Wireless Mesh Network for Pesticide Spray Monitoring and Mapping

Design Document

Team Number: sdmay25-04
Client: Claussen Labs, Nathan Jared & Griffin Ellis
Advisor: Nathan Neihart

Team Members/Roles

Henry Hingst: Group Leader & Software Lead
Ashley Falcon: Software Engineer
Drew Scheidler: Software Engineer
Hector Perez Prieto: Hardware Lead
Yok Quan Ong: Hardware Engineer
Wesley Smith: Hardware Engineer

Team Email: sdmay25-04@iastate.edu
Team Website: https://sdmay25-04.sd.ece.iastate.edu

Revised: 5/4/2025

Executive Summary

With the global population projected to reach 9.7 billion by 2050, the demand for increased food production has never been more critical. Pesticides play a pivotal role in achieving higher crop yields, accounting for a 20–40% increase depending on the crop. However, proper pesticide application is essential to maximize efficiency while minimizing waste and environmental harm. Understanding pesticide spray distribution is vital for optimizing application techniques and ensuring coverage in target areas.

Our project focused on developing a wireless network using ESP32 microcontrollers to measure pesticide spray distribution across different levels of the crop canopy. Interdigitated electrodes (IDEs) designed by Claussen Labs act as sensors to measure resistance, which correlates with pesticide coverage. By deploying IDEs at three canopy levels and integrating them with ESP32-based nodes, our project provides real-time, localized data on pesticide distribution under different application methods (e.g., drones, booms, airplanes).

Key design requirements include:

- Collecting resistance data from three IDEs at each canopy level to calculate an average.
- Ensuring each post is monitored by three ESP32 microcontrollers.
- Establishing a centralized node for data retrieval, enabling user-friendly data collection.
- Developing a robust wireless network consisting of six to twelve nodes for reliable communication across the canopy.

Our design employs ESP32 microcontrollers due to their cost-effectiveness, and an additional LoRa module for long-range Wi-Fi capabilities. The resistance measurements taken from the IDEs are fed through a separate PCB made up of three identical circuits consisting of a voltage divider circuit and a comparator. The circuitry allows us to accurately convert a wide range of resistance measurements to voltage. The ESP32s are connected to the output of the three analog to digital converters dedicated to the three measurement circuits. They convert digitized voltage values to resistance measurements. The final data is stored on an SD card and then transmitted via our LoRa module. An additional microcontroller will serve as the centralized hub to collect and store the data in .txt files for post-analysis by our clients at Claussen Labs.

The final design aligns well with our clients' updated requirements and addresses user needs. The clearest evidence of this is its fulfillment of the essential requirements. Firstly, the final product is able to take resistance measurements over an extremely large range, an update

made by our clients in December. Data from three separate IDEs is collected by an individual microcontroller and stored locally on an SD card, as requested. Additionally, our LoRa module is able to communicate well-over the required distance, ensuring our nodes can individually communicate to a base node. The measurement nodes also utilize unique and identifiable addresses to allow researchers to accurately map pesticide levels across the field. The centralized nature of our design is by far the most important requirement, as it takes into account the users' need for efficient data collection and mapping.

Originally, we anticipated using a mesh network that transmitted data between nodes and then to a central station. However, we now have employed a separate LoRa module that allows for long range communication and does not require mesh network capabilities. The implications of this design decision is that it is far easier to implement additional nodes. This may also eventually limit ranges deployed outside of a research environment. Thus, our design will require refinement if ever implemented on a commercial scale. Another step to be taken could be to implement a graphical user interface (GUI). Our current interface, while user-friendly for a research group, could use simplification if ever accessed by the average consumer.

Learning Summary

Development Standards & Practices Used

While pushing ahead in our project, we have implemented good practices in the hardware and software facets. Some hardware practices include the following:

- Proper grounding measures
- Implementation of safeguards such as diodes to stay within manufacturer-provided voltage inputs
- Verification of pull-up and pull-down resistors to avoid floating states and therefore unexpected digital responses.
- Conduction of hardware simulation before building physical circuits
- Selection of energy-efficient parts

Similarly, it is important to consider software practices, including:

- Adherence to C programming principles for ESP32 microcontrollers.
- Modulation of code to improve performance, flexibility, and decoding.
- Use of a Git library to collaborate among group members.
- Utilization of manufacturer-provided libraries
- Standardization of data text files to ensure reliability and readability.

The usage of standards is at the heart of any engineering practice related to our field. There are numerous IEEE standards applicable to our project. One is IEEE 802.11 lr, which governs Wi-Fi communication across the LoRa network. Another of relevance is IEEE 1588, which enables precise synchronization of clocks in measurement systems related to networks.

Summary of Requirements

As aforementioned, our key requirements include the following:

- Collecting resistance data from three IDEs at each canopy level to calculate an average.
- Ensuring each post is monitored by three ESP32 microcontrollers.
- Establishing a centralized node for data retrieval, enabling user-friendly data collection.
- Developing a robust wireless network consisting of 6–12 nodes for reliable communication across the canopy.

Applicable Courses from Iowa State University Curriculum

Some of the most fundamental courses for our degree areas apply to our project. For instance, the core principles of circuit design covered in EE 2010 and EE 2300 have been invaluable to our hardware development. Without a firm grasp of operational amplifiers, diodes, voltage regulators, and voltage divider, our team would not have been nearly as successful this semester. The basics of troubleshooting and simulation techniques covered in these courses have also made the team far more efficient. EE 3030 has also been useful in determining power requirements and battery selection.

On the software side, one of the most influential classes has been CprE 2880. The class covered the fundamentals of embedded systems programming. It has been incredibly helpful in our development of low-level code that allows the microcontroller to read data, write it to a text file, and then transfer it. Other basic programming classes, such as ComS 2270 and EE 2850, have enforced our good practices in documenting code clearly and pulling from APIs. Finally, ComS 3090 has bolstered our understanding of software development. It has allowed us to more easily navigate the complexities of managing front-end and back-end team structure, which mirrors our software-hardware interaction and communication.

New Skills/Knowledge Acquired

Both as a team and as individuals, we have acquired an array of new skills and knowledge that were not explicitly taught in our curriculum. As a team, we have acquired plenty of knowledge specific to our microcontroller and overall project scheme. Previous to senior design, none of us had had the opportunity to work with large networks. It has taken plenty of time, effort, and focus just to understand the concept. In particular, it has challenged our software team to design communication schemes that can track data with time stamps and indications of which microcontroller or node data has been derived. The process requires the use of synchronous clocks, structured Wi-Fi packages, and other practices that we were either unaware of or simply not equipped for. We also have had to learn how to use the libraries manufactured for our specific microcontroller to utilize the external ADC, store data locally on an SD card, and communicate between microcontrollers. It has taken patience to attain new skills and knowledge, but our software team has taken leaps and bounds to expand our understanding and begin putting ideas into action.

Our hardware team has also acquired plenty of new experience and skills in order to guide our project toward success. The members were required to learn new practices related to PCB

design, layout and assembly. Through simulation and testing, our team has engineered a highly accurate circuit to measure the most fundamental data (i.e. sensor saturation). The team has also learned how to do worst-case analysis sweeps. The strategy allows us to account for and calibrate our errors that we anticipate in our final design, specially related to fluctuations in resistance measurements. The hardware team has made strides to develop the most precise circuit through patience, research, and asking for advisor feedback.

Table of Contents

List of Figures/Tables/Symbols/Definitions

1. Introduction

1.1 Problem Statement

With the global population projected to reach 9.7 billion by 2050, the demand for food will require a 60%-110% increase in production. Improved pesticide use will be necessary to improve crop yields and achieve this goal. However, increasing the efficiency of pesticide application requires new techniques to spread pesticides more precisely and minimize waste.

Our project focuses on the creation of a wireless network using microcontrollers to measure the pesticide spray distribution of a corn crop. Sensors developed by the Claussen Labs research group will be positioned at three levels of the canopy to gather resistance readings. These readings will provide a numerical value for the amount of pesticide reaching these three levels. Data collected from the network will be transmitted to a central node, a microcontroller, allowing users to analyze pesticide spray efficiency and optimize future pesticide application practices.

1.2 Intended Users

Several key user groups will interact with or benefit from the product, including researchers, graduate students, and farmers. Each of these has distinct needs related to the project, and the product will provide them with valuable data to help optimize pesticide usage and improve agricultural productivity.

Nathan Jared, a researcher and PhD student at Iowa State University working with Claussen Labs, has a background in Mechanical Engineering and experience in Chemical Engineering research. His role involves developing sensors to measure pesticide distribution. Nathan needs accurate data from the IDEs (interdigitated electrodes) placed throughout the cornfield to evaluate the efficacy of his designs. By obtaining precise resistance readings from the sensors, Nathan can identify inefficiencies in the current design and adjust the sensors accordingly to enhance their performance. The wireless mesh network we are developing will allow Nathan to receive data from many nodes at once, minimizing the need for manual data collection and reducing the time needed for troubleshooting. This data will ultimately help him refine the sensor design, improving pesticide measurement accuracy, and contributing to more efficient crop management practices.

Griffin Ellis, a graduate student also working at Claussen Labs, collaborates closely with Nathan Jared and the Senior Design Project team. As a Mechanical Engineering student, Griffin focuses on the technical aspects of the project, such as system calibration and programming. His main goal is to ensure the sensors function correctly and that their data can be used to refine his calibration algorithms. Griffin needs access to data from the IDEs to fine-tune the software responsible for analyzing pesticide distribution. The wireless mesh network we are building will provide Griffin with the data necessary for calibration, making his process more efficient by providing consistent and reliable feedback from the

sensors. This will help him better understand the real-world performance of his program and optimize the system for precise pesticide application.

Farmers can majorly benefit from the technology of our project. Farmers rely on efficient farming practices to maintain the health and productivity of their crops. Their primary need is to ensure that their fields are evenly coated with pesticides, preventing both under-application, which could lead to pest issues, and over-application, which would waste resources and harm the environment. The product we are developing offers a significant benefit to farmers by providing real-time data on pesticide distribution. This information will help them adjust their pesticide application methods, ensuring that their fields receive the proper coverage and reducing the risk of crop damage or pesticide runoff. Ultimately, this will lead to improved crop yields, reduced costs, and more sustainable farming practices goals that align with the overarching problem of increasing food production efficiency to meet global demand.

Appendix

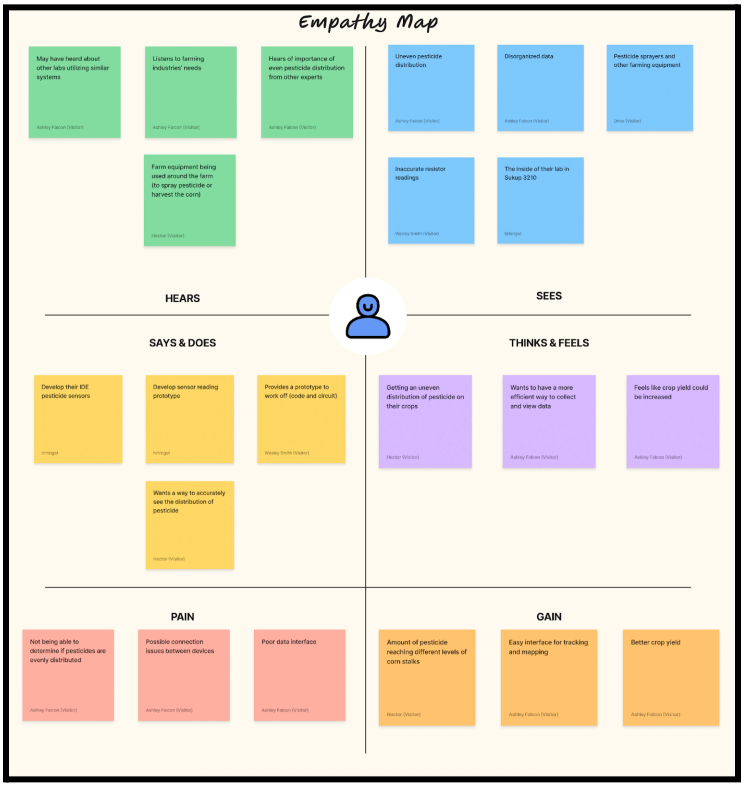


Figure 1.2.1: Empathy map of the user base for this project

2. Requirement, Constraints, and Standards

2.1 Requirements & Constraints

Functional Requirements:

The wireless network for pesticide spray monitoring and mapping will have to store the measurement of resistance across the interdigitated electrodes (IDEs). These IDEs will be placed at three levels of the crop canopy with ESP32 microcontrollers which will transmit the data (resistances) to the master node. The wireless network will be implemented using ESP32 microcontrollers (MCU) and the long range wifi capabilities of LoRa modules. The transmitted data will be stored in a user-friendly format, such as a .txt file, for future analysis. The network supports real-time data monitoring, allowing users to check measurement consistency. The system will also adhere to low power consumption. The network will have a minimum of six to twelve nodes.

Resources Requirements:

Sensors: The sensors (IDEs) will be placed on three levels of a corn canopy. Each will be connected to an ESP32 microcontroller. Each level will have a minimum of three IDEs, so each pole will consist of nine IDEs and three ESP32 microcontrollers, and the sensor's accuracy will be ±1% of actual resistance with minimal noise. The sensor will be operated for three hours on battery power and will send voltages to the microcontrollers that are connected which will communicate and send information to the master node.

Master Node:We must have a master node and a centralized device to communicate to all the nodes and receive data from them, this device will be another ESP32 microcontroller. This master node will collect all the data being received and organize it in a user-friendly format that can be accessed at any time. The master node will have a sleep/wake command for the nodes to verify that the nodes are functioning.

Long-Range Communication: Each ESP32 will act as a node in the networking system. The ESP32 will deploy LoRa modules that have the capabilities of long-term and reliable communication and communicate directly to the master node. The transmitted data should not be lost and will be communicated with a minimum of nine sensor nodes. Each of the nodes will have a distance of two-hundred feet from other nodes, this way the nodes are still able to communicate with one another. The master node will periodically send commands to acquire information relating to the actual data and the operational status of the nodes. A useful component will be the ability to determine if data packets are lost and determine errors in data accuracy. This networking system will support real-time data transmission, which allows users to receive data from the master node.

2.2 Engineering Standards

IEEE 802.11 [1]

This standard defines protocols for creating a wireless network using Wi-Fi and supports broadcast/multicast and unicast data delivery. This will apply to our transmission on the data line. Multiple data transmissions can occur to and from the master node.

IEEE 1588 [2]

This standard defines synchronizing time across distributed systems using precision time protocol (PTP) and ensuring high-accuracy timing. This is important to our project because we are going to be receiving multiple values from the sensors which will be transmitted to the master node. Our system requires consistent and accurate timing which will provide the user with accurate and reliable real-time data analysis.

IEEE 802.15.1 [3]

This covers Bluetooth BR/EDR (Basic Rate/ Enhanced Data Rate). This standard is designed for wireless networks and low-power wireless communication between devices. In our project, Bluetooth LE is designed and will be used for low-energy and periodic communication.

IEEE 1801 [8]

This standard is designed for power management and low power in integrated circuits (IC). Our circuit contains integrated circuits which help with the power efficiency of our circuit as a whole.

3. Project Plan

3.1 Project Management/Tracking Procedures

We've chosen the waterfall approach as our project management plan because this will allow us to achieve our goal step by step and effectively. This approach fits best in our project because it involves iterative execution and does not advance to the next phase until we finish the current phase.

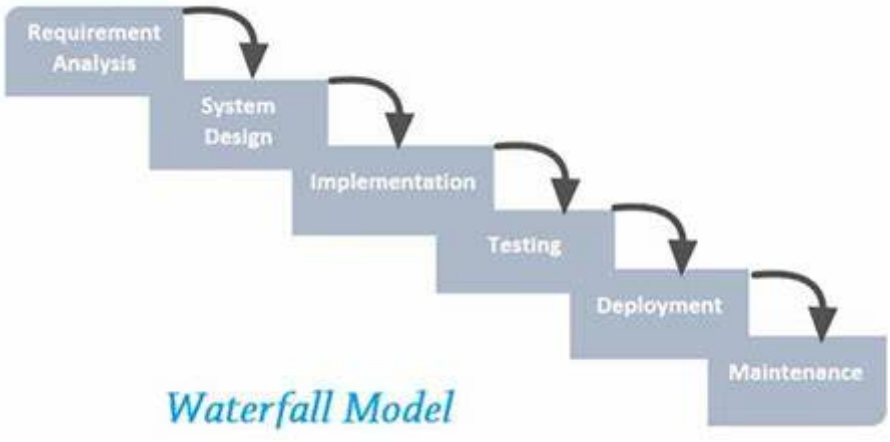


Figure 3.1.1: Project Management Style: Waterfall Model

Our team is divided into two groups: software and hardware. Software groups leveraged platforms like GitHub as the primary repository. The source code is primarily uploaded to a GitHub repository to share with all the other team members. Meanwhile, the hardware group shares the simulation files among team members. They collaborate closely on circuit testing, and ensure that hardware and software components are aligned through each project phase. For general documentation and notes, a shared Google Drive is utilized.

