Smart RC Rescue Rover

A Senior Project Report

Presented to

Department of Electronics and Computer Engineering Technology

Indiana State University

Terre Haute, Indiana

In Fulfillment

of the Requirements for the Degree

B.S. Computer Engineering Technology

by

Lea Tice, Edgar Fernandez, & Gavin Schumaker

May 2026

Keywords: Autonomous, Rover, Search & Rescue, Unmanned Ground Vehicle, Thermal Imaging

ABSTRACT

This project addresses the safety risks faced by first responders when entering structurally unstable or hazardous environments. The primary objective is to design, engineer, and prototype a "Smart RC Rescue Rover", a semi-autonomous, remote vehicle capable of acting as a "first-in" scout to provide real-time situational awareness, thereby reducing human exposure to immediate threats. This project contributes to the field of applied robotics by demonstrating a cost-effective rover that integrates thermal sensing logic, low-latency video transmission, and active payload deployment within a customizable platform for specific first responder duties and tasks. The methodology employs an iterative engineering design process divided into parallel hardware and software development tracks. This means we aren't just building the whole rover in one go and hoping it works at the end but completing the different sub-systems and integrating them into one full system. Expected results include a fully functional prototype capable of navigating uneven terrain, transmitting real-time visual and thermal feedback to a handheld operator unit, and successfully actuating a payload depending on the task (e.g. fire suppression, search and rescue, etc.). Performance validation will involve field testing the rover's torque efficiency on inclines, communication latency in non-line-of-sight conditions, and autonomous obstacle avoidance reliability. These

findings will evaluate the feasibility of deploying low-cost robotic platforms to enhance operational safety in dynamic hazardous scenarios.

PREFACE

This report documents the development of a smart RC rescue rover, a semi-autonomous unmanned rover designed to assist first responders in hazardous environments. This project was conducted over the course of the Fall 2025 and Spring 2026 semester. The goal of this project is to integrate theoretical knowledge of embedded systems, mechanical engineering and robotics into a useful in the real-world functional platform.

The completion of this project relies on an iterative engineering design model which allows us all to work simultaneously on different tasks at different times. This required a multidisciplinary approach that forced us to rely on each individual members' talents.

Division of Responsibility:

- **Lea Tice** - circuit design, firmware programming, and system integration
- **Edgar Fernandez** - circuit installation
- **Gavin Schumaker** - mechanical design and 3D printing

ACKNOWLEDGMENTS

This project is being completed with the support of the Department of Electronics and Computer Engineering at Indiana State University. I would like to sincerely thank Dr. Javaid for her invaluable guidance, electrical and robotics expertise throughout this project. Her feedback and knowledge on the subject were driving forces in what shaped this project and helped us tackle obstacles that left us stumped. I would also like to thank Dr. McLeod and Dr. Li for their mentorship and classes on C programming and microcontrollers, which are essentially the core of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

BACKGROUND

The basis of this project is to provide a new alternative to help first responders combat potentially risky or hazardous situations. Most of the time, first responders are unable to enter these environments because of the unstable or risky conditions that the problem might be in such as intense fires or damaged structures. There are also scenarios where our Navy's sailors must combat these situations in extremely tight quarters making efficient fire suppression a challenge. The solution that we came up with is a smart RC rescue rover that could help with these intense scenarios. The rover will be able to extinguish fires at a safe distance and be able to traverse in different terrains to complete its task. The rover will be able to navigate its environment via time-of-flight sensors autonomously without input from the operator making the operator available for other high priority tasks. During manual navigation, the operator will be able to use a live-feed stream to view the rovers' immediate surroundings as well. There will be an isolated fire suppression payload that the operator can manually dispense using the controller when the rover detects a flame.

CHAPTER 2

INTRODUCTION

The members of this Rover project are Lea Tice, Edgar Fernandez, and Gavin Schumaker. Lea Tice, the Project Manager and Lead Systems Engineer, is responsible for keeping the group and project on track; she is also responsible for engineering the embedded systems of the Rover. Edgar is responsible for installing the components of the Rover. Gavin is responsible for modeling and printing any necessary parts for the Rover. This Rover's purpose is currently a prototype for an autonomous drone that's able to drive around hazardous environments; eventually the final rover product should be able to be used for emergency situations.

Table 1

Student Project Team Composition.

Team Member	Roles
Lea Tice	Project Manager & Lead Systems Engineer. Responsible for firmware development, circuit design and system integration.
Edgar Fernandez	Hardware Integration. Focused on component installation.

Gavin Schumaker	Mechanical Design. Responsible for CAD modeling and 3D printing custom chassis, controller, and sensor housing.

Deliverables

To validate the success of the Smart RC Rescue Rover project, we have defined a comprehensive set of physical and digital deliverables. These items encompass the functional electromechanical hardware and the technical documentation necessary for operation and future replication. The project deliverables are categorized into three primary components.

Smart Rescue RC Rover & Controller

This deliverable consists of the fully assembled functional rover. The custom-designed vehicle will feature a 3D printed chassis, high-torque DC motors, and integrated sensor housing. The controller will be a handheld remote unit that will feature two analog joysticks for speed and direction control, a button to release the payload, a thermal feedback visual cue, and the live feed.

Technical Documentation

This will be a comprehensive set of engineering documents that detail the full design process of the rover. This will include all the source code, electronic design schematics, CAD files, bill of materials and any included reports of significance.

User Manual

The user manual will be designed for the end-user (first responders) to operate the rover safely and effectively. The manual should include two main sections, which are the operations and maintenance procedures. The operations section will include step-by-step instructions on the basic operation of the rover. The maintenance section will include tier 1 instructions on maintenance and troubleshooting as well as next steps if issues are beyond tier 1 and repair must be completed by tier 2 personnel (project subject matter experts).

Features & Functions

The Smart RC Rescue Rover features distinct capabilities designed to enhance operational safety and efficiency in the field.

Driving Capabilities (Wheeled)

The rover's propulsion system is engineered to navigate unpredictable and extreme terrains. It utilizes two gearbox motors with a 1:90 gear ratio. This high-torque gearing allows the rover to overcome static inertia without stalling. This decision was made because the rover's main concern is not speed but maneuverability. The wheels feature a bead-lock design for secure tire seating, a 1.9" wheel and tires are 30A shore silicone rubber tires. The decision to choose rubber ensures maximum traction across a variety of surfaces, including concrete, tile, and uneven debris. These tires do not rely on tire pressure but instead use the softness and malleability of the tire for better conformity to uneven terrains. This also helps with better acceleration and braking.

Autonomous Driving

The rover is primarily operated by the controller however the user will have an option to allow for autonomous driving. This feature allows the operator to switch the rover to "auto" mode, freeing up personnel to focus on other critical tasks during emergency situations. The rover utilizes VL53L0X time-of-flight sensors to detect and avoid obstacles within a calibrated range. When an obstacle is detected, the collision avoidance logic halts motor operation and executes a redirection maneuver to prevent impact.

Flame Sensing

A critical safety feature will include integrated flame sensing logic. This system will utilize a SunFounder ST0253 Flame Sensor. This sensor should detect infrared light between 700 nm and 1000 nm. Fire gives off a significant amount of infrared light which can be physically experienced as radiant heat. The module can provide a digital output, to detect if a flame is present, and analog output, to detect the flames intensity.

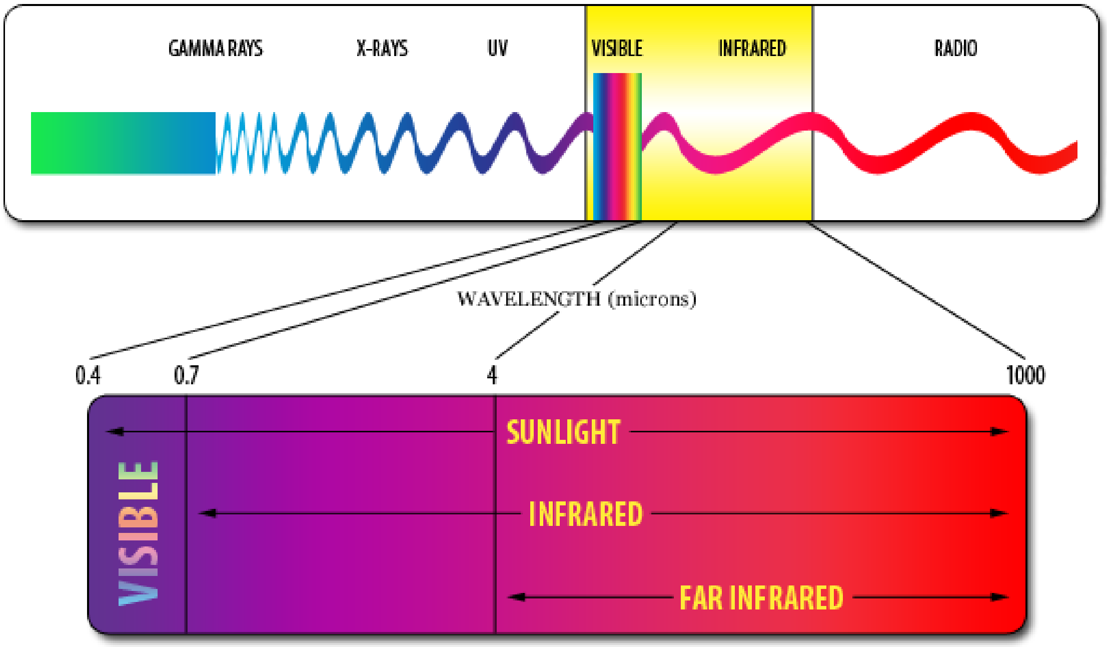


Figure 1 Electromagnetic spectrum and range of infrared light

Payload

The rover will be equipped with a modular payload adaptable to the rover's specific area of operations such as firefighting, search & rescue, police & military operations, etc. In this project, we are primarily focusing on firefighting support where our payload will feature a water discharge system for small-scale fire suppression. This function will be able to be actively triggered remotely via a dedicated input on the controller unit. The rover has a water tank installed on the back underneath which creates a physical firewall between the circuitry and water for safety. The water will be pumped using a 12V DC motor from the water tank to the nozzle, which is custom made to create a pressure effect so the water will have travel distance.

Live-Feed Camera

To provide real-time situational awareness to the rover's operator, the rover will transmit a low-latency video feed to the controller. The first-person view will allow the operator to see exactly where the rover is and its surroundings from a safe distance without having to maintain line of sight with the rover. The camera is an ESP32-CAM microcontroller with a camera module built in. This camera connects to the rovers WiFi network and streams the live feed from there.

