would make. First, we would make the body of our rover longer. This would impair our rovers precise turning capabilities however it would give us more room for easier PCB design, easier wiring, and more possibilities for additional components. We also had plenty of room in the canister to elongate our rover so there would be no conflicts with the rocket /payload team.

Another change we would make is the redesign of the PCB and IR boards. We ran into multiple challenges with the soldering and wiring of such a small PCB and this led to many unexpected hours of work. The compact nature of our PCB also led us to using components we only picked because of their size, which led to the selection of subpar parts. As mentioned earlier, the IR boards were also design incorrectly for our purpose so we would need to redesign a board with the XSHUT pin available for use.

One last change we would make, though there are many more, would be the manufacturing of our rover's frame. Realistically we would like to aim for some light metal variant of our frame to make it more reliable if it was placed in a rocket's payload bay. Our current frame is a great proof of concept that everything fits and functions, however the PLA plastic would be questionable to use for a payload that needs to withstand high temperatures and pressures.

End Section 9

10 Administrative

10.1 – Project Milestones

SD1 Milestones

Task Name	Start Date	End Date
Investigate needs	9/1/2021	10/10/2021
Develop Eng. Requirements	9/7/2021	10/7/2021
Hardware/Software Research	9/7/2021	11/1/2021
Structural/Body Research	11/1/2021	12/22/2021
ANSI/IEEE/Standards Research	10/13/2021	10/24/2021
Choose Standards for Project (6/10 pages)	10/24/2021	12/7/2021
Divide & Conquer V1	9/1/2021	9/17/2021
Divide & Conquer V2	9/17/2021	10/1/2021
45 Page Draft	10/1/2021	11/5/2021
Obstacle Avoid Algo. Base design	11/4/2021	11/19/2021
75 Page Draft	11/5/2021	11/19/2021
Rover PCB schematics	11/19/2021	12/7/2021
Rover Structural Design	11/29/2021	12/22/2021
Final Document due	11/19/2021	12/7/2021
Section 1 Objectives	10/13/2021	12/22/2021
S1-Live Video, Research	10/13/2021	10/20/2021
S1-Live Video, Design	10/21/2021	10/28/2021
S1-Live Video, Order Parts	10/29/2021	12/22/2021
S1-Parts Received (0/3)	12/7/2021	12/22/2021
Section 2 Objectives	10/31/2021	12/22/2021
S2-Sensors, Research	10/31/2021	11/7/2021
S2-Sensors, Design	11/8/2021	11/15/2021
S2-Sensors, Order Parts	11/16/2021	12/22/2021
S2-Parts Received (0/12)	12/7/2021	12/22/2021
Section 3 Objectives	10/25/2021	12/22/2021
S3-PowerSystems, Research	10/25/2021	11/1/2021

S3-PowerSystems, Design	11/2/2021	11/9/2021
S3-PowerSystems, Order Parts	11/10/2021	12/22/2021
S3-Parts Received (0/12)	12/7/2021	12/22/2021
Define Hardware Env and Testing Plan	12/1/2021	12/7/2021
Define Software Env and Testing Plan	12/1/2021	12/3/2021
Layout Final Coding Plan	12/1/2021	12/7/2021

Table 15 – Project milestones (SD1)

SD2 Milestones

Task Name	Start Date	End Date
Purchasing parts (Round 1)	1/12/22	1/21/22
Critical Design Review	1/12/22	2/24/22
Purchasing parts (Round 2)	1/22/22	2/16/22
Project Summary	2/13/22	2/3/22
Mid Term Demo	2/15/22	3/22/22
Testing components	2/28/22	4/15/22
Frame fabrication	3/19/22	3/25/22
8 Page conference paper	3/20/22	4/20/22
Project Showcase	3/20/22	4/20/22
Breadboarding	3/21/22	4/20/22
Final Presentation	3/22/22	4/20/22
PCB building	4/11/22	4/17/22
Final document	4/18/22	4/26/22
Functionality tests	4/17/22	4/24/22

Table 16 – Project milestones (SD2)

Gantt Chart (SD1)

