Highlighted in yellow - needs to be done, general

Highlighted in pink- needs to be done, coding related

EGR 314 - Embedded Systems Design II
TerraGuard Rover: Team Final Report
Prepared By Phoenix Force: Vinny Panchal, Emonie France, Amanda Pizarro, Danitza Jimenez, James Austin, and Erjan Baigenzhin
December 3rd, 2025

Abstract

The Phoenix Force team is developing *TerraGuard*, an autonomous rover designed to help detect wildfires earlier and provide useful environmental data for fire crews. Our system focuses on collecting real-time information using several sensors, including temperature, heat pattern, air quality, humidity, and pressure modules. We also added a GPS and camera so the rover can report its exact location and send visual updates. All data is sent through a mesh-network connection to a monitoring station, where the system checks if any readings hit dangerous levels. If they do, TerraGuard activates LEDs and a speaker to immediately warn responders.

Throughout the semester, our team learned how to combine sensing, microcontrollers, and wireless communication into one working prototype. We made design choices based on what would be reliable outdoors, easy to test, and realistic for an early-stage wildfire detection system. The rover shows that low-cost, ground-based systems can support firefighters by giving them faster information and helping them react before conditions get worse.

Our final prototype can collect environmental data, check it against safety thresholds, and send alerts when something looks hazardous. The project also taught us how to organize subsystem roles, work with limited resources, and test hardware in real scenarios. Each team member received $60 to spend toward parts, giving us a total project budget of $360.

I. INTRODUCTION

Wildfires are becoming more common and more severe, and one of the biggest challenges for responders is catching early hot spots before they grow into large fires. Many of these early warning signs happen in remote areas where people aren't always able to check in person. Our project focuses on solving this problem by creating *TerraGuard*, an autonomous rover that can patrol an area and collect real-time environmental data that may point to potential wildfire activity. The main purpose of the system is to help detect changes in temperature, smoke, air quality, and other conditions so responders can react faster and more safely.

The target users for this system are firefighters, fire-watch teams, and emergency response groups who need reliable information from the field without having to physically be in dangerous areas. We also considered secondary users like researchers and community partners who could use the data for studying fire behavior or improving prevention strategies. These users need technology that is easy to understand, simple to operate, and durable enough for outdoor environments.

The scope of our project includes designing and prototyping a complete embedded system using sensors, a microcontroller, wireless communication, and a mobile rover platform. TerraGuard uses ESP-Mesh networking to send data to a base station, where alerts can be viewed in real time. If the rover detects something past a safe threshold, it triggers LEDs and a speaker to warn responders.

Our prototype includes multiple environmental sensors, a GPS module, and a camera mounted on a mobile rover. The system reads data, checks it against preset safety limits, and communicates the information through a mesh network. Each team member received $60 toward the project, giving us a total budget of $360 to build and test the rover.

This report explains how our team organized the work, how we made design decisions, the system architecture, hardware and software development, test results, lessons learned, and suggestions for improving the design in the future.

II. TEAM ORGANIZATION & DESIGN PROCESS

A. Team Roles and Responsibilities

Vinny – Project Manager / Team Lead – Screen Module, Communication, System Integration
Vinny managed the overall schedule, task assignments, and team coordination. He worked mainly on the screen module subsystem, including the ESP32-WROOM-32U screen unit, LCD display, receiving HTTP requests, and controlling the LED strip, button, speaker, and DFPlayer board. He helped integrate all subsystems into one working rover system by designing the main PCB, and he coded all sensor, drive, GPS, and antenna-related functionality into the rover's main board. He also developed the HMI display software and implemented the LED functionality.

Emonie – Documentation Lead – Wireless Antenna Subsystems & Communication Support
Emonie organized written deliverables, formatting, and version control. She also worked on the 2.4 GHz external antenna subsystem for both the rover and screen module. She supported testing and adjustments to wireless communication performance.

Amanda – Design Lead and Sensors – Gas Sensor & BME688 (Temperature) Subsystems
Amanda created diagrams, graphics, and UI layouts while also working on the gas sensor and the BME688 temperature channel. She contributed to sensor mounting choices, wiring paths, and how sensor data is visually represented on the screen module.

Danitza – Research Lead – BME688 (Humidity & Pressure) Subsystems
Danitza researched wildfire behavior, communication protocols, and environmental sensing. She focused on the BME688 humidity and pressure channels, helping verify sensor accuracy and reliability for outdoor environments.

Erjan – Electronics Lead – GPS Module & Motor Driver / Mobility Subsystems
Erjan designed and tested the rover's electronics, including PCB layout and wiring. He handled the GPS NEO-6M subsystem, the L298 motor driver, and the four-motor assembly. He supported firmware integration and hardware debugging.

James – Test & Validation Lead – Power Subsystem & System Testing
James developed test procedures and validated performance across subsystems. He worked mainly on the power subsystem, including the fuse, USB connector, charger, battery pack, and voltage regulators (5V and 3V). He ensured stable power delivery and checked overall system readiness

B. Mission Statement & Charter (Updated)

Team Charter Statement

Phoenix Force commits to building a reliable and realistic wildfire-detection prototype that can identify early warning signs, communicate without public Wi-Fi, and present clear alerts through a simple, glove-friendly interface. Our team aims to meet measurable engineering goals such as low-latency alerts, accurate sensor performance, and consistent communication reliability. We agree to document our decisions, justify our trade-offs, and deliver a final package that demonstrates our technical growth and reflects a meaningful engineering effort.

Product Mission Statement

Our mission is to design a rugged, easy-to-use wildfire-response rover that senses early changes in the environment, delivers clear alerts, and stays connected even in areas without public networks. The goal is to help wildfire crews make faster, safer decisions by giving them straightforward and dependable real-time information.

C. Decision-Making Process

At the start of the project, our team brainstormed over a hundred different ideas. We eventually narrowed them down to a top fifteen list that included things like a thermal-camera drone, smoke sensors, LoRa mesh communication, rugged weather stations, swappable-battery systems, and even a hybrid firefighting drone. While going through these ideas, we noticed that many of them pointed toward the same need, which was early wildfire detection with good mobility and reliable communication.

Instead of choosing a fully stationary weather station or going with an expensive UAV platform, our team decided to combine the idea of a mobile sensing unit with a simple HMI. We felt this gave us a unique direction while still being realistic for our skill level and budget. We also chose a rover instead of a drone because drones are much more difficult to operate, cost more to maintain, and have very short battery life. A rover can carry more sensors, run longer, and avoid flight restrictions, which made it the better choice for us.

Instructor feedback also played a big role in shaping our concept. We were asked to put our concept generation directly on the website instead of only submitting a document. The suggestion to use pop-ups or tabs pushed us to make the site more interactive. We were also told to cite any AI-generated images and to add callouts to our product sketch so viewers could

easily tell what each subsystem was. This helped us clean up our presentation and make our design clearer.

The biggest pivot we made was shifting from drone-based sensing to a ground-based rover that collects temperature, heat-pattern, air-quality, humidity, and pressure data. Once we decided on this approach, we focused on adding mesh-network communication and a separate screen module for displaying alerts. This final concept fit the course requirements, stayed within our budget, and gave us something original that was not copied from online projects.

III. SYSTEM OVERVIEW

A. Final System Architecture Overview

Our final system is made up of two main parts that work together: the Rover Unit and the Commander Screen Unit. The Rover Unit uses an ESP32-CAM as the main controller, and it is responsible for collecting all sensor data and streaming live video. It reads information from the BME688 for temperature and humidity, the MQ-2 sensor for smoke and flammable gases, the ultrasonic sensor for distance measurements, the GPS module for coordinates, and the MPU6050 for tilt and terrain feedback. The L298N motor driver is also connected to the rover so it can handle forward, reverse, and turning commands. All of these components run through the ESP32-CAM, which processes the data and manages how the rover responds.