Acceptance Criteria

Acceptance criteria are based on a pass/fail condition. These systems must be no-fail as they are the main functions of the rover. Without these main functions, the rover will not operate as intended with the mission in mind.

Driving Capabilities

The rover will utilize a tank-drive algorithm that will give the rover more control and maneuverability in extreme terrains. Depending on terrain, the rover must be able to correct itself using a gyroscope to measure its tilt. The rover must successfully go forward, reverse, and turn left & right, depending on joystick input. The propulsion system must generate enough torque to move across and climb surfaces with its intended weight and payload.

Real-time Camera Feed

Video transmission must provide continuous visual feed to the controller with a low enough latency to allow for operations. The target latency should be less than 500 milliseconds.

Payload Deployment

The payload actuation mechanism must be able to be triggered remotely from the handheld controller. The system must discharge the intended payload (water, fire suppressant, etc.). The payload system must not leak or misfire during operation.

Autonomous Obstacle Detection & Navigation

When placed in its ‘autonomous mode’, the rover must be able to successfully detect an obstacle and go around it. It should make use of its thermal sensor as well to detect heat signals above a certain threshold to simulate a human body or fire.

Controller Communication with Rover

The handheld controller must maintain a stable connection with the rover. Line of sight should not be a factor in reliable communications as the rover is intended to navigate hard to reach and hazardous areas that pose too much of a risk to human life.

CHAPTER 3

PROJECT MANAGEMENT

Effective project management is essential to the success of complex engineering tasks, particularly when integrating mechanical, electrical, and software sub-systems. This chapter outlines the organizational strategies, financial planning, and scheduling methodologies employed by the team to ensure the Smart RC Rescue Rover is completed within the academic year. The project operates under strict time constraints, necessitating a structured approach to resource allocation, risk mitigation, and milestone tracking.

Methodology

To ensure the timely completion of the Smart RC Rescue Rover, the team has adopted an iterative engineering design process. This methodology facilitates a highly efficient, parallel workflow, allowing hardware construction and software development to occur simultaneously rather than sequentially. This approach minimizes "downtime" where one member is waiting for another to finish a task.

As the Lead Systems Engineer, Lea Tice focuses on the initial design and firmware coding for individual sub-systems (e.g., the propulsion logic or thermal sensing logic). Once a sub-system's circuit design is validated, Edgar Fernandez immediately proceeds with component installation and physical wiring. Simultaneously, Gavin

Schumaker manages the CAD modeling and manufacturing (3D printing) of custom chassis components. This ensures that critical elements, such as sensor mounts and the controller housing, are fabricated "just-in-time" for final assembly. By testing and refining each sub-system independently before full-scale integration, the team can identify and resolve technical issues early in the prototyping phase.

Development Tools & Resources

To facilitate seamless development and design collaboration, the team utilizes the Microsoft 365 ecosystem. Microsoft Teams serves as the primary hub for daily communication, while the suite's cloud integration allows for real-time collaborative editing of project documentation. For software version control, GitHub is used to manage the Yuumi Rescue Rover repository hosted at https://github.com/llcesselx. This enables the safe implementation of new algorithms through branching, ensuring that updates can be tested without disrupting existing logic. Autodesk Fusion 360 was used in modeling the main body and payload of the rover. The design engineer has opted to use another software to model the controller. Arduino IDE serves as the primary development environment, chosen for its native compatibility with the project's microcontroller architecture.

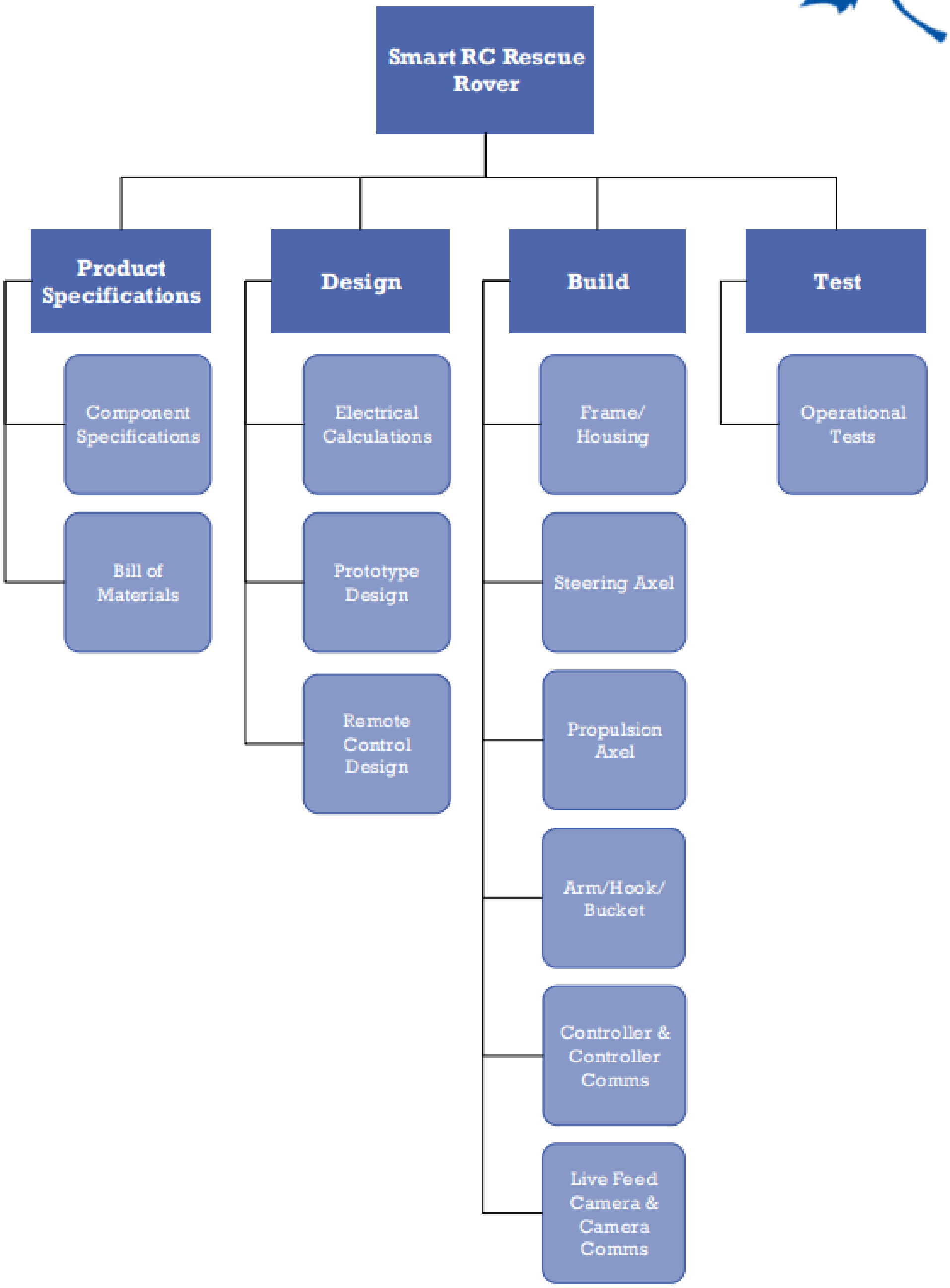
Gantt Chart

Smart RC Rescue Rover	Semester 1																Semester 2															
Task	September				October				November				December				January				February				March				April			
Precedent Research	🟫	🟫	🟫																													
Component Specifications				🟦																												
Design Prototype Rover					🟨	🟨	🟨	🟨																								
Design Prototype Controller					🟨	🟨	🟨	🟨																								
Define Electrical Needs					🟦	🟦	🟦																									
Obtain Component Measurements					🟥	🟥	🟥																									
Define Mounting Pts & Weight Distribution								🟩																								
Model Main Body in CAD Software								🟨	🟨	🟨																						
Develop Code for L. Propulsion System								🟦	🟦	🟦	🟦																					
Design Circuit for L. Propulsion System								🟦	🟦	🟦	🟦																					
Print Main Body										🟨	🟨	🟨																				
Install & Test L. Propulsion System												🟥	🟥	🟥																		
Model Payload in CAD Software												🟨	🟨																			
Develop Code for R. Propulsion System										🟦	🟦	🟦																				
Design Circuit for R. Propulsion System										🟦	🟦	🟦																				
Print Payload													🟨	🟨	🟨	🟨	🟨															
Install & Test R. Propulsion System												🟥	🟥	🟥																		
Develop Code for Fire Extinguishing Payload												🟦	🟦	🟦																		
Design Circuit for Fire Extinguishing Payload												🟦	🟦	🟦																		
Model Controller in CAD Software																	🟨	🟨	🟨	🟨												
Install & Test Payload															🟥	🟥	🟥															
Develop Code for Heat Sensing System															🟦	🟦	🟦															
Design Circuit for Heat Sensing System															🟦	🟦	🟦															
Install & Test Heat Sensing System																		🟥	🟥	🟥												
Develop Code for Communication Link System																		🟦	🟦	🟦												
Design Circuit for Communication Link System																		🟦	🟦	🟦												
Install & Test Communication System																					🟥	🟥	🟥									
Print Controller																					🟨	🟨	🟨									
Develop Code for Rover Live Feed																					🟦	🟦	🟦									
Design Circuit for Rover Live Feed																					🟦	🟦	🟦									
Install & Test Live Feed																								🟥	🟥	🟥						
Design Smart Features Mount for Rover																						🟨	🟨	🟨								
Model Mount in CAD Software																								🟨	🟨							

Develop Code for Smart Features																								🟦	🟦	🟦						
Design Circuit for Smart Features																								🟦	🟦	🟦						
Print Smart Features Mount																										🟨	🟨					
Install & Test Smart Features																											🟥					
Develop Final Test Plan & Success Criteria																											🟦	🟦				
Full System Benchtop Integration Test																												🟦	🟦			
Final Bug Fixes																												🟦	🟦			
Controlled Mobility & Maneuverability Test																														🟥		
Field Performance & Endurance Test																														🟨		
Document Test Results																												🟫	🟫	🟫		
Prepare Final Test Reports																												🟫	🟫	🟫		
Gather All Source Code																															🟦	
Gather All Schematics																															🟥	
Gather All CAD Files																															🟨	
Develop Final Bill of Materials																															🟫	
Develop Troubleshooting, System and User Manuals																															🟫	🟫
Develop Slide Deck & Presentation																															🟫	🟫

Key	Color
Lea Tice	🟦
Edgar Fernandez	🟥
Gavin Schumaker	🟨
Lea/ Edgar	🟩
Edgar/ Gavin	🟧
Lea/ Gavin	▨
All Three Members	🟫

WBS Chart


Smart RC Rescue Rover
Product Specifications
Component Specifications
Bill of Materials
Design
Electrical Calculations
Prototype Design
Remote Control Design
Build
Frame/ Housing
Steering Axel
Propulsion Axel
Arm/Hook/ Bucket
Controller & Controller Comms
Live Feed Camera & Camera Comms
Test
Operational Tests

Budget & Costs

This rover is designed to be a cost-effective alternative to market-ready first-responding units, which often rely on expensive proprietary hardware and specialized service contracts. By leveraging commercial components, such as standard TT gearbox motors and Arduino microcontrollers, this project demonstrates that functional search-and-rescue capabilities can be achieved at a fraction of the industrial price point. This open-architecture approach not only reduces initial build costs but also simplifies future maintenance and repair for the end-user.