3.2 Task Decomposition

Our focus is building a prototype with a single pole with three levels of microcontrollers each with three IDEs rather than replicating the whole system setup. The final system will easily be scaled by duplicating our initial design. The diagram shows a stage approach for our project. Tasks range from designing a circuit to read the sensor data to receiving and transmitting data through the completion of a reliable network.

Following the diagram from the bottom up, the hardware group members will start by designing circuits that allow sensors to read accurate data. We designed a voltage divider circuit with our sensor, providing an output range of 0.2V - 2V due to the use of external ADC. After testing and analysis, the circuit was refined and transformed into a PCB. Software group members then write firmware to implement the

voltage divider circuit that can capture the sensor reading accurately and stably. This firmware will facilitate the gathering of data from an external ADC with a 16 bit capacity. After that, we will design an SD card circuit to write and store our data in the SD card for backup. Once again, the software team will collaborate to create the firmware.

Next, we will implement the data transfer with 802.11 LR, setting up the LoRa network with dummy data and testing the functionality of the network using heartbeat packets and range testing to ensure distance requirements are met. Once the network is functioning properly, we will start collecting real-time data from the sensor and aggregating the data into a .txt file into the SD card and the base station.

Once we are confident that our data is transmitting correctly, we will set up a server in which an external PC can download the text file from the base station. We will need to ensure the file is formatted in a way that's easily readable by those using the data.

Figure 3.2.1: Flowchart of Our Project

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Goal 1: Collect data from poles and combine them into one .txt file on the base station.

- **Task 1:** Data transfer with LoRa
    - Achieve a data transmission rate of at least 95% packet success over a distance of at least 200 feet of the base node. The progress will be evaluated by ensuring successful heartbeat packages are received from the base node.
    - Design package structure.
- **Task 2:** Implement Network with Dummy Data
    - Achieve a packet reliability of at least 90% with a redundancy built-in in the system. The progress will be measuring the reliability of the network by the ratio of the successfully routed packets and the total packets sent.
    - Create directory in order to accurately save measurements to a .txt file that are easy to interpret

Goal 2: Base Nodal Control

- **Task 1:** Send and receive commands
    - We will send out heartbeat packages that ensure measurement nodes are online and able to communicate with the base station
    - User will be able to interact with it via an interface

Goal 3: Write Sensor Data to SD Card

- **Task 1:** Read Data from Sensor
    - Design and implement a voltage divider circuit and coding correspond to the firmware.
        - **Subtask 1:** Design Voltage Divider Circuit
            - Design a circuit that can output consistent voltage readings less than an error of 1%.
            - Utilize 16-bit external ADC
        - **Subtask 2:** Write firmware to implement Voltage Divider Circuit and ADC
            - Design a firmware that can read resistance values with an error of less than 0.5%, taking advantage of an external ADC and conversion calculations. The value will be compared to the expected value to ensure the accuracy.
- **Task 2:** Test Writing Data to an SD Card
    - Design a circuit and firmware for the SD card reading and writing.
        - **Subtask 1:** Design a circuit for SD card
            - The circuit will be able to let the SD card write and read data with an error of 0.5%. This will be monitored by testing the error rate through the writing operation and compared to the total operations.
        - **Subtask 2:** Write firmware for the SD card functionality
            - The firmware should be able to write data with an error rate of less than 0.5%. This will be measured by the speed of data writing for fixed size and verify the error of less than 0.5% data corruption across multiple write and read operations.

3.4 Project Timeline/Schedule

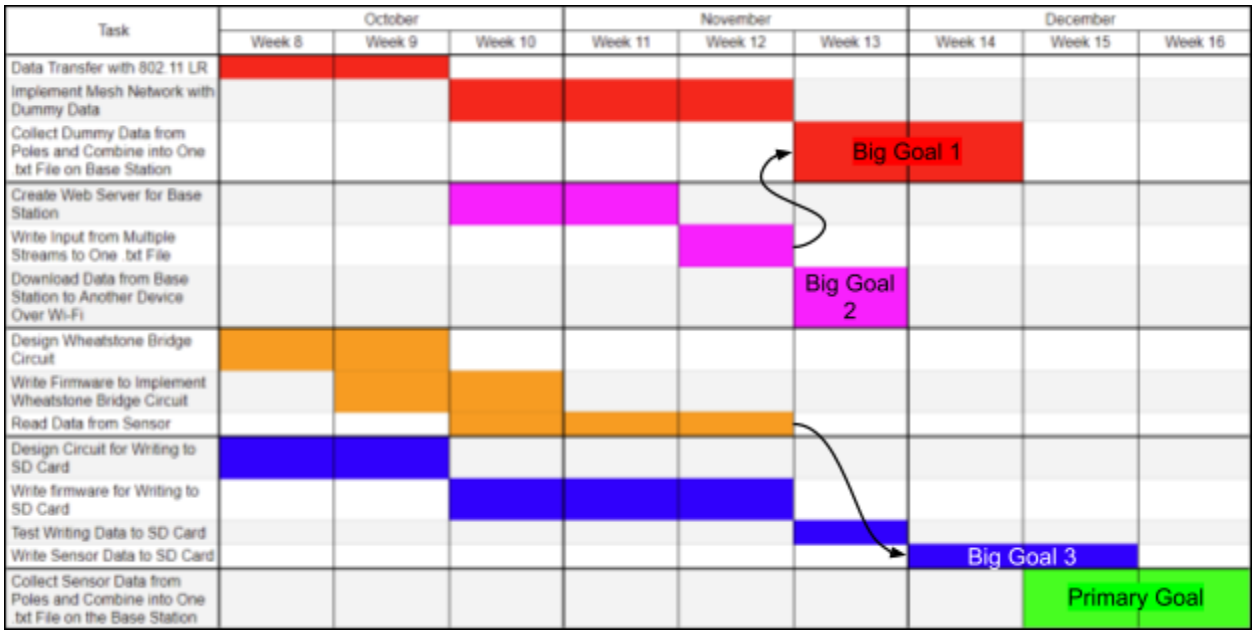


Figure 3.4.1: Semester 1 Timeline of Our Project

This is the rough timeline that we created for this class at the beginning of the first semester. We made one Primary Goal, collecting sensor data from the poles and combining it all into one text file at the base station, as it was the endpoint of our project. We divided this Primary goal into 3 “Big Goals,” collecting dummy data from the poles and combining it all into one text file at the base station (complete by the end of week 14), downloading data from the base station to another device (complete by end of week 13), and writing sensor data to an SD card (complete by end of week 15). Each of these 3 “Big Goals” were broken down into sub-goals, with sequential deadlines.

Our timeline for the second semester changed drastically, as seen below in Figure 3.4.2. Almost none of our originally completed tasks from the first semester directly impacted our timeline for our second. This was a result of four drastic changes: 1) we had to update the circuit entirely to account for a larger range of resistance values, thus forcing us to abandon the wheatstone bridge design, 2) we switched to the utilization of a LoRa module rather than a mesh network, 3) we decided to utilize an external rather than internal ADC, rendering our original code null, and 4) we decided to use a direct connection rather than wireless one to download data off of the base station. For further elaboration on these design changes, please refer to Appendix 2.

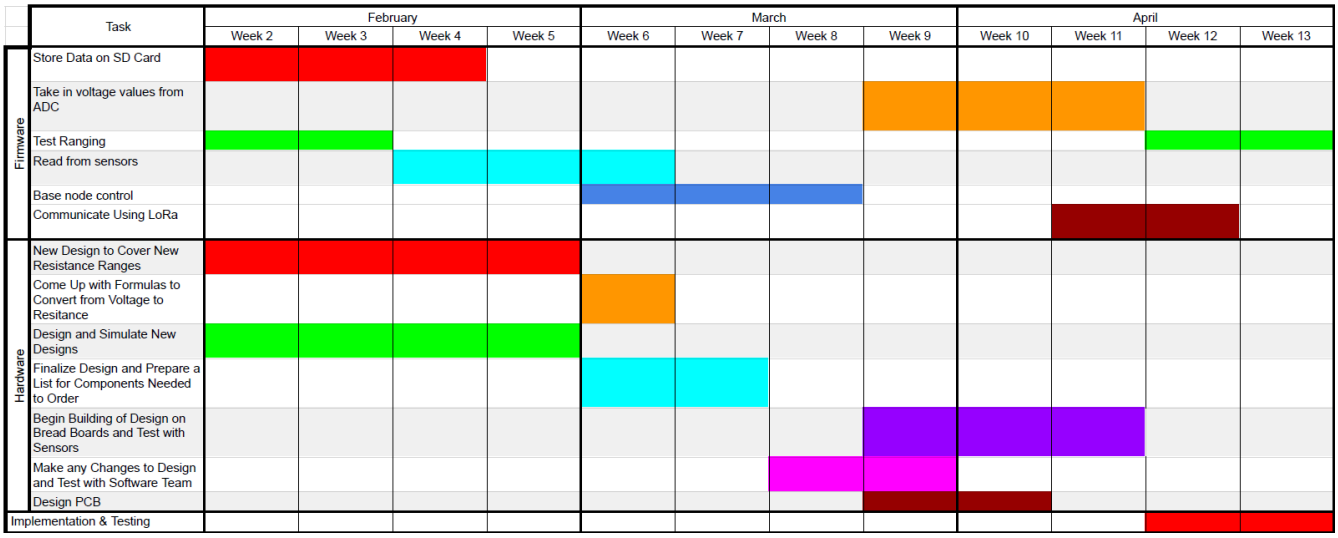


Figure 3.4.2: Semester 2 Timeline of Our Project

3.5 Risks and Risk Management/Mitigation

Risk	Probability	Mitigation Strategies
Not being able to complete the project on time	5%	• Regularly adjust the project timeline based on the progress • Break down the task into smaller, easier-to-track progress • Adjust team members or resources as needed
The connection of the mesh network is unstable	20%	• Conduct extensive testing in both lab and field • Test with alternative wireless protocol if having interference issues • Consider using a longer range of communication modules • Improve the communication of the firmware and sensor
Incorrect sensor reading	10%	• Implement signal filter circuit/ averager & summer circuits • Replace sensors if needed • Revise the designed circuit that captures sensor outputs

Power Management	10%	● Select batteries designated for the circuit ● Design the circuit with a battery holder for easier replacement
Inability to make process or text files user-friendly	5%	● Collaborate with client to see how straightforward it is ● Make clear instructions for retrieving text files

Table 3.5.1: Risk Management and Mitigation

One risk we eventually experienced was significant issues with establishing a mesh network. The documentation and implementation was difficult. We addressed this by completely switching our approach, abandoning the original mesh network. In the end, we utilized a LoRa module for long-range communication which rendered the need of a mesh network useless. We also experienced some difficulties in sensor readings. Our clients at the end of the previous semester significantly increased the resistance range requirements. This led us to needing to completely redesign the board, resulting in project delays. The additional requirement also added hurdles to the software development, as the addition of an external ADC required additional driver code in order to acquire data. We responded to these challenges by altering project milestones and timelines. The transitions were relatively smooth due to active communication among the team and with our advisor and clients.

3.6 Personnel Effort Requirements

Task	**Done?**	**Total Hours**
Design Wheatstone Bridge circuit	Y	35
Finalize circuit PCB	N	30
Download ESP32 repositories and Linux	Y	5
Initialize and calibrate ADC to take voltage values	Y	10
Implement IDE into the voltage divider circuit	N	15
Design circuit for SD card writing	N	30
Write firmware store data on SD card	N	20
Test and troubleshoot SD card data	N	15

Establish protocol framework for data transfer	N	15
Troubleshoot data transfer errors	N	25
Implement and test mesh network with known values	N	15
Combine known values into text file	N	8
Create a web server for base station	N	5
Write input from multiple streams to text files	N	15
Determine how to download data from base station	N	5
Format and simplify text values	N	5

Table 3.6.1: Original Task Decomposition List

Via our task decomposition list, we established individual tasks seen on the left side of the table. By the end of our project, our personal effort requirements were the following:

Task	**Done?**	**Total Hours**
Design voltage divider circuit	Y	70
Finalize circuit PCB	Y	30
Download ESP32 repositories and Linux	Y	5
Initialize and calibrate ADC to take voltage values	Y	20
Implement IDE into the voltage divider circuit	Y	15
Design circuit for SD card writing	Y	10
Write firmware store data on SD card	Y	20
Test and troubleshoot SD card data	Y	15
Establish protocol framework for data transfer	Y	15
Troubleshoot data transfer errors	Y	25
Implement and test network with known values	Y	30
Combine known values into text file	Y	8
Create a server for base station	Y	5

Write input from multiple streams to text files	Y	20
Determine how to download data from base station	Y	10
Format and simplify text values	Y	5

Table 3.6.2: Update Task Decomposition List

There were significant hurdles and changes in project requirements that resulted in subtantial mismatches between our actual input and what we anticipated. Some of these included the redesign of our PCB and change in communication scheme. Additionally, there were some unaccounted for tasks in our original plan, such as creating driver code for an external ADC.

3.7 Other Resource Requirements

There are several tangible and intangible resources we required. Some included software, Wi-Fi, IEEE codes, online databases, and people.

As mentioned, our project requires various forms of software. The Software team relied on coding environments and Linux terminals to initialize and flash to the microcontroller. It also applied to the base station and the ability to properly output a text file. The Hardware team also relied on various simulators and CAD software. They have used LTspice to mock and test the voltage divider. They also eventually needed to replicate it in KiCAD so that a final version of the PCB could be ordered.

This network also relies on Wi-Fi and wireless communication. It was important that we understood the limitations of wireless “reach.” For instance, the corn stalks limited our communication ranges.

IEEE resources and codes are imperative for our project. Specifically, Wi-Fi has stringent rules we must abide by to maintain best practices. We referenced the 802.11 IEEE standard to ensure our use of a wireless network was properly and ethically implemented.

Next, it was important to take advantage of existing resources. For instance, Espressif (the manufacturer of the ESP-32 microcontroller) has endless repositories and code outlines that reduce man hours. For instance, much of the existing SD card initialization and implementation code could be easily drawn from their GitHub repository. Since we pulled from the same databases, our code is more uniform, increasing readability and compatibility.

Finally, and maybe most importantly, we leaned on our clients and advisors for information. There was only so much that we could research and troubleshoot, and at a certain point, it was critical to approach the people guiding our project. They have far more knowledge and experience that has proven to be invaluable. Roadblocks were far easier to overcome by relying on our mentors.

4. Design

4.1 Design Context

4.1.1 Broader Context

List of relevant considerations related to our project in each of the following areas:

Area	Description	Examples
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., the solution is implemented in their communities)	Reducing overuse of pesticides and increasing the effectiveness of food production.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Development or operation of the solution would not violate a profession's code of ethics, implementation of the solution would not require an undesired change in community practices
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	By studying effective pesticide distribution, our product would decrease waste and runoff
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Product needs to remain affordable for target users, create opportunities for economic advancement

Figure 4.1.1: Broader Context

4.1.2 Prior Work/Solutions

John Deere Operations Center: This system integrates precision agriculture tools with IoT and provides comprehensive field monitoring. However, it relies on centralized cloud-based data processing, which may introduce latency issues and limit scalability for remote fields without robust internet access. [10]

CropX Soil Monitoring System: This solution uses wireless sensors for real-time soil moisture and temperature monitoring. While effective for soil conditions, it lacks capabilities for monitoring pesticide application metrics. [11]

SmartSpray Technology: Recent advancements in smart spraying technologies include systems that use camera-based sensors and AI to optimize pesticide application. These systems, while innovative, are often expensive and limited to proprietary platforms. [12]

4.1.3 Technical Complexity

Microcontroller Network (ESP32-C6 Nodes)

Engineering Principles:

- Implementation of IEEE 802.11s for wireless communication
- Configuring the ESP32-C6 microcontrollers to handle real-time data acquisition and transmission
- Synchronization of multiple nodes within a network to ensure reliability and scalability

Mathematical Principles:

- Signal processing techniques to filter and normalize data from sensors before transmission
- Optimization algorithms for dynamic routing and load balancing in the mesh network

Scientific Principles:

- Electromagnetic theory for efficient base station placement and Wi-Fi signal propagation

IDE Sensors and Voltage Divider Subsystem

Engineering Principles:

- Designing a voltage divider to measure changes in resistance due to pesticide spray deposition
- Engineering PCBs in a way such that they are compatible with the limitations of other technology (ex. Pin counts)
- Integrating sensors with microcontrollers via analog-to-digital converters (ADC)

Mathematical Principles:

- Ohm's Law and voltage divider equations to calculate resistance changes accurately
- Calibration curves to correlate sensor resistance with pesticide spray concentration

Scientific Principles:

- Principles of material science to understand the IDE sensor's sensitivity and response to the sprayed chemicals

Power Management System

Engineering Principles:

- Utilizing batteries with voltage regulators to maintain stable power for sensors and microcontrollers
- Implementing energy-efficient designs to prolong system operation in remote agricultural fields

Mathematical Principles:

- Power budgeting to ensure all components operate within their specified voltage and current limits
- Efficiency calculations for voltage conversion and energy storage

Scientific Principles:

- Electrochemistry for battery performance and longevity under variable environmental conditions

Base Station Subsystem

Engineering Principles:

- Configuring an ESP32-C6 as a base station to collect, aggregate, and transmit data to a remote server
- Implementing error correction and secure data protocols to ensure data integrity and privacy

Mathematical Principles:

- Compression algorithms to minimize data size during transmission
- Statistical analysis for preliminary data insights at the base station

Scientific Principles:

- Principles of wireless communication for seamless connectivity between nodes and the base station

Conclusion

The integration of multiple subsystems, each relying on distinct scientific, mathematical, and engineering principles, demonstrates the technical complexity of our project. By addressing a range of challenging requirements from real-time data processing to energy efficiency our system aims to provide a robust, scalable, and precise solution that aligns with and surpasses existing industry benchmarks.