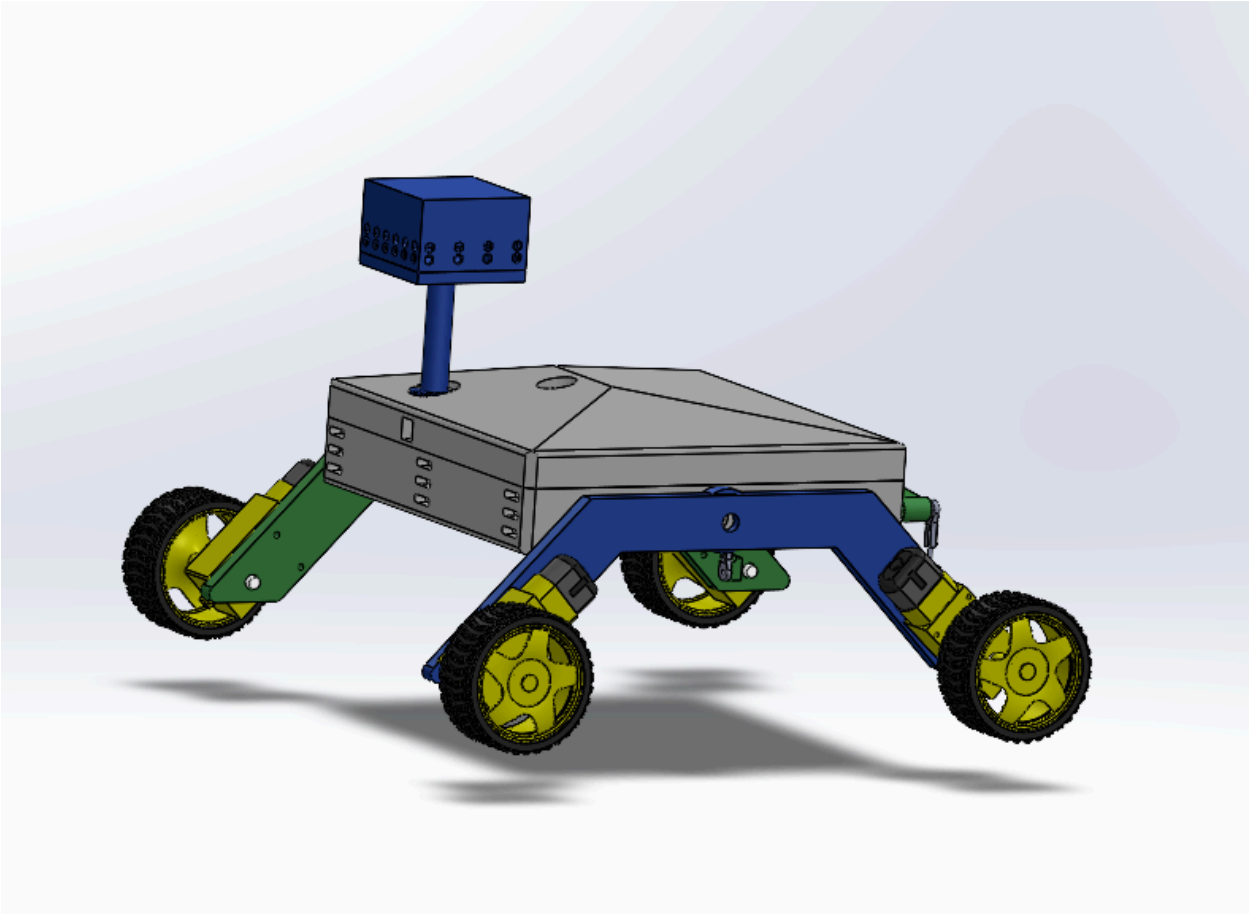
A key part of how the system works is that the Rover ESP32 creates its own Wi-Fi Access Point. This allows the system to work even in remote areas because it does not depend on outside Wi-Fi or internet. The Commander Screen Unit, which uses a CrowPanel ESP32-S3 display, connects directly to this Access Point. After it connects, the screen starts polling the rover for updated sensor readings while also displaying the MJPEG video stream. The user interface was built in LVGL, and it shows the main metrics including gas readings, GPS position, humidity and temperature, and ultrasonic distance. The screen also plays audio alerts and changes LED effects when there is a warning such as high gas levels, an obstacle too close to the rover, or poor camera visibility.

Movement commands go back to the rover through simple HTTP requests sent from the screen. When the user presses a button on the interface, the Commander sends a command to the rover using the same Access Point connection. The rover then activates the correct motor pins. We also added safety checks so the rover can cancel movement on its own if something dangerous is detected. For example, if the ultrasonic sensor detects an obstacle in the stop zone, the rover refuses to move and notifies the screen.

Overall, the two units form a closed system where the rover continuously sends data and video to the screen while the screen sends control commands back. Because everything runs through the rover's Access Point, the system stays self-contained and reliable in the field. This made it possible for us to create a real-time monitoring rover that supports environmental sensing, obstacle detection, and user control in a simple and effective way.

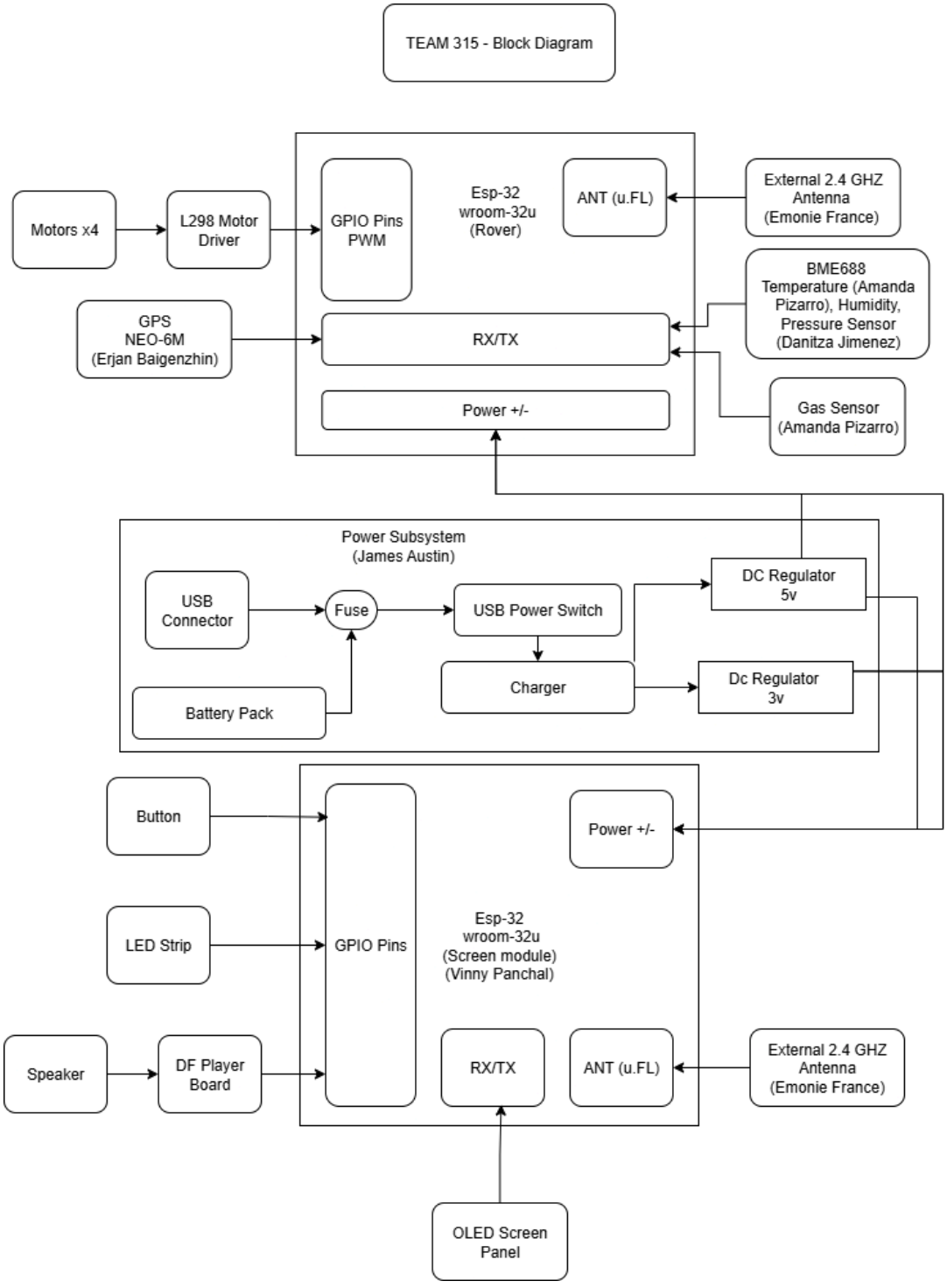
B. Final System Image

(Insert final picture of prototype. Live picture here as well)



IV. BLOCK DIAGRAM & SYSTEM DESIGN

A. Final Block Diagram



B. Block Diagram Rationale

We structured our block diagram in a way that separates each major subsystem so we could clearly understand how the rover, power system, and screen module all connect. At the time we created it, we already had our main idea but not the full execution plan, so the goal was to map out how every part would communicate and share power. Breaking the system into three main sections helped us organize our thinking: the rover electronics, the power subsystem, and the screen/HMI module. This made it easier for the team to see which parts depended on each other and which components needed dedicated pins, power rails, or communication lines.

This structure also lines up with the product requirements for EGR 314. We needed at least one complex actuator subsystem, so the L298 motor driver and motors satisfy that requirement. We also needed serial sensors and analog sensing, which we met through the BME688 (I^2C) and the gas sensors. The Rover ESP32 and the Screen ESP32 communicate over UART and mesh networking, which meets the microcontroller communication requirements. Keeping the block diagram simple and modular allowed us to plan for our PCB design, pin mapping, voltage regulation, and the final 100 × 100 mm board limit.