Funding

This project is self-funded by the students in the project team. Some resources were also funded by the department of electronic and computer engineering technology. Each student will contribute to the total cost of the project depending on hardware needed.

Table 2

Bill of Materials

Component	**Category**	**Cost**	**Qty**	**Total**	**Funded By**
TT Motor All-Metal Gearbox – 1:90 Gear Ratio	Propulsion	$5.95	4	$23.80	Lea
3D Print Filament	Resources	X	X	X	ISU
48 AA Batteries	Resources	$9.99	1	$9.99	Lea
4 x BOJACK L298N Motor Drivers	Propulsion	$9.99	1	$9.99	Lea
3 x HiLetgo MPU-6050 Accelerometer/Gyroscope Sensor	Propulsion	$9.99	1	$9.99	Lea

HiLetgo MLX90614ESF Infrared Temperature Module	Fire Detection	$10.99	1	$10.99	Lea
5x ACEIRMC VL53L0X ToF Sensors	Autonomy	$15.99	1	$15.99	Lea
3 x HiLetgo ESP32	Communication	$17.99	1	$17.99	Lea
2 x Hosyond ESP32-CAM	Live Feed	$17.99	1	$17.99	Lea
3x KEXIN 32GB Micro SD Card	Live Feed	$15.99	1	$15.99	Lea
Creality K1C 3D Printer	Resources	$379.00	1	$379.00	Lea
Creality Hyper PLA CF 2KG Bundle	Resources	$34.00	1	$34.00	Lea
BBDINO 30 Shore A Silicone Rubber	Resources	$24.67	2	$49.34	Lea
20 Pin Rainbow Flat Ribbon Cable Wire 28G	Resources	$8.99	1	$8.99	Lea
Zeee 3S Lipo Battery 3000mAH 11.1 V 50C	Power	$42.99	1	$42.99	Edgar
T Plug Connectors	Power	$8.99	1	$8.99	Lea
10 x Rocker Switches	Resources	$6.99	1	$6.99	Lea
Arduino Uno WiFi	Microcontroller	$44.00	1	$44.00	Lea
Arduino Nano ESP32	Microcontroller	$19.30	1	$19.30	Lea
Arduino 33 IoT	Microcontroller	$24.90	1	$24.90	Edgar

Risk Management

Every engineering project has risks. Some of the potential risks to the successful completion of the rover identified include technical, safety and scheduling risks.

Technical Risk

This primary technical risk of this project is integration of the different hardware systems and creating firmware for the different systems that communicate with each other. The hardware and software will be developed in parallel due to the functions and feedback of the sensors relying on each other.

Another risk we already tackled was the current handling of the L298N motor drivers may draw high currents during stall conditions or steep climbs that may exceed torque specifications, potentially leading to component failures. To mitigate this risk, logic in the propulsion system firmware constrains the values of the PWM pins used to prevent stalling.

A critical risk identified is the overheating of electronic components, specifically the L298N motor drivers under load. To mitigate this, the chassis features a ventilated, open-airflow architecture. This design maximizes passive convective cooling, ensuring effective heat dissipation without the need for active fans or additional power consumption.

Latency of the line-of-sight video feed is a secondary risk we are predicting. High latency would make remote operation of the rover extremely difficult and increase the likelihood of damage to the rover if tested in certain environments. The plans for mitigation are prioritizing connection stability over resolution.

Safety Risk

The rover is powered by a LiPo battery. These are known to be dangerous due to their self-sustaining fires which can be caused by different factors which can easily be recreated due to negligence or ignorance. Fires and explosions can occur from damage, overcharging, and overheating. These batteries also have to be balanced or they could fail.

The fire suppression system that is planned for the rover will utilize water. This will introduce the risk of water encountering some of our exposed circuitry which could potentially cause permanent damage to the components or a short circuit. The plan to mitigate this issue is a physically compartmentalized area for the fire suppression system. The fluid reservoir and pump are planned to be housed underneath the circuitry on the bottom of the rover, and the payload delivery system (hose) will be traced along the bottom suppurts of the rover away from the components and into the payload housing.

Schedule Risk

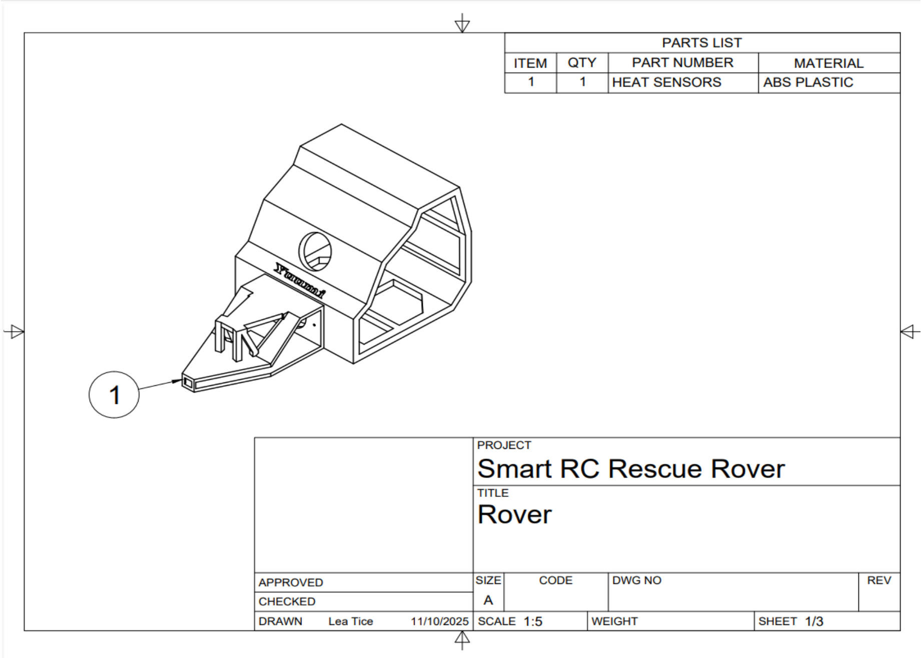
Fabrication bottlenecks present a significant risk because this project relies heavily on 3D printed components. Since the team will be utilizing university facilities for production of the models, fabrication timelines are subject to print lab availability and student demands from other courses. To mitigate this, early coordination with the university print lab staff needs to be prioritized by the design engineer. Regular status updates will need to be maintained with the staff as well to anticipate delays. Scheduling of these timelines has been given longer timeslots than other tasks in the project to help with this issue as well.

Most components to be used in the rover will be pre-built modules such as the 1:90 gearbox motors. Variable shipping times and availability of components could throw the schedule off track. All critical hardware has been sourced during the research and design phase, but not all have been ordered. A list of approved venders has been created and maintained for component selection during development. This will ensure alternative procurement options are available should the stock of an item be out from a primary vendor.

CHAPTER 4

ROVER DESIGN

The rover consists of two major parts, the main body and the payload. The main body will be housing most of the circuitry and hardware with the payload housing the modular features for the different applications of the rover. As we have said, this prototype will focus on firefighting support and will feature a fire suppression system. The fire suppression system is traced from the bottom of the rover which holds the water in a custom tank and into the payload. Modeling of the rover and payload kept openness in mind. There should be ample space to install the different mounts and supports for each system but still maintain structural integrity. Some lessons learned during this were the need to separate the frame and 'wall's of the rover. Due to the closedness of the top and bottom of the rover's payload, it was hard to install some of the sensors. Due to this, next time I would keep a more open bare bones design for the rover itself and design a separate body for the rover like a cars frame and body.



PARTS LIST

ITEM	QTY	PART NUMBER	MATERIAL
1	1	HEAT SENSORS	ABS PLASTIC

PROJECT
Smart RC Rescue Rover

TITLE
Rover

APPROVED
CHECKED
DRAWN Lea Tice 11/10/2025

SIZE	CODE	DWG NO	REV
A			

SCALE 1:5 WEIGHT SHEET 1/3

Figure 2. Isometric view of completed rover model.

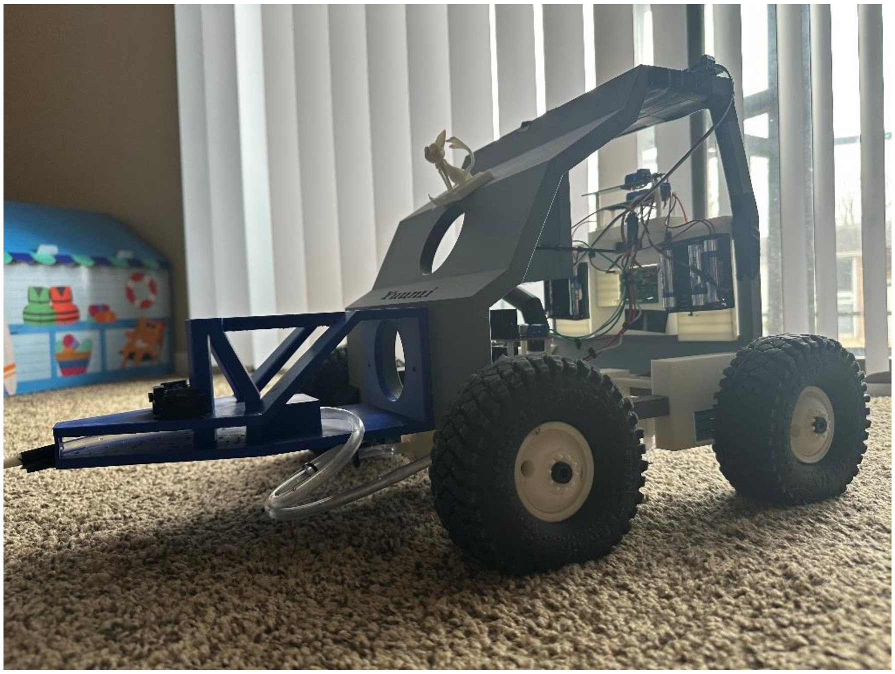


Figure 3 Yuumi support rover body

CHAPTER 5

ELECTRONIC CIRCUIT DESIGN

The electronic architecture of the Smart RC Rescue Rover is centered around an Arduino microcontroller. This platform was selected for its 5V logic compatibility, extensive library support for C firmware, and sufficient General-Purpose Input/Output (GPIO) pins to handle both the propulsion and future sensor sub-systems.