4.2 Design Exploration

4.2.1 Design Decisions

AD7171: One major design decision was to utilize an ADC external to the microcontroller, rather than the one within the ESP32. This decision was an important factor in our success. The ADC has a 16-bit capacity, thereby allowing us to have an increased resistance range when compared to the 12-bit ADC within the microcontroller. Without this design choice, we would not have been able to meet the clients' need to measure over such a large saturation range.

Placement of Interdigitated Electrode (IDE): We placed three sensors on three different levels. This allows us to get a more comprehensive analysis of the pesticide spray distribution. The sensor at different heights will capture different data during the spray. The sensor's placement is critical because the data we get will affect the analysis of the given spray techniques.

Base Station: We will use an additional ESP32-C6 as our base station to store the data from multiple sensor nodes. A central node will allow us to manage and analyze data from the whole network. The microcontroller has the capability to store the data from six to twelve nodes from the network in a .txt file.

Voltage DIvider Circuit: The voltage divider in our circuit is used to convert the varying resistance of the sensor reading into a measurable voltage. This approach offers a simple and effective solution for handling a wide resistance range from 100Ω to 200kΩ. A stable 2.5V reference voltage is provided by the ADR5041 precision diode and buffered by a low noise AD8628 op-amp to minimize errors caused by fluctuations in supply voltage. The sensor resistance (Rx) is placed in series with one of three precision resistors (R1 =330Ω, R2 = 2.7kΩ, R3 = 27kΩ), depending on the target measurement range. These resistors, each with a 1% tolerance, are selected through ADG801 switches, enabling accurate switching between three distinct measurement ranges. The resulting voltage from the divider is buddered and fed into a 16-bits external ADC to ensure the high resolution and low noise digital conversion. This design

enables accurate resistance measurement while maintaining less than 1% full-scale error, making it suitable for monitoring pesticide distribution through sensor response.

4.2.2 Ideation

Raspberry Pi ● Built-in Wi-Fi ● Locally stored/ external storage ● Access remotely ● Low power consumption	**Arduino** ● Low power consumption ● Support basic data storage ● Lack of real-time processing	**Industrial Standard** ● Might require specific soft for data access ● High cost ● High reliability ● Weather proves
ESP32-C6 ● Limited data storage ● Low cost ● Low power consumption	**Data Collection For Wireless Mesh Network**	**Cloud-Based** ● Real-time data collection ● High cost ● Depends on cellular coverage
Laptop with Data Software ● Large data storage ● High power consumption ● Less durable ● Complex data analysis	**Custom Build(SD Card)** ● Low power consumption ● Customizability ● Limited Capability ● Data management challenge	**LoRa** ● Long-range data collection ● Low power consumption ● Depends on the range

Table 4.2.2: Comparison of Data Collection Solutions

4.2.3 Decision-Making and Trade-Off

During our decision-making, we focus on five aspects based on the project requirements:

Criteria	Weightage
Data Storage Capacity	20%
Power Efficiency	15%
Real-time Data Processing	25%
Measurement Accuracy	20%
Cost	20%

Table 4.2.3: Criteria and Weightage of Data Collection

Data Storage Capacity: Based on these criteria, we have chosen the ESP32-C6 as our central device for the data collection in our mesh network system. The ESP32-C6 has built-in flash memory, which is enough for moderate data logging. Pairing this with external storage such as an SD card enhances its capacity for larger data sets, especially for long-term data collection. In our case, we need our system to collect data for 3 hours. This decision allows us to ensure this happens.

Power Efficiency: The ESP32-C6 is designed for low-power applications. It supports various power-saving modes, making it an energy-efficient choice. This microcontroller is especially useful in remote installations, where battery life is crucial. This also reinforced our decision.

Real-time Data Processing: With a relatively high processing power and Wi-Fi 6 capability, this microcontroller is suited for low-latency data processing applications. The Wi-Fi feature allows faster data rates and reduced latency, which is essential for real-time processing in a distributed network. Although we ended up not using the mesh network feature, this could be a useful quality of the ESP32 for future modifications to our project.

Measurement Accuracy: Accuracy largely depends on our sensors provided by Claussen Labs. However, the ESP32-C6's input interfaces support accurate sensor data capture from our external ADC, making it a solid choice for environmental sensing applications such as pesticide spray monitoring.

Cost: The ESP32-C6 is known for its affordability compared to other microcontrollers with similar capabilities. Priced at nine dollars, it allows for the scaling of our project to cover multiple nodes without exceeding budget constraints.

4.3 Final Design

4.3.1 Overview

Our design will collect and conglomerate pesticide data. Essentially, we will scatter sensors throughout a corn field at different crop canopy levels that record pesticide saturation at that point. Each sensor will have a microcontroller to which it will feed data. In basic terms, the microcontroller acts as a digital log to collect the sensor measurements. Data can then be sent between these microcontrollers and eventually to a central node (see *Figure 4.3.1*). The central node acts as a base station where all sensor data ends up. It allows researchers to remotely pull all pesticide saturation data from one station. The measurements will be organized into a user-friendly text file.

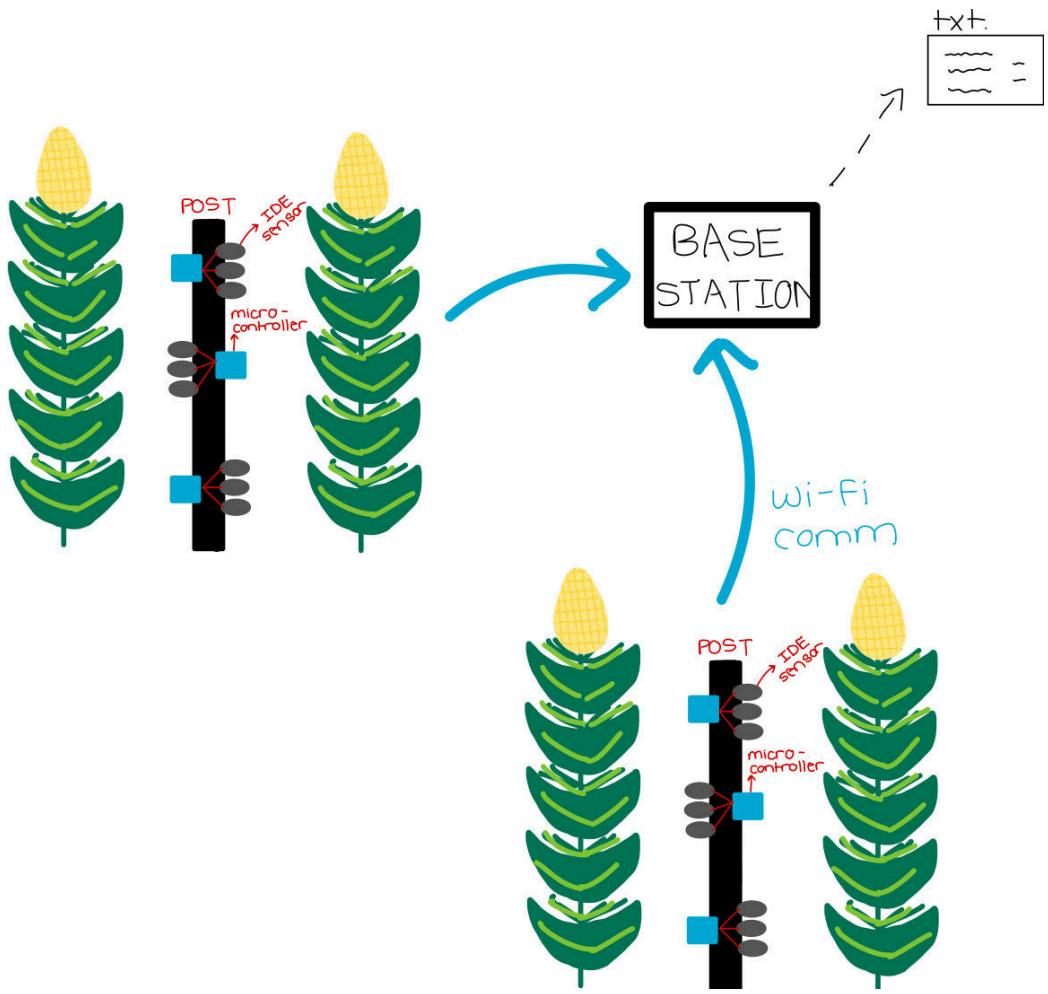


Figure 4.3.1: Simplified overview of the communication system

4.3.2 Detailed Design and Visuals

The goal of our project was to develop a wireless network that assists in the monitoring and mapping of pesticide spray. Using interdigitated electrodes (or IDEs) developed by Claussens Labs, we can collect resistance values at different levels of the crop canopy. These values correlate to the pesticide saturation at that level of the post.

Each post would have nine different sensors (IDEs) connected to three different microcontrollers or nodes. Acting as a gate between the sensors and microcontrollers is the voltage divider circuit. We have designed it to take in a resistance value and convert it to a corresponding voltage. It is an imperative step since the microcontroller takes in data via voltages rather than resistance values. It does this via an external ADC (analog-to-digital converter). The digital voltage values will be converted back into resistance measurements through programming. Next, the output will be written to a physical SD card in addition to being transmitted (see *Figure 4.3.2.1*)

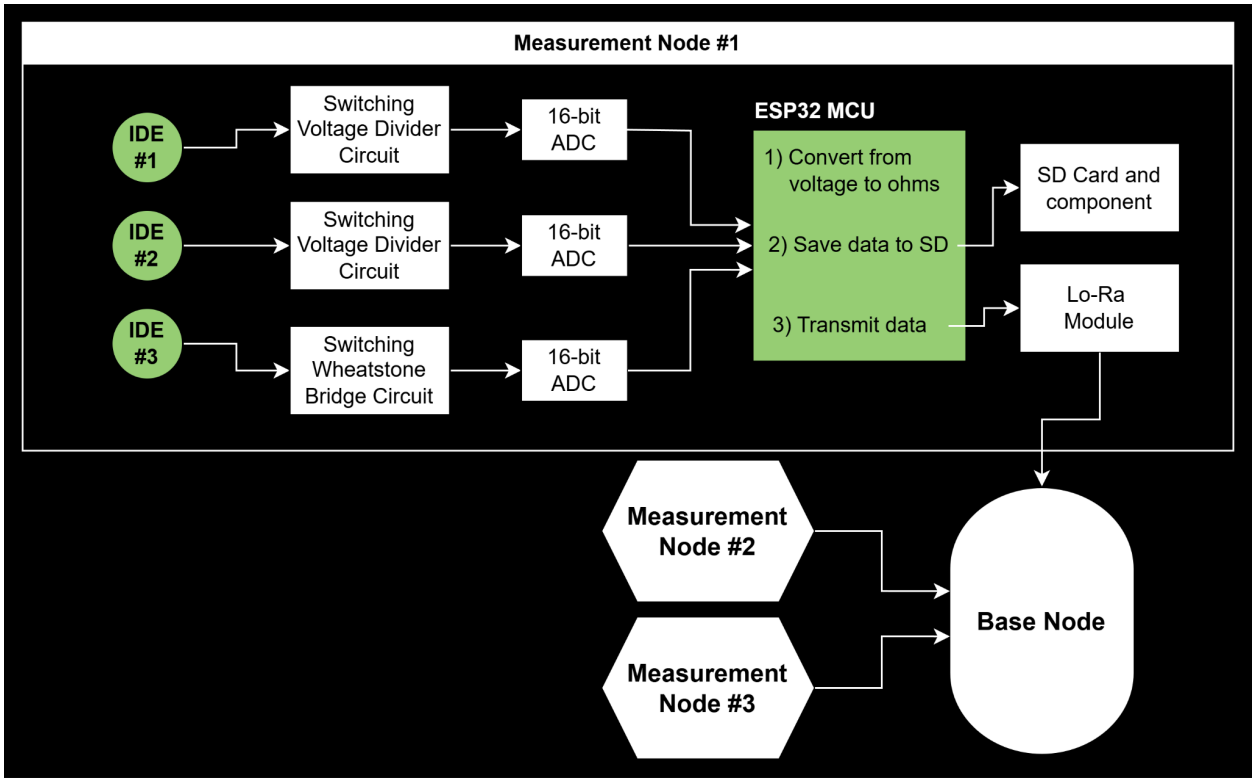


Figure 4.3.2.1: Block diagram of the sensor data path

The communication between measurement nodes and the base station is achieved via a RFM95W LoRa module. This module allows us to transmit data on the 900mhz ISM band. This module allows us to realistically transmit data over distances of over a kilometer, greatly exceeding the 200ft requirement for our project. The LoRa module, however, is less feature rich than something like wifi. It does support error checking, which allows us to trust that whatever data we receive on the radio is correct and uncorrupted. Other than that it simply broadcasts binary data frames. This meant that we needed to create our own simple network. Our network is a star network, as the base station is the ultimate authority and all communication is done between the base station and one of the measurement nodes (a measurement node will never send data to another measurement node). We ensure that no two boards transmit at the same time by having the base station initiate all communication. In other words, the measurement nodes will only respond to the base station. They will never transmit anything on their own. We defined some data frames in order to stand in for the wifi packets we planned to use previously (see *Figure 4.3.2.2*) Given that the LoRa modules broadcast data, it is likely that more boards than simply your intended receiver will receive the data. Thus, we include who the data is to and who its from. This allows a board to determine whether it needs to process the data or not. Each board is assigned a number between 0 and 255 which functions as its identifier. 0 is reserved for the base station, and 255 is reserved for broadcasting to all boards (used to start and stop recording). The start and stop recording frames are used by the base station to get the measurement node to either start or stop recording. The reading request frames are used to get the sensor readings from each measurement node to the base station. The heartbeat frames are intended to be used when initially setting up the network so that the

researchers will be able to determine if the base station is able to communicate with each measurement node.

Packet Types	Packet Type #		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 9	Byte 10	Byte 11	Byte 12
start recording	1		to	from	packet type									
stop recording	2		to	from	packet type									
reading request	3		to	from	packet type	sensor id								
reading request response	4		to	from	packet type	sensor id	# of last reading	# of readings	flags	reading time	reading time cont'd	reading value	reading value cont'd	reading value cont'd
heartbeat request	5		to	from	packet type									
heartbeat response	6		to	from	packet type									

Figure 4.3.2.2: Definition of Each Data Frame used in the Network

Our project will propel Claussen Labs forward in their expedition to determine which pesticide distribution methods are most efficient. With an ever-growing world population, determining the best means of pesticide application is necessary for creating dependable and high-yield food sources.

4.3.3 Functionality

In this design, researchers or farmers can efficiently monitor pesticide distribution across a crop field. To begin, the user places sensor-equipped poles in different field areas, each with ESP32 microcontrollers that form part of a network. After powering on the devices, the user goes to a central node accessible via a laptop with an SSH connection over Wi-Fi and sends a wake signal that activates all sensor nodes. The central node displays the status of each node, ensuring they are ready for data collection.

As the user applies pesticides, each sensor node records data on pesticide levels at various canopy heights. The user can monitor network health and confirm that all sensors are operational via terminal commands on their laptop. When the application is complete, the user returns to the central node, issues a command to stop data recording, consolidates data at the central node, and puts all the nodes into sleep mode to conserve power.

