SONOMA STATE UNIVERSITY | ENGINEERING SCIENCE

Senior Design Project Progress Report
EE 492 Senior Design Project Planning

ORRHA

Open Range Ranch Hand Assistant
an IoT Smart Ranch System

By:

Jeff Camilletti
Juan Menendez
Dano Lopez

15 MAY 2024

Faculty Advisor: Dr. Farahmand , SSU Professor
Industry Advisor: Roberto Medina, SSU Consultant
Client: QuarterWave

Project Website: https://camilletti4.wixsite.com/orrha

Acknowledgments

We would like to thank Professor Farid Farahmand and Professor Mohamed Salem for guiding us in pursuit of this project. Thank you to our Industry Advisor Roberto Medina for advising us about the LoRa protocol and a special thank you to the QuarterWave corporation, our customer, for sponsoring this project. We would also like to thank Mr. Steve Price, Mr. Daniel Spangler, and Mr. Tyler Walters.

Abstract

In response to the challenges involving remotely monitoring ranch conditions, this engineering project aims to achieve a low power long range IoT system that communicates the status of gates and water tanks to the customer. The three subsystems of this project include the sensor node circuitry and mechanical design, the firmware implementation, and the graphical user interface. The expected outcomes of this project include a reliable sensor network that will update the ranch managers of their ranch conditions utilizing the LoRa protocol, The Things Network, and TagoIO. These outcomes were achieved and the result is a well rounded IoT system that contains a low power modular sensor node capable of using an ultrasonic sensor or hall effect sensor, implementation of LoRa protocol capable of two way transmission, and a user friendly graphical interface that automatically updates with new sensor data.

Table of Contents

Abstract **2**

List of Figures **4**

List of Tables **4**

Introduction and Background **5**
Problem Statement 5
Literature Review
Motivation and Impact 7

Project Requirements **8**
Marketing Requirements (MR) 8
Engineering Requirements (ER) 9

Design and Methodology **9**
Software Design and Dataflow 12
PCB Design and Implementation 19
Mechanical design 20

Challenges & Risks **21**

List of Tests **25**

Ethics of the Engineering Profession and Our Project **35**

References **36**

Appendices **37**

List of Figures

Fig. 1 Firmware flowchart for gate node
Fig. 2 Firmware flowchart for the water tank node
Fig. 3 Block Diagram of the System Architecture
Fig. 4 Hardware Block Diagram of the System Architecture
Fig. 5 Illustration of Data Storage in Buckets
Fig. 6 Inputting tank parameters
Fig. 7 Python Utilization for Calculating Remaining Tank Gallons
Fig. 8 Action within Tago
Fig. 9 Information the end user will see
Fig. 10 The gate status, battery level, signal strength, and data log
Fig. 11 Time intervals available for a downlink message
Fig. 12 PCB Schematic
Fig. 13 PCB Layout, Manufactured, and Assembled
Fig. 14 3D Printed Sensor Node Housing
Fig. 15 Aluminum Milled Sensor Node Housing
Fig. 16 Procedure for testing the hall effect sensor
Fig. 17 Procedure for testing the ultrasonic sensor
Fig. 18 Error graph Vs Distance (Ultrasonic Sensor)
Fig. 19 Equation for ADC resolution
Fig. 20 Google Earth Pro image of distance traveled with LoRa Module
Fig. 21 Graph showing non-line-of-sight
Fig. 22 Procedure for testing Battery Measurement Test
Fig. 23 Error vs Voltage for ADC measurements
Fig. 24 Resolution equation for ADC measurements
Fig. 25 Image showing the Nordic Power Profiler Kit II connected to the Gate Sensor Node
Fig. 26 Block diagram for the setup of the power efficiency test
Fig. 27 Nordic power profiler kit application showing an average current of 3.62 mA
Fig. 28 Equation for battery life in hours and days
Fig. 29 Image showing the Nordic Power Profiler Kit II connected to the Water Tank Sensor Node
Fig. 30 Block diagram for the setup of the power efficiency test
Fig. 31 Nordic power profiler kit application showing an average current of 4.13 mA
Fig. 32 Equation for battery life in hours and days

List of Tables

Table 1: Bill of Materials for PCB
Table 2: Budge/Parts list for the ORRHA project
Table 3: Gantt chart of the ORRHA project
Table 4: Summary of conducted tests.
Table 5: Table of thresholds when sensor status has changed
Table 6: RSSI vs Distance

1. Introduction and Background

Problem Statement

Ranches are used for the breeding, raising, and selling of livestock. These ranches are wide open spaces consisting of 100's of acres of land where the livestock live. Operating a ranch is hard work and there is always a lot that needs to be done on a daily basis. The people who assist ranch managers with operating and maintaining the ranch are called ranch hands. Ranch hands typically provide care for livestock, maintain trails, repair fencing and ensure the overall safety and security of the ranch is adequate. Since the ranches tend to be 100's of square acres performing the duties of a ranch hand involves traversing large areas in as little time as possible. The current standard for traversing a ranch are horses, ATV or other offroad vehicles. Even with these modes of transportation the hours of travel time these ranch hands must perform takes away time from other important tasks.

Ranch hands could save time and energy if they could remotely monitor some of the more mundane tasks. Currently there is residential home monitoring equipment that can be purchased at most technology retailers. The current monitoring equipment on the market requires a WiFi or cell tower connection. WiFi has a range of 150 feet indoors and 300 feet outdoors, this is not nearly enough to cover the wide open ranges that ranches occupy, so WiFi enabled residential monitoring nodes will not work on a ranch. The other option is to use cellular data enabled monitoring nodes. Cell towers have a range of 40 kilometers but they are located where the cell tower companies put them and may be out of range for the ranch manager's needs. The cellular data also requires a cellular data plan subscription and that could be expensive.

Ranch managers desire a way to inexpensively monitor ranch conditions without the need to travel to and physically monitor ranch conditions. They wish to have an easy to use monitoring system that will allow them to remotely check the conditions of the ranch without WiFi or cellular data enabled monitoring nodes. In response to this problem our project aims to provide the ranch managers with an easy to use monitoring system that utilizes the LoRa protocol which offers data transmission of several kilometers to a WiFi connected device. The collected node data will then upload the data to a web server that the ranch managers can view. This allows the ranch managers to remotely monitor ranch conditions from their home or even when they are away from their property with any internet connected device.

Literature Review Long Range Data Transmission in IoT

Long range, low power communication networks are becoming exponentially more common in the age of IoT. The increasing demand for such protocols is caused by the need for machine to machine communication [3]. Specifically the data transmission between "Things" and gateway devices. "Things" refers to connected devices in the Internet of Things (IoT). In our case we are focusing on sensor networks and how different Long Range, low power compare and contrast in terms of distance of transmission, power consumption, and reliability. This literature review will examine three main communication protocols LoRa, Sigfox, and LTE to ultimately determine the advantages and disadvantages for each technology.

On to LoRa in terms of its applications, its advantages and limitations in the world of IoT systems such as sensor networks. LoRa is an inviting option due to its long distance transmission at a low energy budget [2]. It is possible to transmit many kilometers in an outdoor environment while obeying FCC regulations [2]. This is extremely beneficial for one-hop sensor networks. One-hop meaning the transmission of packets being contained in a single transmission without the need of routing or switching. One-hop transmission can be cost effective because only one edge device would be needed for many sensors in a network. Another advantage of LoRa is its configurability [2]. Depending on how it's configured,meaning the carrier frequency, spread factor, bandwidth, and coding rate, LoRa can be tailored to a range of applications. Through the lens of a low power and low data transmission sensor network, Lora can be optimized with low bandwidth, high spread factor, high carrier frequency, and lower coding rate to make a reliable, long range, and low power system [2]. LoRa can even have channel separation if needed by using different spread factors which are orthogonal from each other [2]. LoRa is also characterized by its “resilience to noise”, which is important for message delivery [2]. A big disadvantage of LoRa is the trade off of low data rate. Since the protocol was designed for low power applications, it is impossible to transmit through LoRa at a high data rate. So it would not be applicable to image or video transmission.

Now on to SigFox communication protocol, again through the lens of a sensor network to identify this protocol's advantages and limitations. SigFox is most commonly used for low power IoT applications that involve hundreds if not thousands of nodes [1]. In this setting SigFox operates at ultra narrow band (UNB), operating in the range 100 bps to 600 bps [1]. This is certainly a limitation of the communication protocol, but UNB is effective in sensor network systems. An even larger disadvantage of SigFox appears when too many sensor nodes are communicating on the same channel, ultimately causing many collisions. This limitation can only be overcome by providing many gateway devices so that if a transmission to one gateway becomes corrupt the transmission to another gateway may still be in tack [1]. This will increase the cost of equipment and power consumption to have a reliable transmission.

Finally we will discuss LTE and its application to an IoT sensor network to determine its advantages and limitations in such an environment. Unlike the previous two protocols, LTE can be configured for higher data rate and lower latency making it the only protocol ,of the three, capable of video and image transmission[3]. This may not be useful in a sensor network application, but is important to mention nonetheless. LTE is a licensed technology unlike LoRa and SigFox [3]. Although there are low power versions of LTE, it is ultimately going to use more power then both LoRa and SigFox. Another drawback to LTE is the need of a sim card and a subscription to a cellular service provider. This would increase the cost of each sensor node. If the sensor network contains hundreds of nodes, it would not be feasible to provide each node with a sim card. So in this case the sensor network would need many gateway devices surrounding the nodes or the nodes would need to be close enough together to have local communication to a base station where it would be transmitted via LTE.

Some previous works include “Monnit”[6], “Ranch Bot”[5], and “Ranch Systems”[7]. Monnit has a vast amount of sensor options that use “ALTA" long range communications. Many of these sensors claim 10 years of battery life! The sensors range from temperature, ultrasonic, movement, humidity, door status, voltage meter, light intensity, water detection, and more. The sensors can communicate with Monnit's proprietary gateway device up to 1200 feet. This system

is the most similar to our project. Our project will stand out due to longer range data transmission and a more focused application that will be tailored to a ranch specifically.

Ranch Bot is purely a water surveying/managing system. Water remains the “lifeline” of agriculture. Sense water is such an integral part of ranch management, It is imperative that ranchers know the status of their water tanks often. Without the use of technology a rancher’s only way of monitoring their water tank is by physically checking it. Here is where the problem arises. Many water tanks are not easily accessible and checking them often becomes timely and frustrating. Ranch Bot supports ranchers by allowing them to remotely monitor their water tanks. The features of Ranch Bot include low water level alert, high water level alert, and rapid decreasing water level alert. Some cons about Ranch Bot is the upfront cost and annual fees are expensive. Our project will do long range tank monitoring and more.

Ranch Systems was a company that specialized in providing technology solutions for precision agriculture and farm management. They offered a range of products and services designed to help farmers monitor and manage their operations. Ranch Systems allows the user to access key information about their crop such as soil moisture, weather conditions and crop health. This information is then provided to the user through an online application allowing them to make data driven decisions vs instincts. Ranch systems uses two forms of wireless communication to get this information to its users, long range bluetooth and Wifi. This is where the biggest con of the ranch system can be found. Each sensor must have their RS10 long range bluetooth that then send information to their host telemetry unit RM400 and RS330 that is connected to the the web via wifi or cellular. These telemetry units can only 4 soil probes per unit with an approximant distance of 1000 ft with clear line of sight. Although with trees or foliage between the RS10 and host the typical range is 350ft. This then goes to a second con that is cost, the user with a large property has to get a second host for every 350ft and limited with 4 sensors. The cost grows quickly for a 500 acre farm with multiple host units required along with the cellular service that can get expensive fast. Our project will be able to handle 100s of different sensors to a single gateway with a range of at least 2 km.