Propulsion System

The system operates on a dual-voltage standard: a high-current rail for the electromechanical components and a regulated low-voltage rail for signal processing. The schematic documented in Appendix B illustrates the completed wiring for the propulsion system. The circuit utilizes a BOJACK L298N Dual H-Bridge motor driver to manage the high-current demand of the two DC gear motors.

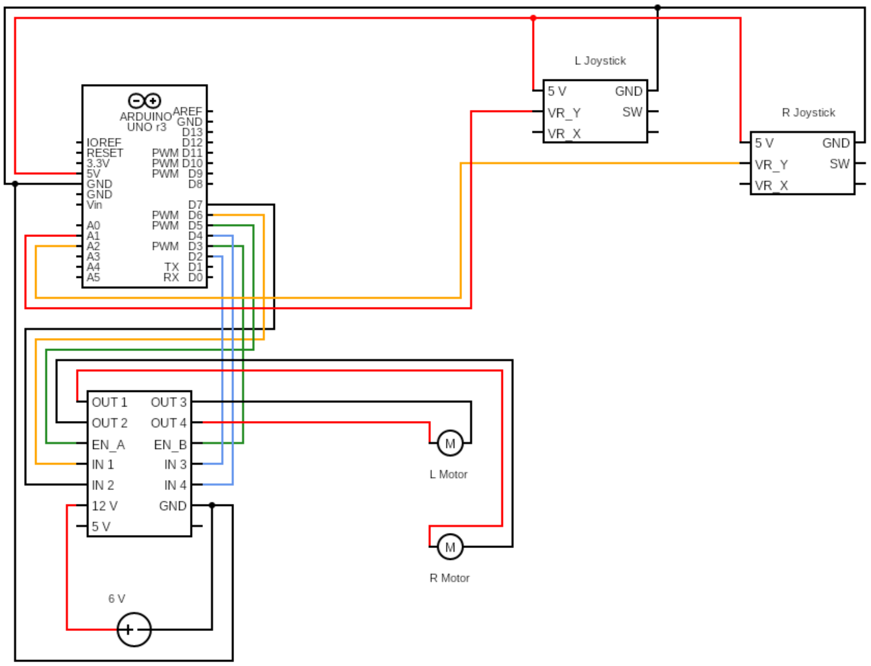


Figure 4. Wiring schematic of propulsion system.

Motor Power

A dedicated 6V DC supply (derived from the AA battery bank) is connected to the 12V input terminal of the L298N driver. This ensures the motors have direct access to the battery current.

Common Ground

A common ground (GND) reference is established between the 6V battery source, the L298N driver, and the Arduino Uno to ensure accurate signal transmission.

L298N Interface

The L298N interfaces with the Arduino via six digital control lines. The driver's Enable pins (EN A and EN B) are connected to Arduino PWM-capable pins 3 and 5, respectively. This allows the firmware to adjust the duty cycle of the voltage sent to the motors, controlling their speed. The H-Bridge logic pins determine the polarity of the motors. IN 1 and IN 2 (controlling the Left Motor) are wired to digital pins 2 and 4. IN 3 and IN 4 (controlling the Right Motor) are wired to digital pins 7 and 6.

CHAPTER 6

SOFTWARE DESIGN

The firmware for the Smart RC Rescue Rover is developed in C using the Arduino Integrated Development Environment (IDE). The software architecture utilizes standard control structures and logic loops [1]. In this design, the main loop() function sequentially calls distinct sub-routines for propulsion, sensor data acquisition, and payload management.

Propulsion System

The core mobility logic, documented in Appendix A, implements a differential steering ("tank drive") algorithm. This system translates user inputs from the handheld controller into PWM signals for the motor drivers.

Input (Data Acquisition)

The firmware reads the analog voltage from the joystick potentiometers using the analogRead() function. This returns a raw integer value between 0 and 1023, representing the physical position of the stick.

Signal Processing

The raw input data is processed to match the motor driver's requirements. The map() function is used to convert the 10-bit input range (0-1023) into an 8-bit PWM output range (0-255). To prevent "ghosting" (unwanted movement when the sticks are neutral), the code includes a conditional check: if (abs(value) > 20). This ensures that minor sensor fluctuations near the center position are ignored, and the motors remain idle until a deliberate command is input.

Output (Motor Actuation)

The firmware reads the analog voltage from the joystick potentiometers using the analogRead() function. This returns a raw integer value between 0 and 1023, representing the physical position of the stick. The processed speed and direction values are sent to the L298N driver via the digitalWrite() and analogWrite() functions. The logic splits the signal: the magnitude of the value determines the PWM duty cycle (speed), while the sign (positive/negative) toggles the H-Bridge pins to determine direction (Forward/Reverse).

CHAPTER 7

HARDWARE DEVELOPMENT & PROCUREMENT

PLA (Polylactic Acid) was selected as the primary material for all structural bodies, mounts, and brackets in this phase. as driven by the material's high dimensional accuracy and cost-effectiveness. Furthermore, the team increased its manufacturing agility by utilizing personal 3D printing equipment alongside university resources, allowing for rapid, low-cost iteration of design prototypes without being limited by lab hours or external lead times. There was a delay with the controller development which caused a huge delay in testing the system as each subsystem is installed. The controller design model was due on January 12th, and the print was supposed to be completed by January 30th. Therefore, a prototype controller was designed to use in the testing environment and if worse comes to worse, will be the one the project uses.

Tires and Wheels

The rovers wheels and tires are custom made. They were designed by a BuildItBetter on makerworld.com and features a bead-lock design like road-worthy tires and wheels. The tires are made from a carefully selected silicone mix that took into account the hardness of the finished product.

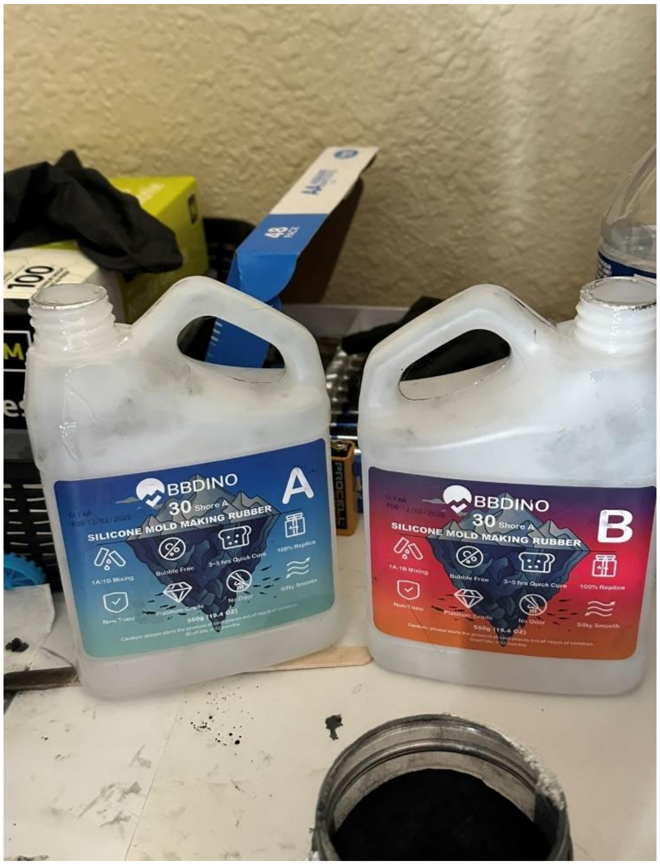


Figure 5 BBDINO 30 Shore A Silicone Rubber Mix

Tires

The tires were molded using a custom made 3D printed mold from BuildItBetter. The mold consisted of a bottom piece, top piece, middle pieces for the inside of the tire and a funnel.

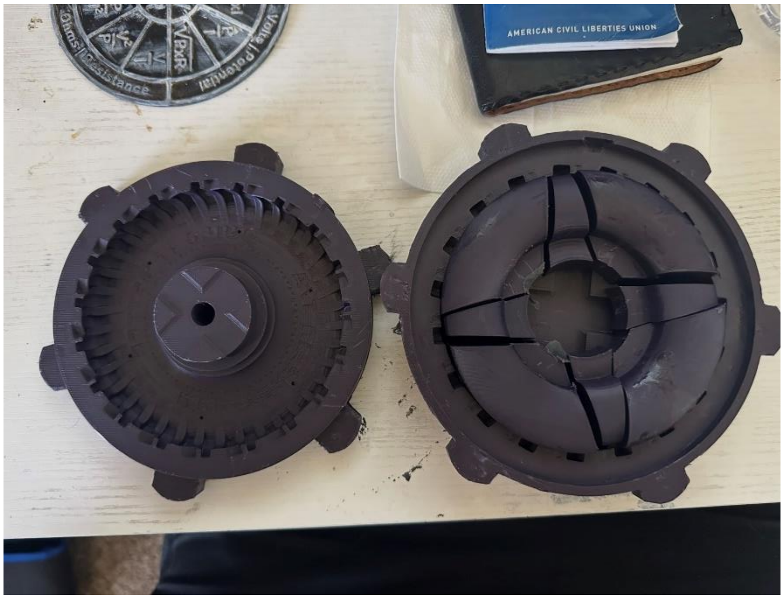


Figure 6. BuildItBetter Custom Tire Mold

The mixture for the silicone consisted of a 1:1 ratio. I also added black pigment for a more realistic tire aesthetic but the black pigment itself does nothing else for the tires integrity. After mixing properly for about 4-5 minutes, I would then pour the mixture into the mold. A few issues were found when following the direct instructions for this mold. The silicone instructions were to let it sit for about 4 hours and the maker of the mold suggested 4 hours as well. During the first pour, the tire ripped after only letting it cure for the 4 hours. I waited 6 hours the next time and the tire came out better. So instead of wasting more silicone, I decided to do an overnight cure. In the morning, I would make the mixture, pour it into the mold and not pull it out until the next morning.