Next, the user retrieves the data by downloading a .txt file from the central node, which contains all recorded readings for analysis. Additional terminal commands allow the user to monitor network status and troubleshoot as needed, providing feedback on each node's connectivity. Overall, this design streamlines field monitoring, allowing the user to control the network and data collection from a single interface, reducing manual handling of each sensor and making the system easy and efficient for field operations.

Timeline			
Step	Action	System Response	User Benefit
1	Pole Setup	Nodes initialized in sleep mode	System conserves energy until needed
2	Wake-Up Signal	Nodes activated and confirm presence	User confirms all nodes are active
3	Pesticide Application	Nodes continuously collect data	Insights into pesticide application over time
4	Application Completion	Nodes sleep; data compiled to .txt file	Full record of pesticide distribution
5	Data Retrieval	User obtains .txt file for analysis	Enables post-analysis and optimization

Table 4.3.3: Timeline of system use/functionality

4.3.4 Areas of Challenge

We faced quite a few challenges while creating our final design. One of our largest roadblocks was the mesh network itself. Our software team found it technically challenging to implement due to the lack of clear resources provided by the ESP32's manufacturer, Espressif. After speaking with our client, our team shifted focus to a simplified communication scheme in which each node speaks directly with the central base station. We did this by enacting the far more documented and utilized long-range wifi module, LoRa. Since the LoRa increased communication range drastically, our clients did not find it necessary to have a mesh network any longer.

Another challenge that corresponded with client requirements changing was our voltage divider circuit. This circuit is a part of a larger PCB that acts as the segue between the sensor and the microcontroller. Initially, we expected a much smaller range of resistance values. The smaller range enabled us to use a simple circuit with the built-in ADC on the ESP32 microcontroller. However, in December, our clients advised that they needed a far larger range of resistance values to be accounted for, rendering our original design null. To accommodate this change, our hardware team engineered a more complex switching circuit, which allowed us to account for a larger range of resistance values. The new range did cause another complication in that our design required a 16-bit ADC rather than the 12-bit one located within the microcontroller. We addressed this using an external ADC connected to the ESP32 via pin configurations.

Most of our challenges were addressed through direct and timely communication with our advisor, client, and one another. The notion of voicing issues and roadblocks early was a driving force behind overcoming obstacles and is definitely a lesson our team will take away from our senior design project.

4.4 Technology Considerations

For our project, we are using both internal and external technologies. Both forms of technology will still be used together and are ultimately going to be connected. First, the ESP32-C6 microcontroller, as explained before in our design decisions, has various strengths such as built-in Wi-Fi capabilities, low power consumption, and energy efficiency. One weakness is the complexity of enacting a mesh network on ESP32s. This has led to our use of a LoRa board as well, which allows for long range communication to avoid network errors. Next, our circuit design includes a voltage divider which allows us to precisely measure the resistance of an unknown resistance, which will be coming from our pesticide sensor. While a Wheatstone bridge can offer higher sensitivity, the voltage divider was chosen for this project because it provides accurate resistance measurements in a simpler and more efficient way. This makes it a practical solution for the sensor system. The pesticide sensors provided by the Clausssen Lab are connected as part of the voltage divider as Rx, allowing changes in resistance to be easily translated into measurable voltage signals. When the time comes to spray pesticide on the crops, this sensor will be doused with a certain amount of pesticide depending on the location of the sensor. The amount of pesticide on our sensor will give us a certain resistance measurement which will be stored in an ESP32-C6.

When it comes to the technology available that has similar functions to our project, there is a good amount of products to look into. First, the Libelium [6], allows the monitoring of multiple environmental parameters involving a wide range of applications, from plant growing analysis to weather observation. The strengths of this technology are; that it supports 30 different sensors covering critical environmental parameters such as soil moisture (can also do temperature, humidity, solar radiation, wind speed, and rainfall), easy to deploy, functional wireless mesh network, and energy efficient. The weaknesses of the Libelium are; high cost ($5,000 - $20,000) for the whole system, complex maintenance, connectivity issues in remote areas, and low data security. Another available technology is the iMETOS [7], which has the strengths of real-time data access from the platform, alerts for critical weather events, durability to harsh weather, and integration with other sensors. The weaknesses of this technology are it doesn’t directly monitor pesticide spray, poor connection in some areas, high initial cost ($1,500 - $3,000), and a complex setup.

For our project, we looked at these available technologies and found possible solutions and designs that could be implemented into our project. Including the solutions and resources provided to us by our client, we were able to narrow down and start designing our own circuits and networks. For our circuit, which will be connected to the sensors and microcontroller, we initially started with a simple voltage divider. This worked but had to have such a high resistance that ultimately caused a lot of room for unexpected error. We then found the solution of the voltage divider. This new circuit allowed us to lower the resistances and lower the room for error, it did not erase it but it allowed us to be able to control it easier. Additionally, the new feature of the voltage divider resulted in more accurate voltages and it also made it easier to find certain voltages due to being able to easily change resistance values on either side of the bridge.

5. Testing

5.1 Unit Testing

For the circuit we created, the unit that is being measured is voltage. With the design used in this circuit, the amount of pesticide on the sensor changes its resistance. This varying resistance is part of a voltage divider, which produces a corresponding output voltage. This voltage is then buffered and sent to an external ADC, where it can be accurately measured and analyzed. The output will be a voltage in the range of 0.1 - 2 volts, to ensure the microcontroller is not damaged. The tools we used to get the voltage values are LTSpice and the multimeters in the labs in Coover Hall. On the software side, we had to ensure each microcontroller functioned properly in terms of flashing, connectivity, and identification. We did this by utilizing error codes and iterative variables, to ensure sensor data and packets were not being lost. We also will require physical identification on the researchers parts to keep track of where each microcontroller is located in the field.

5.2 Interface Testing

One interface we used in our project was for the RFM95W LoRa module. The module itself is its own little computer that we communicate with over SPI. We found a low level library online that helps handle the SPI transactions. We did a few different things to test this interface. First we wanted to test the basic functionality that we were going to use of the library. We tested this by sending data back and forth between a few of the modules and checking if any of that data was corrupted or incorrect. Another thing we wanted to test was the range of the interface and whether the increased distance lead to increased data corruption. To test this we had one module constantly sending out the same data, and then had another module listening for that data. If it received the data it expected it would blink a green LED, if it received the wrong data then it would blink a red LED, and if it didn't receive anything it wouldn't blink anything. Testing this way we were able to get reliable transmission over a distance of 800m. So long as the two modules were able to maintain line of sight, distance didn't increase the amount of data corruption.

Another interface that needed to be tested was the AD7171 analog-to-digital converter on the ESP32-C6 chip that we are using to collect sensor data. We needed to test how accurate the measurements of the ADC were, how it handled any mistakes in connecting the sensor and voltage divider, and how repeatable its measurements were. To do this testing, we used an ESP32-C6 board, voltage source, multimeter, waveform generator, and an oscilloscope. We found that our resistance measurements were well under the 1% error required. Our measurements were within 0.25% of the expected value.

5.3 Integration Testing

Our project has three major integration parts. These include the circuit that consists of our sensor which is connected to the analog-to-digital converter (ADC). Following this connection is the connection to the

master node, which is another microcontroller that collects and stores the information being received from other microcontrollers. These integration parts were tested by looking at the data being collected by the microcontrollers and verifying that the voltage values the ADC is converting to resistance values are within our 1% acceptance range. Following this test, we tested the range of the LoRa module..We used the LoRa boards and tracked the success rate of package transfer at increasingly larger discrepancies in distance between the two.

5.4 System Testing

For the hardware side of our project, we tested many different circuits in two ways, in the lab and through LTSpice. We conducted tests on this circuit using the voltage sources and multimeter provided to students in the Coover Hall labs. LTSpice was used to test the margin of error that our circuit could potentially have, which helped with tuning the components of our circuit to give us the desired voltage results. Our software systems were integrated by sending predictable dummy data across the network. We also implemented heartbeat packages. These consist of a call and response that determine if the microcontroller is online, and can be used in the future for troubleshooting in the research lab. We utilized the microcontroller and the LoRa board for testing our system, in addition to code that kept track of individual functions that are called in the main program and alert the user if behavior is not what is expected. For instance, we have the capability to determine if the ADC had an error while reading data or if a microcontroller is not connecting to the system via status bits located in the data register.

5.5 Regression Testing

As we planned our hardware implementation, we looked into ways to prevent hardware malfunctions. These malfunctions could occur when too much voltage is introduced to the ADC component. After the hardware implementation and verifying that the circuit is reliable and accurate, it is also important to ensure the code for our microcontrollers, both the microcontrollers that will be collecting data and the master node that will be communicating with all others and storing the data, is organized to find any potential regressions. For instance, the modulation of our code into discrete functions that could be individually tested was critical to our success. Since the hardware and software teams directly worked together, it was crucial that both were reliable, as well as stable.

5.6 Acceptance Testing

We ensure that our design requirements, both functional and non-functional, are being met by utilizing a spreadsheet with all of our test cases and requirements. This is used to keep track of the testing status and results, such as voltage accuracies and communication errors. It also ensures that all team members have somewhere to see what functionality to consider outside of their own assigned work. We have involved our client in the acceptance testing process by creating prototypes for them. Additionally, we utilize our clients' input as a marker of success for our overall software interface; their ability to use our design easily is one of our most critical requirements.

5.7 User Testing

In our project, our clients double as our users. Thus, user testing is very straightforward. We have done this by creating a user manual for the software. We will provide this manual to the researchers and ask to receive feedback on ease of use and functionality. Thus far, their reactions have been positive. This is not unexpected, as we have received continuous feedback concerning their needs and experience with software throughout the entirety of our project. Overall, we have observed our users to be quite adept with our design as they are researchers and therefore not unfamiliarity with programming.

5.8 Other Testing

After consideration, we do not deem any additional testing necessary. For instance, security testing isn't necessarily important to our project. The data that is being sent over our network isn't sensitive as it is simply sensor data, and commands used to start and stop recording data. The mesh network framework from Espressif also implements the same security features one would expect from any other Wi-Fi device, such as WPA2 encryption.

5.9 Results

For the hardware side, our goal was to find a way to relate the output voltage of our circuit to a resistance value. The core measurement is based on voltage divider configuration. An unknown sensor resistance, Rx, is connected in series with some known reference resistors, R1 =330Ω, R2 = 2.7kΩ, R3 = 27kΩ, to form a voltage divider across a stable input voltage, VIN. The output of the voltage divider (VDIV) is fed into an AD8628 voltage buffer op amp. To ensure minimal error we are using an external ADC. This allows us to use the VIN as VREF, this results in any noise or variation being canceled out.

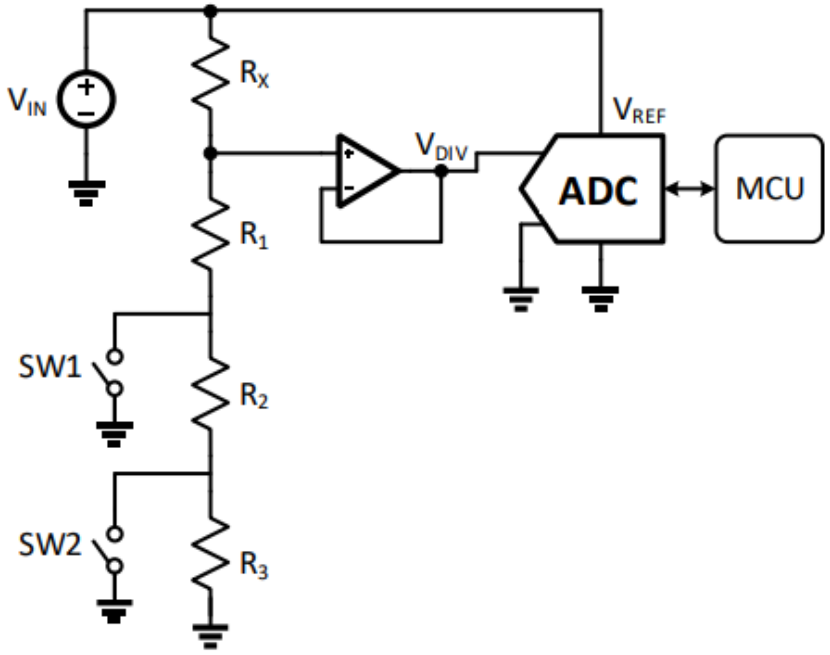


Figure 5.9.1: Voltage Divider Circuit with the ADC and MCU

Once we received our PCB, we conducted unit testing to ensure components were working as expected. Below is a small snippet of the organization of our tested and expected data. Our results are pretty telling; every measured value was extremely close to what was expected.

Range 1 Testing

Nominal Resistance (Ω)	Fixed Resistance (Ω)	Expected V (with tested values)	Circuit 1 - Tested Voltage	Circuit 2 - Tested Voltage	Circuit 3 - Tested Voltage	Average
100	98.859	1.923	1.90333	1.90258	1.90401	1.90331
150	149.523	1.721	1.70431	1.70406	1.70637	1.70491
220	218.558	1.503	1.49221	1.49072	1.49382	1.49225
330	324.933	1.259	1.25132	1.25411	1.25437	1.25327
470	466.689	1.036	1.03041	1.02917	1.03109	1.03022
570	565.750	0.921	0.91712	0.91638	0.91923	0.91758
660	650.290	0.842	0.83908	0.83811	0.84062	0.83927
680	675.231	0.821	0.81730	0.81609	0.81950	0.81763
800	791.658	0.736	0.73342	0.73231	0.73476	0.73350
900	893.896	0.674	0.67201	0.67123	0.67402	0.67242
1000	983.567	0.602	0.62419	0.62359	0.62587	0.62455
1100	1083.171	0.575	0.58027	0.58000	0.58218	0.58081

Figure 5.9.2: PCB Testing of First Range (100 Ω - 1000 Ω)

For the software side our results can largely be broken down into two groups: sensor reading correctness, and network effectiveness. As for sensor reading correctness, our results are very positive. We've been able to meet the requirement of reading sensors with less than 1% error of their actual

value. From our own testing, we've been getting an average error of around 0.25% of the expected values of resistors connected. We've successfully gotten the switching of ranges to work regularly. On the networking side, we've successfully tested each of the data frames and they behave as expected. The function of getting all the sensor readings onto the base station ultimately doesn't work incredibly well. Over the course of a couple hours, it's basically guaranteed that at least one of the frames of sensor readings will be dropped. This is due entirely to the primitive nature of the LoRa network as opposed to WiFi. We simply didn't have time to implement a more complex algorithm to account for data loss and to attempt to fix any missing data. So ultimately it works, but it's more of a beta feature. Command data frames (start/stop recording & heartbeat frames) work better since they are only one frame, and I've been able to implement some logic so that measurement nodes respond to commands. That way the base station knows whether a measurement node has heard its command or not. One case that this doesn't cover however, would be if the measurement node heard the command from the base station but the base station didn't hear the response from the measurement node. This leads the networking to work, but it's certainly not perfect.

6. Implementation

The overall design was implemented in a way that we felt was most effective in meeting client goals. There are multiple subsystems to our design that are important to describe individually:

1. Power System
   The circuit is powered by a 9 V alkaline battery. After the battery, we use a LM1117 linear voltage regulator that converts the 9 V to 5 V, this is then used to power our ESP32 microcontroller and the positive power supply of our AD8628 op amps. Once the microcontroller is powered, it further regulates the 5 V to 3.3 V. We then use the 3v3 pin of our microcontroller to supply voltage to other components and sections of our circuit.
2. Voltage Input Generation

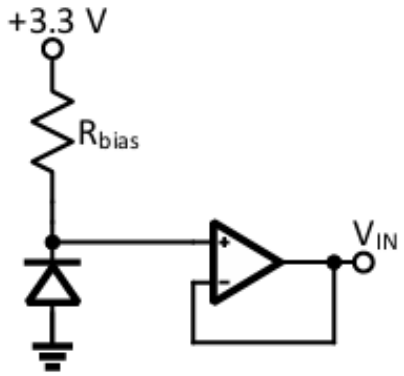


Figure 6.0.1: Reference Voltage Circuit

A stable V_{IN} voltage is required for accurate measurements in this circuit. For this reason, we use the above circuit to generate a value of V_{IN} = 2.5 V. The diode is the ADR5041 precision, micropower shunt voltage reference by Analog Devices. This device generates a very stable 2.5 V drop across the diode, which is then buffered by the op amp before being applied to the voltage divider.

The value of R_{bias} is chosen to allow a current of approximately 100 µA to flow through the diode. The circuit uses an R_{bias} = 8.2 kΩ which results in a current of 97.56 µA.