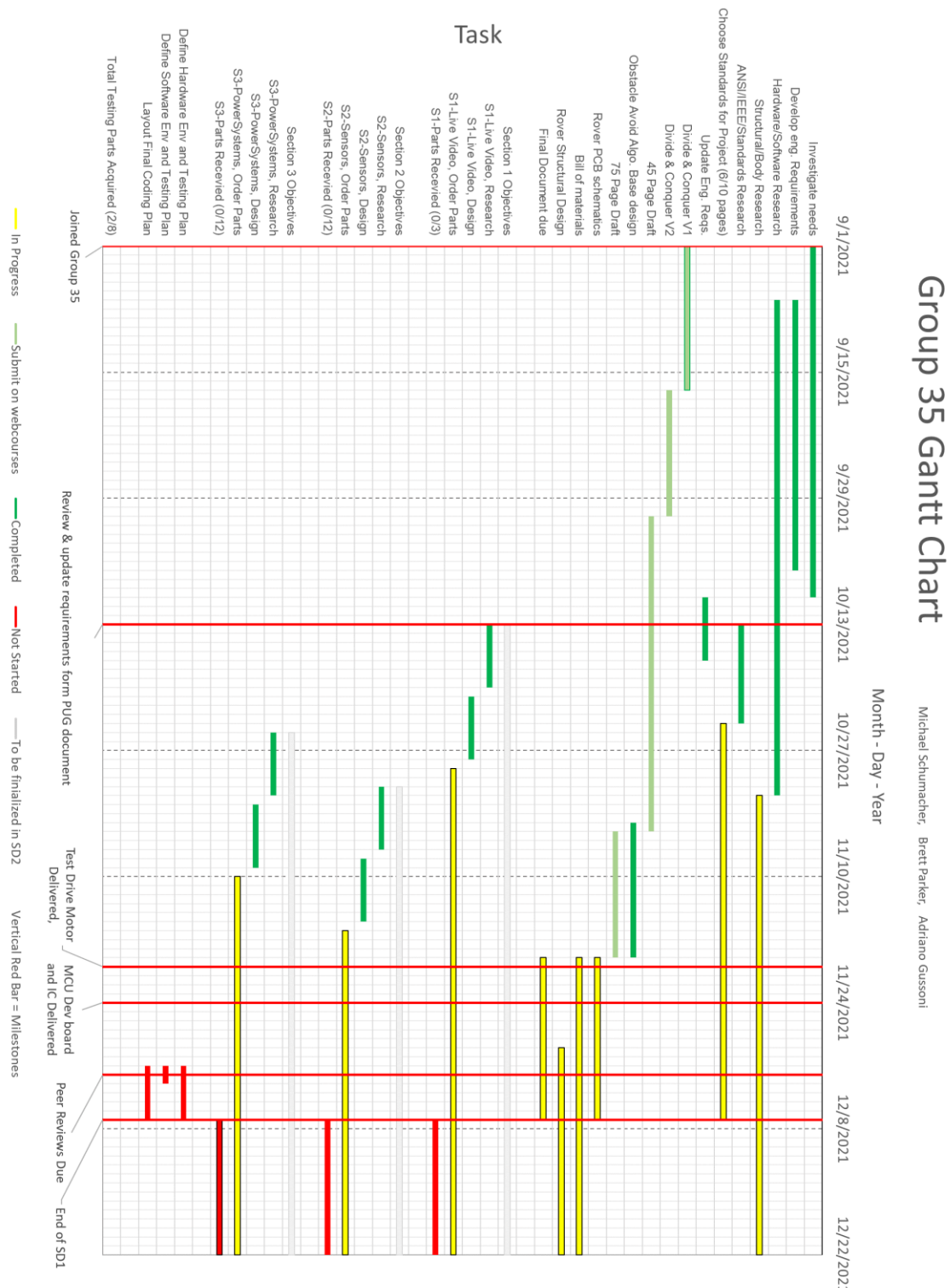


Figure 67 – Project Gantt Chart, with featured milestones

Gantt Chart (SD2)

Figure 68 – Gantt Chart 1 (SD2)