This made the tires much more stronger and durable during the process of clearing it from the mold. When pulling the tire away form the mold, I would have issues of it ripping and so the extra time helped. As well as the extra time, it was better to overfill the mold rather than underfill. I noticed when I filled it just enough, the tires would have air pockets but when I would pour the silicone into the molds until an excess would come out of the small air vents, the tire came out perfect. Unfortunately, I was only able to get one perfect tire which was the last one I made so in the meantime, the rover will have less than perfect tires until we can get more silicone.

Figure 7 Black Pigment for Silicone Rubber Mix

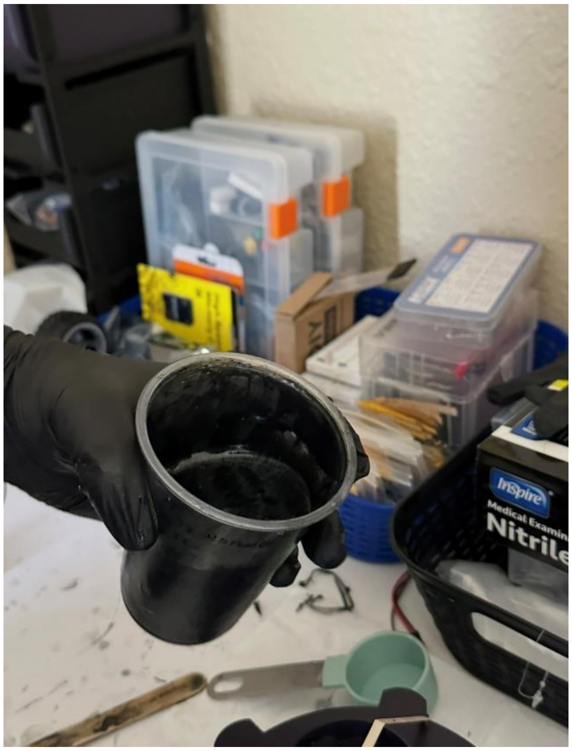


Figure 8 Silicone Rubber Tire Completed Mixture

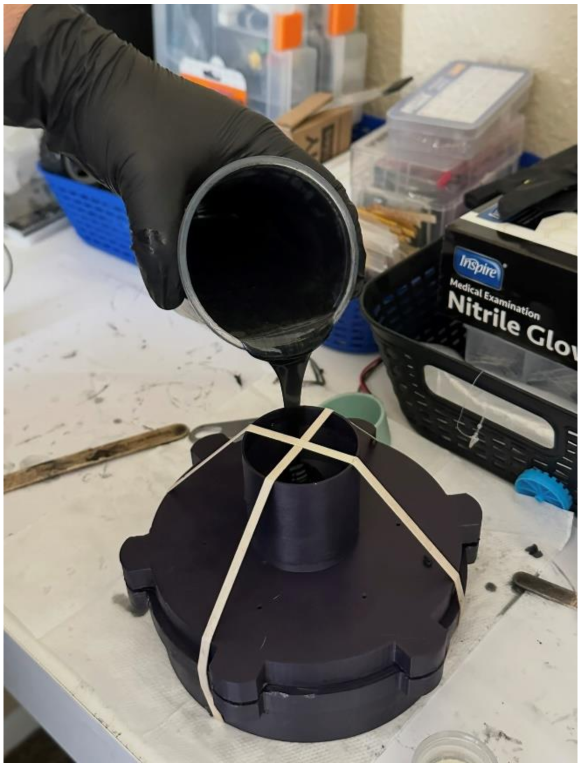


Figure 9 Silicone Rubber Mixture Pour

Wheels

The wheels for the custom tires were made from the same maker as the tire, BuildItBetter. The wheels feature a 1.9” diameter with a bead-lock system found on the wheels and tires of cars on the road today. The bead of a tire is the inner diameter of the tire, so the donut hole part. If you feel it, you’ll notice it isn’t as hard as the rest of the tire (unless they are summer tires) and not as flexible. This bead snaps onto the wheel of the car which then holds it into place. The bead-lock allows for extremely low tire pressure in a tire while keeping it seated on the rim of the wheel which is essentially what we are dealing with in this design because there is zero air pressure in our tires.

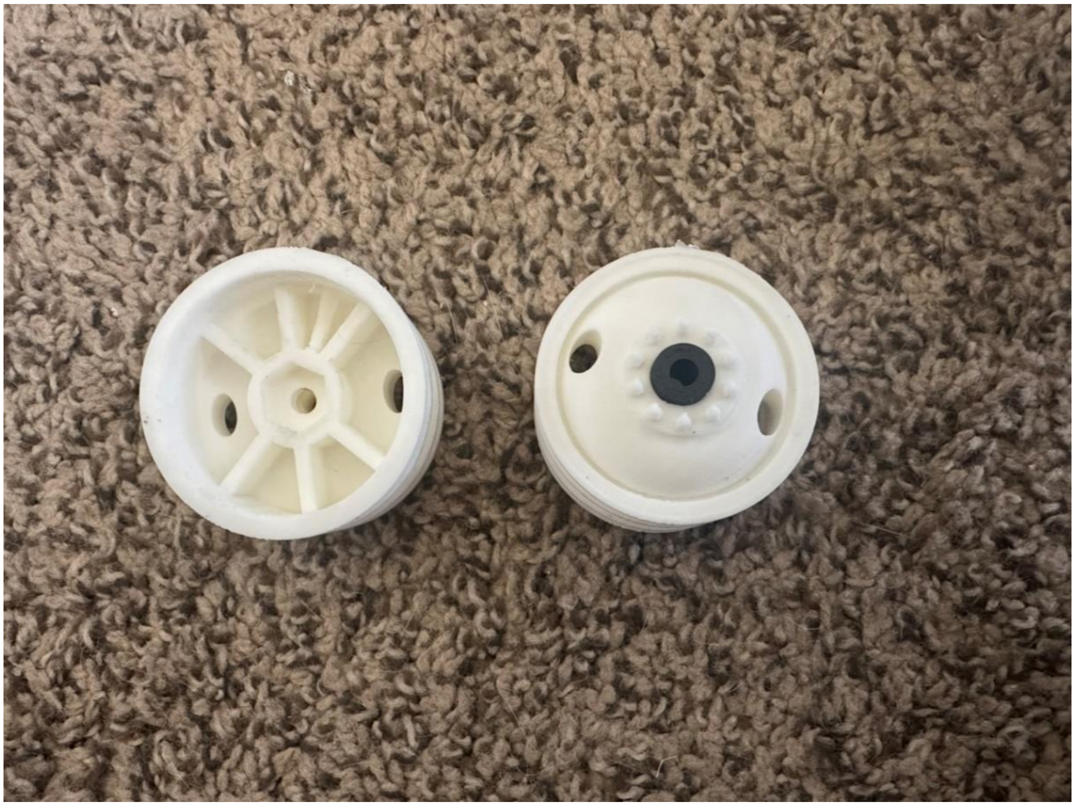


Figure 10 3D Printed Bead-Lock Wheels

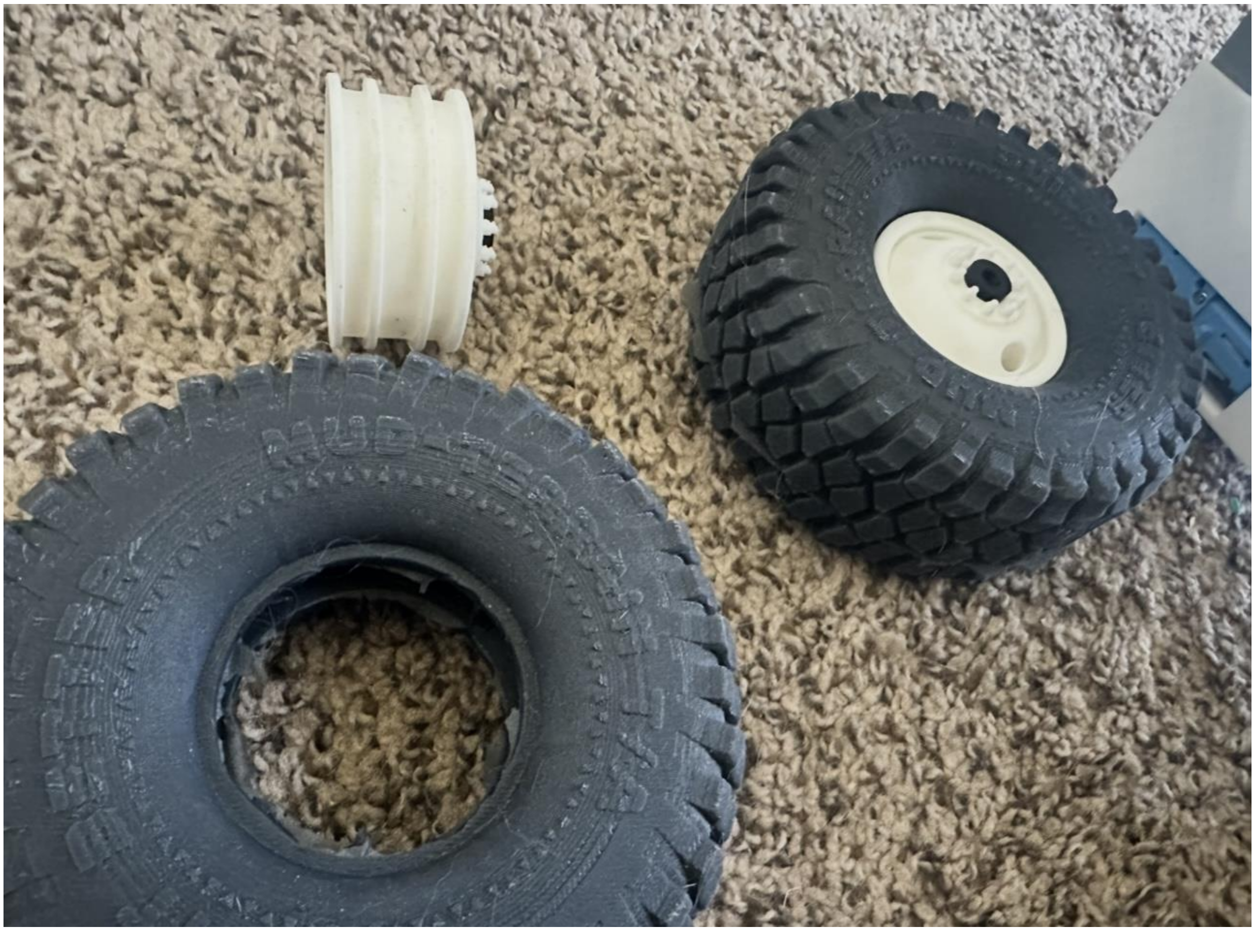


Figure 11 Finished Tires and Wheels

Mounts & Housings

Specialized housing was developed for the VL53L0X Time-of-Flight sensors and the ESP32-CAM module in Fusion 360.