The input buffer is an AD8628 zero-drift op amp from Analog Devices. This opamp can be powered from a single supply and has rail-to-rail inputs and outputs. It is primarily used to provide current required by the voltage divider. Assuming V_{IN} = 2.5 V, the maximum current that will flow through the voltage divider is 2.5 V / 330 Ω = 7.6 mA if the sensor resistance is 0 Ω. The AD8628 has an output current limitation of ±10 mA, meaning that it can handle the minimum sensor resistance of 100 Ω. With the minimum resistance the maximum divider current is 2.5 V / 100 Ω = 5.8 mA.

3. Voltage Divider Measurement

   As specified previously, the core measurement is based on voltage divider configuration. An unknown sensor resistance, R_x, is connected in series with some known reference resistors, R_1, R_2, and R_3, to form a voltage divider across a stable input voltage, V_{IN}.

   The output of the voltage divider (V_{DIV}) is fed into an AD8628 voltage buffer op amp. To ensure minimal error we are using an external ADC. This allows us to use the V_{IN} as V_{REF}, this results in any noise or variation being canceled out.

4. Reference Resistor and Range Switching

   The circuit in Figure 2 measures the unknown measured resistance in three different ranges:

   - Range 1 : 100 Ω to 1000 Ω
   - Range 2 : 1,000 Ω to 10,000 Ω
   - Range 3 : 10,000 Ω to 200,000 Ω

   The circuit in Figure 2 has the ability to switch between the ranges. To do this we set:

   - R_1 = 330 Ω
   - R_2 = 2,700 Ω
   - R_3 = 27,000 Ω

   For each of these resistors we will use a tolerance of 0.1% or better. The switches will be realized using ADG801 CMOS SPST switches from Analog Devices. These switches are normally open when the input voltage is LOW, and the corresponding reference resistor is disconnected from the circuit. When the digital input voltage is set HIGH, the switch closes, and the corresponding reference resistor is connected into the circuit. Attached is the truth table for the ADG801 CMOS SPST and a table showing what switch states correlate to which measurement ranges.

ADG801 (Pin IN)	Switch Conditions
0	Off
1	On

Table 6.0.2: ADG801 Truth Table

SW1 State	SW2 State	Measurement Range
OPEN	OPEN	Range 1
CLOSED	OPEN	Range 2
CLOSED	CLOSED	Range 3

Table 6.0.3: Switch States and Measurement Ranges

5. ADC Conversion
   The ADC is the AD7171 16-bit, low power, ΣΔ ADC from Analog Devices. This ADC has a differential input that provides an output data rate of 125 Hz. In addition, an external reference voltage can be supplied to the chip. Valid supply voltages range from 2.7 V to 5 V. We use the 3v3 pin of our microcontroller to supply 3.3 V to the positive supply voltage of the ADC. We need to use 3.3 V because the ADC communicates with the MCU using a simple 2-wire communication interface and the digital logic levels are dictated by the supply voltage.
6. Circuit Connection
   The PCB design connects an ESP microcontroller (MCU) to:
   - ADC modules
   - LoRa communication module
   - SD card module
   - Switches (for voltage divider)

   All the connections are handled through direct GPIO interfacing, allowing flexibility and modular testing for each sensor circuit.
7. SD Card
   The client required local storage on each individual node to ensure the protection of data measurement in the case our network does not transmit it successfully. We utilized FreeRTOS functions included in the ESP32 documentation in order to format the SD card and write data. Our clients required individual directories per run as well, which we were able to successfully implement. Overall, we feel confident in our local storage design.
8. Network communication:
   Both the base station and each measurement node has a RFM95W LoRa module attached ot them. They use these to form a star network with the base station at the center. The measurement nodes are constantly listening for data from the base station. The base station is able to send different commands to either a specific node or broadcast that command to all nodes. Once sensor recording has started, the base station will periodically poll each measurement node for sensor readings. Once sensor recording has ended, the base station will go through a cleanup loop where it keeps polling each node for sensor readings until that node is out of readings to send it.

6.1 Design Analysis

While we had a successful design that meets the base needs of the client, one functionality that we lacked was the implementation of a mesh network. Instead, we used a LoRa module specifically designed for long-range wireless communication. Our choice to switch communication schemes hinged on our inability to successfully communicate over the ESP32's mesh network as documented. In the interest of time, it was imperative to move forward with a protocol that worked. Luckily, the LoRa module was an excellent replacement as we did not have to sacrifice distance requirements.

7. Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

This discussion is with respect to the paper by J. McCormack and colleagues titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012 [9]

Area of Responsibility	Definition	Corresponding IEEE Ethics Code	Team Interaction
Work Competence	Deliver high-quality work with integrity, timeliness, and professional expertise	6) Maintain and improve our technical competence and undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations	Our team ensured that all tasks were carried out with the highest quality, adhering to timelines, and maintaining professional competence, while also ensuring that any limitations were fully disclosed and addressed
Financial Responsibility	Provide products and services that offer tangible value while ensuring they are delivered at a cost-effective price	5) Seek, accept, and offer honest criticism of technical work, acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others	Our team ensured that the products and services delivered were of realistic value and cost-effective by actively seeking and incorporating honest feedback, correcting any errors, and maintaining transparency in claims and estimates based on available data
Communication Honesty	Provide accurate, transparent, and clear reports of work to stakeholders, ensuring no deception or misrepresentation		Our team maintained clear and honest communication with all stakeholders, ensuring that technical work, progress, and estimates were reported accurately while acknowledging the contributions of all members and addressing any errors transparently
Health, Safety, Well-Being	Actively reduce risks to the safety, health, and overall well-being of	1) Hold paramount the safety, health, and welfare of the public, strive to comply with ethical design and sustainable	Our team primarily focused on the technical development of the wireless mesh network, ensuring that the project

	all stakeholders	development practices, protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment	met the functional requirements while minimizing risks where possible
Property Ownership	Respect the property, ideas, and information of clients and others by ensuring their proper use and protection	9) Avoid injuring others, their property, reputation, or employment by false or malicious actions, rumors, or any other verbal or physical abuses	Our team has maintained a focus on the technical aspects of the project while ensuring respect for the property, ideas, and information provided by the client
Sustainability	Safeguard the environment and natural resources at both local and global levels	1) Hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and disclose promptly factors that might endanger the public or the environment	Our team has made efforts to ensure the project adheres to ethical design practices by considering the environmental impact of pesticide applications and aiming to improve the efficiency of spray distribution
Social Responsibility	Develop products and services that positively impact society and contribute to the well-being of communities		Our team has considered social responsibility by ensuring that the wireless mesh network for pesticide monitoring is designed to improve agricultural efficiency, with an emphasis on minimizing environmental impact through precise pesticide application

Table 7.1.1: Mapping Area of Responsibilities

Our team performed well in the area of social responsibility by focusing on developing a system that benefits society and communities. Our project addresses the critical need for sustainable food production by reducing pesticide waste and environmental contamination through precise monitoring and data collection. We created a scalable, user-friendly system that empowers farmers to make informed pesticide application decisions, ultimately improving resource efficiency and minimizing environmental impact. This approach upholds our ethical and professional responsibilities by ensuring data accuracy, enhancing sustainability, and providing equitable access to technology for diverse agricultural communities.

One area we have discovered in which our team could have improved is sustainability, specifically in protecting the environment and natural resources both locally and globally. While our project aimed to optimize pesticide application and reduce environmental damage by preventing overuse and minimizing pesticide drift, our focus was primarily on precision through IDE sensors and a wireless mesh network for efficient pesticide spray measurement. To enhance sustainability, we could have considered optimizing energy use in our system, selecting more sustainable materials for our sensors and components, and accounting for the long-term environmental impacts, including how we manage and dispose of equipment. This would not only make our solution more eco-friendly but also contribute to more sustainable agricultural practices. As we transition to the workplace, we will be more mindful of how to reduce the negative impacts that are associated with the disposal of electronics.

7.2 Four Principles

	Beneficence	Nonmaleficence	Respect for Autonomy	Justice
Public health, safety, and welfare	Increase crop yield and accessible food sources.	The design will ensure certain foods are not oversaturated with pesticides.	Enables individuals to feed themselves despite the population growth.	Positive outcomes will affect all individuals evenly and fairly.
Global, cultural, and social	Promote a culture of affordable and sustainable food.	The FDA will further govern to protect from negative impacts once brought to the public.	Some anti-pesticide culture could be infringed on.	Since we're in the research stages, public opinion can still be taken into consideration.
Environmental	Pesticides are directly applied to plant life.	Avoidance of oversaturation and therefore damage.	Some concern for crops applied with pesticide.	Ensure design does not interfere with the health and layout of crops.
Economic	Affordable food due to high yields from even pesticide distribution.	Design is simple and affordable.	Will enable small farmers to have high yield crops.	Simplicity of design will make it affordable to small farmers.

Table 7.2.1: Four Principles for Pesticide Design

One important broader context-principle pair for this project is Public health, safety, and welfare - Nonmaleficence. The project aims to develop a system that ensures pesticides are distributed efficiently and evenly within targeted crop canopies, avoiding oversaturation that could lead to harmful pesticide residues in food or environmental contamination. This promotes public health by reducing the risks associated with improper pesticide application, such as health issues in consumers and environmental damage. To ensure these benefits, the design prioritizes accurate, reliable measurements from IDE sensors and robust data transmission to a centralized node. This will enable stakeholders to make informed decisions about pesticide application rates and methods, minimizing unintended harm.

Conversely, the project currently lacks in the Global, cultural, and social - Justice area. While the system promotes efficient pesticide use, it may not fully account for the diverse perspectives and cultural practices of anti-pesticide communities. These groups may feel marginalized or see the project as a challenge to their values. However, this limitation is counterbalanced by the project's broader positive impacts, including increased food accessibility and sustainability. To address this gap, our team could engage with diverse stakeholder groups, including those skeptical of pesticide use, to incorporate their feedback. This could involve designing the system to be adaptable for use with organic or alternative pest control methods, ensuring the project aligns with a broader range of cultural values.

By the end of this project, we also realized that there was an additional economic impact we had not considered. This is the financial impact successful research could have on Iowa State's research labs. If our design is useful for Claussen labs, they can have more efficient data collection and produce results faster. That way, less money will be spent on the current research, providing more funds for future opportunities at Claussen Labs.

7.3 Virtues

3 Virtues Important to the Team

1) **Integrity**

Definition: Upholding honesty, transparency, and ethical behavior in all project-related tasks and interactions.

Actions to Support:

- Ensure clear and honest communication within the team and with stakeholders, reporting progress truthfully and acknowledging limitations or errors.
- Commit to ethical design practices, prioritizing environmental and societal impact alongside technical functionality.
- Foster accountability by encouraging team members to take ownership of their tasks and responsibilities.

2) **Collaboration**

Definition: Working effectively as a team by valuing diverse perspectives, leveraging individual strengths, and maintaining mutual respect.

Actions to Support:

- Hold regular team meetings to share updates, brainstorm solutions, and provide constructive feedback.
- Promote open dialogue and equal participation to ensure all voices are heard, regardless of discipline.
- Use collaborative tools like shared project management platforms to track tasks and foster transparency.

3) **Excellence**

Definition: Striving for high-quality outcomes through diligence, professionalism, and continuous improvement.

Actions to Support:

- Set clear goals and timelines, ensuring the team meets project milestones with attention to detail and precision.
- Review and test designs rigorously to ensure the final product meets or exceeds client expectations.

Virtues Important to Team Members

Henry

- **Demonstrated Virtue:** Broad Perspective
    - **Importance:** All team members, especially those in leadership roles, must be able to "zoom out" their view of the project in order to understand the full scope of the project better in order to make proper design decisions and do their work with the rest of the system as a whole in mind.
    - **What I Have Done:** My major, Computer Engineering, has given me a broad exposure to engineering topics from Electrical Engineering to Computer Science. This has allowed me to, at least partially, understand every part of our project and make proper decisions on our system design.
- **Undemonstrated Virtue:** Sociability
    - **Importance:** Ultimately, engineers are human and thus social. A team of friends is more likely to be successful and productive than a team of strangers. The rest of my team were already friends before our project and I hadn't met any of them before this semester.
    - **What I Could Do:** In my future career, I will likely need to continue to collaborate with coworkers. Improving and continuing my sociability skills will allow me to form professional connections and thus receive more opportunities.

Ashley

- **Demonstrated Virtue:** Communication
    - **Importance:** Healthy, productive, and consistent communication is fundamental to a team structure. It is imperative that we speak to our accomplishments, shortcomings, and areas needing improvement.
    - **What I Have Done:** I have acted at the group liaison between our team and our advisor. I also ensure tasks are actively assigned so that we are all on the same page. Open communication has allowed us to avoid the trap of unclear responsibilities.
- **Undemonstrated Virtue:** Confidence
    - **Importance:** It is important to maintain respect for oneself in all situations. Confidence allows us to be less easily toppled by challenges, and enables us to be unafraid of failure.
    - **What I Could Do:** In my career, I could work on asking more questions and being comfortable vocalizing confusion. I need to trust my capabilities and accept shortcomings with optimism rather than defeat.

Drew

- **Demonstrated Virtue:** Altruism
    - **Importance:** Altruism emphasizes contributing to the greater good, fostering collaboration, and advancing projects for the benefit of all.

- **What I Have Done:** Actively assisted team members with tasks, such as troubleshooting code and discussing project strategies.

- **Undemonstrated Virtue:** Eloquence
  - **Importance:** Eloquence enables clear, impactful communication, especially when explaining complex technical concepts and ensures ideas are conveyed effectively, fostering understanding and collaboration with clients and team members.
  - **What I Could Do:** Focus on presenting ideas clearly and confidently in team meetings and client interactions and prepare well-organized talking points and practice delivering explanations in a concise, understandable manner.

Hector

- **Demonstrated Virtue:** Adaptability
  - **Importance:** The ability to adapt to any shift regarding any new client demands, small problems, or working around the schedules of each member is important when working as a team.
  - **What I Have Done:** I have helped the hardware team create a circuit that meets the needs of our client. As a team we had to redesign our circuit numerous times to fit the needs of our client as well as be as efficient and accurate as possible.
- **Undemonstrated Virtue:** Creativity
  - **Importance:** Creativity is important because it allows us to improvise and act in innovative ways to solve problems.
  - **What I Could Do:** I could help my future professional teams by thinking outside of the box to hopefully find a solution for the problem that arose.

Yok

- **Demonstrated Virtue:** Perseverance
  - **Importance:** It is important to help me to solve the problem and make a better version that will fit into our design.
  - **What I Have Done:** When I need to troubleshoot and change the value of the Wheatstone bridge, our team keeps changing the design of the Wheatstone bridge since the first couple weeks till now, even though some of the design is within the range, I would like to get a better range.
- **Undemonstrated Virtue:** Proactivity
  - **Importance:** It helps us to identify the potential problems that I will meet in the future. This is make the progress of our project become more smoother
  - **What I Could Do:** While completing tasks, I will better anticipate changes and communicate these with my team. By sharing the plan we will align our goal and make sure we meet the project deadlines.

Wesley

- **<u>Demonstrated Virtue:</u>** Collaboration
    - **Importance:** Collaboration is essential in team-based projects to leverage diverse skills and perspectives, especially when designing complex systems like a mesh network.
    - **What I Have Done:** I have actively participated in team meetings, shared insights, solutions, and contributed to collective problem-solving.
- **<u>Undemonstrated Virtue:</u>** Initiative
    - **Importance:** Initiative encourages proactive problem-solving and taking on responsibilities without waiting for direction, which can move the project forward efficiently.
    - **What I Could Do:** I will need to continue showing initiative in my career. I will do so by setting up meetings with mentors and bosses, in order to show my desire to grow and learn.

8. Conclusions

8.1 Summary of Progress

In the end, our team was successful in designing a product that is useful to our clients, while documenting it well enough to be sophisticated by future teams. Our results can be summarized most easily by first focusing on the individual accomplishments of the hardware and software teams. These accomplishments consist of not only design decisions, but testing methodology and results as well.

Hardware Accomplishments:

The hardware team worked on designing, simulating, and testing several parts of the circuit, including the voltage divider, voltage regulator, signal buffering stages, and expected output vs actual output measurement. Instead of using Wheatstone bridge, the voltage divider was chosen to convert the sensor's resistance into a readable voltage. The voltage was then buffered and sent to the ADC for accurate measurement. This design kept things simple while still providing reliable and accurate resistance readings. Worst case analysis and unit testing on the PCB were performed to ensure that everything worked as expected. Documentations on design file, assembly, and testing are included for our client to help them have a better understanding how the output responds under different conditions and how to assemble them if they needed.