Motivation and Impact

Keeping livestock in their designated areas is a major responsibility of the ranch hand. They maintain very large areas of fenced in space. The expansive fencing has various gates throughout its configuration and the ranch hand needs to ensure the gates are closed when they are supposed to be. Keeping gates closed is important to ensure the livestock stay in their designated areas. Ensuring the gates are closed would usually require the ranch hand travel to the gates and ensure they are closed which involves a lot of travel time and man hours. To relieve the ranch hand of this arduous duty we will create a gate monitoring system that will tell the ranch hand when the gate is open or closed without the need for the ranch hand to be present at the gate. We will utilize low power LoRa enabled sensor nodes which will transmit the gate data up to several kilometers to a LoRagateway located in the rancher's home or place of business. This data will then be uploaded to TagoIO where the ranch hand can then check the status of the gate from their own internet connected device.

Maintaining the health and wellness of the livestock is another important duty of the ranch hand. They do this by providing enough clean water and food for the livestock. The ranch

hand will fill large tanks with potable water that is used to hydrate the livestock. Without drinkable water the livestock will perish so this task is performed often. A lot of time is used to ensure the water tanks remain plenished. We aim to alleviate the hands-on monitoring of the water tank levels from the ranch hands, saving them time and money. We will monitor the water level of the tanks using a LoRa enabled sensor node much like the nodes used for the gates. This node will provide the ranch hand with how much water is left in the tank and how quickly the level is draining. This sensor can also be used as an early warning system of any tank leakage.

We are calling our smart ranch system, Open Range Ranch Hand Assistant (ORRHA). ORRHA will provide the LoRa based sensor nodes to monitor gates and water tank levels. ORRHA will also provide a graphical user interface that the ranch managers can then utilize to have an easy to use monitoring interface.

2. Requirements

A well designed project needs requirements to structure design and business choices. The two types of requirements that define this project are “marketing requirements” and “engineering requirements”. Marketing requirements are formed through customer discovery. Engineering requirements are design specifications that fulfill the already existing marketing requirements. By fulfilling the engineering requirements, the marketing requirements are also met and the project is successful. After our customer discovery we decided on eleven marketing requirements, which resulted in nine engineering requirements. From our discovery we found that customers would benefit from a sensor network that is connected to the internet for easy data access. Accordingly MR1 acknowledges this by stating that the system must include sensor nodes, a gateway, a database, and a graphical user interface. We also discovered that our customers had large areas of open land, which means that sensor nodes must be able to transmit data long distances and must be able to withstand the weather. For efficiency the nodes must be low power consuming, so that trips for battery recharge can be minimized. After searching the market we have discovered two products that are similarly related to ours, Ranch Systems and Rancheyes. The first system, Ranch Systems, is both power hungry low range. Rancheyes is also power hungry and unable to detect water tank levels. Our marketing and engineering requirements acknowledge and solve the limitations of existing solutions.

Marketing Requirements:

1. The system must include two kinds of sensor nodes, a gateway, a database, and a graphical user interface.
2. The sensor nodes must be capable of long distance data transmission
3. The sensor nodes and gateway must be weatherproof
4. The sensor nodes must be able to operate on battery power at least 1 month without recharge
5. The system must have a gate status detection sensor node and water tank level sensor node
6. The gate status detection sensor must accurately determine the status of a gate
7. The water tank level sensor must be able to measure water level to within 35 gallons in regards to a 4000 gallon tank
8. The sensor nodes must be compact

9. The gateway must transmit data to the database and store data reliably
10. The gateway must be capable of a large sensor network
11. The graphical user interface (GUI) must automatically update the user with new data and notify user of any hazards

Engineering Requirements:

1. The sensor nodes must transmit data at least 2 Km at 35% line of sight or better (MR2)
2. The sensor nodes must be IP45 rated (MR3)
3. The sensor nodes must be able to operate with an average current draw of less than 1.4 mA when operating on a battery with a 1000 mAh capacity (MR4)
4. The sensor node PCB and circuitry must fit within a housing with a volume of 16 cubic inches (MR8)
5. The water tank sensor node must detect the water level to ± 3.5 cm (MR1, MR5, MR7)
6. The gate sensor node must show a status of closed while within a 1 inch inflection point of the gate post center (MR1, MR5, MR6)
7. The data collected by the sensor node must be stored in an online database with 99% reliability (MR9, MR11)
8. The system must be able to send an alert to the user of a leak detection if the water level decreases more than 100 gallons an hour (MR11)
9. The gateway must be capable of syncing 10 nodes per channel (MR10)

3. Design & Methodology

In order to alleviate the ranch managers of the problems outlined above we plan to utilize battery powered LoRa enabled nodes with attached sensors to gather the appropriate data that the ranch managers require. To alleviate the difficulties of using WiFi over long distances or having to pay massive fees for cell service we plan to send sensor data using the LoRa protocol. LoRa enabled systems can transmit data over several kilometers because it transmits data at a lower frequency then WiFi. While WiFi transmits data at 2.4 or 5 GHz, LoRa transmits data at 915 MHz which enables data to be transmitted much farther and with less susceptibility to noise as the lower frequency passes through objects much easier. The downside to the lower frequency and long range is that the data rate is much lower, typically 0.3 to 50 kbps. This is not enough to send pictures or videos in a timely manner but is enough to send sensor data.

We are offering two different types of sensor nodes. We will offer a gate detection node that will detect whether a gate is opened or closed and we will offer a water level detection node that will detect the water level in a tank. The gate detection node will utilize a digital hall effect sensor. The digital hall effect sensor can detect the presence of a magnetic field. The hall effect sensor will be placed on the gate post and a magnet will be placed on the gate. When the gate is open a magnetic field will no longer be present and this data will be uploaded to a RAK3172 LoRa enabled STM32 microcontroller. The water level sensor will upload data to the microcontroller in a similar way except the way the water level sensor detects the water level is

different. We will utilize an ultrasonic sensor which can detect how far away the water is from the sensor. This can be accomplished because the ultrasonic sensor sends out sound waves, typically above 20 kHz, and times how long it takes for the sound waves to bounce back. This will allow us to measure how full the water tank is.

The sensor node is a printed circuit board (PCB) that has been designed by us to be compatible with both the ultrasonic sensor and hall effect sensor. The PCB has two microcontrollers, a lithium ion battery charger, a voltage regulator, connector for a battery, connector for a solar panel, and two more connectors that give us access to all available GPIO ports for both microcontrollers for future development. The ultrasonic sensor we chose is the MB1643 because of its narrow beam path, accuracy, low current consumption, and humidity rating. The hall effect sensor we chose is the 55140-3H-02-A because of its high sensitivity and low operating current. The main microcontroller is the Seeed Xiao Samd21[9] which takes the sensor measurements and communicates the data to the second microcontroller, the RAK3172 over UART. The Seeed Xiao Samd21 was chosen for its small footprint, familiarity, and low operating current and voltage. The RAK3172 is used as a LoRa "module" by taking advantage of the factory firmware. The RUI3 firmware allows us to use AT commands to control the RAK3172 [8], which means we can easily send and receive data over the air as well as from one microcontroller to the other.

Below you will find our firmware flowcharts for both the gate detection node, Figure 1, and the water tank monitoring node in Figure 2.

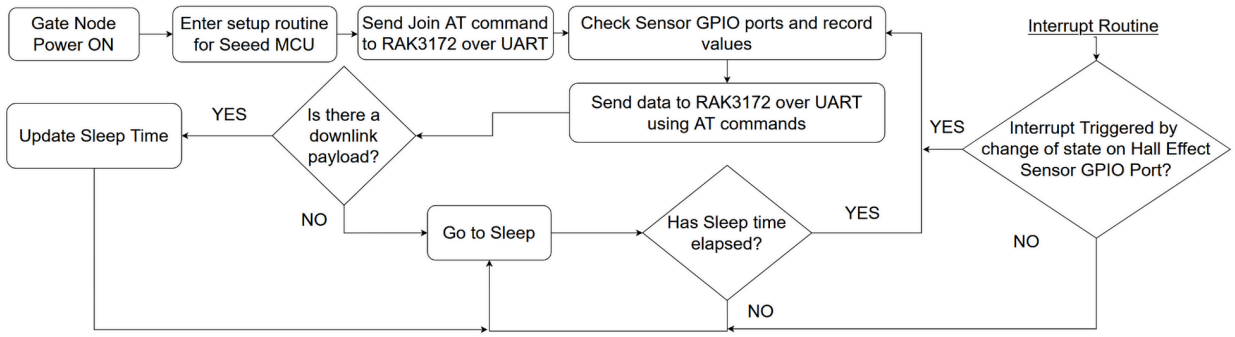


Figure 1. Firmware flowchart for gate node. Full code can be viewed in the appendix.

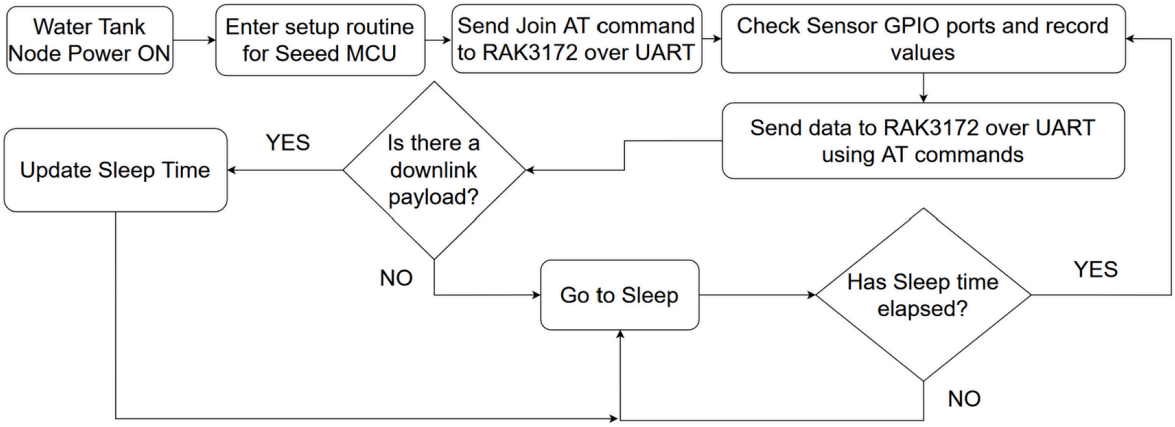


Figure 2. Firmware flowchart for the water tank node. Full code can be viewed in the appendix.

The gateways will be connected to the user's network and powered by a standard wall outlet. The gateway is constantly listening for incoming LoRa messages at the 915 MHz frequency. When the gateway receives a message it will upload the data to The Thing Network which is a LoRa based network server that stores the sensor node data. The Things Network will be connected to TagoIO which is an Internet of Things (IoT) cloud platform that will visualize the sensor data in a graphical user interface. TagoIO will make it easy and comfortable for the ranch manager to monitor the conditions of their gates and water tanks. Below you will find a high level system block diagram in Figure 3 and a more detailed block diagram, Figure 4, which outlines the exact components used for the project.