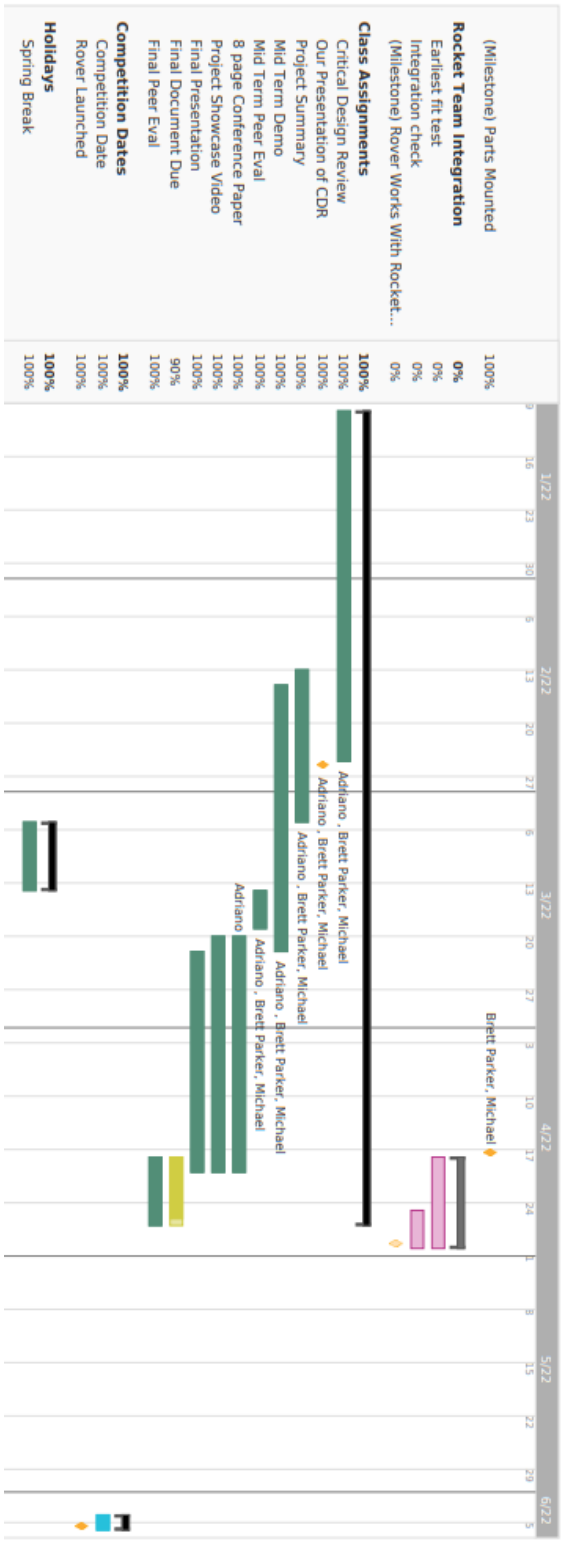


Figure 69 – Gantt Chart 2 (SD2)

10.2 – Budget Overview & Financial Planning

$500 is estimated to be provided by our sponsor, Aerojet Rocketdyne Coleman Aerospace to aid us in the development of the rover project, to incorporate into the final project.

Project budget can be divided into two categories: Structural and Component level systems. Table 4 shows an overall aspect of the distribution of funds. These estimates were well before the known scope of our project. This project cost did not estimate that the rover was going to be smaller than 6 inches wide. The original budget below, was suited for a 6-inch-wide rover should be well above the end cost of the project. Preliminary budget is does not reflect the end cost of the design. It is only a rough estimate needed for the contribution of funds from the team.

Preliminary Budget Estimation and High-End Estimates of System Costs

Electronics, Drive, & Power Systems	
• MCU PCB (Fab./Populated)	Up to $ 250
• Microprocessor (E.g., FPGA)	~$ 100 (may have on-hand)
• Video camera	$ 50-100
• Main power supply	~$ 80
• Various sensors	Included in MCU PCB
• Voltage regulation	Included in Main power supply
• Electromechanical Motors	~$ 25
• EM Motor Drivers	TBD
Structural / Canister Budget	
• Rover frame	Up to $ 50
• Payload canister	Up to $ 100
• Supports to sled	Up to $ 50
To date total (Maximum):	**$ 755**
Less sponsor contribution	**-$ 500**
Rover team total	**$ 255**

Table 17 – Project Budget

10.3 – Bill of Materials

This is the bill of materials for all the major components to our project. Major components include discrete add-ons and integrated circuits. This bill of materials does not show the smaller, more fundamental SMD components such as resistors and capacitors. The team will commit to a SMD component size in mid-December to finalize the overall sizing of decoupling capacitors and general resistors used in the design. As of now, the total batch price of these SMD components is being calculated to find the most cost-efficient way of purchase (Multiples of 10, a sample book of said components, etc.).