Software Accomplishments:

The software team configured the development environments that our clients will eventually use to start the system. We successfully set up the FreeRTOS for the ESP32 programming in C. We developed, debugged, and tested the external ADC functionality to capture and transfer the data (resistance value) to the ESP32 for further processing. We then created a network via the LoRa module in order to transmit the data from the measurement nodes to the central base station. The team also created a software to enable data logging onto the SD card for the backup purposes; in the meantime all the data is transmitted to the base station microcontroller. By sending dummy data, we have also been able to ensure the data collection and transmission is stable. We do so by monitoring wifi packets in addition to comparing the base stations received data to the expected data.

Overall, our team has made significant progress in meeting the goals we set out to accomplish at the beginning of our project. We satisfied the primary client requirements, which were to create a network of nodes that can individually measure resistance of the IDE sensors and transmit that data to a central node. The central node data is easily pulled as a .txt file, thus being user friendly for data analysis. Our project overall will allow the researchers to be far more efficient in their data collection, in order to come to accurate conclusions on which methods of pesticide application are most effective.

8.2 Value Provided

Our project has undeniably provided value to our client. The primary objective of our design was to simplify data collection for researchers and thus increase the efficiency of research operations. The easiest way to verify our claim is to reflect on our clients' requirements.

The first requirement was to create a measurement node that can take in data from three different interdigitated electrodes (IDEs). We did this by designing a switching circuit that converted resistance values to voltage. Using this circuit, we were able to utilize an analog to digital converter (ADC) that translates voltage data to a code that can be processed by the microcontroller. By processing, we mean that the ADC can toggle in the digital voltage value bit-by-bit that can be translated back into an analog voltage via our code. We were then able to convert the voltage back into a resistance value. This final value should be within +/- 1% of the actual resistance. By utilizing a switching circuit in addition to a 16-bit ADC, we were able to maintain the precision of our measurements.

That resistance measurement can then be stored locally, another requirement. We utilized a separate SD component that has enough memory to directly store resistance measurements at each node (i.e. microcontroller). Our clients needed these measurements to then be transmitted to a base station over a range of at least 200 feet. We selected a LoRa module specifically designed for long range communication, which allowed us to transmit data over this distance. With this capability, our base node could collect data from a number of nodes and therefore be accessible to the researchers.

A final requirement was that the data should be easily obtained and interpreted by researchers. By creating an operation manual (as referenced in Appendix 1) and an easy-to-read text file that compiled all measurements, we met this final requirement.

8.3 Next Steps

There are numerous additional steps that could be taken following the completion of our project to increase the robustness and applicability of our design. They fall within the following areas:

Mesh Network Implementation: As aforementioned, one of the original requirements of creating a mesh network was not implemented. After collaborating with our client, we found it to be not necessary in the scope of this project as our long range wifi module met the distance needs. Nonetheless, this mesh network frame could be used in the future, especially if distance requirements are increased.

Graphical User Interface: Our final interface was quite simplistic and lacked an official interface. Instead, there was a basic console that required specific commands in a programming fashion. If this design is used outside of research, a sleeker user interface could be useful and even required.

Increased Scope: In our project, we focused on creating a singular node that interacted with a base node. Due to time constraints, we were unable to expand the number of measuring nodes. This is definitely an aspect of our project that could be updated by a future team.

Overall, there are a number of updates that could be made, especially if this project is deployed on a commercial scale.

9. References

[1] “IEEE SA - IEEE Standard for Information Technology--Telecommunications and information exchange between systems--local and metropolitan area networks--specific requirements part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications amendment 10: Mesh Networking,” IEEE Standards Association, https://standards.ieee.org/ieee/802.11s/4243/ (accessed Dec. 5, 2024).

[2] “IEEE SA - IEEE Standard Profile for use of IEEE 1588 Precision Time Protocol in Power System Applications,” IEEE Standards Association, https://standards.ieee.org/ieee/C37.238/4609/ (accessed Dec. 5, 2024).

[3] “IEEE SA - IEEE standard for Telecommunications and information exchange between systems - LAN/man - specific requirements - part 15: Wireless Medium Access Control (MAC) and Physical

Layer (PHY) specifications for Wireless Personal Area Networks (wpans),” IEEE Standards Association, https://standards.ieee.org/ieee/802.15.1/1180/ (accessed Dec. 5, 2024).

[4] Esp32-C6 series, https://www.espressif.com/sites/default/files/documentation/esp32-c6_datasheet_en.pdf (accessed Dec. 5, 2024).

[5] Esp32-C6, https://www.espressif.com/sites/default/files/documentation/esp32-c6_technical_reference_manual_en.pdf (accessed Dec. 5, 2024).

[6] “Waspmote,” Libelium, https://www.libelium.com/iot-products/waspmote/ (accessed Dec. 5, 2024).

[7] “IMETOS 3.3,” METOS® by Pessl Instruments, https://metos.global/en/imetos33/ (accessed Dec. 5, 2024).

[8] “IEEE SA - IEEE standard for design and verification of low power integrated circuits,” IEEE Standards Association, https://standards.ieee.org/ieee/1801/4189/ (accessed Dec. 7, 2024).

[9] L. J. McNeill and M. M. Bellamy, "Contextualizing professionalism in capstone projects using the IDEALS professional responsibility assessment," *International Journal of Engineering Education*, vol. 28, no. 2, pp. 416–424, 2012.

[10] R. Kumar and M. P. Singh, "Wireless Sensor Networks for Precision Agriculture," IEEE Communications Magazine, vol. 54, no. 3, pp. 34-40, March 2018.

[11] S. P. Mohanty, U. Choppali, and E. Kougianos, "Everything You Wanted to Know About Smart Agriculture: Sensors, Systems, and Applications," IEEE Transactions on Consumer Electronics, vol. 62, no. 1, pp. 45-54, Feb. 2019.

[12] J. Smith, P. Lee, and R. Patel, "IoT-Based Pest Monitoring and Control Systems in Agriculture," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1234-1242, April 2020.

10. Appendices

Appendix 1 - Operation Manual

Throughout our creation of our senior design project we’ve created many forms of documentation elaborating on the setup, demo, test, and use of our system. We’ve provided these to our client in forms

of powerpoints, pdf files, etc. but this is a summarization of all of those documents. Previously in parts 5 and 6 we had discussed the inner workings of the hardware and the software. To begin we will start with the hardware operation manual and move to the software operation manual.

Hardware Operation Manual:
Firstly, to assemble our system you will need all the components below. The only thing excluded from this BOM is the PCB. The PCB file can be found in our KiCAD documentation.

Index	Quantity	Part Number	Manufacturer Part Number	Description	Vendor
1	1	490-GRM21BR60G107ME11KCT-ND	GRM21BR60G107ME11K	CAP CER 100UF 4V X5R 0805	DigiKey
2	1	490-5523-1-ND	GRM21BR61E106KA73L	CAP CER 10UF 25V X5R 0805	DigiKey
3	3	1276-1029-1-ND	CL21B105KBFNNNE	CAP CER 1UF 50V X7R 0805	DigiKey
4	4	311-1140-1-ND	CC0805KRX7R9BB104	CAP CER 0.1UF 50V X7R 0805	DigiKey
5	3	S8.2KCACT-ND	RNMF14FTC8K20	RES 8.2K OHM 1% 1/4W AXIAL	Digikey
6	3	S330CACT-ND	RNMF14FTC330R	RES 330 OHM 1% 1/4W AXIAL	Digikey
7	3	S2.7KCACT-ND	RNMF14FTC2K70	RES 2.7K OHM 1% 1/4W AXIAL	Digikey
8	3	MFR-25FBF52-27K-ND	MFR-25FBF52-27K	RES 27K OHM 1% 1/4W AXIAL	Digikey
9	1	S1KCACT-ND	RNMF14FTC1K00	RES 1K OHM 1% 1/4W AXIAL	Digikey
10	1	S3KCACT-ND	RNMF14FTC3K00	RES 3K OHM 1% 1/4W AXIAL	Digikey
11	1	LM1117MPX-5.0/NOPBCT-ND - Cut Tape (CT)	LM1117MPX-5.0/NOPB	IC REG LINEAR 5V 800MA SOT-223-4	DigiKey
12	3	J10126-ND	111-0703-001	CONN BIND POST KNURLED BLACK	DigiKey
13	3	J10125-ND	111-0702-001	CONN BIND POST KNURLED RED	DigiKey
14	1	P687-ND	6LF22XWA/B	BATTERY ALKALINE 9V	Digikey
15	1	36-84-8-ND	84-8	BATT CONNECTOR SNAP 9V 8" LEADS	Digikey
16	1	S7014-ND	PPTC161LFBN-RC	CONN HDR 16POS 0.1 TIN PCB	Digikey
17	2	ED10561-ND	OSTVN02A150	TERM BLK 2P SIDE ENT 2.54MM PCB	Digikey
18	3	AD7171BCPZ-REEL7CT-ND	AD7171BCPZ-REEL7	IC ADC 16BIT SIGMA-DELTA 10LFCSP	Digikey
19	6	BKCT3805-30-ND	CT3805-30	TEST LEAD BANANA TO GATOR 12"	DigiKey
20	1	4265-DESDM-04GS02AW1ST-ND	DESDM-04GS02AW1ST	4GB Micro SD Card	Digikey
21	1	1528-4682-ND	4682	MICROSD SPI/SPIO BREAKOUT BOARD	Digikey
22	6	505-AD8628WARZ-R7CT-ND	AD8628WARZ-R7	IC SW SPST-NOX1 300MOHM SOT23-6	Digikey
23	3	505-ADR5041BRTZ-REEL7CT-ND	ADR5041BRTZ-REEL7	IC VREF SHUNT 0.1% SOT23-3	Digikey
24	6	ADG801BRTZ-REEL7CT-ND	ADG801BRTZ-REEL7	IC SW SPST-NOX1 300MOHM SOT23-6	Digikey
25	11	36-5006-ND	5006	Test point	DigiKey
26	1	3072	RFM95W-915S2	AdaFruit FRM95W LoRa Radio Transceiver Breakout - 868 or 915MHz	AdaFruit

Appendix 1: Bill of Materials

Secondly the assembly portion, this portion is all well documented in a detailed powerpoint presentation that we've provided to our client. To assemble our system we segmented it into two portions, reflow soldering and through hole soldering / finalizing. Here are the steps.

<u>Reflow Soldering -</u>

1. Tape down PCB, follow this by taping the stencil over top of the PCB leaving only metal exposed.
2. Apply and scrape solder paste (Kester R500 water soluble solder paste) across the stencil applying paste to all exposed areas on the PCB.
3. Remove stencil and tape, identify surface mount components and add surface mount components where specified. *if using an un-updated version of the KiCAD document do not populate C9, C10, C11*
4. Place the PCB in a reflow solder oven and begin the reflow soldering process.
5. Ensure components are soldered correctly, any component legs that may have been soldered together only need to have flux applied and set under a heat gun to re-settle the solder. Other complications are outside the scope of this guide and will need to be identified and resolved by the user through their own troubleshooting.

<u>Through Hole Soldering / Finalizing -</u>

1. Preheat soldering iron and solder through hole components where specified in design ensuring flux is applied for efficient soldering.

2. Solder pins to SD card module and LoRa module, then add them to the board.
3. Screw on binding posts and screw in battery terminal wires.

To test that the system is functioning as intended we recommend that a testing regime similar to the one we conducted is done. This is all documented in our testing documentation but here is a summary of our testing process.

1. Document set test resistor values of resistors ranging from 100 Ω - 200 kΩ.
2. Plug a 9V battery, ESP32-C6 microcontroller able to switch between resistance ranges.
3. Document voltage regulator output ensuring a 5 V output voltage.
4. Document reference voltage circuit outputs ensuring a 2.5 V output voltage.
5. Calculate expected resistances according to these measurements.
6. Measure voltage output of circuit with resistors in series with the circuit utilizing the binding posts.
7. Ensure expected voltages align with measured voltages.
8. Complications regarding testing are outside the scope of this guide and will need to be identified and resolved by the user through their own troubleshooting.

With these steps completed we've discerned the circuit functions as intended and can measure resistances in the specified range. Thus, the circuit is ready to be tested using sensors or anything else the user intends to measure.

Software Operation Manual:
Operating our project in the field consists of interacting with the base station over a cable connection that supports a command line. The command line will provide information about what is happening on the network, as well as what commands are available to the user at any point in time.
To deploy the network

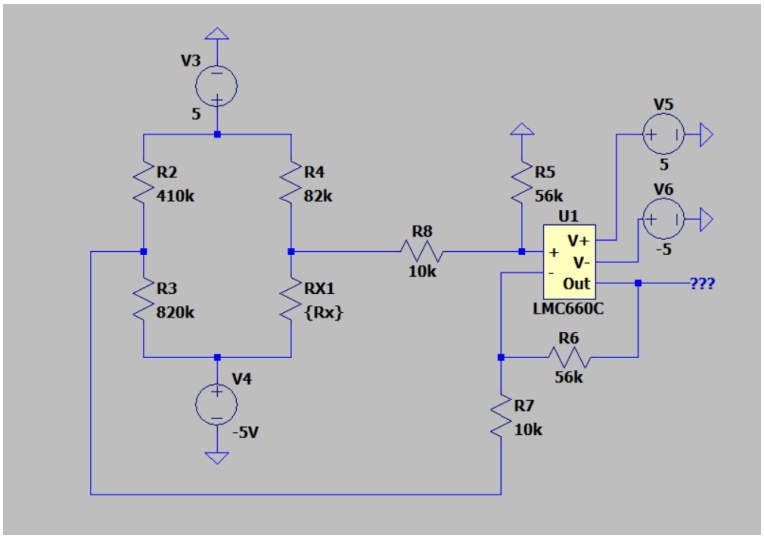
1. Place measurement nodes and their sensors in the field & turn each measurement node on
2. Use the Espressif idf tool to flash the code to the base station and monitor its command line with a laptop and usb c cable
3. Upon starting up you will have two commands available to you, Check Connections & Start Recording
4. Use Check Connections to have the base station query each measurement node to ensure that it can communicate with it
5. If any connections fail, move that node or its antenna to attempt to improve the connection
6. Once all connections have been validated, start recording
7. While you are recording, the only command available to you will be Stop Recording. Once you have finished spraying the field and are ready to conclude your test, run Stop Recording
8. Allow the base station to run until you get a message on the terminal that it has finished its cleanup and is ready to be shut down
9. Once the base station is ready to be shut down, simply disconnect it from your laptop and collect all the measurement nodes form the field and turn them off
10. The data from each run is stored on a file on each board's SD card. The base station's will contain all of the sensor readings and each measurement node's SD card will contain the sensor readings it took (to be used as a backup)

Appendix 2 - Alternative and Initial Versions of Design

We had numerous design changes as a result of requirement changes, obstacles, and advisor input. These changes occurred on both the software and hardware sides, and thus have been organized accordingly.

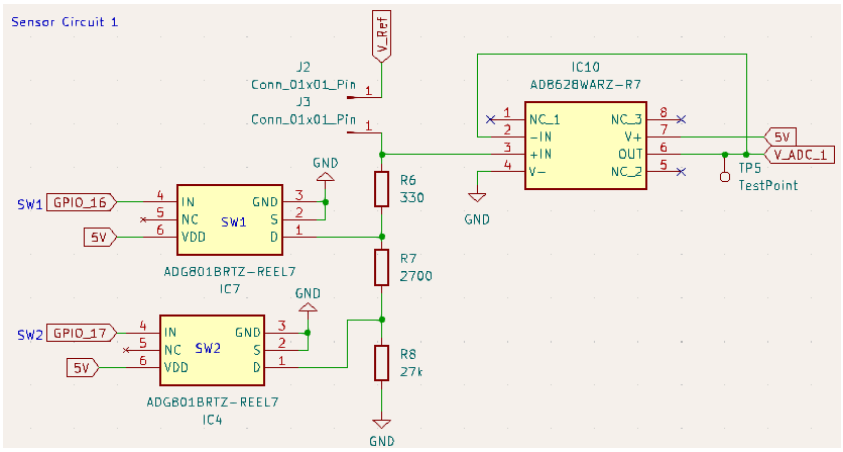
Hardware Iterations:

1. Voltage Divider to Wheatstone Bridge: We initially utilized a basic voltage divider circuit as means to convert resistance measurements to voltage that could be fed into the ADC. However, after speaking to our advisor, we determined we needed a circuit with far more accuracy than what a voltage divider would allow. Hence, we utilized Wheatstone Bridges, instead. The wheatstone bridge is far more sensitive to small changes in resistance and thus was a better design choice. The implementation of the wheatstone bridge can be seen in the first iteration of our circuit below.