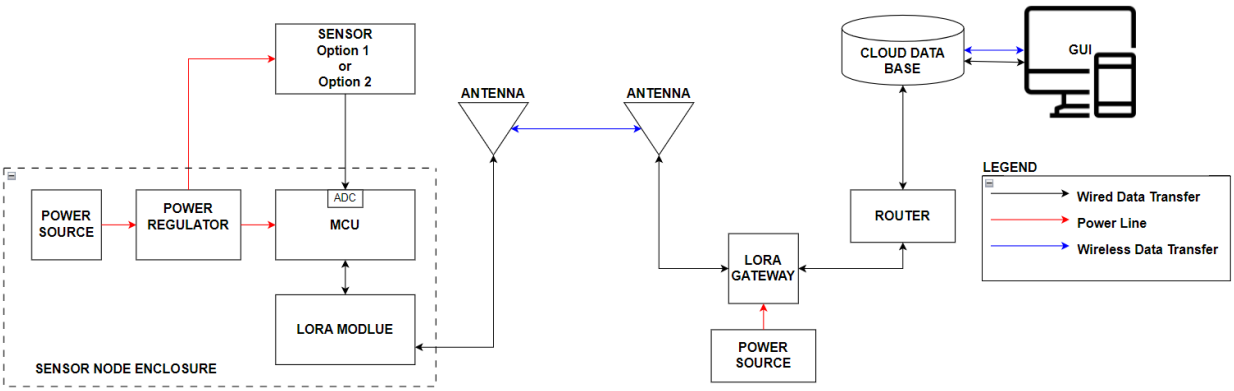


Figure 3. Block Diagram of the System Architecture

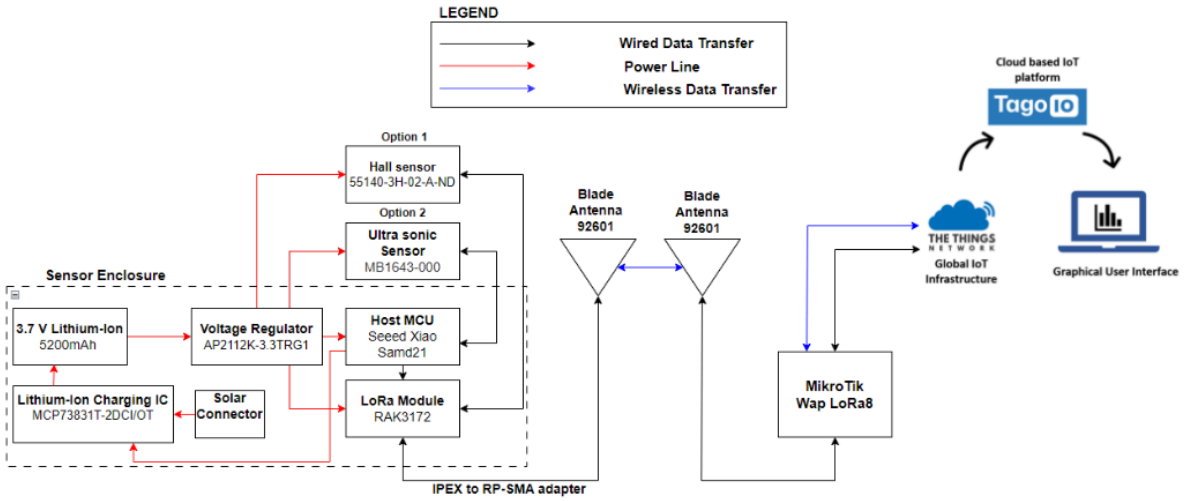


Figure 4. Hardware Block Diagram of the System Architecture

Software Design and Data Flow:

For our project, we utilized Tago.io for data management and a user interface. Tago.io is an Internet of Things (IoT) platform that provides comprehensive tools for collecting, analyzing, and visualizing data from connected devices. This cloud-based platform enables users to create and manage IoT applications, offering features such as real-time data visualization, analytics, and notifications. With support for various devices and protocols, Tago.io allows users to automate actions and gain insights through customizable dashboards and applications, making it suitable for industries like smart agriculture and industrial automation.

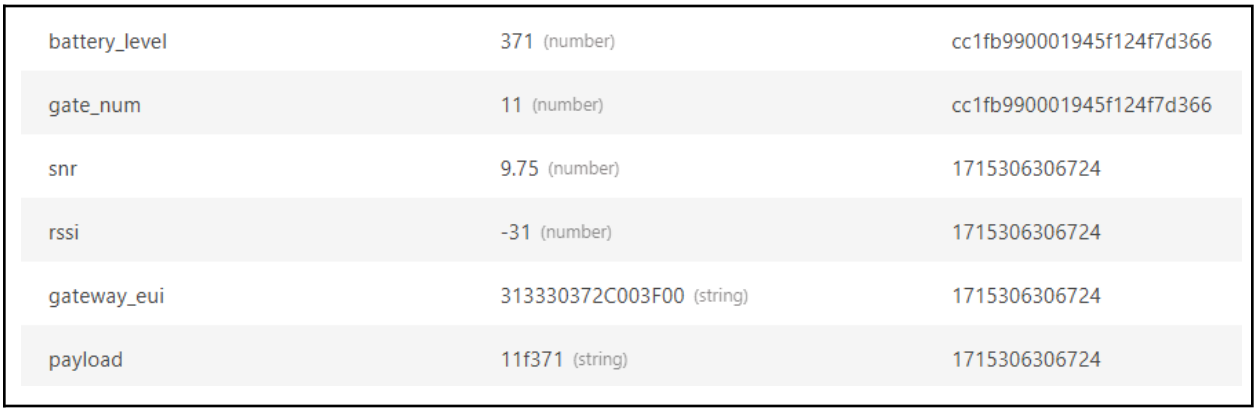
For our LoRaWAN support we used The Things Network (TTN) is a global, open-source LoRaWAN (Long Range Wide Area Network) that enables devices to connect wirelessly to the internet. It provides the necessary infrastructure and tools for deploying and managing IoT applications over long distances with low power consumption. TTN supports a collaborative and decentralized model, allowing widespread participation and benefits from a global IoT ecosystem. It is ideal for applications requiring wide coverage and long battery life, such as smart agriculture, environmental monitoring, and asset tracking.

In our project, TTN and Tago.io work together to provide data from the end node to a user-friendly GUI. IoT devices equipped with LoRaWAN technology send data to nearby TTN gateways, which forward the data to TTN servers for secure and efficient transmission. Once processed by TTN, the data is integrated with Tago.io through built-in connectors or APIs. Tago.io stores and provides tools for visualizing and analyzing the data, allowing users to create custom dashboards, set up real-time alerts, and apply advanced analytics for actionable insights. This integration ensures efficient monitoring and management of IoT solutions across various applications.

Initially, our project involved sensor nodes, including water level and gate sensors, which sent data to our gateway. Communication between a gateway and an end node occurs over LoRa using long-range, low-power wireless technology. The end node transmits data packets to the

gateway over a specific frequency (915 MHz in our case), and the gateway forwards these packets to a network server for further processing. We have two different nodes transmitting distinct packets. For instance, our gate node sends packets such as [11 F2 50], where "11" indicates the gate is open, and "10" indicates it is closed. The "F" serves as a filler, allowing us to easily utilize Tago's payload parser.

Our water tank node sends packets like [110 F25 0], where the first 3 bits represent the distance from the water to the top of the tank in centimeters. This is followed by the battery level, indicated by the next 3 bits. Both segments are separated by an 'F', similar to our gate node. This data is transmitted to The Things Network, where it is temporarily stored before being imported into Tago using the built-in API. Tago's payload parser employs JavaScript for processing.



battery_level	371 (number)	cc1fb990001945f124f7d366
gate_num	11 (number)	cc1fb990001945f124f7d366
snr	9.75 (number)	1715306306724
rssi	-31 (number)	1715306306724
gateway_eui	313330372C003F00 (string)	1715306306724
payload	11f371 (string)	1715306306724

Figure 5. Illustration of Data Storage in Buckets Before and After Parsing this data is stored in .JSON formatting

Once the data is received in Tago, it is stored in a device-specific database called a bucket, as shown in Figure 5. These buckets serve as our data storage and allow us to manipulate the data using Python through Tago's Analysis tool. Data can be saved in these buckets for 1 month or up to 1 million data points whichever comes first. In Figure 3, we can observe the data storage process. The variable payload represents the data sent from the end node, while gate_num and battery_level are the values stored after parsing the data. These variables then allow us to create analysis and actions.

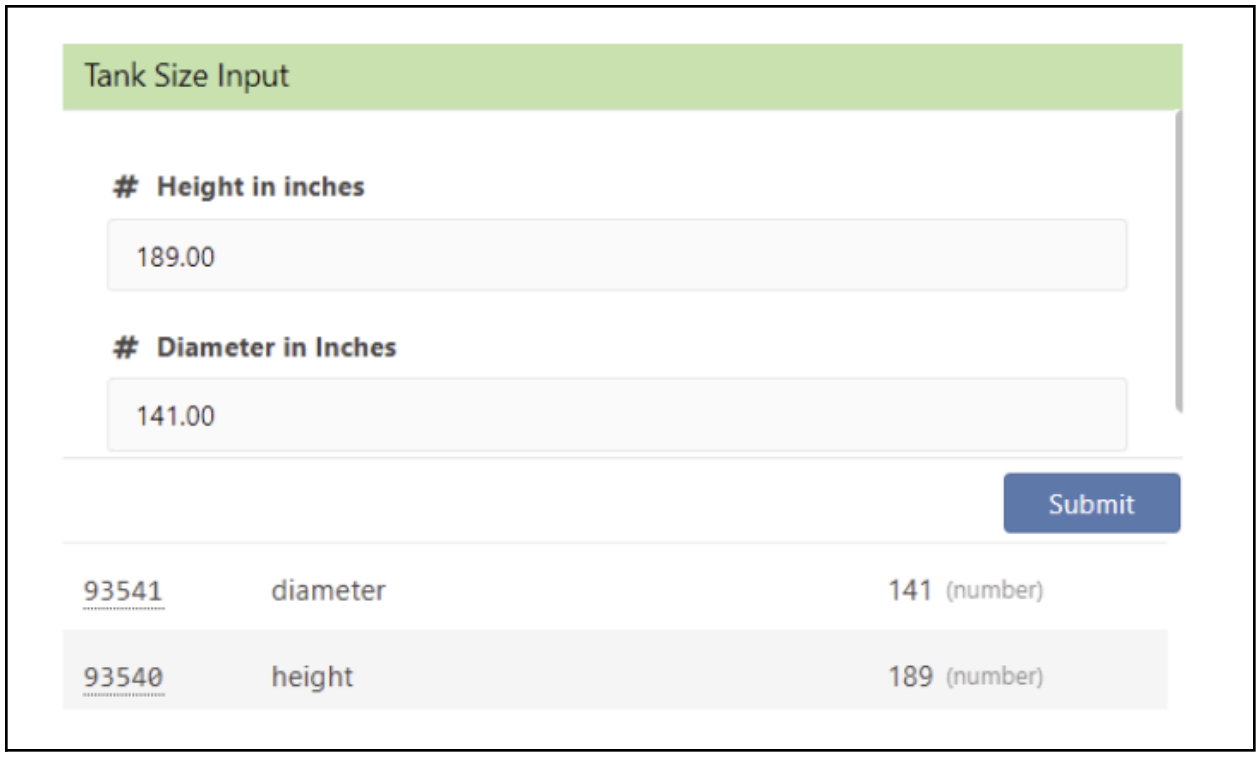


Figure 6. The figure above shows the user interface for inputting tank parameters as well as how it is saved in the bucket