Bill Of Materials for known, major components

Qty	Desc.	MPN	Distributor	Price	Total
1	GPS	MAX-M8C-0	Digi-Key	$21.00	$21.00
1	MCU	STM32F303ZET7	Digi-Key	$13.97	$13.97
1	12v regulator IC	LM2587S-12/NOPB	Rochester Ele.	$8.88	$8.88
1	3.3v Regulator	LM1085ISX-3.3/NOPB	Texas Inst.	$1.81	$1.81
1	Schottky Diode	BAS40-06-7-F	Farnell	$0.21	$0.21
1	SAW Filter	B39162B9417K610	Mouser	$0.65	$0.65
1	GNSS LNA	BGA123L4E6327XTSA1	Rochester Ele.	$0.27	$0.27
1	Gyroscope IC	FXAS21002CQR1	Rochester Ele.	$2.30	$2.30
1	Accelerometer IC	MXC4005XC	Mouser	$1.49	$1.49
1	Magnetometer IC	MLX90395KDC	Mouser	$2.60	$2.60
2	H-bridge drive motor IC	DRV8833PW	Texas Instr.	$1.91	$3.82
3	5v Regulator	LM1085IT-5.0/NOPB	Rochester El.s	$1.18	$3.54
8	IR Sensor	VL53L0CXV0DH/1	Arrow Electronics	$4.57	$36.55
9	6-pin connector	BM04B-SRSS-TB	Utmel Electronic	$0.18	$1.58
1	Bipolar transistor	SMMUN2114LT1G	Digi-Key	$0.31	$0.31

*Jolly logic cute release may be revere engineered by the team instead of spending $130.
** The team may try to 3d print 43 mm wheels.

Table 18 – BOM for major components

Bill of materials per project system			
System	**Actual Cost**	**Over/under Budget**	**Allocated Budget**
• Processing	$ 74.36	-	$ 80.00
• PCB Fabrication	$ 33.94	+	$ 20.00
• Video System	$ 44.98	-	$ 50-100
• Power System	$ 93.40	+	$ 80.00
• Sensing	$ 123.99	+	$ 70.00
• Drive System	$ 109.60	-	$ 90.00
• Miscellaneous	$46.68		$100.00
• Tools	$100.00	+	$0.00
• Testing Devs/Modules	$122.60	+	$0.00
• Mistakes	$13.00	+	$0.00
TOTAL	**$ 762.55**		**$ 800.00**
Sponsor contribution	**-$500.00**		**$ 500.00**
Team contribution	**-$262.55**		**$ 300.00**

Table 19 – Systematic Cost

10.4 – Project Inspiration

Each member of this team, at the beginning of the semester, had a decision to make: To attempt to gain a place in a sponsored project or come up with a project without sponsorship. This project was posted among a small list of others which had attached sponsorships. Each posted sponsored project allowed opportunity to work with a company for solutions to a potential real-world problem. Each member of the team has listed some of their motivation and inspiration for choosing to join this Aerojet Rocketdyne Coleman Aerospace sponsored payload project.

"My motivation to join the team, stems from my past; having interest in topics such as aviation, electronics, and guidance systems. This project "hits home" and is well aligned with my studies as a graduating computer engineer with electives geared toward embedded, real-time control systems; eager to one day to work in guidance and control on weapon systems. I did intend on starting amateur rocketry as a hobby after graduation with the intent of making scale self-landing rockets; which made me favor this project more! Although I'm not working on the high-powered rocket direct[ly], I figured this would be a challenge to prepare myself for my hope of quickly rising in industry."

- *Michael Schumacher*

"Working in such a large project with multiple disciplines involved will be a great starting experience for working in the industry. We will rarely be working within a single team that doesn't rely on another discipline. As a Power Track EE, being able to expand my knowledge with electronics and mechanical components will be very interesting. Learning more about the software side of the operation from my teammates will be great experience as well. Since we will be going for the autonomous rover route, this means I will have to practice my control systems and get more experience apply linear controls to real world applications which is something I wouldn't get normally in a lecture."

- *Adriano Gussoni*

"I think this project will be a great opportunity to learn about not only embedded systems but also how those systems are worked into an overall project. I'm excited to work with a diverse group of all types of engineering majors and hope to learn a lot about the hardware and software that goes into building a rover."