Autonomous Sensor Suite Mount

An iterative design process was utilized when developing the mounts and housing. Three housings were developed for the VL53L0X sensors as well as a mounting system to allow for 45-degree angles offsets on the left and right-side mounts to allow for smarter autonomous movement. Measurements were acquired using digital calipers to ensure a press-fit for the sensors. Through several iterations of rapid prototyping, we successfully eliminated early dimensional errors, resulting in a finalized mounts and housings.

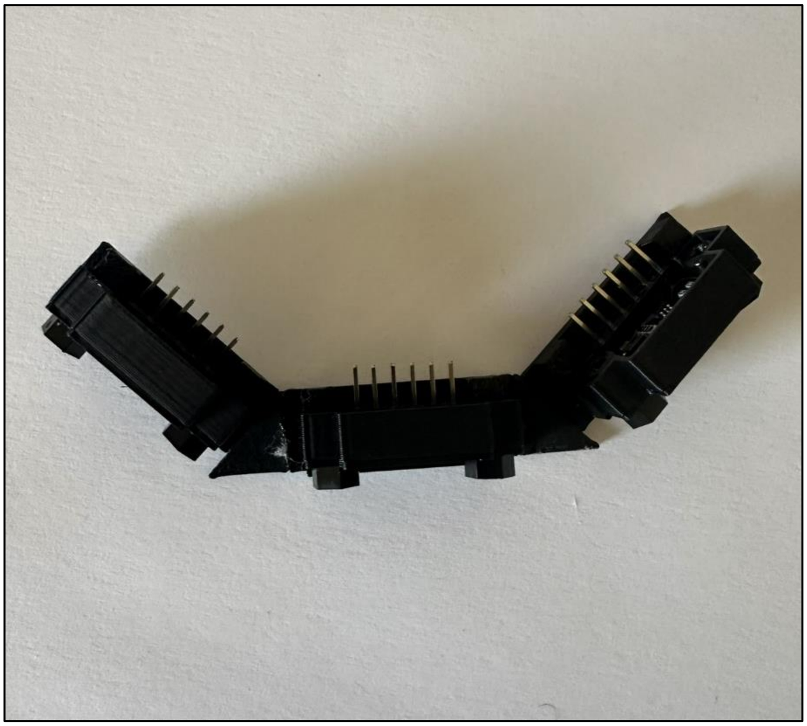


Figure 12 VL53L0X ToF Sensors Custom Mount

Similarly, the ESP32-CAM mount underwent several design revisions to address camera stability issues. Early prototypes suffered from component "wobble" inside the housing. The final design iteration features a four-point mounting system and a recessed internal channel, allowing the camera module to sit flush against the aperture.

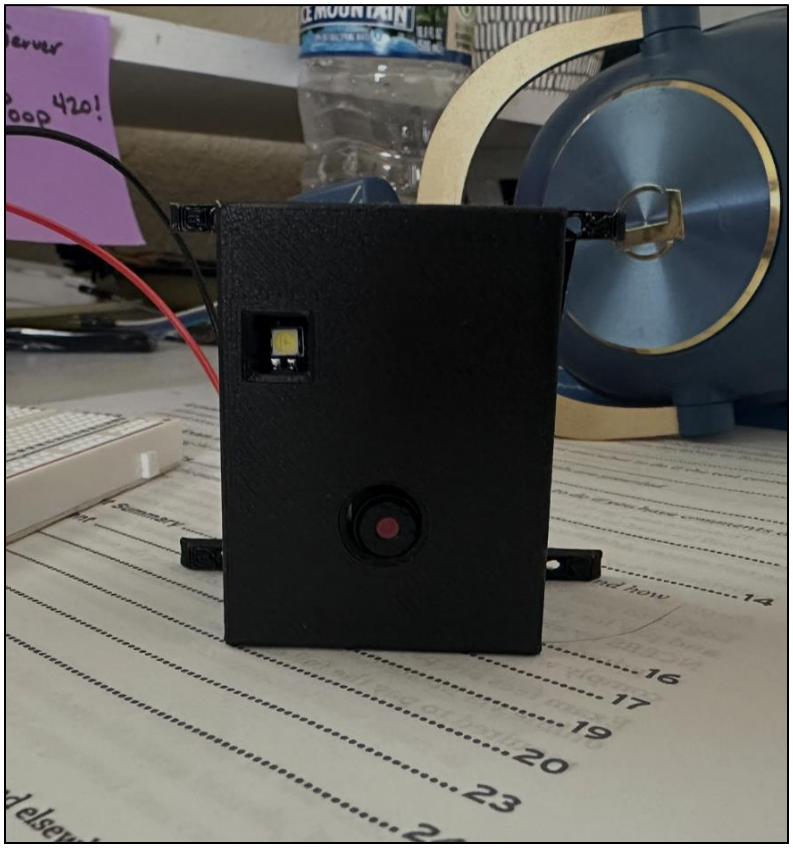


Figure 13 ESP32-CAM Custom Mount

Ultrasonic Sensors to VL53L0X Time-of-Flight Sensors

During the initial planning phase, the team selected the standard HC-SR04 Ultrasonic sensors for our autonomous logic. We decided to pivot to another sensor, the VL53L0X sensors for one critical reason. In a rescue environment, there could be tight corners, debris and other obstacles that create a close quartered environment. These tight spaces allow for errors with ultrasonic sensors due to ‘ghosting’ where the sensor detects obstacles that aren’t there. The VL53L0X sensor utilizes light which mitigates this issue.

Motor Mounts

The design of the motor mounts has been slow and therefore temporary mounts were designed and printed in the meantime. These mounts are not made to be permanent and just feature a slide-on design for now. Due to the rover needing to be mobile, these temporary mounts were needed. The installation phase of the propulsion system has been delayed, which was due last year.

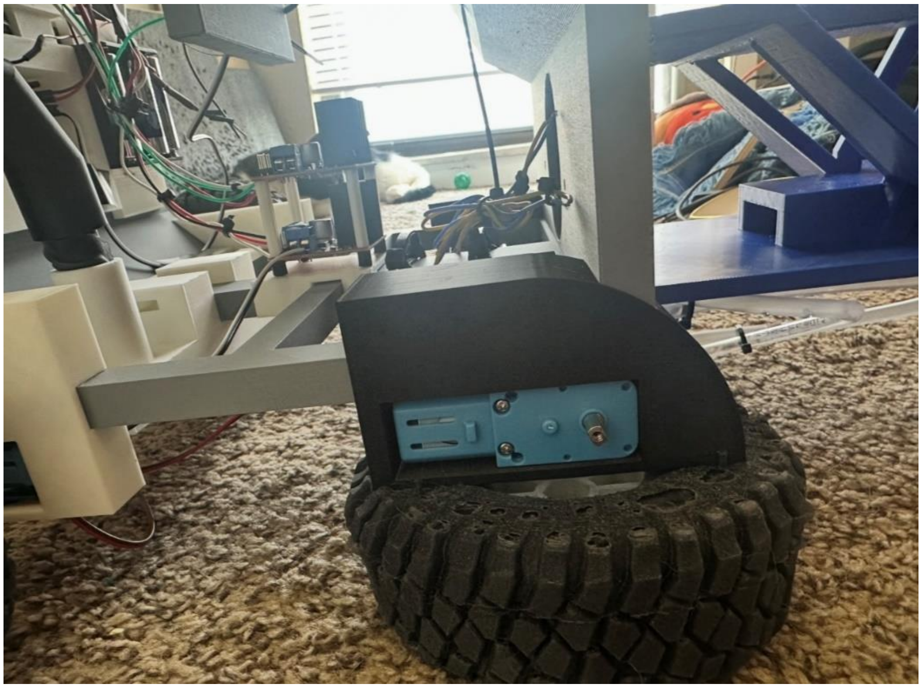


Figure 14 Temporary Front Motor Mount

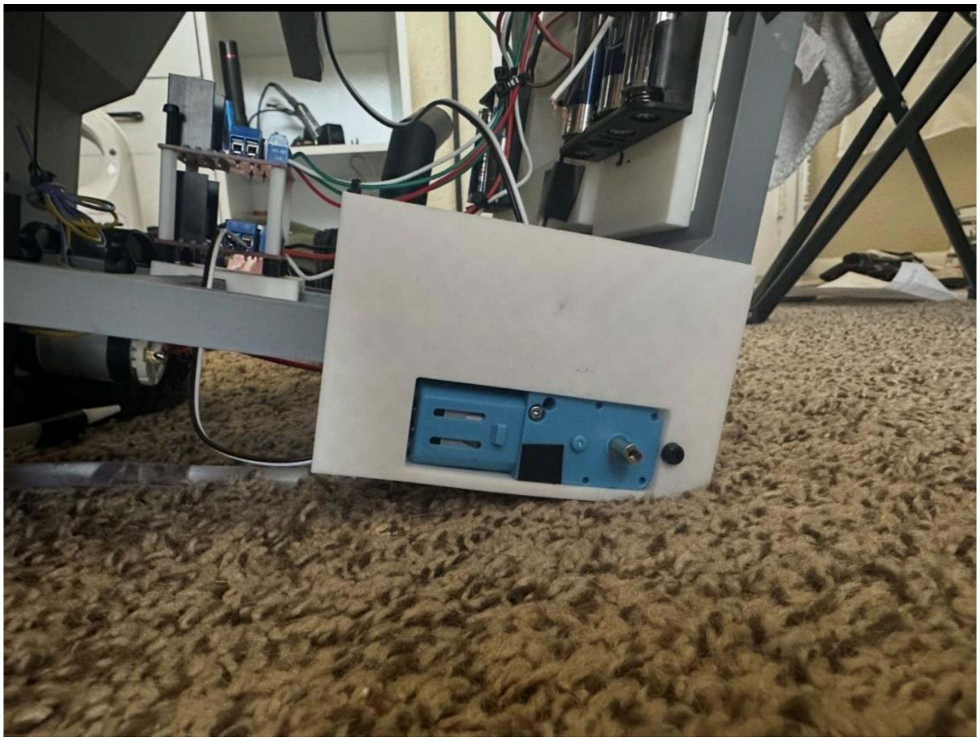


Figure 15 Temporary Rear Motor Mount

CHAPTER 8

NETWORK ARCHITECTURE

The Smart Rescue Rover utilizes a distributed embedded system architecture to manage a WiFi access point and high-bandwidth video streaming. During this period, the network topology was established. The original design called for the ESP32-CAM to act as the primary Access Point (AP). However, initial testing revealed that the camera module suffered from recurrent power-instability resets (Brownouts) when simultaneously managing the Wi-Fi stack and high-resolution video processing. To resolve this, the team implemented a Hub-Client topology.