Tago also had the capability to create its own variables based on user inputs as shown in Figure 6. This feature is used for setting the tank parameters and downlink that will be covered in a later section. These variables allow for the user to quickly set up their water level nodes and adjust on the fly.

```
# Calculate radius in inches
radius_in = payload_diameter_in / 2

# Convert water level from centimeters to inches
water_level_in = payload_water_level_cm / 2.54

# Calculate volume of cylindrical segment in cubic inches
volume_in3 = int(3.141592653589793 * radius_in ** 2 * (payload_height_in - water_level_in))

# Calculate total volume capacity of the tank in cubic inches
total_volume_in3 = int(3.141592653589793 * radius_in ** 2 * payload_height_in)

# Convert total volume from cubic inches to gallons (1 gallon ≈ 231 cubic inches)
total_volume_gallons = int(total_volume_in3 / 231)
adjusted_total_volume_gallons = int(total_volume_gallons * 0.93)  # Subtract 7%

water_volume_gallons = int(volume_in3 / 231)

# Calculate the percentage full as an integer
```

Figure 7. Illustration of Python Utilization for Calculating Remaining Tank Gallons

Using the Tagos Analysis tool, we can leverage Python to create scripts that perform various operations with internal data and external services in real-time. Figure 7 demonstrates an example of how we employed this analysis tool. To complete these analyses, we extracted both parsed variables and user-generated variables that were created in Figure 6. This script executes when an Action is triggered, as shown in Figure 8. Similar steps are configured for our gate node, though a different script is used. Once this Python script runs, we can create the widgets for the final GUI.

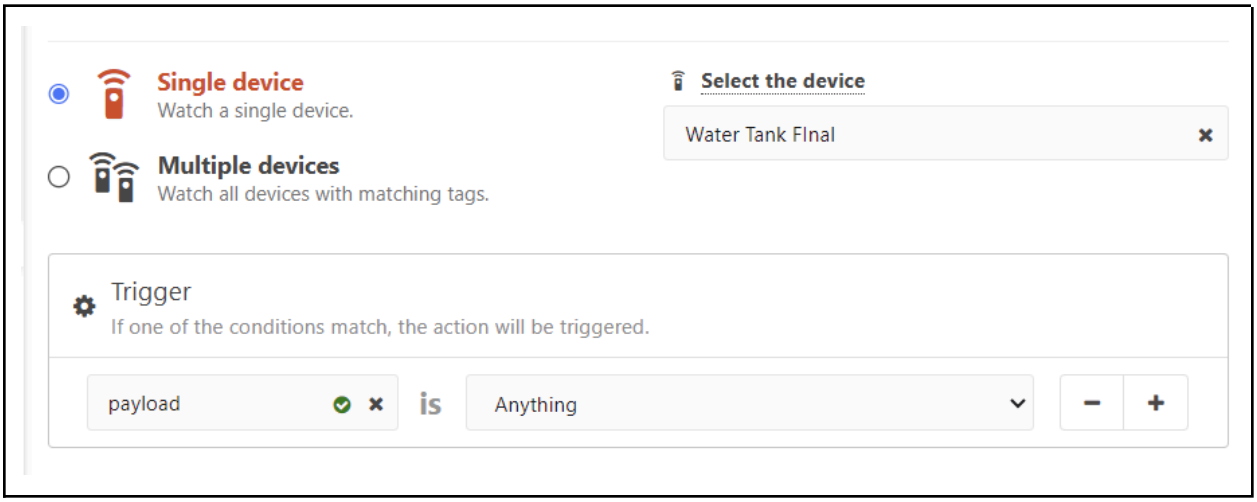


Figure 8: An example of an Action within Tago, this action runs the python script shown in Figure 7 anytime the payload variable is updated.

Tago's widgets are interactive elements that enhance the user interface by displaying real-time data and facilitating user interaction. These widgets can be configured to visualize data in various forms, such as charts, graphs, gauges, maps, and more. For the water level node, we used a tank gauge to show the percentage of water remaining, along with a chart to display water levels over time and a record of the last 50 entries. These can be seen in Figure 9

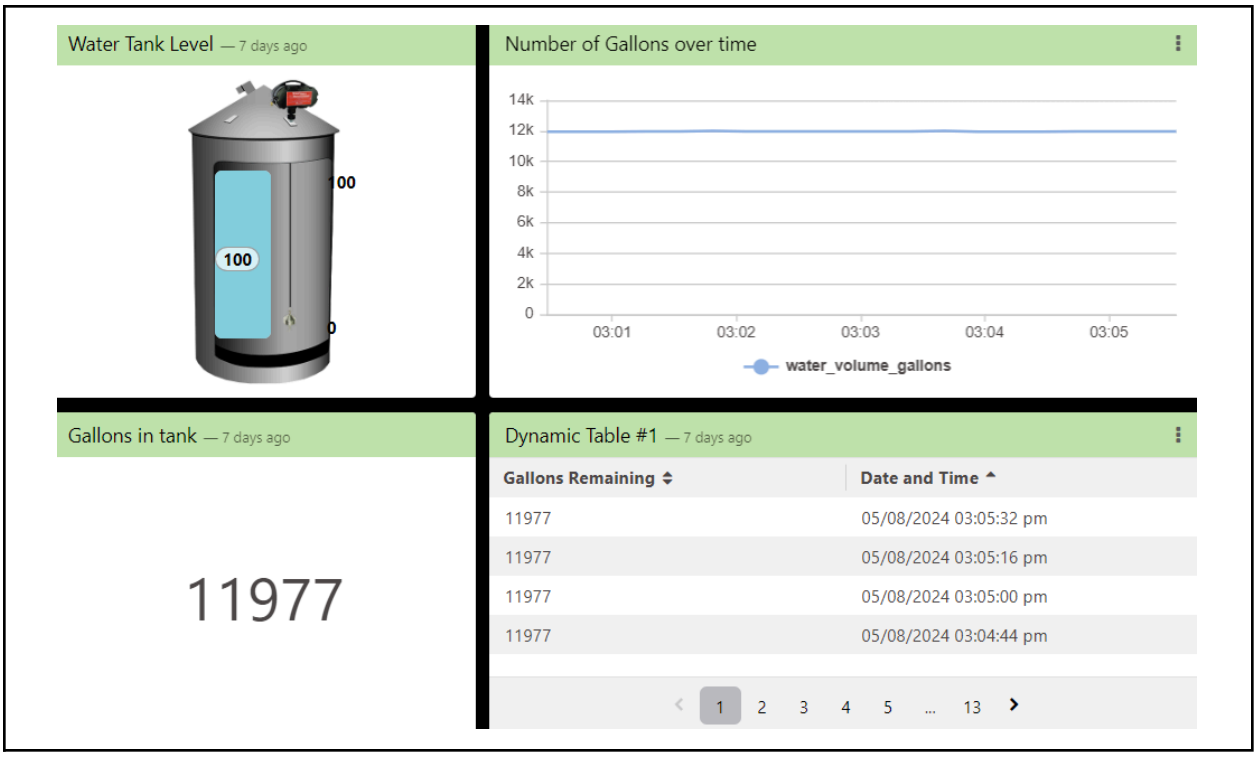


Figure 9. Displayed above is all the information the end user will see from their water tank node

The same procedure is applied to the gate sensor node, where we utilized the text widget alongside a log displaying the last 500 records. This setup is illustrated in Figure 9, which also shows our battery level indicator and signal strength. The battery level indicator updates in colors ranging from green to red based on the voltage and the

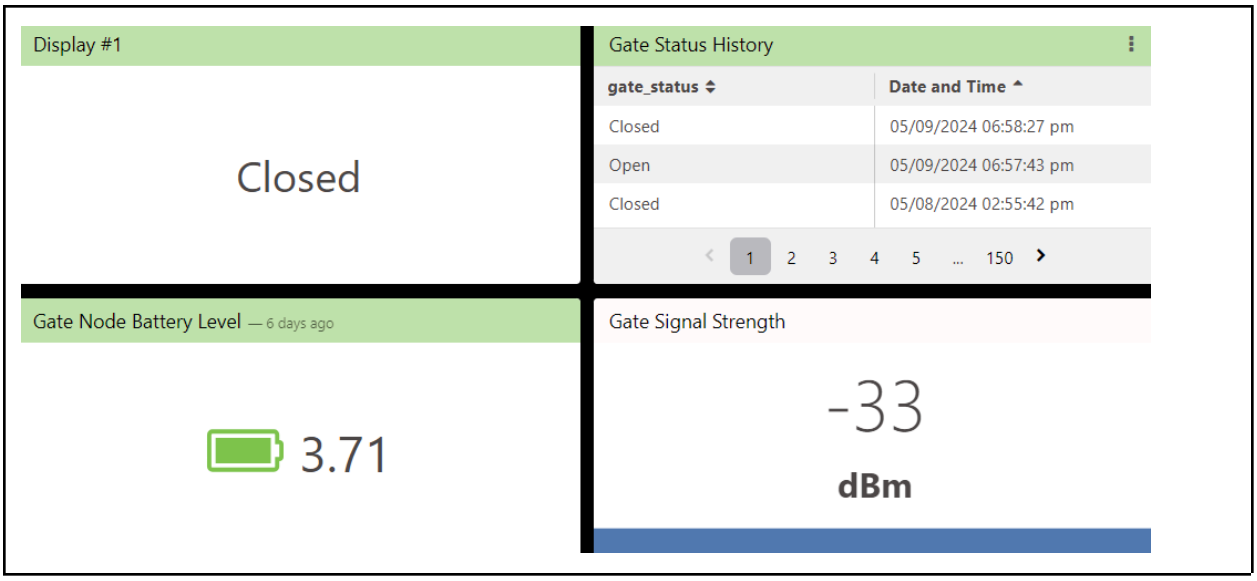


Figure 10. Illustrates the gate status, battery level, signal strength, and data log.

Lastly, we have the downlink. The downlink process from Tago to The Things Network (TTN) to the end node is a seamless communication flow designed to transmit commands and data efficiently. When an action is triggered in Tago, a downlink message is generated and sent to TTN. In our case the message is how often the user would like their nodes to update and transmit data. TTN then handles the message routing and ensures its delivery to the appropriate end node. Once the end node receives the downlink message, it executes the specified command or updates its configuration accordingly. We provided the user with update times every 15 mins ranging from 15mins to 2 hours shown in Figure 11.