- *Brett Parker*

Appendices

A – Citations

A.1 – Email Responses for image permissions

Subject	**Re: Permission to use image in report.**
From	**Team Hōkūlele Rocketry**
To	**Michael Schumacher**
Sent	**11/18/2021, 12:33:09 AM**

Aloha Michael,

My name Efren , I am currently serving as the project manager for Team Hōkūlele. I'm glad that the work of last year's team has been stumbled upon on Google! With that said, go right ahead and use it. If you need any other information, don't hesitate to reach out and we'll answer to the best of our abilities. Have a good rest of your semester and best of luck with all future endeavors.

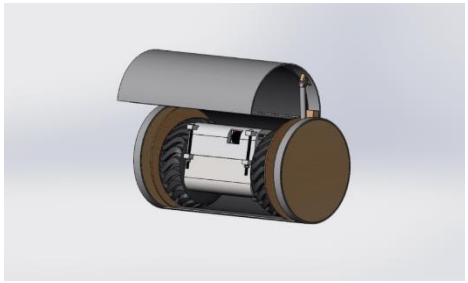
Mahalo,
Efren

On Wed, Nov 17, 2021, at 1:08 PM Michael Schumacher <
> wrote:

Hello!

. . .

Picture that permission for use is requested for:



. . .

-Michael Schumacher
University of Central Florida

A.2 – Image Permissions

FAIR USE NOTICE:

Copyright Disclaimer under Section 107 of the copyright act 1976, allowance is made for fair use for

Copyright Disclaimer under Section 107 of the copyright act 1976 (see https://www.copyright.gov/title17/92chap1.html#107), allowance is made for fair use for purposes such as criticism, comment, news reporting, scholarship, and research. Fair use is a use permitted by copyright statute that might otherwise be infringing. Non-profit, educational / research purposes only. This report may contain copyrighted material; the use of which has not always been specifically authorized by the copyright owner. We are making the material available in our efforts to spread awareness and understand of robotic technologies and the tools which help promote its advancement. While permissions for use emails have been sent, it is not guaranteed that the direct citation of these permissions be included before this report submission date.

A.2.1 – Non original Images

[F5] Figure 5

By NASA/Dave Scott; cropped by User:Bubba73 - http://www.hq.nasa.gov/office/pao/History/alsj/a15/images15.html (direct link), Public Domain, https://commons.wikimedia.org/w/index.php?curid=6057491

[F6] Figure 6

By NASA - https://www.jpl.nasa.gov/news/press_kits/mars_2020/assets/images/04_mission/04C_science/rover-instruments@2x.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=91370608

[F7] Figure 7

"Case IH autonomous tractor" by Mark Bridge is licensed under CC BY-NC-ND 2.0

[F9] Figure 9

Permissions in – Email Responses for image permissions

[F10] Figure 10

Rework of various IR sensing depictions

[F11] Figure 11

By Simple_feedback_control_loop2.png: Coronaderivative work: Rehua (talk) - This file was derived from: Simple feedback control loop2.png:, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=18126556

[F12] Figure 12

Unknown Author

[F13] Figure 13

FreeRTOS direct link

[F14] Figure 14
By Unknown author - https://www.ros.org/about-ros/, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=78446932

[F15] Figure 15
Unknown Author direct link

[F16] Figure 16
Direct link

[F19] Figure 19
Codekaizen, CC BY 3.0, via Wikimedia Commons

[F21] Figure 21
By Zithan - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=6899029

[F22] Figure 22
By en:User:Cburnett - Own work This W3C-unspecified vector image was created with Inkscape, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1476502

[F23] Figure 23
By Tim Mathias - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=111975651

[F25] Figure 25
Unknown Author

A.2.2 – ORIGINAL FIGURES

[F1] [F2] [F3] [F4] Figures 1, 2, 3, 4
Diagrams realized and created by Michael Schumacher

[F17] [F18] [F24] [F26] Figure 17, 18, 24, 26
Diagrams realized and created by Brett Parker

[F27] [F28] [F29] [F30] [F31] [F32] [F33] Figure 27, 28, 29, 30, 31, 32, 33
Diagrams realized and created Brett Parker and visual rework by Michael Schumacher

[F34] Figure 34
Diagram realized and created by Michael Schumacher

[F35] [F36] [F37] [F39] [F40] [F41] Figure 35, 36, 37, 39, 40, 41
Diagrams realized and created by Adriano Gussoni and visual rework by Michael Schumacher