Responsibilities were divided based on each person's subsystem in the diagram. Erjan took the GPS and motor driver, Amanda and Danitza handled the BME688 and gas sensor branches, Emonie handled the external antennas, James handled the entire power subsystem, and Vinny handled the screen module and RX/TX communication. The block diagram layout made it clear who owned what and how all the pieces would come together.

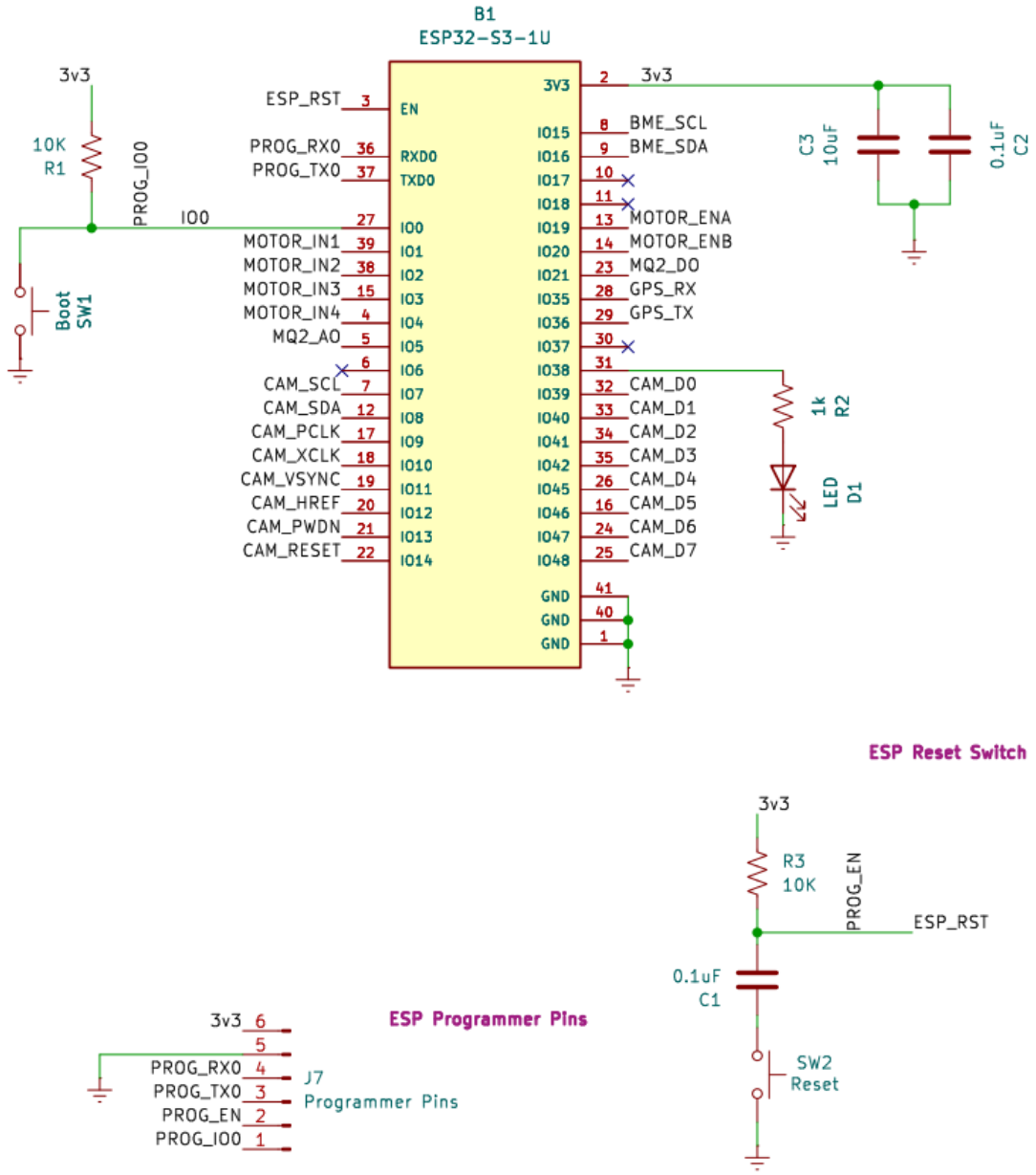
We also improved the diagram based on instructor feedback. We were asked to add the minimum and maximum voltages for each subsystem so we could decide whether a 12V, 9V, or 7V battery would be needed, and this pushed us to look deeper into voltage regulation and power routing. We were also told to upload the updated diagram to our website, which helped us clean up the labels and present the diagram in a clearer and more readable way.

C. Component Selection Summary

Number	Item Name	Subsystem	Link	Price	Quantity	Line Total
1	7" Touch Display	Screen / HMI	Link	$52.99	1	$52.99
2	LED Strip	Screen / HMI	Link	$13.99	1	$13.99

3	ESP 32 Cam 2.4 GHz	Communication, Screen / HMI, Rover	Link	$19.99	1	$19.99
4	2.4 GHz Antenna	Communication, Rover, Screen / HMI	Link	$9.99	1	$9.99
5	GY-NEO6MV2 NEO-6M GPS	Drive, Rover	Link	$9.99	1	$9.99
6	BME688 GAS SENSOR	Sensors, Rover	Link	$8.65	3	$25.95
7	2 L298n and 4 motors kit	Drive, Rover	Link	$15.99	1	$15.99
8	Battery	Power, Screen / HMI	Link	$15.99	2	$31.98
9	Camera	Rover, Sensors	Link	$27.99	1	$27.99
10	Rover Battery	Power, Rover	Link	$37.99	1	$37.99
11	18650 Holder	Power, Rover	Link	$9.99	1	$9.99
12	Backup MQ2 Gas Sensor	Sensors, Rover	Link	$11.99	1	$11.99
13	Buck Convertor	Rover, Power	Link	$7.39	1	$7.39
14	SHT40-CD1B-R3	Sensors, Rover	Link	$1.80	3	$5.40
15	I2C to UART	Communication	Link	$5.30	1	$5.30
16	Voltage regulator	Power	Link	$0.91	1	$0.91
17	Low-Dropout Regulator	Power	Link	$1.68	1	$1.68
Total						**$293.71**

D. Microcontroller Configuration / Pin Mapping



E. Power Budget

Power Budget

A. List ALL major components (active devices, integrated circuits, etc.) except for power sources, voltage regulators, resistors, capacitors, or passive elements							
All Major Components	**Component Name**	**Part Number**	**Supply Voltage Range**	**#**	**Absolute Maximum Current (mA)**	**Total Current (mA)**	**Unit**
	Gas Sensor	MQ-2	+3.3 - 5V	1	n/a	150	mA
	Camera	KC04	+5 - 12V	1	n/a	6	mA
	ESP32 programing	S006	+3.3 - 5V	1		80	mA
	Motor Driver	L298N	+5 - 35V	4	n/a	8000	mA
	Temp/Pressure Sensor	BME688	+1.2 - 3.6V	1	n/a	0.9	mA
	GPS module	GY-NEO 6MV2	+3.3 - 5V	1	n/a	67	mA
	Touch Display	ELECROW ESP32 Display 800×480	5V	1	n/a	2000	mA
	LED Strip	SEZO WS2812B ECO	5V	1	n/a	18000	mA

		LED Strip					
	Antenna	B00ZBJNO9O	+3.3 - 5V	2	n/a	0	mA
	GPS module	GY-NEO6MV2 NEO-6M GPS	+3.3 - 5V	1	n/a	45	mA
	ESP32 microcontroller	ESP32-S3-1U	+3.3 - 5V	1	n/a	260	mA

B. Assign each major component above to ONE power rail below. Try to minimize the number of different power rails in the design.

Add additional power rails or change the power rail voltages if needed.