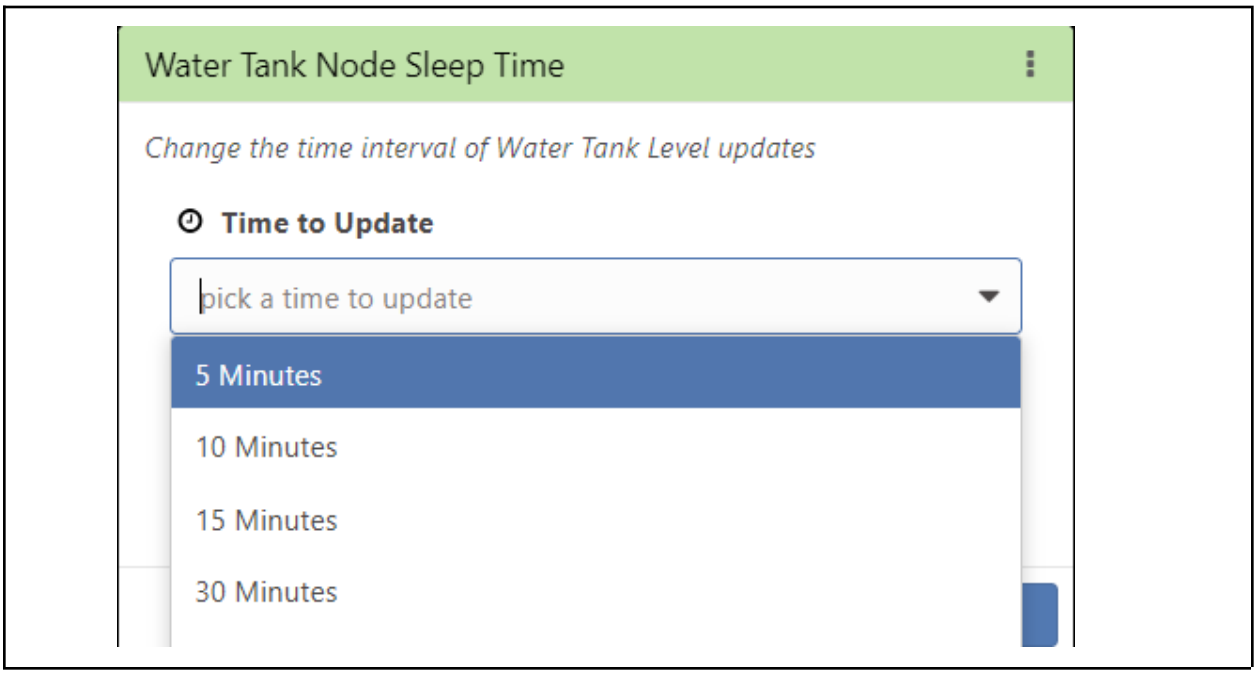


Figure 11. Illustration of the time intervals available for a downlink message from the Tago dashboard to the end node.

PCB Design & Implementation:

The printed circuit board (PCB) was designed for low voltage and current consumption, to be small in size, and easily interfaced with many sensors. To accomplish low power consumption we chose components that operate at 3.3 V or less and have low operating currents. The circuit is powered by a 3.7 V lithium-ion battery connected to J2. The battery voltage is regulated by an LDO for minimum power loss. To accomplish a small sized PCB we chose small components such as 0805 sized resistors and capacitors, and low profile connectors to minimize board space. We accomplished modularity by routing all available I/O ports to connectors, so that we have the opportunity to use multiple kinds of sensors. The schematic can be seen on Figure 12 and the final PCB can be seen on Figure 13.

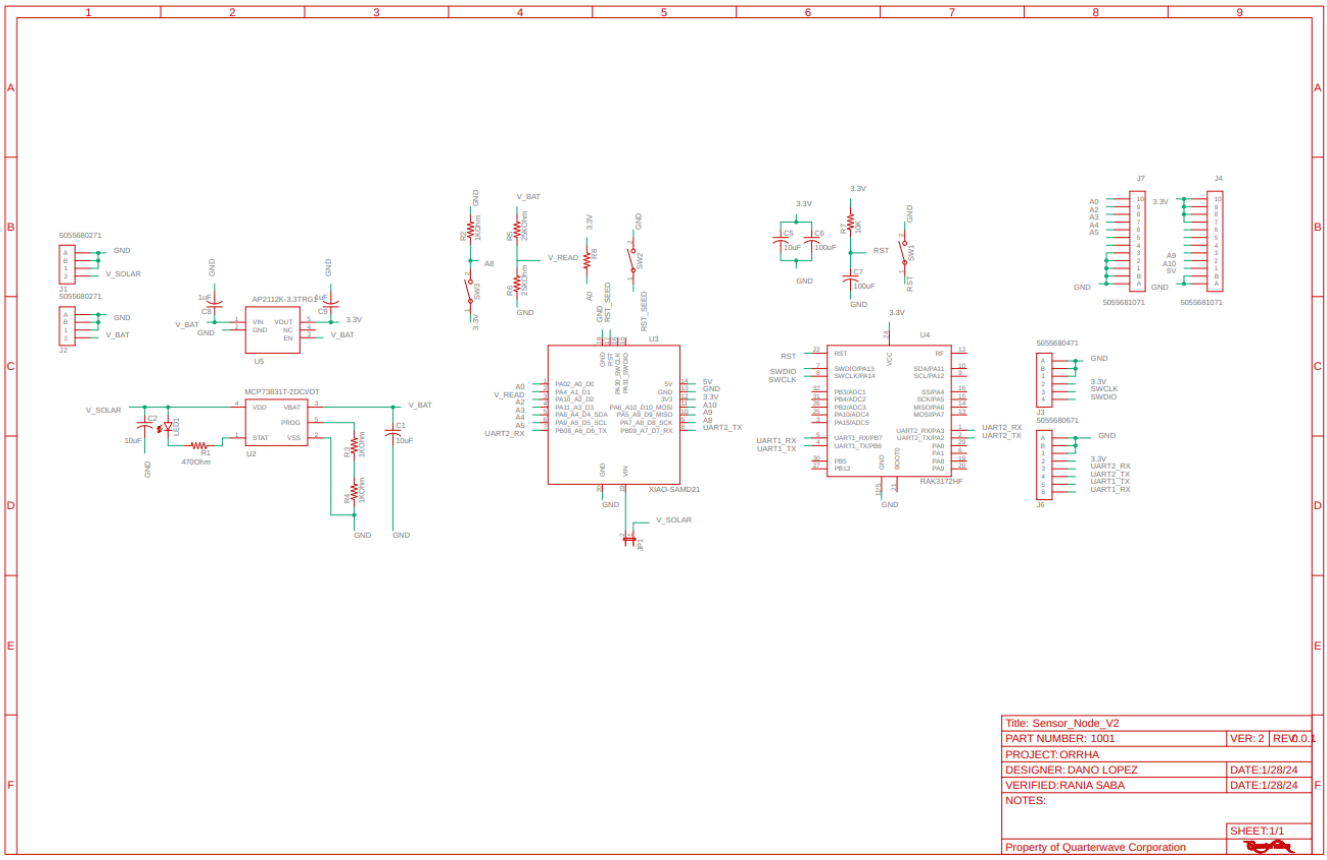


Figure 12. PCB Schematic created on Autodesk Eagle

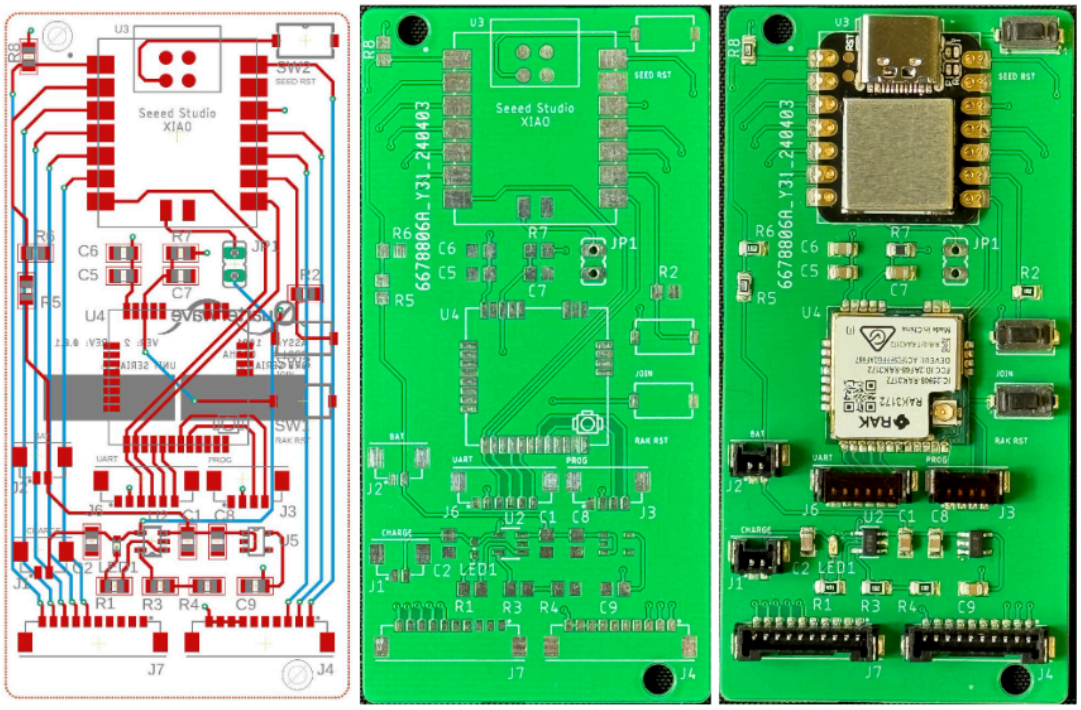


Figure 13. PCB Layout visualized using Autodesk Eagle, A black PCB after being manufactured by PCB way, and lastly as fully assembled PCB with all surface mount components soldered on

Mechanical Design

Our project is designed for ranchers which means the typical environment will be outdoors in the elements. We designed an aesthetic yet functional housing that fits the PCB, a 5200 mAh lithium-ion battery and holes for sensor connections, a solar connection, and a RF connection. The 3D printed housing can be seen in figure 14 and an aluminum milled housing can be seen in figure 15.