[F42] Figure 42
Diagram realized and created by Brett Parker and visual rework by Michael Schumacher

[F43] Figure 43
Diagram realized and created by Michael Schumacher

[F44] [F45] Figure 44, 45
Diagrams realized and created by Brett Parker and visual rework by Michael Schumacher

[F46] Figure 46
Diagram realized and created by Brett Parker and visual rework by Michael Schumacher

[F47] Figure 47
Diagram realized and created by Brett Parker

[F48] Figure 48
Diagram realized and created by Brett Parker and visual rework by Michael Schumacher

[F49] Figure 49
Diagram realized and created by Brett Parker

[F50] Figure 50
Diagram realized and created by Michael Schumacher

[F51] Figure 51
Diagram realized by Rover team and created by Michael Schumacher

A.3 – Reference Citations

[AT]
Army Technology. (2012, August 14). *Squad Mission Support System (SMSS)*. From Army Technology: https://www.army-technology.com/projects/squad-mission-support-system-smss/

[3.3.4]
"Radio Electronics: Transmitters and Receivers." *Dummies*, https://www.dummies.com/programming/electronics/components/radio-electronics-transmitters-and-receivers/.

[4.1.3.A]
"(PDF) Overview of Software Testing Standard ISO/IEC/IEEE 29119." *ResearchGate*, https://www.researchgate.net/publication/323759544_Overview_of_Software_Testing_Standard_ISOIECIEEE_29119.

[P2020]
NASA. (n.d.). Mission overview. NASA. Retrieved September 23, 2021, from https://mars.nasa.gov/mars2020/mission/overview/

[ING]

NASA. (n.d.). Mars helicopter. NASA. Retrieved September 23, 2021, from https://mars.nasa.gov/technology/helicopter/#Overview

[PATH]

What really happened on Mars Rover Pathfinder. (n.d.). Retrieved September 29, 2021, from http://www.cs.cornell.edu/courses/cs614/1999sp/papers/pathfinder.html

[APP]

Dunbar, B. (2018, January 9). Apollo 15. NASA. Retrieved October 2, 2021, from https://www.nasa.gov/mission_pages/apollo/missions/apollo15.html

[FAR]

Documentation from FAR-1030 June 2021 amateur rocket competition https://secureservercdn.net/198.71.233.138/2af.f63.myftpupload.com/wp-cont ent/uploads/2021/03/FAR-1030-Competition-2021-June-5-edited-v5-FINAL.pdf

[TH1030]

TEAM HŌKŪLELE. (n.d.). 2021 far 1030. 2021 FRIENDS OF AMATEUR ROCKETRY (FAR) 1030. Retrieved October 31, 2021, from https://www.teamhokulele.com/2021-far-1030

End Appendix A

B – Datasheets

B.1 – Links to major PCB component datasheets:

Microcontroller Unit (STM32F303ZET7):
https://www.mouser.com/datasheet/2/389/dm00118585-1798068.pdf

Motor drivers:
https://www.ti.com/lit/gpn/DRV8833

Accelerometer:
https://media.digikey.com/pdf/Data%20Sheets/MEMSIC%20PDFs/MXC400xXC_Rev.B_4-24-15.pdf

Gyroscope:
https://www.mouser.com/datasheet/2/302/FXAS21002-1126255.pdf

Magnetometer:
https://www.mouser.com/datasheet/2/734/MLX90395_Datasheet_Melexis-2885859.pdf

GPS Integrated Circuit (MAX-M8C):
https://www.u-blox.com/sites/default/files/MAX-8-M8-FW3_HardwareIntegrationManual_%28UBX-15030059%29.pdf

Global Navigation Satellite Systems Low Noise Amplifier:
https://www.infineon.com/dgdl/Infineon-BGA123L4-DS-v01_00-EN.pdf?fileId=5546d4625fe36784015ffd006b653a7c

Surface Acoustic Wave Filter (B9415):
https://www.mouser.com/datasheet/2/842/RF360_04172020_B39162B9415K610-1838137.pdf

Infrared Sensor (VL53l0X):
https://www.mouser.com/datasheet/2/389/dm00279086-1798836.pdf

Low Noise Amplifier (BGA855N6):
https://www.mouser.com/datasheet/2/196/Infineon-BGA855N6-DS-v01_00-EN-1544704.pdf

End Appendix B