+5V Power Rail	**Component Name**	**Part Number**	**Supply Voltage Range**	**#**	**Absolute Maximum Current (mA)**	**Total Current (mA)**	**Unit**
	Motor Driver	L298N	+5 - 35V	4	n/a	8000	mA
	ESP32 microcontroller	ESP32-S3-1U	+3.3 - 5V		n/a	260	mA
					Subtotal	8260	mA

					Safety Margin	25%	
					Total Current Required on +5V Rail	10325	mA
c2. Regulator or Source Choice	+7.4V Regulator	LM2596	+3.3 - 40V	1	3000	3000	mA
					Total Remaining Current Available on +5V Rail	7325	mA
-7.4V Power Rail	**Component Name**	**Part Number**	**Supply Voltage Range**	**#**	**Absolute Maximum Current (mA)**	**Total Current (mA)**	**Unit**
				1	100	0	mA
						0	mA
						0	mA
						0	mA
					Subtotal	0	mA
					Safety Margin	25%	

	Total Current Required on -5V Rail					0	mA
c3. Regulator or Source Choice	-7.4V Regulator			1	500	500	mA
	Total Remaining Current Available on -7.4V Rail					500	mA
+3.3V Power Rail	**Component Name**	**Part Number**	**Supply Voltage Range**	**#**	**Absolute Maximum Current (mA)**	**Total Current (mA)**	**Unit**
	Gas Sensor	MQ-2	+3.3 - 5V			150	mA
	Temp/Pressure Sensor	BME688	+1.2 - 3.6V			0.9	mA
	Camera	KC04	+5 - 12V			6	mA
	GPS module	GY-NEO 6MV2	+3.3 - 5V			67	mA
	ESP32 programing	S006	+3.3 - 5V			80	mA
					Subtotal	303.9	mA
					Safety Margin	25%	

	Total Current Required on +3.3V Rail					379.875	mA
c4. Regulator or Source Choice				1	500	500	mA
	Total Remaining Current Available on 3.3V Rail					120.125	mA

C. For each power rail above, select a specific voltage regulator using the same process as for major component selection. Confirm that the Total Remaining Current Available on each rail above is not negative.

D. Select a specific external power source (wall supply or battery) for your system, and confirm that it can supply all of the regulators for all of the power rails simultaneously. If you need multiple power sources, list each separately below and indicate which regulators will be connected to each supply. Confirm that the Total Remaining Current Available on each power source below is not negative.

External Power Source 1	**Component Name**	**Part Number**	**Supply Voltage Range**	**Output Voltage**	**Absolute Maximum Current (mA)**	**Total Current (mA)**	**Unit**
Power Source 1 Selection	Rover Battery	Zeee 7.4V 2S	7.4V	-7.4V		260	mA

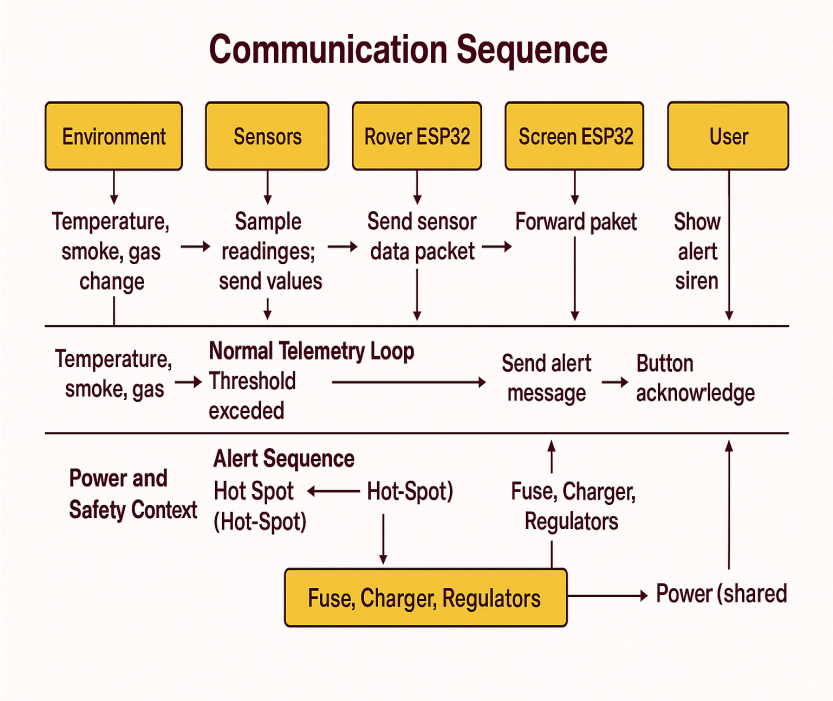
Power Rails Connected to External Power Source 1							mA
				1	0	0	mA
				1	0	0	mA
	Total Remaining Current Available on External Power Source 1					260	mA
External Power Source 2	**Component Name**	**Part Number**	**Supply Voltage Range**	**Output Voltage**	**Absolute Maximum Current (mA)**	**Total Current (mA)**	**Unit**
Power Source 2 Selection	18650 Rechargeable Battery	PAOWANG	3.6-4.2V		500	3500	mA
Power Rails Connected to External Power Source 2					500	3500	mA

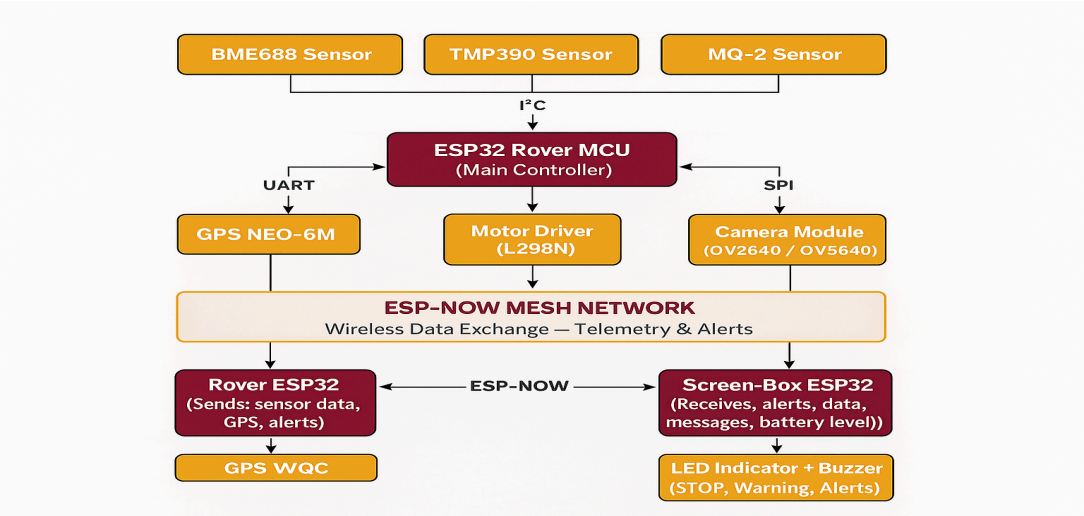
	Total Remaining Current Available on External Power Source 2					0	mA
E. Calculate Battery Life (if applicable). For each battery, also check the worst-case lifetime of the battery by indicating the capacity in mAh.							
	Component Name	**Part Number**	**Supply Voltage Range**		**Capacity (mAh)**	**Required By Regulators**	
	Battery	(full part number)	+12V		500	3500	
					Battery Life	0.1428571429	hours
Notes							
External Supply Voltage should be determined by the dropout voltage for highest-voltage regulator (e.g., +14V for a +12V regulator).							
If you have multiple units in your design (e.g., a base unit and remote unit) then you need a separate power budget for each unit							

The above power budget sheet shows how we estimated our power needs. As we have many components we ran the risk of not having sufficient enough power. Through this chart we were able to calculate the required power sources for each subsystem effectively.

V. COMMUNICATION ARCHITECTURE

A. Communication Sequence Diagram





B. Message Structure

We use ESP-NOW for wireless communication between the rover ESP32 and the screen-box ESP32. All data is sent as small binary packets to keep latency low and avoid wasting bandwidth.

We use two main packet types:

1. Telemetry Packet (Normal Loop)
Sent from Rover → Screen about 5–10 times per second while conditions are normal.

Telemetry Packet

Field Description

---

hdr_type 0x01 (telemetry)

device_id Rover ID (1 byte)

seq_num Packet counter for debugging

timestamp_ms Time since boot (uint32)

gps_lat Latitude (float)

gps_lon Longitude (float)

temp_c Temperature in °C (float)

humidity_pct Relative humidity (float)

pressure_pa Pressure (float)

gas_level Gas / smoke reading (float or int)

alert_state 0 = normal, 1 = warning, 2 = critical

battery_pct Rover battery estimate (0–100)

crc8 Simple checksum for packet integrity

2. Alert Packet (Hot-Spot Event)

Sent from Rover → Screen as soon as any threshold is exceeded. Also repeated a few times in case one packet is lost.

Alert Packet

Field	Description
hdr_type	0x02 (alert)
device_id	Rover ID
seq_num	Packet counter
timestamp_ms	Time of alert
gps_lat	Latitude at alert
gps_lon	Longitude at alert
max_sensor_id	Which sensor triggered first
max_sensor_val	Value that crossed the threshold
alert_level	1 = warning, 2 = critical
battery_pct	Rover battery estimate
crc8	Checksum

3. Acknowledge / Control Packet

Sent from Screen → Rover when the user presses the button to acknowledge an alert.