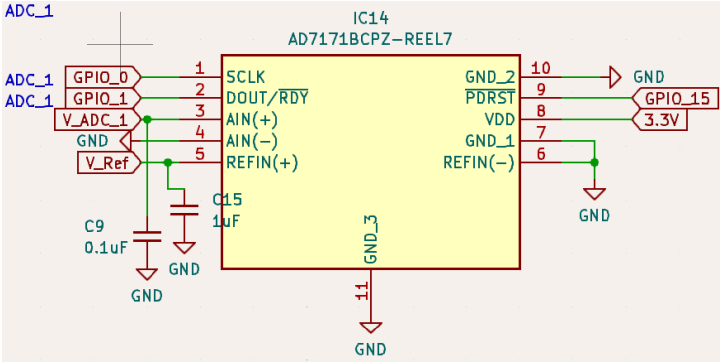


2. Change in Resistance Requirements: Our next iteration of our conversion circuit came about from a client change in the resistance we were required to account for. The range was drastically increased to account for 100 Ω (fully saturated) to 200k Ω (barely saturated). While our original design could technically accommodate this range, it would not do so with the precision required. This resulted in us making the following changes:

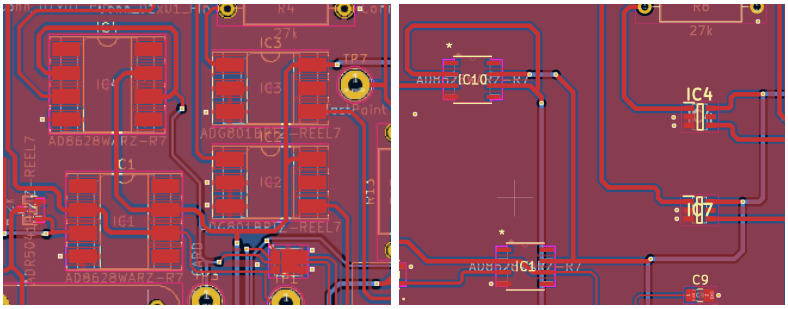
    a. Switching circuit: we increased the complexity of our design to account for three different ranges of resistance values that can be digitally switched between. That allows us to maintain precision for three different smaller ranges.



b. In order to account for the precision, however, we had to alter our ADC design. Originally, the 12-bit ADC internal to the microcontroller was to be utilized by taking in the voltage output from the circuit. To account for the larger ranges with an accuracy still within the clients' parameters, we opted for an external 16-bit ADC. While not a major change to our hardware, it did result in relatively significant software changes.



3. <u>PCB Alterations:</u> After creating an initial PCB design, we submitted it to be manufactured. After receiving the physical board, we realized there were multiple issues.
   a. Components footprint
      In the first version of our PCB design, two component footprints were incorrect. The footprints for both the op-amp and the switches were larger than the actual components, which could have caused placement and soldering issues. The figure below shows the difference between the original footprints and the corrected version.



b. Component selection
During the initial component order through ETG, incorrect parts, wrong PCBs and stencil were selected, which led to delays in both building and testing the PCB . The issue affected our timeline and required reordering the correct components, pushing back our development schedule. Besides, there were several other instances where incorrect parts were ordered due to the overlooked specifications. These setbacks highlighted the importance of carefully cross-checking component details, including datasheets, and part numbers before placing order.

Software Iterations:

1. Mesh Network to LoRa module: Initially, we anticipated utilizing the mesh network functionality of the ESP32 microcontroller. However, we ran into quite a few issues with implementing the design. The microcontroller's mesh network documentation proved to be difficult to follow and adapt for our application. After spending an extensive amount of time attempting to make the network work, we made the decision to sway from the network entirely. After speaking with our advisor and clients, we made the decision to utilize an external LoRa module in conjunction with the microcontroller. The LoRa module is designed for long range communication networks. The range capacity rendered a mesh network design unnecessary, as its original purpose was to account for distance between the measurement nodes and base nodes.

2. External ADC: Per above, we decided to use an external ADC rather than the original internal one. Below, is an excerpt of the original ADC code:

```
void app_main(void)
{
    // GPIO setup
    gpio_set_direction(GPIO_NUM_1, GPIO_MODE_OUTPUT);

    // ADC configuration
    adc_oneshot_unit_init_cfg_t unit_cfg = {
        .unit_id = ADC_UNIT_1,
    };
    adc_oneshot_new_unit(&unit_cfg, &adc_handle);

    adc_oneshot_chan_cfg_t config = {
        .bitwidth = ADC_BITWIDTH_DEFAULT,  // Use the default bit width
        .atten = ADC_ATTEN_DB_12,  // Using the updated attenuation constant
    };
    adc_oneshot_config_channel(adc_handle, ADC_CHANNEL_0, &config);
    adc_oneshot_config_channel(adc_handle, ADC_CHANNEL_4, &config);
    adc_oneshot_config_channel(adc_handle, ADC_CHANNEL_5, &config);

    int adc_raw = 0;
    char userInput;
```

As seen, the original ADC code was simplistic as it could easily take advantage of built in functions, due to the ADC being internal to the microcontroller. There was example code in Espressif's documentation that made the coding process far easier. However, the switch to an

external ADC complicated our process significantly. The basics of our code took advantage of a strategy called “bit-banging”. The ADC itself has the ability to take in a digitized 24-bit code that corresponds to voltage values. To feed this data into the microcontroller, we toggled the clock and manually read in one bit at a time.

```
// Read ADC data manually
uint32_t ad7171_read_data_manual(int dout_rdy_pin, int sclk_pin) {
    uint32_t result = 0;

    for (int i = 0; i < 24; i++) {
        gpio_set_level(sclk_pin, 1);
        delay_us(50);

        result <<= 1;
        if (gpio_get_level(dout_rdy_pin)) {
            result |= 1;
        }

        gpio_set_level(sclk_pin, 0);
        delay_us(50);
    }



    return result;
}
```

Once manually read, our microcontroller could convert the ADC code to a voltage value and then into a resistance value. The final resistance value is saved to an SD card and then transmitted via our communication system.

Overall, our iterations were important to account for requirement changes, challenges, and compatibility. Although sometimes frustrating, these design changes were imperative for our success and allowed us to individually grow. The iterative process required creative thinking, resiliency, and communication with our client, all virtues that are relevant to our success in engineering.

Appendix 3 - Other Considerations

Upon the completion of senior design, it’s helpful to look back and reflect on our experiences. One of our most important takeaways is the power of resiliency. We had many moments of mishaps, misunderstanding, and flat out frustration. One, now funny, frustration was our inability to get our SD card component to interact with our microcontroller properly. After hours of troubleshooting and refining our code, we could not for the life of us figure out where we went wrong. Turns out, we forgot to ground the microcontroller. Our team could not accept that we messed up the simplest part of our project and had to laugh or we would cry.

Another irritation was our inability to get the correct PCB delivered. Our first order came back as we anticipated, although it was not compatible with the rest of our design. Hence, we decided to redesign it and order a new iteration of our design. However, ETG decided to send off our first file again, and we received an exact replica of our PCB that we had already determined did not work. Eventually we got the correct board, but not without the squandering of precious time. In the future, we will almost certainly take into account small but impactful errors such as this when making project timelines.

Overall, we look back positively on our senior design project. Our team bonded over silly mistakes and late nights in the lab. Without each other, our sanity may have been lost throughout this project.

Appendix 4 - Code

Our team took advantage of GitLab in order to have version control tools and increase collaboration effectiveness. We have included the link below, in addition to a link to our shared CyBox which will have finalized code and documentation:

GitLab link: https://git.las.iastate.edu/hrhingst/cpre491
CyBox link: https://iastate.box.com/s/axzvw7fagl9qmyijiuvoo1d2a1vrj5q1

We will describe a few interesting/important snippets of our code, but for our full code and descriptions of all the systems involved, we encourage you to visit our CyBox repository.

Device Registration and Recognition

To manage a network of ESP32 devices, we use a system that registers each device by its MAC address and assigns it a role—either a base node or a measurement node. This setup ensures that only known devices can join the network and operate as expected.

```c
#include "device_config.h"

DeviceConfig configTable[] = {
    {{0xF0, 0xF5, 0xBD, 0x02, 0x75, 0xF4},
        0,
        {-1, -1, -1}
    },
    {{0x40, 0x4C, 0xCA, 0x57, 0x87, 0xB8},
        1,
        {0, 1, 2}
    },
    {{0xF0, 0xF5, 0xBD, 0x0D, 0xC3, 0x44},
        2,
        {3, 4, 5}
    },
    {{0xF0, 0xF5, 0xBD, 0x0E, 0x62, 0x1C},
        3,
        {6, 7, 8}
    },
    {{0xF0, 0xF5, 0xBD, 0x02, 0x72, 0xA0},
        4,
        {9, 10, 11}
    },
    {{0xF0, 0xF5, 0xBD, 0x0E, 0x63, 0x3C},
        5,
        {12, 13, 14}
    },
};

const int configCount = sizeof(configTable) / sizeof(configTable[0]);
```

This file lists all approved devices by MAC address. Each entry includes a unique ID and sensor slots. The base node has ID 0; others are measurement nodes. This list is how we identify and set up each device.

```c
/* REGISTRATION CHECKER */
    uint8_t mac[6];
    // esp_efuse_mac_get_default(mac);
    esp_read_mac(mac, ESP_MAC_WIFI_STA);
    DeviceConfig *myConfig = NULL;
    //
    // Retrieve device's config data from device_config
    for (int i = 0; i < configCount; i++) {···
    //
    // If device not recognized (Not registered)
    if (!myConfig) {
        // If no base node has been registered, ask user to register base node in device_config.c
        if (configCount == 0) {···
        // Ask user to register measurement node in device_config.c
        else {···
        return;
    }
    //
    // printf("Device Recognized as Device %d!\n", myConfig->deviceID); // HACK

/* MAIN PROTOTYPE */
    /* BASE NODE */
        if (myConfig->deviceID == 0) {···
    /* MEASUREMENT NODE */
```

At startup, the device reads its MAC address and compares it to the list. If it’s found, it loads its config. If not, the program prints out the code needed to add the device. This blocks unregistered devices.

```c
/* BASE NODE */
    if (myConfig->deviceID == 0) {
        char userInput = 0;
        while (1) {
            printf("\n"
                   "What would you like to do?\n"
                   "Type one of the following to continue:\n"
                   "\t\"H\" to send a Heartbeat request to all devices to check communication\n"
                   "\t\"S\" to Start recording sensor values from all operational measurement nodes\n"
                   "\t\"E\" to Exit the program\n"
                   "> ");
            //
            userInput = 0;
            while (userInput != 'H' && userInput != 'S' && userInput != 'E' && userInput != 'h' &&
                   userInput != 's' && userInput != 'e') {
                scanf("%c", &userInput);
                vTaskDelay(1);
            }
            printf("%c\n", userInput);
            //
            // Perform action based on userInput
            if (userInput == 'H' || userInput == 'h') {···
            } else if (userInput == 'S' || userInput == 's') {···
            } else if (userInput == 'E' || userInput == 'e') {···
        }
        return;
    }
```

If the device is the base node, it shows a simple menu. The user can check if other nodes are online, start recording data, or exit. This menu lets the base node control the network.

Base Node Main Loop

Another important segment of our code is the main loop of the base station.

```
//The base node still waits to receive the reading response
//Has a timeout of 1 second so that the base node won't get stuck here
while(1){
    time(&currentTime);
    if(currentTime - packetSentTime > BASE_STATION_READING_TIMEOUT){
        printf("Ran out of time waiting for a reading response from %d\n", measurementNodeIDs[i]);
        printf("\n");
        break;
    }
    if(lora_received()){
        //Get the message size and contents from the lora module
        const int messageMaxSize = 252;
        uint8_t message[messageMaxSize];
        int messageSize = lora_receive_packet(message, messageMaxSize);

        //The smallest packet we have is 3 bytes so if message is less than that then it's not one of ours
        if(messageSize < 3){ ···

        //Pull the default header info off of the packet
        uint8_t destinationAddress = message[0];
        uint8_t senderAddress = message[1];
        uint8_t packetType = message[2];
        printf("Received packet of size %d from %d to %d of type %d\n", messageSize, senderAddress, destinationAddress, packetType);

        //Quit processing if the packet isn't for us
        if(destinationAddress != baseStationID && destinationAddress != LORA_BROADCAST_ID){ ···

        //Process the rest of the packet based off of what type it is
        if(packetType == START_RECORDING_TYPE){ ···
        }else if(packetType == STOP_RECORDING_TYPE){ ···
        }else if(packetType == READING_REQUEST_TYPE){ ···
        }else if(packetType == READING_RESPONSE_TYPE){ ···
        }else if(packetType == HEARTBEAT_REQUEST_TYPE){ ···
        }else if(packetType == HEARTBEAT_RESPONSE_TYPE){ ···
        }else{ ···

        printf("\n");
    }else{
        vTaskDelay(pdMS_TO_TICKS(100));
    }
}
```

It consists of a while loop that checks two things. First, if enough time has passed since the last time it polled for sensor readings, then it will send reading requests to each measurement node for each sensor, wait for their response, and save those readings to the SD card. Second, it checks if it has received anything on the LoRa module. If it has then it attempts to deal with it. This part is mostly a failsafe designed to reduce errors from missing a reading request response due to waiting too long. If neither of these are true then it simply delays the task for 100ms and loops again.

SD Card Directory and File Initialization

To organize and store sensor data, our system uses an SD card. For each recording run, a new directory is created, and individual files are set up for each sensor. This approach keeps data organized, avoids overwriting, and ensures that each run's data is saved separately.

```h
#pragma once
#include <stdio.h>
#include "esp_err.h"

// Function prototypes
esp_err_t sd_init(void);

void sd_info(void);

esp_err_t sd_format(void);

int directory_exists(const char *path);

esp_err_t sd_create_directory(const char *path);

esp_err_t sd_write_to_file(char *text, char *file_path, char *write_or_append);

void sd_dinit(void);
```

This header defines the interface for SD card operations implemented in sd_driver.c. It includes functions to initialize and deinitialize the SD card, check for directories, create directories, write to files, and optionally format the card. These functions help manage file I/O and directory structures safely and cleanly.

```c
/* AT START OF RECORDING RUN */
    xTaskCreate(sd_init_task, "sd_init_task", 4096, NULL, 5, NULL);
    vTaskDelay(pdMS_TO_TICKS(1000));
    //
    /* DEDICATE NEW DIRECTORY FOR THIS RECORDING RUN */
        while (1) {
            snprintf(run, sizeof(run), "/RUN%d", runNum);
            if (!directory_exists(run)) {
                // ESP_LOGI(TAG, "Directory \"%s\" does not yet exist!", run); // HACK
                break;
            } else {
                // ESP_LOGE(TAG, "Directory \"%s\" already exists!", run); // HACK
            }
            runNum++;
        }
        ret = sd_create_directory(run);
        if (ret != ESP_OK) return;
    //
```

At the start of a recording run, we launch a task to initialize the SD card. Then we generate a unique directory name (e.g., /RUN0, /RUN1, etc.) by checking for existing directories and incrementing a counter until a new one is found. Once a unique directory name is confirmed, it's created on the SD card to store that run’s data.

```
int dataSize = 0;
if (WRITE_READING_NUM_TO_SD) {
    dataSize += 5 + 1; // Include room in data for reading # (max 65535) + ","
}
if (WRITE_DELTA_TIME_TO_SD) {
    dataSize += 5 + 1 + 1; // Include room in data for delta time (seconds since last reading, with precision PRECISION) (max 65535) + "." + ","
}
if (WRITE_ABSOLUTE_TIME_TO_SD) {
    dataSize += 10 + 1 + 1; // Include room in data for absolute time (seconds since recording start, with precision PRECISION) (max 4294967295)
}
dataSize += 6 + 1 + 1; // Include room in data for reading value (max 999999) + "\n" + "\0"
char data[dataSize];
//
// Generate Each Sensor's File Path
char *sensorFilePaths[3];
for (int i = 0; i < 3; i++) {
    int filePathLength = 1 + strlen(run) + 1 + 5 + strlen(FILE_TYPE) + 1;
    sensorFilePaths[i] = malloc(filePathLength);
    // snprintf(sensorFilePaths[i], filePathLength, "/%s/%d%s", run, i, FILE_TYPE);     // Sensors 0,1,2
    snprintf(sensorFilePaths[i], filePathLength, "/%s/%d%s", run, (i + 1), FILE_TYPE);  // Sensors 1,2,3
}
```

After the directory is created, we prepare file paths for each of the three sensors. Each file is named using its sensor number and a fixed file type (e.g., .txt). These paths are stored in an array for use during data logging. We also calculate the maximum buffer size needed for each data entry, depending on what data fields are being written (e.g., reading number, timestamps, sensor value).