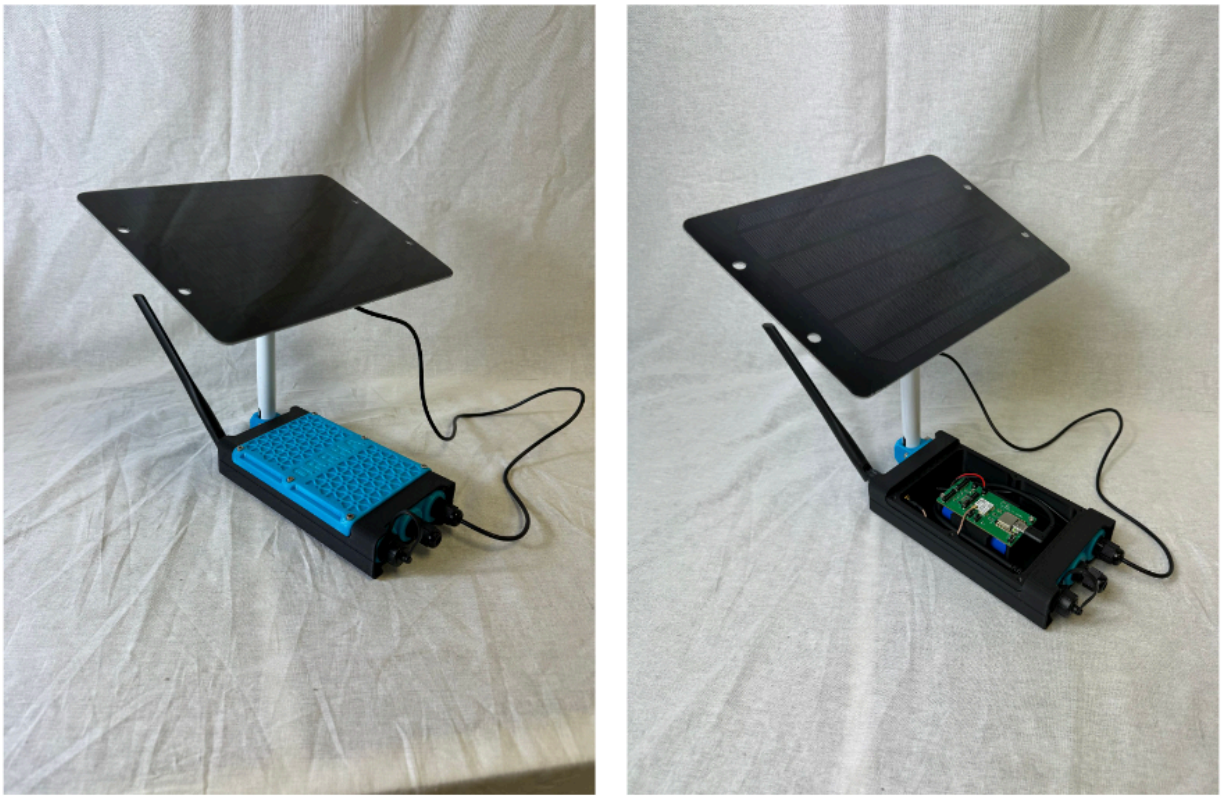


Figure 14. 3D Printed Sensor Node Housing fully assembled with a Solar panel attached.

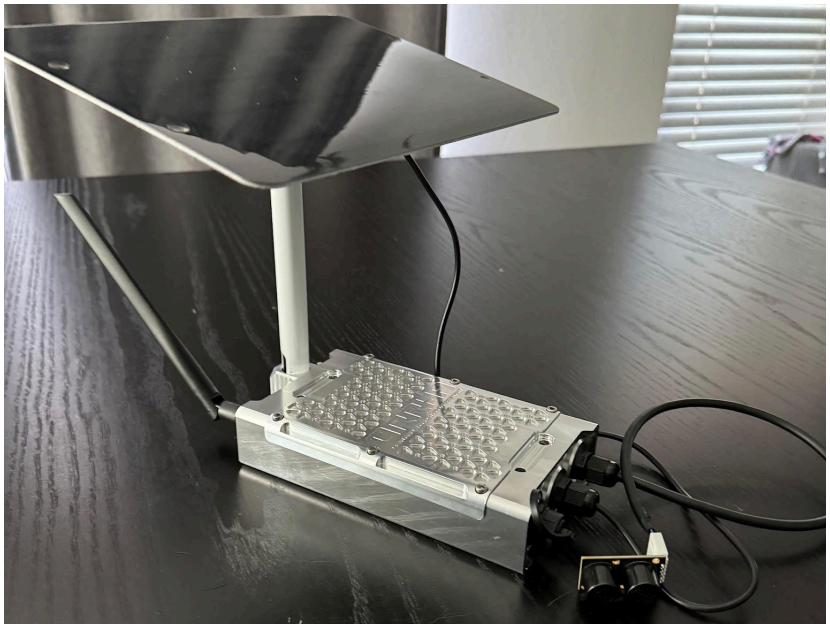


Figure 15. Aluminum Milled Sensor Node Housing

4. Challenges and Risks

A major challenge for our project is to achieve the desired transmission distance for our sensor nodes. LoRa communication can transmit up to 3 km and possibly 16 km if there is no interference (line of sight transmission). Our customers may not need the sensor nodes to transmit at these distances but we would like to accurately advertise our actual transmission distances. The challenge will result in finding our transmission distance threshold as we would require very large open spaces which may be difficult to find in our current location. The challenge comes in finding an appropriate place to test our transmission distances.

Another challenge we will encounter will be to determine the approximate time the batteries will last for each sensor node. An easy way to test this would be to let the system run and wait until the batteries run out. Our sensor node batteries may last several weeks or months so this may not be a viable option. To remedy this situation we plan to include a battery level monitor into our sensor nodes. We can then operate our system over several days and then use the battery level monitor to see how much the batteries have depleted. We can use this information to extrapolate and determine an approximate battery life expectancy.

Our sensor nodes will be located outside and because of this our sensor nodes must be weather resistant. We plan to meet the IP45 standard of weather resistance. In order to achieve this we plan to 3D print our own housing for the sensor nodes. They will have all the appropriate gaskets to ensure the sensor nodes remain dry in wet weather conditions. The challenge arises in testing the level of weather resistance. We could place a moisture sensor inside the housing and conduct a series of tests to see which test activates the moisture sensor. The challenge will arise if the tests fail and we must find where the weather resistance is failing. Correcting the failure through redesigns could prove to be troublesome as testing for weather resistance will take physical prototypes.

We face multiple risks that pose some threat to our project's completion and to the project's lifetime. The list of our risks are animals dislodging our gate sensor nodes, lightning strikes the outdoor gateway antenna, and the internet going out due to a client's power going out. Our first risk is possible and minor giving it a risk score of 6. Our mitigation plan is to design a secure fasten system to make knocking the node off the gate less likely. Our contingency plan is to separate the sensor itself from the rest of the node using power and data lines to connect them so that there is less surface area for the animal to make contact with. Another risk we face is lightning striking the gateway antenna which is a rare occurrence, but catastrophic to the system giving this risk a score of 5. The mitigation plan would be to attach a lightning protection kit to the antenna to hopefully protect the gateway from a surge of current and voltage. Our contingency plan is to choose an indoor gateway and near a window for best transmission of data. Another risk we face is the internet going out due to a power outage which we rated as a10 with a likelihood of almost certain and a consequence of minor. Our mitigation plan is to have a battery bank to keep the gateway alive for as long as possible to keep data transmission alive locally. Our contingency plan is to optimize our system to get back up and running as soon as possible as soon as the power comes back on.

Table 1. Bill of Materials for PCB (Total parts to assemble a single PCB)

Qty	Value	Package	Parts	Component
1	Option	R0805	R8	Resistor
1	10 KOhm	R0805	R7	Resistor
1	470 Ohm	R0805	R1	Resistor
1	5055680471	5055680471	J3	Connector
1	5055680671	5055680671	J6	Connector
1	AP2112K-3.3TRG1	SOT-25	U5	Voltage Regulator
1	MCP73831T-2DCI/OT	SOT-23-5	U2	Charger
1	RAK3172HF	32-SMD Module	U4	IC
1	XIAO-SAMD21		U3	IC
2	100uF	C0805	C6, C7	Capacitor
2	1uF	C0805	C8, C9	Capacitor
2	25 KOhm	R0805	R5. R6	Resistor
2	5055680271	5055680271	J1, J2	Connector
2	5055681071	5055681071	J4, J7	Connector
3	10uF	C0805	C1, C2, C5	Capacitor
3	1KOhm	R0805	R2, R3, R4	Resistor
3	PTS636 SM50J SMTR LFS	PTS636 SM50J SMTR LFS	SW1, SW2, SW3	Switch

Table 2. Budge/Parts list for the ORRHA project (Total cost of the entire project)

Part	Quantity	Price	Description
RAK3172	8	$6.49 x 8	RAK3172 is a low-power long-range transceiver module for LoRa and LoRaWAN applications based on STM32WLE5CC chip
Mikrotik KNOT LR9	1	$185 x 1	The gateway supports 8 LoRa channels, multi backhaul with Ethernet, Wi-Fi, and Cellular connectivity.
Seed Xiao Samd21	10	$5.71 x 10	ARM Cortex-M0+ CPU running at up to 48MHz
MB1643-000	2	$34.95 x 2	The HRLV-ShortRange-EZ sensor line is the most cost-effective solution for close range applications where precision range-finding, low-voltage operation,
55140-3H-02-A	2	$10.79 x 4	Digital Hall Sensor
3.7V 5200mah	2	$16.00 x 2	3.7 V Lithium-Ion Battery Pack
PCB	10	$15	Custom PCBs from PCBway
Node housing	2	$30	Custom printed housings for sensor node
Surface Mount Components	280	$70	Resistors: 10 KOhm, 470 Ohm, 1 KOhm Capacitors: 10 uF, 1 uF, 100uF Switches, Connectors, Bat charger, Voltage regulator
Total Cost		**$554.08**	

Budget/Parts List

We are fortunate to have a project that is funded by Quarterwave because we will not experience small budget limitations. Although we want to keep the budget in a reasonable range because we want our end product to be reasonably priced afterall. What we found is a good balance between price and performance using our alternative design matrices. Our budget/parts list includes the expenses of a LoRaWAN system of sensor nodes. This parts list includes the components needed for eight sensor nodes, four water level and four gate detection, an outdoor gateway, and the housings for the sensor nodes. It also includes an estimated cost of the PCBs we will order from PCBway, which is the company Quarterwave orders their PCBs from. The total

cost of our budgeted parts comes out to $701.40. Table 1 outlines the parts used on the PCB and Table 2 outlines all items bought for this project.

Project Schedule

We outlined our project schedule utilizing a Gantt chart last semester. We described four major milestones we wished to accomplish: PCB design, gateway design, mechanical design and testing. Dano was in charge of designing a compact PCB that will fit within a small housing that will be 3D printed by Juan. Jeff was in charge of programming and design of the graphical user interface. We planned to have the project completed by the end of April 2024.

We did not follow the Gantt chart as planned. After running into unforeseen challenges we as a group had come up with a new plan to execute. Ultimately Dano was in charge of PCB development and testing. Jeff was in charge of firmware development and testing. Juan was in charge of creating the database and graphical user interface as well as the mechanical design. Some challenges we ran into included overcoming mistakes made to the first revision of the PCB, implementing a sleep function to reach our power requirements, and making TagoIO intuitive and scalable. These unforeseen challenges set behind schedule. We made up for this set back with redirecting of tasks, communication and collaboration between all three team members. Table 13 outlines our Gannt chart.

Table 3. Gantt chart of the ORRHA project

Activity	Assigned to	November	December	January	February	March	April	May
PCB Desgin								
Design Sensor Node Circuitry	Dano							
Design PCB on Eagle	Dano							
Order PCB and BOM	Dano							
Manufacture PCB and test	Dano/Jeff							
Gateway Desgin								
Reaseach a gateway	Juan							
Interface sensors and test the gateway	Juan							
Design a functinal GUI	Jeff							
Make sensors easy added to system	Everyone							
Interface sensors with GUI	Everyone							
Mecanical Desgin								
Research materials suitable for IP65	Jeff							
Desgin chassis for sensor nodes	Juan							
Manufacture chassis	Juan							
Test								
Test sensor node battery life	Jeff/Dano							
Test materials suitable for IP65	Juan/Jeff							
Test transmission distance of the node and GW	Jeff							
Test the PCB housing for animal interference	Juan/Jeff							
System Integration and Test	Jeff							
Final intergration								
Final Assembly and intergration	Everyone							

5. List of Tests

Below, we present a summary of tests that are conducted so far.