Ack / Control Packet

Field	Description
hdr_type	0x03 (ack / control)

device_id Screen ID

seq_num Packet counter

timestamp_ms Time of user action

ack_alert_level Last alert level acknowledged

command_flags Bits for simple commands (reset, silence, test)

crc8 Checksum

Using a small hdr_type byte lets us reuse the same ESP-NOW link while still supporting three different behaviors: normal streaming, high-priority alerts, and user control.

C. Initial Design Rationale

We structured the messages this way so that each packet carries only what is actually needed for that type of communication. Normal telemetry packets include the full set of sensor readings, location, and battery status so the screen can draw graphs and give the user a live picture of the environment. Alert packets remove extra fields and highlight the most important information at that moment: where the rover is, which sensor triggered, how bad it is, and whether the battery can support staying in the field. Ack packets stay very small since they only need to confirm that the user saw the alert or pressed a button.

This communication flow matches what our users need in the field. Firefighters mainly want two things: a quick view of current conditions and a clear signal when something is wrong. The normal telemetry loop supports slow “situational awareness” on the screen, while the alert path is a faster shortcut that jumps straight to warning the user with LEDs, sound, and an alert screen. Including GPS and sensor IDs in the alert packet makes it easier for a crew to know where to go and what kind of hazard they are dealing with.

For bandwidth and reliability, ESP-NOW was a good choice because it does not depend on Wi-Fi infrastructure and has low overhead. Our packets are under the ESP-NOW size limit, and we do not send big images or video, only numeric data. The normal telemetry rate is kept modest so we do not flood the link, while alert packets are repeated a few times to reduce the chance of losing a critical message. We use a sequence number and a simple checksum so we can detect dropped or corrupted packets during testing.

From a timing point of view, the rover samples sensors, builds packets, and sends them in a loop that easily fits within our goal of about one second or less from hazard detection to alert on

the screen. Keeping the packet format simple, fixed-length, and binary helped us meet that target without heavy parsing code or extra delays.

D. Updated Communication Method

Our project does not use ESP-NOW or ESP-Mesh.
Instead, the rover hosts its own dedicated AP network that the screen connects to.

Communication architecture:

- The rover's ESP32-WROOM-32U creates an AP with:
  - SSID: *TerraGuard-AP*
  - Password: *phoenixforce*
  - IP: *192.168.4.1*
- The ESP32-S3 Screen Module connects as a station (STA)
- All communication happens over this AP using HTTP GET.

Data paths in our system:

- Rover → Screen: real-time metrics from /metrics
- Rover → Screen: MJPEG camera stream from /stream
- Screen → Rover: periodic HTTP GET requests for metrics and video frames

This gives intentional two-way communication between two separate PCBs over a wireless network.
Even though the screen does not send commands back, its continuous HTTP requests still count as directional data flow.

E. Top 5 Software Changes Since Proposal

(Use a numbered list. Each item must include: issue → fix → impact.)

1. We originally planned to include full video streaming from the ESP32-CAM → We removed the camera subsystem → This was necessary because our PCB voltage regulator could not supply enough current, and removing the camera prevented brownouts and made the rover stable again.

2. We attempted to stream video to the screen module → We removed the stream viewer from the screen UI → This simplified the interface and prevented crashes caused by the regulator issue, allowing us to focus on reliable sensor reporting instead.

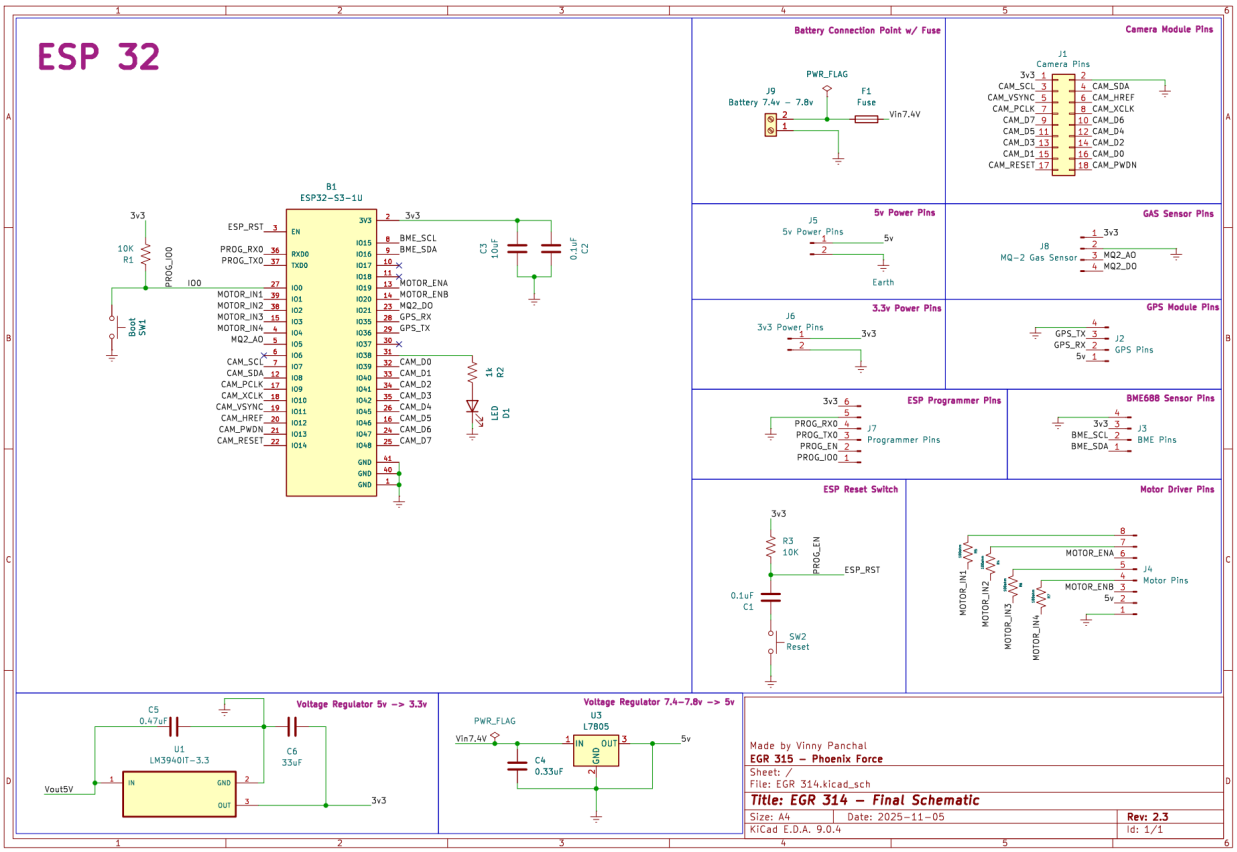
3. Our initial communication plan used ESP-NOW or ESP-Mesh for rover–screen communication → We switched to a dedicated AP hosted by the rover → The AP connection is more reliable, easier to debug, and does not require broadcast pairing or mesh synchronization, which helped the system perform consistently.

4. The screen originally only displayed sensor data → We added full drive controls to the screen UI → This combined command and monitoring into one interface, making the system easier to use and reducing the number of separate components needed for testing.

5. We started with many smaller subsystems, including the speaker and other optional features → We removed non-essential subsystems → This let us put more time and effort into making the core features stable, such as sensing, communication, and rover control.