Hub-Client Topology

The main microcontroller used in the rover for controls is an Arduino Nano ESP32. This acts as the rovers' brain and network host. We chose this microcontroller due to its superior processing capabilities compared to the microcontroller included in our already acquired lab kits. The Nano utilizes its ESP32-S3 chip to broadcast its network RescueRover_HUB.

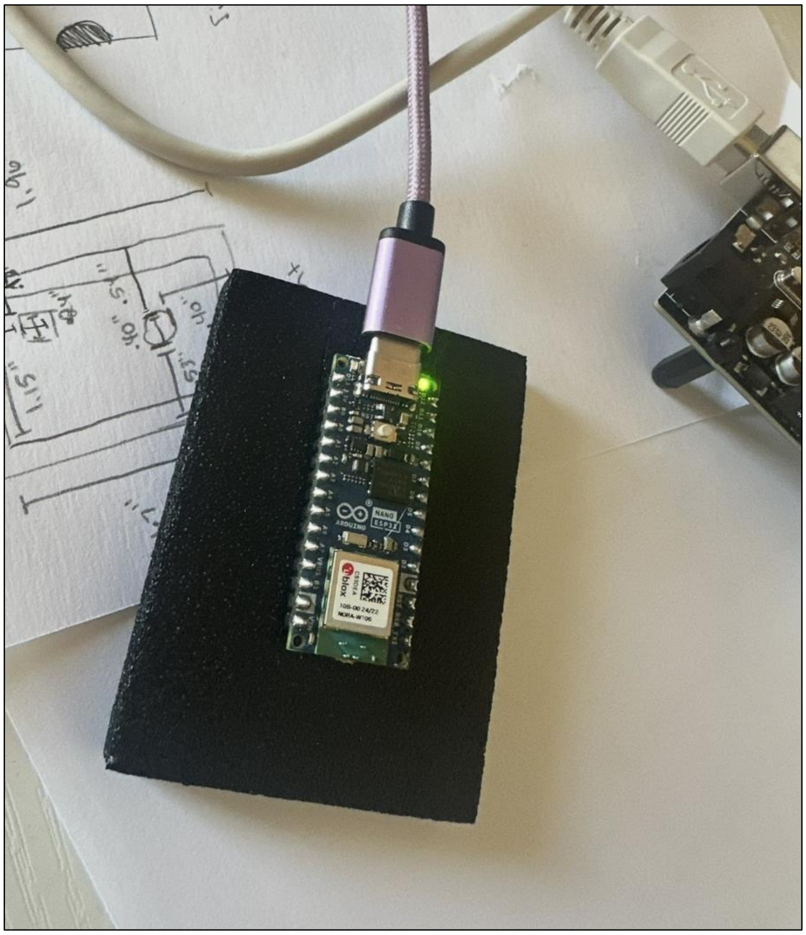


Figure 16 Arduino Nano ESP32

ESP32-CAM

The ESP32-CAM connects to the Nano as a client by connecting to the RescueRover_HUB network and automatically entering the required password using its firmware. This offloads the networking overhead from the camera module so it can solely be used for the live feed. This allows the ESP32-CAM to stabilize the video feed and dedicate its resources solely to MJPEG compression. One of the primary technical hurdles during this period was the lack of an onboard USB-to-Serial converter on the

ESP32-CAM. To overcome this without specialized hardware, the team utilized an Arduino Uno Hardware Bypass technique:

- The Logic: By connecting the Arduino Uno’s RESET pin to GND, the onboard ATmega328P microcontroller is disabled. This effectively transforms the Uno into a dedicated USB-to-TTL Serial converter.
- The Interface: The computer’s serial data is passed directly through the Uno’s RX (0) and TX (1) pins to the ESP32-CAM.
- The Bootloader Trigger: To initiate the firmware upload, GPIO 0 was strapped to GND, forcing the ESP32 into "Flash Mode." This allowed the Arduino IDE to communicate directly with the CAM’s bootloader at a stabilized 115200 baud rate.

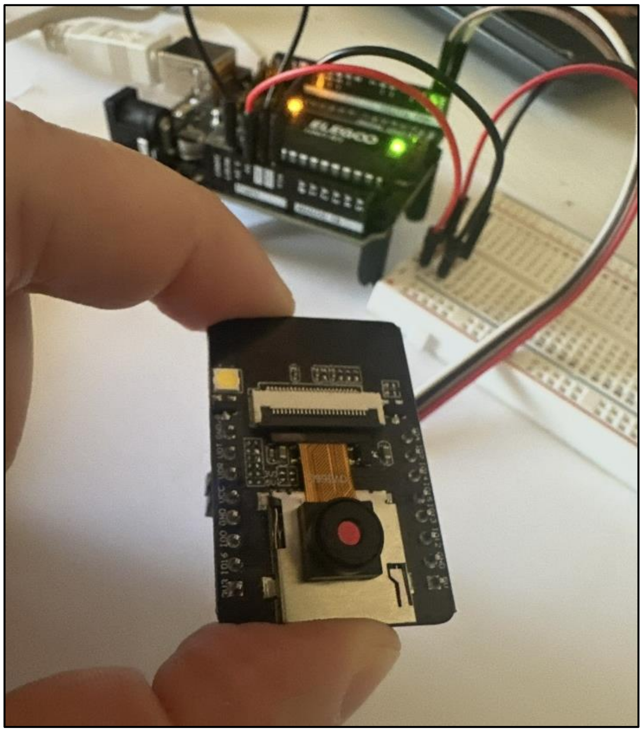


Figure 17 ESP32-CAM Module

ESP32-CAM Firmware

The deployed code (see Appendix A) includes several low-level optimizations to ensure stability in a mobile, battery-powered environment. Due to high instantaneous current spikes during Wi-Fi handshaking, the team implemented a software-level override using WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0). This prevents the chip from entering a "reboot loop" when voltage dips occur. The External Clock (XCLK) frequency was set to 10MHz. While the sensor can handle higher speeds, 10MHz was chosen to minimize electromagnetic interference (EMI) and ensure "clean" image data transmission over the jumper-wire interface. The firmware utilizes MJPEG (Motion JPEG) compression. By setting the fb_count to 1 and the jpeg_quality to 12, the system maintains a consistent frame rate over the UDP stream while staying within the heap memory limits of the ESP32's internal RAM. In the setup() function, the module executes a WiFi.begin() command targeting the RescueRover_HUB SSID. The firmware enters a blocking while loop, polling the connection status until a local IP is assigned by the Nano ESP32. Once connected, an internal HTTP server is initialized on Port 80, specifically listening for the /stream URI request to begin high-speed image buffer transmission.

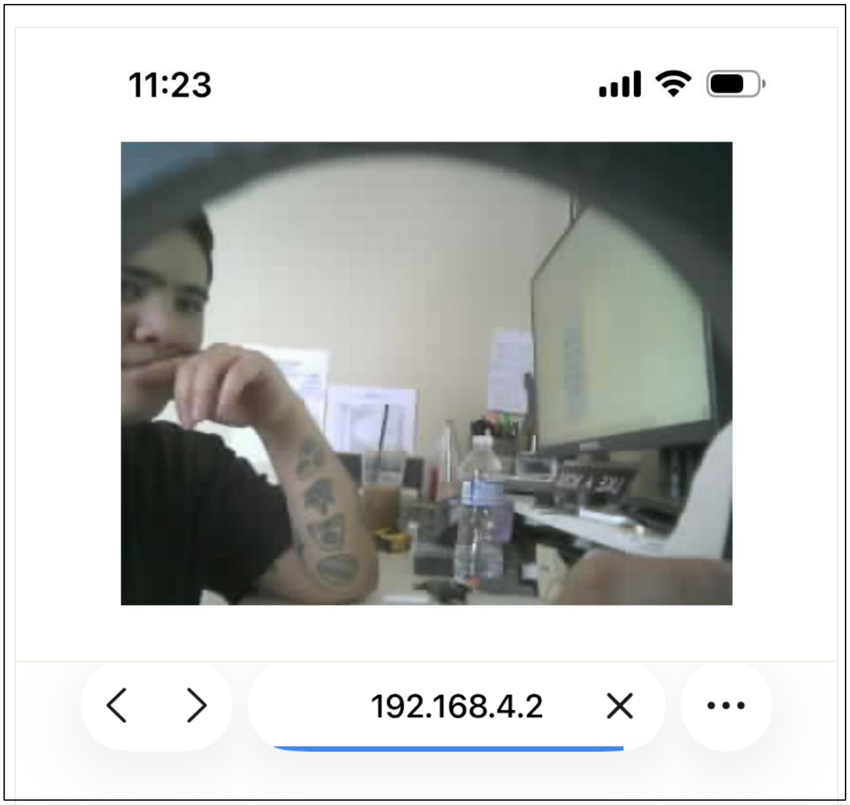


Figure 18 Live Feed View

CHAPTER 9

POWER DISTRIBUTION AND CIRCUIT DESIGN

There have been multiple issues with signal integrity and power throughout the whole system when each subsystem is connected. Due to a blown regulator in the original microcontroller, no progress has been made towards firmware on certain systems. Since this issue, caution needs to be prioritized moving forward, so we do not get behind any more due to broken parts. A new design is being implemented in the rover in which fuse relays will be utilized to protect logic level circuits. A high-current 11.1-volt rail will be used to power the gearbox motors and water pump. A separate 5V rail will be utilized to power the microcontroller and sensors. A new 4 fuse relay module, larger gauge wire, and buck converters were purchased for the new power system.

CHAPTER 10

CONCLUSION

This concludes all our progress on the “Smart RC Rescue Rover” so far. The next three weeks will be focused on transitioning from an installation state to a testing state so we can start debugging and troubleshooting full-system integration issues if they arise. While limited progress has been made recently, advice from senior level engineers like Dr. Javaid, created a stronger understanding of power electronics and the idea of implementing fuse relays. The immediate priority for the next phase is to resolve the communication errors between the access point and the remote control. Once this link is established, testing of the controls wirelessly will be able to begin.