Table 4: Summary of conducted tests.

Test Number	Objective	Setup/Procedure	Pass/Fail Criteria
FT-1	Test the hall effect sensors range in accordance with ER6.	Connect the hall effect sensor to an ESP8266 microcontroller and measure the distance away from a magnet that the sensor will change state. Measure the swing distance in both the X and Y directions of the magnet to determine when the sensor changes state.	PASS = The gate status remains closed when the magnet is within 1 inch from the sensor FAIL = The gate status remains open when the magnet is within 1 inch from the sensor.
FT-2	Measure the accuracy of the ultrasonic sensors in accordance with ER5	Take measurements from 20 cm to 500 cm with the ultrasonic sensor and compare it against a measuring tape.	PASS = The sensor measures a distance less than 3.5 cm of error compared to the tape measure FAIL= The sensor measures more than 3.5 cm of error compared to the tape measure
FT-3	Test the transmission range of the LoRa module in accordance with ER1	Place the gateway at a static location and place the transmitter in a vehicle and travel at least 2 Km away from the gateway. The transmitter will continuously transmit a signal that we can monitor using an online database.	PASS = The gateway continues to receive a signal after the 2 Km threshold FAIL = The gateway stops

			receiving a signal before the 2 Km threshold
FT-4	Battery Voltage ADC Measurement Accuracy	Connect a variable voltage supply to PCB battery connection and change the power supply from 4.0 V to 2.7 V in 10 mV increments. Then Record each voltage read by the ADC and compare that voltage to a multi-meter	PASS = The accuracy of the voltage reading is within 10 mV of the DMM FAIL= The accuracy of the voltage reading is over 10 mV of the DMM

ST-1	Test to observe the power consumed by the sensor node to determine if the battery life will meet the 1 month threshold and the 1.4 mA average in accordance with ER 3 & 4 (water tank sensor node)	Power the sensor node and start LoRa communication protocol. Use a power evaluation board to take data points of current over time. Using the data points, determine the average current draw of the sensor node	PASS = The average current is less than 1.4 mA and the battery lasts for longer than 1 month FAIL = The average current is more than 1.4 mA and the battery life will not last 1 month.
ST-2	Test to observe the power consumed by the sensor node to determine if the battery life will meet the 1 month threshold and the 1.4 mA average in accordance with ER 3 & 4 (gate sensor node)	Place the battery and PCB inside the chassis along with a strip of cobalt chloride water moisture strip. Using a 6.3mm nozzle of water, spray the chassis from all angles for 5 minutes. Open the chassis to observe if any water has made it through	PASS = There is no sign of water ingress on the cobalt chloride strip FAIL = There is evidence of water ingress

Description of Tests

FT-1 Hall Effect Sensor Range

Objective: Test the hall effect sensors range in accordance with engineering requirement 6.

Setup: We connected the digital hall effect sensor to an ESP8266. We programmed the ESP8266 to show the digital hall effect sensor's status on the serial monitor in the Arduino IDE.

Procedure: We placed a magnet on a wall which simulates a fence post and placed the hall effect sensor on a swinging TV mount that simulates the gate opening. We swung the hall effect sensor back and forth and measured the distance between the sensor and the magnet once the sensor status changed.

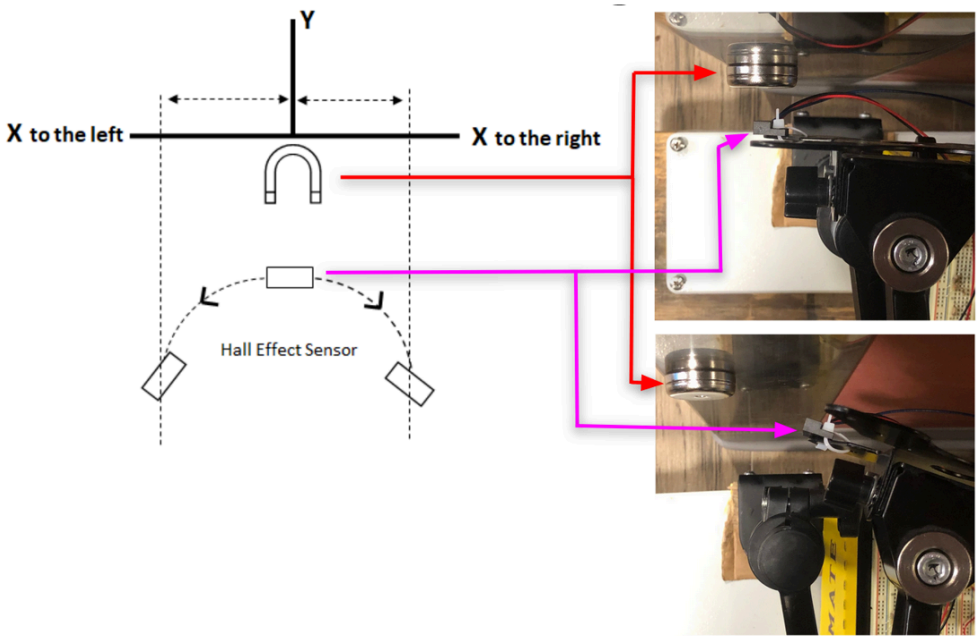


Figure 16. Procedure for testing the hall effect sensor

Results: The average range for change in sensor status we measured was 1.5 inches in both the X and Y directions. The test results indicate the test has passed.

Table 5: Table of thresholds when sensor status has changed

Y [Inches]	X to the Left [Inches]	X to the Right [Inches]	Status
0.5	< 1.5	< 1.3	Closed
1	< 1.75	< 1.5	Closed
1.5	< 1.75	< 1.5	Closed
2	< 1.5	< 1.75	Closed
2.5	< 1	< 1.5	Closed

FT-2 Ultrasonic Sensor Accuracy

Objective: Measure the distance accuracy of ultrasonic sensor in accordance with engineering requirement 5.

Setup: We connected the ultrasonic sensor to the Seed Xiao Samd21 and wrote a program to show the ultrasonic sensor's measurements on the serial monitor in the Arduino IDE. We then layed out a measuring tape from a wall out to 5 meters. Then we took a sensor measurement from 20 cm to 5 m in 10 cm increments and compared the value to the tape measure.

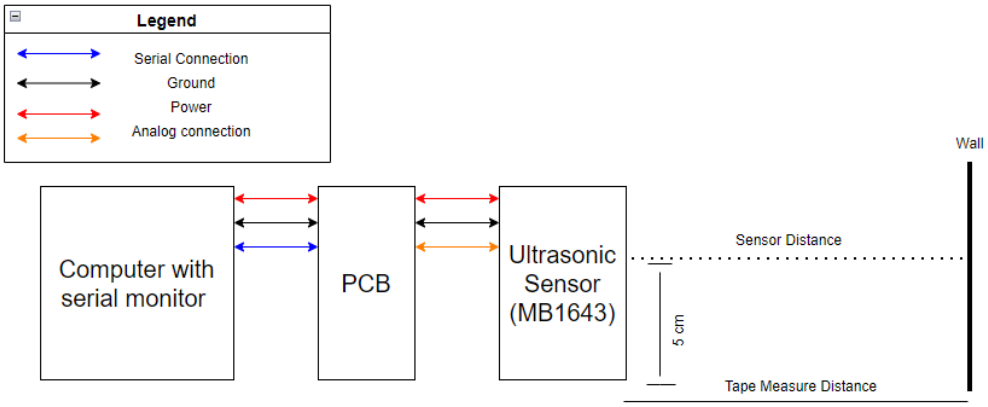


Figure 17. Procedure for testing the ultrasonic sensor

Results:
The average error of the sensor data was 0.26 cm and the maxim error was 0.97 cm

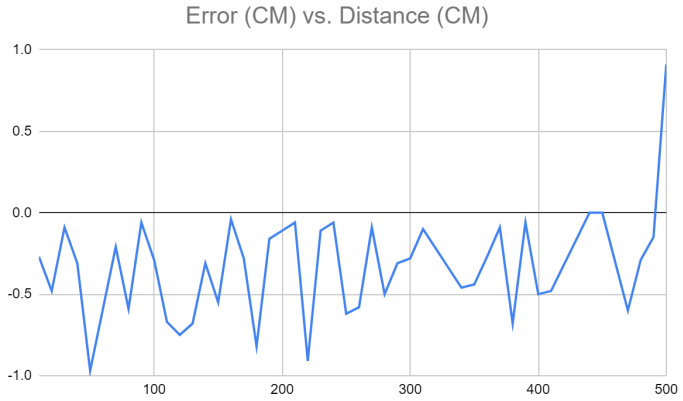


Figure 18. Error graph Vs Distance (Ultrasonic Sensor)

$$Res = \left(\frac{3.3V}{2^{12}}\right) / \left(\frac{3.3V}{500-20}\right) = \frac{0.0008047\,V}{0.006875\,V/cm} = 0.117cm$$

Figure 19. Equation for ADC resolution

FT-3 Transmission Distance

Objective: Verify data transmission range of sensor nodes to gateway is over over 2 km in accordance with ER1

Setup: We used Google Earth Pro to accurately establish the coordinates of our gateway and our transmitter to precisely measure the transmission distance. We then joined our gateway from a close distance to obtain the gateway ID and ensure a proper connection can be made. Knowing our gateway ID we will travel to the location obtained from google earth and establish a connection to the gateway and send data



Figure 20. Google Earth Pro image of distance traveled with LoRa Module

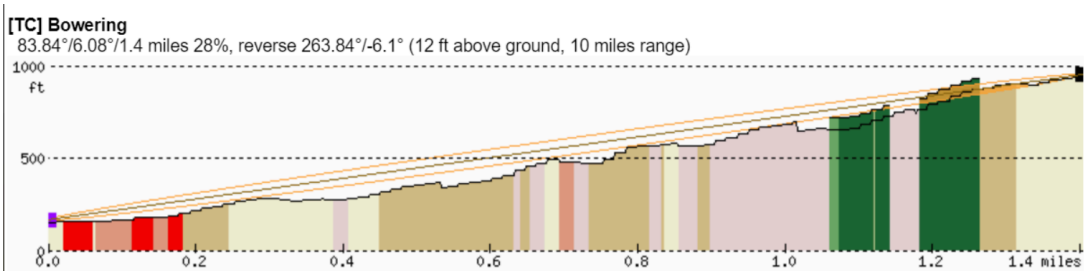


Figure 21. Graph showing non-line-of-sight

Results:
Data transmission by the sensor node was received by the gateway successfully beyond 2 km which can be viewed on table 6

Table 6. RSSI vs Distance

Test Location	Distance	RSSI
1	2.25 km	-89 dBm
2	2.70 km	-101 dBm
3	3.45 km	-98 dBm
4	4.33 km	-89 dBm
5	6.74 km	-91 dBm
6	10.08 km	-99 dBm

FT-4 Battery Voltage ADC Measurement Accuracy

Objective: Measure the accuracy of the ADC reading of the LithIon battery in the expected battery voltage range (2.7 V - 4.0 V)

Setup: Connect a variable voltage supply to PCB battery connection and change the power supply from 4.0 V to 2.7 V in 10 mV increments. Then Record each voltage read by the ADC and compare that voltage to a multi-meter

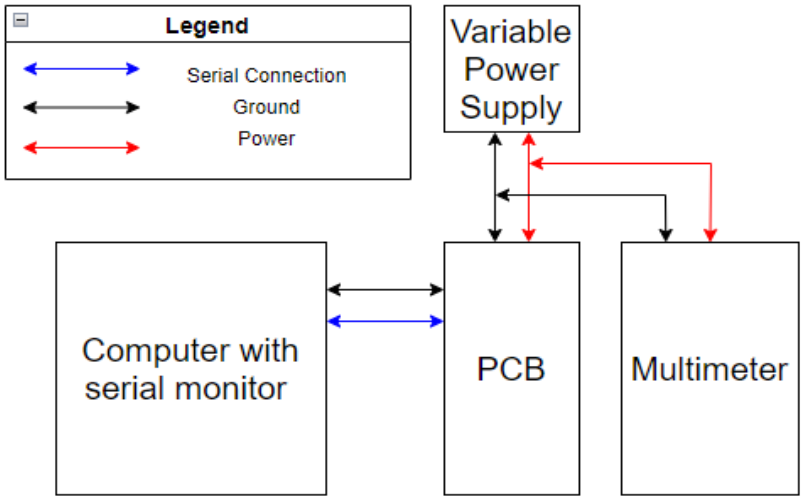


Figure 22. Procedure for testing Battery Measurement Test

Results:
The ADC was able to read the voltage levels with a 0.125% error when compared to the multimeter. With a maximum error of 11 mV between the voltage range 2.7 V - 4.0 V.