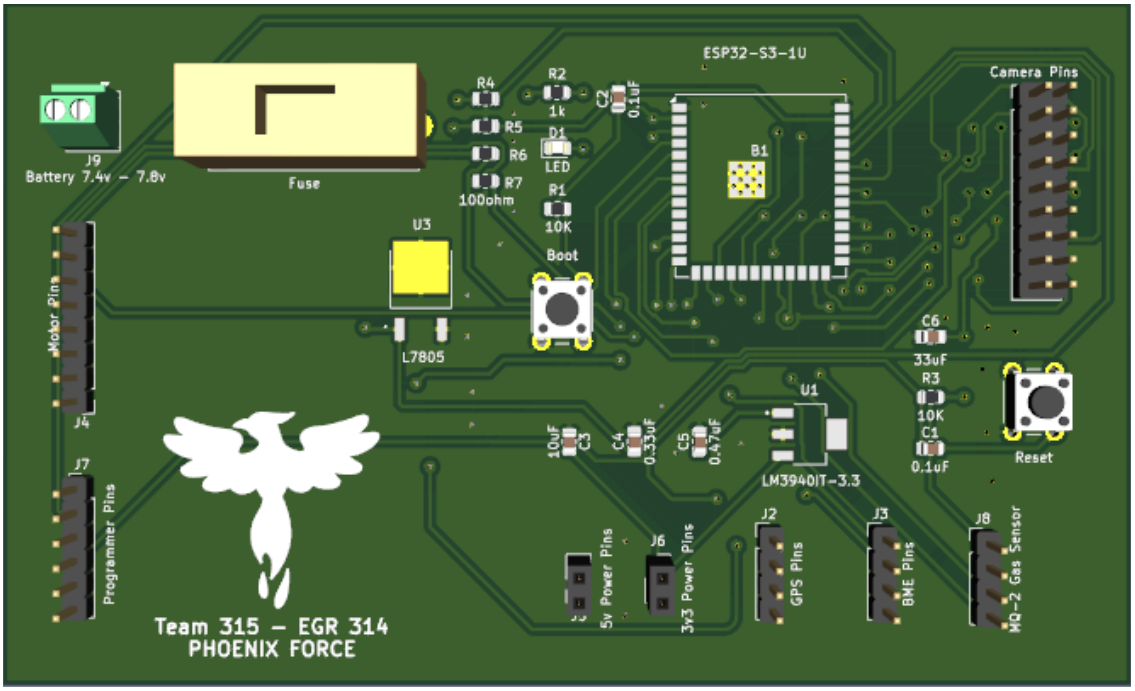
VI. HARDWARE DESIGN

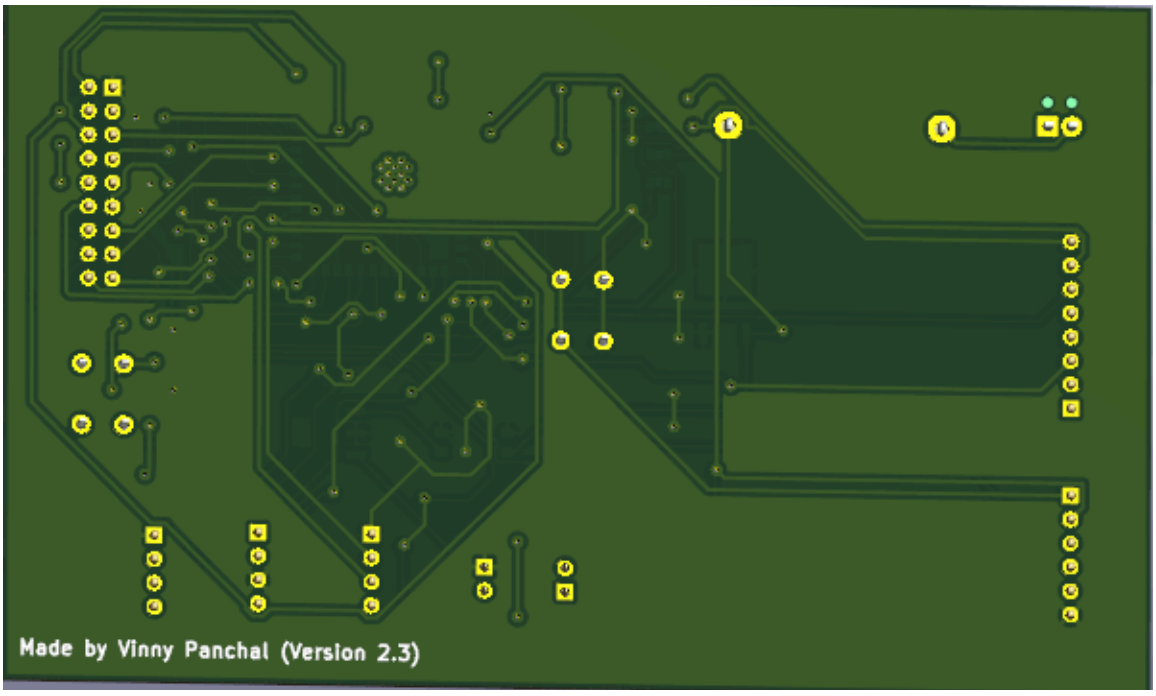
A. Updated Schematic (Final, Fabricated)



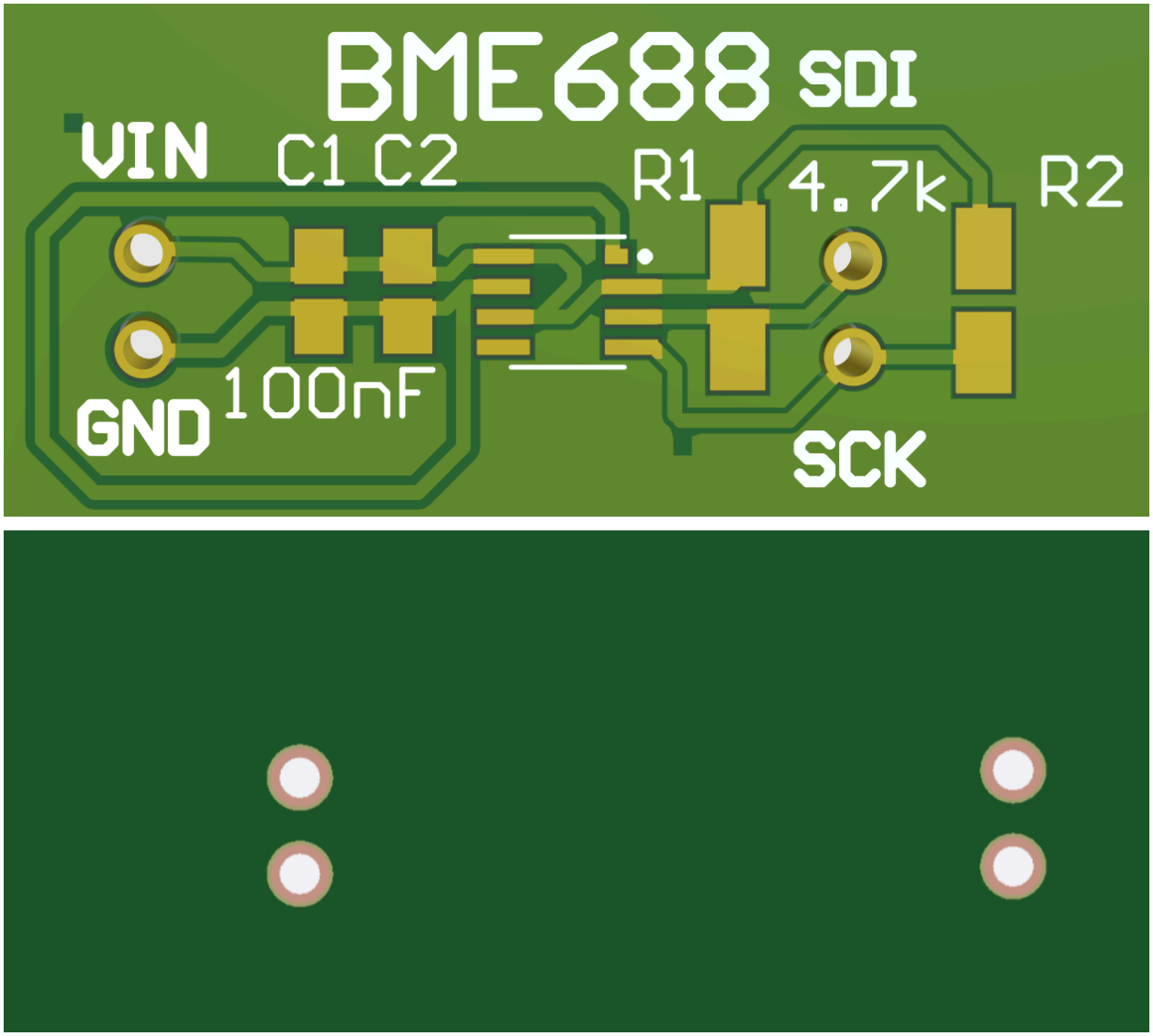
B. PCB (Front & Back Photos)

Main PCB Front and Back





BME688 PCB Front and Back



C. Hardware Functionality Discussion

Explain how the schematic and PCB satisfy product requirements:

- Power regulation
- Sensor integration
- Actuator/motor driver circuitry

- Safety considerations
- Routing and design constraints

D. Hardware Decision-Making Process

As each team member individually designed their subsystem's schematic, we decided to integrate of all of those subsystems into two different boards. The main PCB consists of the battery, a power regulator, a fuse, the ESP32 module, a reset button, and male header pins to interface with the rover's camera, motors, gas sensors, and the GPS. The second PCB contains the BME688 gas sensor. We decided to have a separate board for the gas sensor in order for it to be on the top of the rover instead of on the rover's body. This design choice allows for more accurate readings of the air, pressure, gas, and temperature.

Additionally, standard practices to ensure the efficiency of the PCB were undergone such as the integration of decoupling capacitors. These capacitors were placed as close to the chip as possible tin order to prevent noise and ensure a steady voltage supply.

E. Hardware Version 2.0

Discuss improvements for a future revision:

- Layout fixes
- Power improvements
- Routing changes
- Additional sensors or features
- Reliability upgrades

(½ page minimum.)