REFERENCES

[1] K. N. King, *C Programming: A Modern Approach*, 2nd ed. New York, NY: W. W. Norton & Company, 2008.

[2] T. Gaddis, *Starting Out with Programming Logic and Design*, 2nd ed. Boston, MA: Pearson, 2010.

[3] R. L. Boylestad, *Introductory Circuit Analysis*, 12th ed. Upper Saddle River, NJ: Pearson, 2010.

[4] R. L. Boylestad and L. Nashelsky, *Electronic Devices and Circuit Theory*, 11th ed. Upper Saddle River, NJ: Pearson, 2013.

[5] M. N. Horenstein, *Design Concepts for Engineers*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2010.

[6] Autodesk, "Fusion 360: Integrated CAD, CAM, and CAE software," *Autodesk.com*, 2025. [Online]. Available: https://www.autodesk.com/products/fusion-360/overview.

[7] Indiana State University, "ECT 437 Senior Project Guidelines," *Department of Electronics and Computer Engineering Technology*, 2025.

APPENDIX A: CODE

Appendix A serves as the comprehensive repository for the C++ firmware source code developed for the Smart RC Rescue Rover's embedded control systems. The software architecture is modular, with distinct codebases dedicated to each functional sub-system of the rover. As development progresses through each planned system, this appendix will be updated to include the source code for each sub-system.

Propulsion System v1

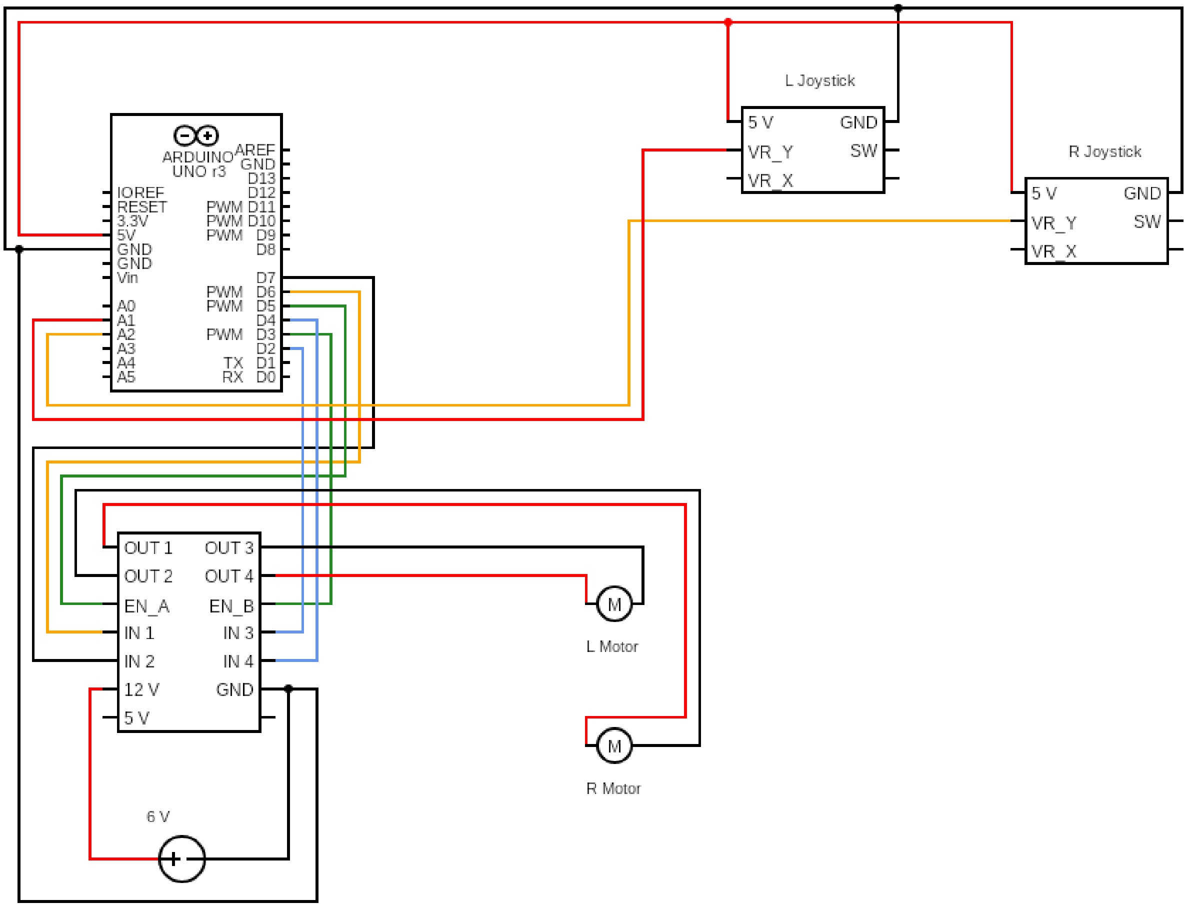
This firmware interfaces with the dual-joystick handheld controller to read analog inputs, apply dead zone correction to prevent signal drift, and map the values to Pulse Width Modulation (PWM) signals. These signals drive the L298N motor controller, allowing for precise forward, reverse, and rotational maneuvering of the high-torque DC motors.

```
/*
  Dual Joystick to L298N Motor Control Test (Tank Drive) - FIXED
  - Added 'constrain()' to prevent motor stall at full stick.
*/

// --- Joystick Pins ---
#define LEFT_JOY_Y_PIN A1
#define RIGHT_JOY_Y_PIN A2

// Default joystick center values (0-255 range)
#define JOY_CORRECTION 128

// --- LEFT Motor Pins ---
#define LEFT_ENABLE_PIN 3   // PWM
#define LEFT_IN1_PIN 2
#define LEFT_IN2_PIN 4

// --- RIGHT Motor Pins ---
#define RIGHT_ENABLE_PIN 5  // PWM
#define RIGHT_IN3_PIN 7
#define RIGHT_IN4_PIN 6

void setup() {
  Serial.begin(115200);
  Serial.println("--- ROVER READY ---");

  // Set Left Motor pins
  pinMode(LEFT_ENABLE_PIN, OUTPUT);
  pinMode(LEFT_IN1_PIN, OUTPUT);
  pinMode(LEFT_IN2_PIN, OUTPUT);

  // Set Right Motor pins
  pinMode(RIGHT_ENABLE_PIN, OUTPUT);
  pinMode(RIGHT_IN3_PIN, OUTPUT);
  pinMode(RIGHT_IN4_PIN, OUTPUT);
}

void loop() {
  // ==================================================
  // 1. READ JOYSTICKS (With Anti-Ghosting)
  // ==================================================
  int rawLeft = analogRead(LEFT_JOY_Y_PIN);
  delay(2); // Tiny delay to prevent crosstalk
  int rawRight = analogRead(RIGHT_JOY_Y_PIN);

  // ==================================================
  // 2. CONTROL LEFT MOTOR
  // ==================================================
  // Map 0-1023 input to 0-255 range
  int leftY = map(rawLeft, 0, 1023, 0, 255) - JOY_CORRECTION;
  leftY = leftY * -1; // Flip Y-axis so UP is positive

  int leftSpeed = 0;

  // Calculate Speed
  if (abs(leftY) > 20) { // Deadzone check
    if (leftY > 0) {
      leftSpeed = map(leftY, 20, 127, 0, 255);
    } else {
      leftSpeed = map(leftY, -20, -127, 0, -255);
    }
  }

  // --- THE FIX: CONSTRAIN VALUES ---
  // Ensure speed never goes above 255 or below -255
  leftSpeed = constrain(leftSpeed, -255, 255);
  // ==================================================
  // 3. CONTROL RIGHT MOTOR
  // ==================================================
  int rightY = map(rawRight, 0, 1023, 0, 255) - JOY_CORRECTION;
  rightY = rightY * -1; // Flip Y-axis so UP is positive

  int rightSpeed = 0;

  // Calculate Speed
  if (abs(rightY) > 20) { // Deadzone check
    if (rightY > 0) {
      rightSpeed = map(rightY, 20, 127, 0, 255);
    } else {
      rightSpeed = map(rightY, -20, -127, 0, -255);
    }
  }

  // --- CONSTRAIN VALUES ---
  rightSpeed = constrain(rightSpeed, -255, 255);

  // ==================================================
  // 4. SEND COMMANDS
  // ==================================================
  // Serial.print("L: "); Serial.print(leftSpeed);
  // Serial.print(" | R: "); Serial.println(rightSpeed);

  controlLeftMotor(leftSpeed);
  controlRightMotor(rightSpeed);

  delay(50);
}

// --- Helper Functions ---

void controlLeftMotor(int speed) {
  if (speed > 0) {
    digitalWrite(LEFT_IN1_PIN, HIGH);
    digitalWrite(LEFT_IN2_PIN, LOW);
    analogWrite(LEFT_ENABLE_PIN, speed);
  } else if (speed < 0) {
    digitalWrite(LEFT_IN1_PIN, LOW);
    digitalWrite(LEFT_IN2_PIN, HIGH);
    analogWrite(LEFT_ENABLE_PIN, abs(speed));
  } else {
    digitalWrite(LEFT_IN1_PIN, LOW);
    digitalWrite(LEFT_IN2_PIN, LOW);
    analogWrite(LEFT_ENABLE_PIN, 0);
  }
}

void controlRightMotor(int speed) {
  if (speed > 0) {
    digitalWrite(RIGHT_IN3_PIN, HIGH);
    digitalWrite(RIGHT_IN4_PIN, LOW);
    analogWrite(RIGHT_ENABLE_PIN, speed);
  } else if (speed < 0) {
    digitalWrite(RIGHT_IN3_PIN, LOW);
    digitalWrite(RIGHT_IN4_PIN, HIGH);
    analogWrite(RIGHT_ENABLE_PIN, abs(speed));
  } else {
    digitalWrite(RIGHT_IN3_PIN, LOW);
    digitalWrite(RIGHT_IN4_PIN, LOW);
    analogWrite(RIGHT_ENABLE_PIN, 0);
  }
}
```

APPENDIX B: SCHEMATICS

Propulsion System v1

The following schematic details the integration of the Smart RC Rescue Rover's propulsion sub-system. This circuit interfaces the Arduino microcontroller with the BOJACK L298N H-Bridge motor driver



APPENDIX C: APPROVED VENDORS

https://www.adafruit.com/

https://www.sparkfun.com/all-categories

https://www.amazon.com/

https://www.digikey.com/

https://www.automationdirect.com/adc/home/home