Measurement Node Sensor Reading and Switching

Our measurement node reads from three sensors that can operate across three resistance ranges (low, mid, high). Since resistance affects voltage differently at each range, the system switches between ranges automatically based on voltage thresholds to maintain accuracy. This is done using GPIO-controlled analog switches and ADC voltage readings, which are then converted to resistance values.

```
#pragma once

#define ADC_SCLK 0

#define SENSOR1_ADC_DOUT_RDY 1
#define SENSOR1_ADC_PDRST 15
#define SENSOR1_SWITCH1_IN 16
#define SENSOR1_SWITCH2_IN 17

#define SENSOR2_ADC_DOUT_RDY 22
#define SENSOR2_ADC_PDRST 23
#define SENSOR2_SWITCH1_IN 21
#define SENSOR2_SWITCH2_IN 20

#define SENSOR3_ADC_DOUT_RDY 13
#define SENSOR3_ADC_PDRST 9
#define SENSOR3_SWITCH1_IN 18
#define SENSOR3_SWITCH2_IN 19

// TODO: Make into a struct instead?
extern const int ADC_DOUT_RDYS[3]; // {SENSOR1_ADC_DOUT_RDY, SENSOR2_ADC_DOUT_RDY, SENSOR3_ADC_DOUT_RDY}
extern const int ADC_PDRSTS[3]; // {SENSOR1_ADC_PDRST, SENSOR2_ADC_PDRST, SENSOR3_ADC_PDRST}
extern const int SWITCHES[6]; // {SENSOR1_SWITCH1_IN, SENSOR1_SWITCH2_IN, SENSOR2_SWITCH1_IN, SENSOR2_SWITCH2_IN, SENSOR3_SWITCH1_IN, SENSOR3_SWITCH2_IN}

#define LORA_RST 8
#define LORA_CS 10
#define LORA_MOSI 11
#define LORA_MISO 2
#define LORA_SCK 3

#define SD_CS_D3 4
#define SD_SI_CMD 5
#define SD_SO_D0 6
#define SD_CLK 7
```

This file defines GPIO pin assignments for each sensor's ADC data line, reset line, and the two control lines used for switching ranges. Arrays are provided to access pins by sensor index. Grouping the pins like this makes it easier to loop through sensors and control them programmatically.

```
/* MEASURED VALUES */
    /* THRESHOLD VOLTAGES */
        // Switch to MID range when voltage is less than LOW_TO_MID
        #define LOW_TO_MID  0.62455 // Exact Bound  (R = 1k)
        // #define LOW_TO_MID   0.58081 // Leeway Bound (R = 1.1k)
        //
        // Switch to LOW range when voltage is greater than MID_TO_LOW
        #define MID_TO_LOW  1.88079 // Exact Bound  (R = 1k)
        // #define MID_TO_LOW   1.92566 // Leeway Bound (R = 900)
        //
        // Switch to HIGH range when voltage is less than MID_TO_HIGH
        #define MID_TO_HIGH 0.58190 // Exact Bound  (R = 10k)
        // #define MID_TO_HIGH  0.54090 // Leeway Bound (R = 11k)
        //
        // Switch to MID range when voltage is greater than HIGH_TO_MID
        #define HIGH_TO_MID 1.87777 // Exact Bound  (R = 10k)
        // #define HIGH_TO_MID  1.92795 // Leeway Bound (R = 9k)
    //
    /* VOLTAGE TO RESISTANCE CONVERSION */
        #define LOW_RANGE_R2_RESISTANCE 330.1106667
        #define MID_RANGE_R2_RESISTANCE 2980.666667
        #define HIGH_RANGE_R2_RESISTANCE 30060
        //
        #define VOLTAGE_SOURCE 2.49914
        //
        #define VOLTAGE_TO_RESISTANCE(R2_RESISTANCE, voltageReading) (((VOLTAGE_SOURCE * R2_RESISTANCE) / voltageReading) - R2_RESISTANCE)
```

Voltage thresholds determine when the system should switch between low, mid, and high measurement ranges. Constants for resistor values and the voltage source allow us to convert voltage readings into resistance using a standard formula. This makes the measurement logic adaptable and precise across varying sensor outputs.

```
/* READ SENSORS */
    for (int i = 0; i < 3; i++) {
        //
        sum = 0;
        //
        for (int j = 0; j < READINGS_TO_AVERAGE; j++) {
            // Sets switches to Mid Range
            gpio_set_level(SWITCHES[i * 2], 0);         // Set Switch 1 (Open (IN=0))
            gpio_set_level(SWITCHES[(i * 2) + 1], 1);   // Set Switch 2 (Closed (IN=1))
            vTaskDelay(pdMS_TO_TICKS(SWITCH_SETTLE_TIME_MS));
            tempVoltageReading = ad7171_read(i + 1);
            if (tempVoltageReading > MID_TO_LOW) { // Use Low Range
                gpio_set_level(SWITCHES[i * 2], 1); // Set Switch 1 (Closed (IN=1))
                vTaskDelay(pdMS_TO_TICKS(SWITCH_SETTLE_TIME_MS));
                //
                tempVoltageReading = ad7171_read(i + 1);
                //
                // Convert voltage to resistance using Low Range equation
                tempResistanceReading = VOLTAGE_TO_RESISTANCE(LOW_RANGE_R2_RESISTANCE, tempVoltageReading);
            } else if (tempVoltageReading < MID_TO_HIGH) { // Use High Range
                gpio_set_level(SWITCHES[(i * 2) + 1], 0);   // Set Switch 2 (Open (IN=0))
                vTaskDelay(pdMS_TO_TICKS(SWITCH_SETTLE_TIME_MS));
                //
                tempVoltageReading = ad7171_read(i + 1);
                //
                // Convert voltage to resistance using High Range equation
                tempResistanceReading = VOLTAGE_TO_RESISTANCE(HIGH_RANGE_R2_RESISTANCE, tempVoltageReading);
            } else {
                // Convert voltage to resistance using Mid Range equation
                tempResistanceReading = VOLTAGE_TO_RESISTANCE(MID_RANGE_R2_RESISTANCE, tempVoltageReading);
            }
            //
            sum += tempResistanceReading;
        }
        //
        now = (uint32_t) xTaskGetTickCount();
        previousAbsoluteTime = absoluteTime[i];
        // Time is measured relative to when recording started
        absoluteTime[i] = (now - startTick) / (configTICK_RATE_HZ * PRECISION);
        deltaTime = (uint16_t) (absoluteTime[i] - previousAbsoluteTime);
        //
        averageReading = (uint32_t)((sum / READINGS_TO_AVERAGE) + 0.5f);
```

For each sensor, we first set the range to mid by default. We read the voltage, and based on its value, we may switch to low or high range using GPIO controls. After letting the circuit settle, we take a new reading and convert it to resistance using the appropriate formula. We repeat this process several times to compute an average resistance. We also track timestamps for each reading to support time-based analysis.

ADC Driver

In our system, we use the AD7171 ADC to convert analog voltage signals from our sensors into digital values that can be processed by the microcontroller. This process involves manually clocking out data from the ADC, interpreting control bits, and converting the result into a usable voltage value. The ADC provides 24-bit data, of which 16 bits represent the measured signal and the remaining bits include status and pattern checks to verify data validity.

```c
// Initialize ADC
void ad7171_init() {
    gpio_reset_pin(ADC_SCLK);
    gpio_set_direction(ADC_SCLK, GPIO_MODE_OUTPUT);

    for (int i = 0; i < (sizeof(ADC_DOUT_RDYS) / sizeof(ADC_DOUT_RDYS[0])); i++) { // Set up DOUT/RDY pins
        gpio_reset_pin(ADC_DOUT_RDYS[i]);
        gpio_set_direction(ADC_DOUT_RDYS[i], GPIO_MODE_INPUT);
        gpio_set_pull_mode(ADC_DOUT_RDYS[i], GPIO_PULLDOWN_ONLY);
    }

    for (int i = 0; i < sizeof(ADC_PDRSTS) / sizeof(ADC_PDRSTS[0]); i++) { // Set up PDRST pins
        gpio_reset_pin(ADC_PDRSTS[i]);
        gpio_set_direction(ADC_PDRSTS[i], GPIO_MODE_OUTPUT);
        gpio_set_level(ADC_PDRSTS[i], 1);
    }

    vTaskDelay(pdMS_TO_TICKS(24)); // Setup time
}
```

This setup function initializes the ADC interface. It configures the clock (SCLK) as an output and sets up the DOUT/RDY and power/reset (PDRST) pins for each sensor. It also ensures that each ADC is powered and ready before readings begin, using a short delay to allow setup time.

```c
// Read ADC voltage
float ad7171_read(int sensor_num) {

    uint32_t raw_data;

    while (1) {

        vTaskDelay(pdMS_TO_TICKS(10)); // HACK Reduces erroneous readings

        raw_data = ad7171_read_data_manual(sensor_num);

        uint8_t adc_flags = (uint8_t)raw_data;

        int RDY = (adc_flags >> 7) & 0x01;
        int ZERO = (adc_flags >> 6) & 0x01;
        int ERR = (adc_flags >> 5) & 0x01;
        int ID1 = (adc_flags >> 4) & 0x01;
        int ID0 = (adc_flags >> 3) & 0x01;
        int PAT2 = (adc_flags >> 2) & 0x01;
        int PAT1 = (adc_flags >> 1) & 0x01;
        int PAT0 = (adc_flags >> 0) & 0x01;

        if ((RDY == 0) && (ZERO == 0) && (ERR == 0)) {
            if ((ID1 == 0) && (ID0 == 1)) {
                if ((PAT2 == 1) && (PAT1 == 0) && (PAT0 == 1)) {
                    break;
                }
            }
        }
    }

    return ad7171_convert_to_voltage(raw_data);
}
```

This high-level function reads a voltage from the ADC for a given sensor. It loops until a valid reading is received by checking specific flag bits in the raw data. Once the pattern and flags confirm the data is good, it converts the result to a voltage using the earlier conversion function. This ensures we only use clean, verified sensor readings.

```
// Read ADC data manually
uint32_t ad7171_read_data_manual(int sensor_num) {
    uint32_t result = 0;

    for (int i = 0; i < 24; i++) {
        gpio_set_level(ADC_SCLK, 1);
        delay_us(50);

        result <<= 1;
        if (gpio_get_level(ADC_DOUT_RDYS[sensor_num - 1])) {
            result |= 1;
        }

        gpio_set_level(ADC_SCLK, 0);
        delay_us(50);
    }


    return result;
}

// Convert ADC data to voltage
float ad7171_convert_to_voltage(uint32_t raw_data) {
    uint16_t adc_code = (uint16_t) (raw_data >> 8);

    return (((float)adc_code / 32768.0) - 1) * VREF;
}
```

This helper function reads 24 bits of data from the AD7171 one bit at a time using GPIO-based clock pulses. It checks the DOUT line of the specified sensor and builds a full 24-bit data word. This raw data includes both the sensor reading and additional bits used for validity checks.

```
// Convert ADC data to voltage
float ad7171_convert_to_voltage(uint32_t raw_data) {
    uint16_t adc_code = (uint16_t) (raw_data >> 8);

    return (((float)adc_code / 32768.0) - 1) * VREF;
}
```

This helper function takes the raw 24-bit ADC output, discards the lower 8 bits (used for flags/patterns), and converts the remaining 16-bit ADC code into a voltage. It uses the reference voltage VREF and a signed conversion formula because the AD7171 operates in bipolar mode.

Appendix 5 - Team Contract

Team Name sdmay25-04

Team Members:

1) Ashley Falcon
2) Drew Scheidler
3) Wesley Smith
4) Henry Hingst
5) Yok Quan Ong
6) Hector Perez Prieto

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. Face-to-face
    b. Team meeting: tentatively Mondays, 2:15 - 3:05 pm @ Parks Library
        i. Main expectation: team meeting 1x per week
    c. Advisor meeting: Wednesday, 2:15 - 3:05 pm @ Parks Library or SICTR

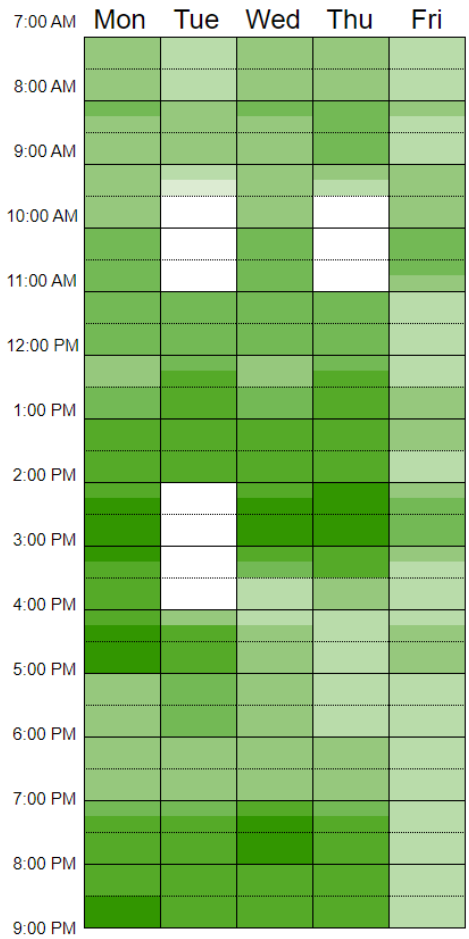


Figure 9.5.1: Time Map to Help Organize Team/Advisor Meeting

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
    a. Phone (Snapchat Group Chat)
    b. Discord Server
    c. Face-to-face at our weekly meetings.
3. Decision-making policy (e.g., consensus, majority vote):

    a. Majority vote (4/6 members agree)
    b. Ideally a compromise should be made prior to vote
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
    a. Project Report done weekly
    b. Notes made by Wesley kept in shared Google Drive folder

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    a. 100% attendance expected, notify of absence ASAP via group chat
    b. Respect for other group members' time
    c. Assigned tasks should be completed, and any roadblocks should be reported
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    a. Timelines, responsibilities, and deadlines should be clear before the end of team meetings
    b. Roadblocks should be reported promptly
3. Expected level of communication with other team members:
    a. Face-to-face communication once a week
    b. Track progress in the Google Drive folder for team reference
    c. Communicate issues ASAP
4. Expected level of commitment to team decisions and tasks:
    a. See “Decision-making policy.”
    b. Before anything is turned in, all members must approve (unless assignment due date reached)

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
    a. Client/Advisor interaction (Emailer): Ashley Falcon
    b. Professional Note Taker: Wesley Smith
    c. Other roles will be assigned to other group members when project advancements are made
2. Strategies for supporting and guiding the work of all team members:
    a. Breaking off into groups that are dedicated to specific tasks could be useful
    b. Regular team meetings to address issues and progress
    c. Encouraging questions
    d. Reaching out to Professor Neihart with daily questions
3. Strategies for recognizing the contributions of all team members:
    a. Documented tasks assigned
    b. Giving positive feedback to team members

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
    a. Drew: Software expertise and interfaces
    b. Henry: Networking. Experience with microcontrollers in previous internship.
    c. Wesley: Power distribution, basic microcontroller knowledge from HABET
    d. Hector: Basic microcontroller knowledge and Power distribution knowledge from previous internships.
    e. Ashley: Experience with microcontrollers and hardware testing in previous internships. Good communication skills. CAD experience.
    f. Yok: PCB design (KiCAD), basic microcontrollers knowledge
2. Strategies for encouraging and supporting contributions and ideas from all team members:
    a. Open discussion at group meetings
    b. Set time aside for questions and kudos for other team members
3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
    a. Set aside kudos time will ensure team members feel comfortable
    b. Open and prompt communication is preferred
    c. Ashley can be a mediator
    d. Make expectations clear to avoid a lack of team member contributions

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    a. Wireless chip set to chosen and drivers written
    b. Schematic of PCB done by the end of the semester
    c. Part number for ESP32 (microcontroller) picked out and blink test complete
2. Strategies for planning and assigning individual and team work:
    a. Allow team members to communicate what they most want to do
    b. Delegate tasks from advisor/client meetings to specific individual/team
3. Strategies for keeping on task:
    a. All members should be ready to contribute any research regarding the project found during weekly meetings
    b. Maximize efficiency during times we meet

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    a. Handle infractions as a team
    b. Preferably communicate infractions before they happen
2. What will your team do if the infractions continue?
    a. Discuss continued infractions to Senior Design TAs

b. Intervention with the team member

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Ashley Falcon	DATE:	9/17/2024
2) Drew Scheidler	DATE:	9/17/2024
3) Wesley Smith	DATE:	9/17/2024
4) Henry Hingst	DATE:	9/17/2024
5) Yok Quan Ong	DATE:	9/17/2024
6) Hector Perez Prieto	DATE:	9/17/2024