VII. SOFTWARE DESIGN

A. Overview of Code Structure

High-level explanation of how the code is organized.

B. Updated API Page

(Insert final API functions, parameters, return values, and message formats.)

C. Key State Machines, Tasks, or Interrupts

(Insert UML or flow diagrams as needed.)

VIII. SYSTEM VERIFICATION & RESULTS

A. Team System Verification Summary

Our system includes multiple feedback loops where sensor data triggers actuator behavior. Listed below are the “sensor → output” feedback-controlled responses

Sensors used:

- BME688 (air quality, gas resistance, temperature, humidity)
- MQ-2 Gas & Smoke Sensor
- GPS (NEO-6M)
- Camera

Feedback behavior:

- Gas readings → gas-type classification (clean air, VOCs, smoke/CO, methane, butane, danger)
- Air quality state → LED pattern logic
  (red, yellow, or green depending on severity)
- Loss of AP connection → hot-pink LED state

These LEDs were present for testing, but the main visual output is now handled on the screen module LEDs. Additionally, the screen has its own LED strip which reacts to rover data shown below

Screen Module:

- Not connected to AP → LED strip shows a pink pattern
- Good air → green LEDs
- Moderate/warning → yellow LEDs
- Poor air/danger → red LEDs

This fulfills the “sensor input → actuator output” requirement.

B. Actuators Demonstrated

Rover PCB Actuators:

- WS2812B LED strip
- Motor driver
- GPS

Screen PCB Actuators:

- LED strip with color-based air quality and connection feedback

C. Power Distribution

Only the rover PCB contains the power subsystem. The screen runs on its own battery and does not share power rails.

Rover Power System

- Battery pack → charger → 5V regulator → 3.3V regulator
- Rails power:
    - ESP32-WROOM-32U
    - BME688 sensor
    - MQ-2 gas sensor
    - GPS module
    - LED strip
    - Motor driver

We have verified the 3.3V sensor power and 5V LED power using a multimeter.

Screen Power

- The screen has its own power system/battery pack
- Not counted in subsystem power checks

D. Subsystem Overview

1. **ESP32-WROOM-32U Communication Core**
   Hosts the AP, metrics endpoint, and camera streaming server

2. **BME688 Environmental Sensor Subsystem**
   Temperature, humidity, gas resistance, air quality

3. **MQ-2 Gas & Smoke Sensor Subsystem**
   Detects smoke, CO, methane, propane/butane

4. **GPS Subsystem (NEO-6M)**
   Provides location and satellite data

5. **Motor Driver Subsystem**
   Part of the mechanical drive system (Erjan)

6. **Power Regulation Subsystem**
   Battery, charger, 5V regulator, 3.3V regulator
7. **Screen displays** real-time video, metrics, alerts, and system status. Connects to the rover's AP as a Wi-Fi station to request /metrics and /stream. Provides a user interface for driving, monitoring sensors, and receiving warnings (visibility alerts, obstacle detection, low battery, etc.).

E. Individual Team Contributions

Vinny Panchal – Screen/HMI Development

- Built the complete ESP32-S3 screen module, including the LCD display, UI layout, and subsystem interfaces

- Implemented the full LVGL UI, live metrics polling, and the MJPEG streaming viewer (before the camera was removed from the final design)

- Added LED color feedback, button controls, and speaker/DFPlayer alert functionality directly from the screen

- Integrated AP-based communication between the rover and the screen for data transfer and control

- Created system diagrams, helped with full-system integration, and supported debugging across the main PCB, sensor logic, GPS, and drive subsystems

Emonie France – Antennas, RF, and Camera Routing

- Managed the dual-antenna placement (u.FL 2.4 GHz) for stable AP communication
- Ensured reliable rover-to-screen connectivity through RF layout review and testing
- Conducted range tests and analyzed network quality throughout the verification area
- Evaluated antenna orientation, gain, and cable placement to reduce interference
- Assisted with **camera module routing** considerations, including signal integrity, layout spacing, and connector stability
- Ensured the rover's communication and camera subsystems maintained consistent performance under movement and load

Amanda Pizarro – Gas Sensing

- Integrated BME688 sensor
- Wrote initial gas-reading logic
- Helped define classification ranges

Danitza Jimenez – Air Quality & Data Interpretation

- Worked on data smoothing/validation for gas readings
- Helped interpret environmental sensor outputs

James Austin – Power Subsystem

- Designed and assembled the entire rover power system
- Implemented charger, regulators, and power rails
- Verified correct voltages on the final PCB

Erjan Baigenzhin – GPS & Drive System

- Integrated the GPS module and handled the UART configuration
- Developed the motor driver subsystem
- Assisted with drive testing and motor functionality

F. Demonstration Summary

During system verification, we demonstrated:

- Wireless communication over the rover AP
- Real-time metrics
- Screen LED responding to rover sensor values.
- Motor driver operation
- Power rails measured on the rover PCB.
- All rover PCB subsystems are functioning as designed.

G. Performance Metrics

Include measurable results (range, speed, accuracy, etc.)

Measure how far away you can b/w stable communication

Speed

Sensor accuracy

H. Images/Videos of Working System

(Embed YouTube videos if applicable.)

I. Discussion of Requirements vs. Achievements

The following is a list of the measurable targets we had initially set for our project (course-fit):

- Detect stimulus in ≤1-2 min (lab surrogate tests)
- Telemetry/alert latency ≤1 s device↔base (lab)
- Perimeter/localization error ≤25-50 m (field surrogate)

- ≥24-48 h battery runtime; weather-resistant enclosure
- ≥99% message delivery (n≥100)

To define our success, we can compare this to the project's achievements

- Successfully detects stimulus in ≤1-2 min (lab surrogate tests)
- Successful Telemetry/alert latency ≤1 s device↔base (lab)
- Perimeter/localization error successfully within ≤25-50 m (field surrogate)
- ≥99% message delivery (n≥100)
- Consistent demo

Overall, each requirement was met except for the water-resistant enclosure goal.

IX. TEAM REFLECTION

A. Lessons Learned

1. One of the most important things our team learned was that we should start system integration much earlier, because individual parts often work on their own but reveal completely new issues once they are combined.

2. We learned that dedicating time early to the PCB and power subsystem is critical, since unstable power rails, brownouts, and late PCB revisions caused delays that affected almost every other subsystem in our project.

3. Our team realized that communication methods need to be chosen and documented early, because switching between protocols or trying to fix routing issues at the end of the semester adds unnecessary stress and slows down testing.

4. We saw how helpful detailed block diagrams are, especially diagrams that include pin assignments, data flow, and subsystem interactions, because high-level diagrams alone did not give us enough information once we started debugging.

5. Another lesson we learned was that using different PCB software across team members created compatibility problems, so agreeing on either KiCad or Altium from the start would have made our final team PCB much easier to assemble.

6. We learned that debugging embedded systems requires patience and structure, and that isolating subsystems, printing logs, and creating simple test sketches is more effective than trying to debug the entire project at once.

7. Our team discovered that the ESP32 has limits when it comes to more advanced processing, especially with gas classification, and that a Raspberry Pi or similar device would be a better choice for machine-learning-based gas detection.

8. Testing the drive system showed us that small differences in motor output can cause drift or unpredictable behavior, which made it clear that using PID control early in development would greatly improve consistency and reliability.

9. We learned that RF performance depends heavily on antenna placement, cable routing, and the physical environment, which affected our metrics updates and camera streaming more than we expected.

10. Finally, we learned how important teamwork and communication are, especially when multiple subsystems depend on each other, because staying organized and communicating clearly helped us push through unexpected issues near the end of the project.

B. Recommendations for Future Students

1. Future students should begin learning PCB design and power regulation early, because these areas take time to understand and affect every other part of the project.

2. Students should choose one PCB tool as a team at the beginning of the semester to avoid problems with converting files or mixing KiCad and Altium designs near the deadline.

3. It is important for students to practice writing detailed block diagrams and system maps that include data flow and pin assignments, since these diagrams guide the entire build process.

4. Students should learn about PID control and basic motor tuning early on, because these skills make the drive system more predictable and save a lot of debugging time later.

5. Future students should review communication protocols such as AP mode and ESP-NOW before starting the project, since understanding these methods early makes integration much smoother.

C. Version 2.0 – Communication Architecture (½ page minimum)

- Improvements

- Debugging tools
- New protocol features
- Reliability and timing enhancements
- Better packet design

X. RESOURCES PAGE ITEMS

(These will go on the GitHub team site, but list them here for completeness.)

Team images

Videos

CAD ZIP files

Code ZIP files

Gerbers & ECAD project ZIP

Final poster JPG & PDF

XI. CONCLUSION

The project's primary objective was to prevent and detect wildfires, and (if ignition occurs) quickly guide responders with clear info. As a result, we had created a reliable and realistic wildfire-detection prototype that can identify early warning signs, communicate without public Wi-Fi, and present clear alerts through a simple, glove-friendly interface. Through our

repeatable and consistent demo, our prototype is able to send low-latency alerts and consistently communicate accurate readings on environmental factors that can indicate a wildfire.

Our design aims to maximize effeciency and eliminate time wasted in wildfire detection and response through simple and sturdy human interfacing. This focus on quickness and efficiency is important when considering how crucial every second is when fighting wildfires. Being able to have reliable connection when utilizing the TerraGaurd rover even without Wi-Fi was also another key factor as many Arizona wildfires take place on land without dependable service.

The TerraGaurd rover has a range of [[SPEED, RANGE, AND METRICS HERE]

This project successfully demonstrates and features the following key capabilities:

- Local sensing + glove-friendly on-device alerts (screen + buttons)
- Perimeter estimation and geo-tagged reports to a base unit/map
- Internet-optional comms with acks/retries and link-loss UX

From here, we could further improve the project's design by creating the rover with fire and water-resistant materials. [more improvements here please]

REFERENCES AND DATASHEETS

[1] "Neo-6 u-blox 6 GPS modules data sheet abstract," datasheethub, https://datasheethub.com/wp-content/uploads/2022/08/NEO6MV2-GPS-Module-Datasheet.pdf (accessed Dec. 5, 2025).

[2] "ESP32-CAM-MB Datasheet", AI-Thinker, https://components101.com/modules/esp32-cam-camera-module (accessed Dec. 5, 2025).

[3] "Technical data MQ-2 gas sensor datasheet", https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf (accessed Dec. 5, 2025).

[4] "LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator," Texas Instruments,

https://www.ti.com/lit/ds/symlink/lmk04832.pdf?ts=1757271304952&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLMK04832%253Futm_source%253Dgoogle%2526utm_medium%253Dcpc%2526utm_campaign%253Dasc-null-null-GPN_EN-cpc-pf-google-ww_en_cons%2526utm_content%253DLMK04832%2526ds_k%253DLMK04832%2526DCM%253Dyes%2526gclsrc%253Daw.ds%2526gad_source%253D1%2526gad_campaignid%253D14388345080%2526gclid%253DEAIaIQobChMIq-DP26nHjwMVlilECB3HAyU8EAAYASAAEgLhF_D_BwE (accessed Dec. 5, 2025).

[5] “Bosch Sensortec | BME688 Datasheet 1 | 60”, Bosch, https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme688-ds000.pdf (accessed Dec. 5, 2025).

[6] “Positive voltage regulator ICs Datasheet,” STMicroelectronics , https://www.st.com/content/ccc/resource/technical/document/datasheet/41/4f/b3/b0/12/d4/47/88/CD00000444.pdf/files/CD00000444.pdf/jcr:content/translations/en.CD00000444.pdf (accessed Dec. 5, 2025).

[7] “LM3940 1-A Low-Dropout Regulator for 5-V to 3.3-V Conversion,” Texas Instruments, https://www.ti.com/lit/ds/symlink/sn74lvch245a.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1746657274528&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fsn74lvch245a (accessed Dec. 5, 2025).

APPENDICES

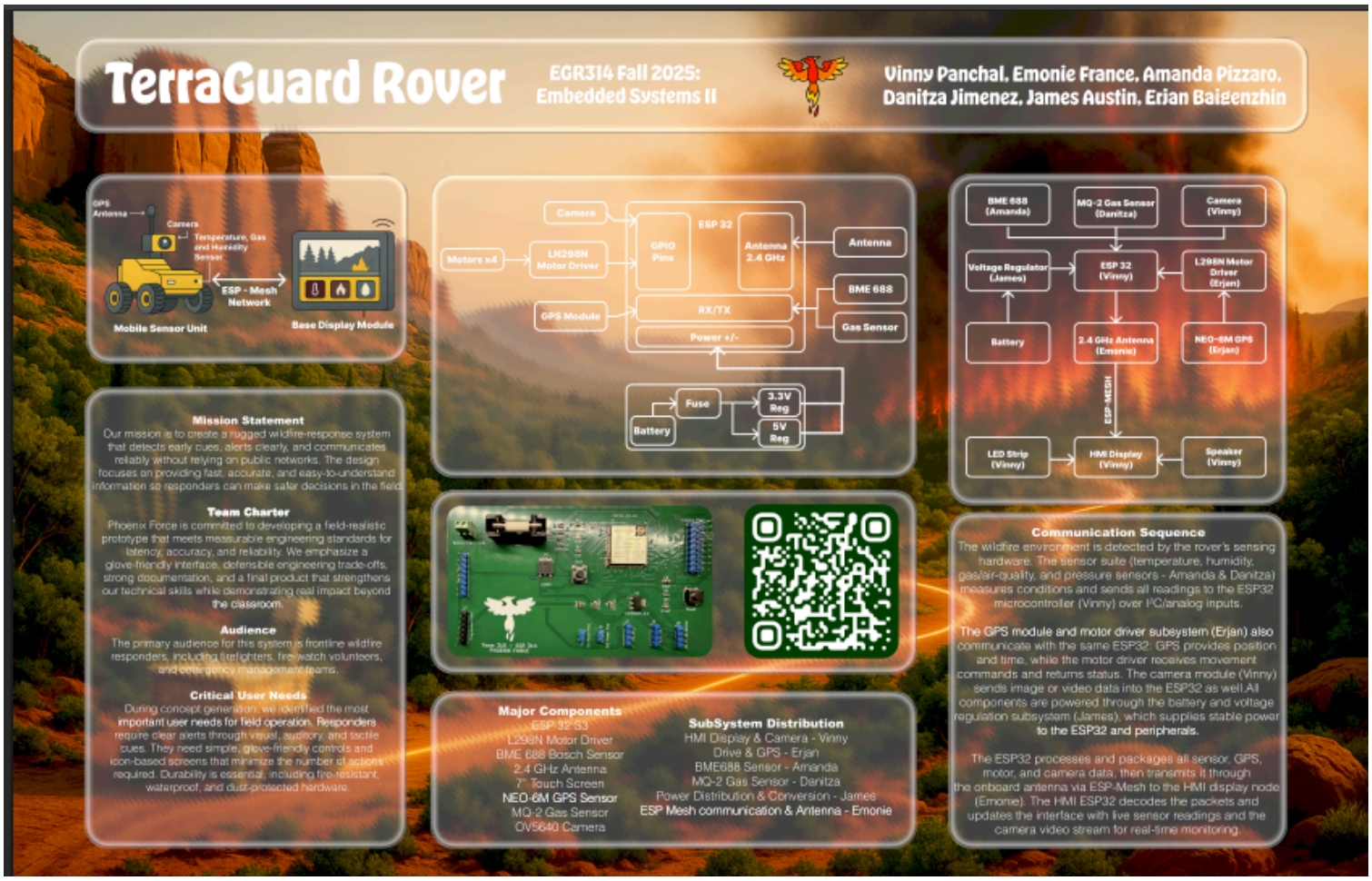
Appendix A — Full BOM

Number	Item Name	Subsystem	Link	Price	Quantity	Line Total
1	7" Touch Display	Screen / HMI	Link	$52.99	1	$52.99
2	LED Strip	Screen / HMI	Link	$13.99	1	$13.99

3	ESP 32 Cam 2.4 ghz	Communication, Screen / HMI, Rover	Link	$19.99	1	$19.99
4	2.4 ghz Antenna	Communication, Rover, Screen / HMI	Link	$9.99	1	$9.99
5	GY-NEO6MV2 NEO-6M GPS	Drive, Rover	Link	$9.99	1	$9.99
6	BME688 GAS SENSOR	Sensors, Rover	Link	$8.65	3	$25.95
7	2 L298n and 4 motors kit	Drive, Rover	Link	$15.99	1	$15.99
8	Battery	Power, Screen / HMI	Link	$15.99	2	$31.98
9	Camera	Rover, Sensors	Link	$27.99	1	$27.99
10	Rover Battery	Power, Rover	Link	$37.99	1	$37.99
11	18650 Holder	Power, Rover	Link	$9.99	1	$9.99
12	Backup MQ2 Gas Sensor	Sensors, Rover	Link	$11.99	1	$11.99
13	Buck Convertor	Rover, Power	Link	$7.39	1	$7.39
14	SHT40-CD1B-R3	Sensors, Rover	Link	$1.80	3	$5.40
15	I2C to UART	Communication	Link	$5.30	1	$5.30
16	Voltage regulator	Power	Link	$0.91	1	$0.91
17	Low-Dropout Regulator	Power	Link	$1.68	1	$1.68
Total						**$293.71**

Appendix B — Full Code (Optional or Link to ZIP)

Appendix C — Presentation Poster



Appendix D — more Additional Images, Diagrams, or Calculations