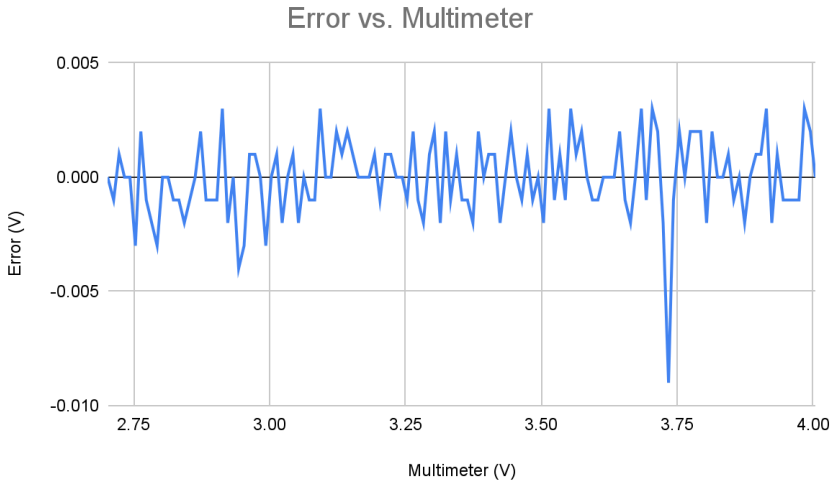


Figure 23. Error vs Voltage for ADC measurements

$$Res = \left(\frac{3.3V}{2^{12}}\right) / \left(\frac{3.3V}{4.0V - 2.7V}\right) = \frac{0.0008047\,V}{2.53846} = 0.32mV$$

Figure 24. Resolution equation for ADC measurements

ST-1 Gate Sensor Node Power Test

Objective: Measure the current consumption of the node with the hall effect switch using the gate detection firmware while transmitting data. A passing test must be under an average of 1.4 mA

Setup: Using the Nordic Power Profiler Kit II we measured and recorded the total current consumption of our system of the PCB. Upload the firmware to gate status readings in an interval of every 10 minutes. Supply voltage through the nordic board and let the firmware run for 30 hours (3 data transmissions). Observe the recorded current consumption data gathered by the Power Profiler.

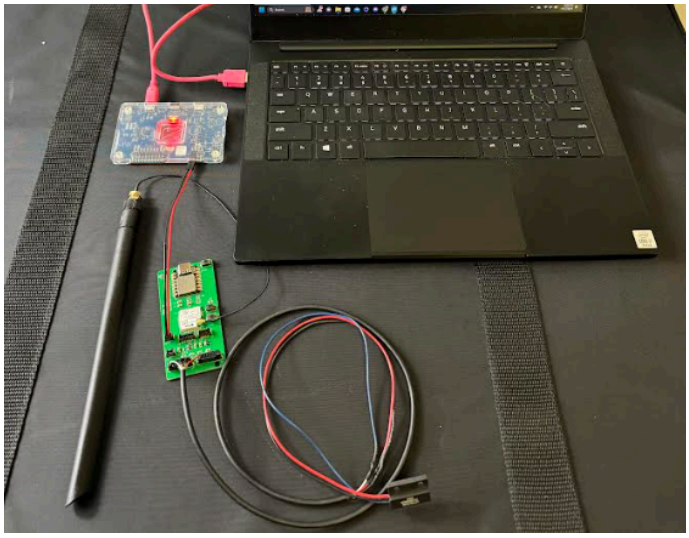


Figure 25. Image showing the Nordic Power Profiler Kit II connected to the Gate Sensor Node

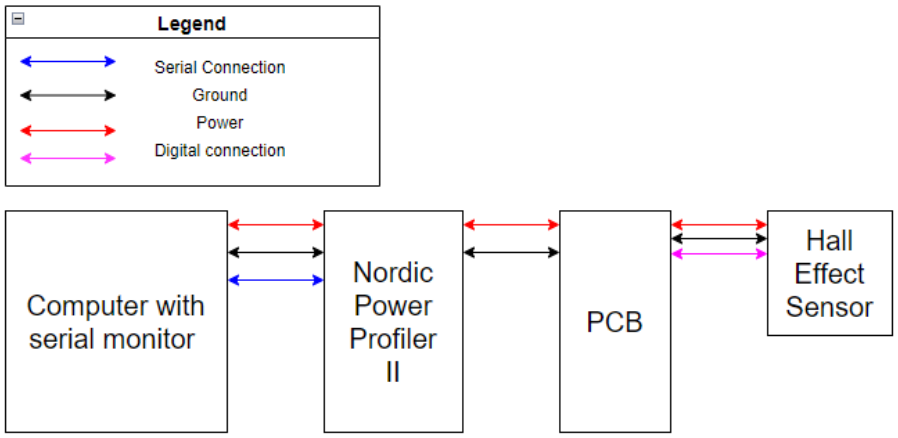


Figure 26. Block diagram for the setup of the power efficiency test

Results:
The Nordic Power Profiler 2 recorded an average of 3.62 mA over the 30 minute long test. A total of 6.88 Coulombs of charge

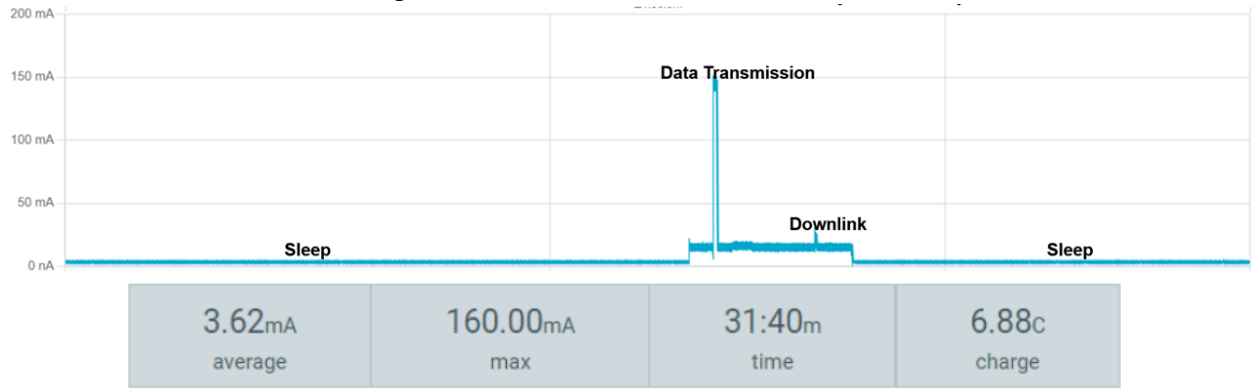


Figure 27. Nordic power profiler kit application showing an average current of 3.62 mA

$$Bat\ Life = \frac{(5200\ mAh * 0.80)}{3.62\ mA} = \frac{4160\ mAh}{3.62\ mA} = 1149.17\ hours = 47.88\ Days$$

Figure 28. Equation for battery life in hours and days showing our sensor node lasting 47.88 days with a 5200 mAh battery.

ST-2 Water Tank Node Power Test

Objective: Measure the current consumption of the node with the ultrasonic sensor using the water tank firmware and transmitting data. A passing test must be under an average current consumption of 1.4 mA

Setup: Using the Nordic Power Profiler Kit II we measured and recorded the total current consumption of our system of the PCB. Upload the firmware to take water tank level readings in an interval of every 10 minutes. Supply voltage through the nordic board and let the firmware run for 30 hours (3 data transmissions). Observe the recorded current consumption data gathered by the Power Profiler.

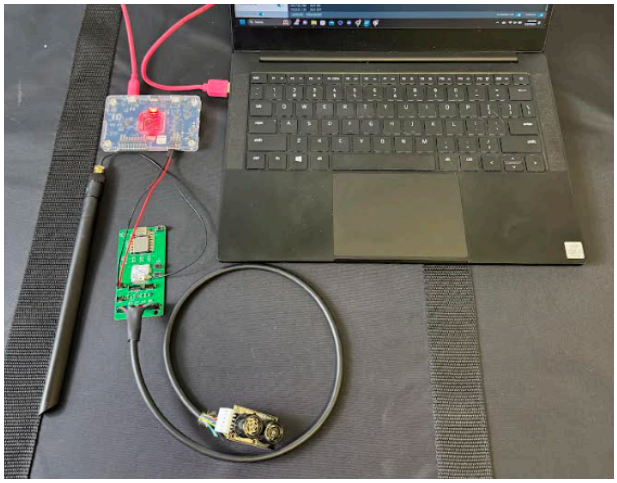


Figure 29. Image showing the Nordic Power Profiler Kit II connected to the Water Tank Sensor Node

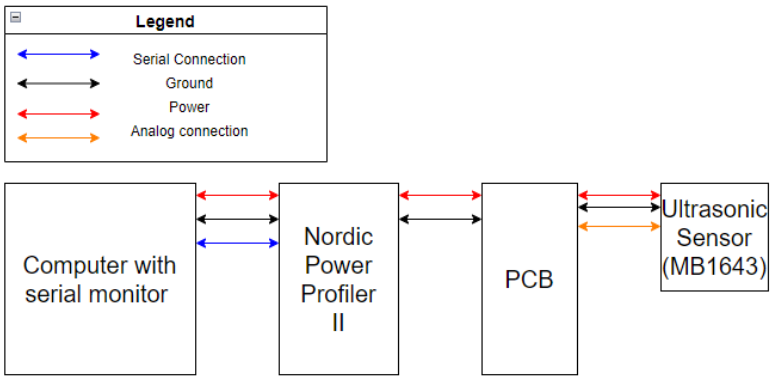


Figure 30. Block diagram for the setup of the power efficiency test

Results:
The Nordic Power Profiler 2 recorded an average of 4.13 mA over the 30 minute long test. A total of 7.01 Coulombs of charge

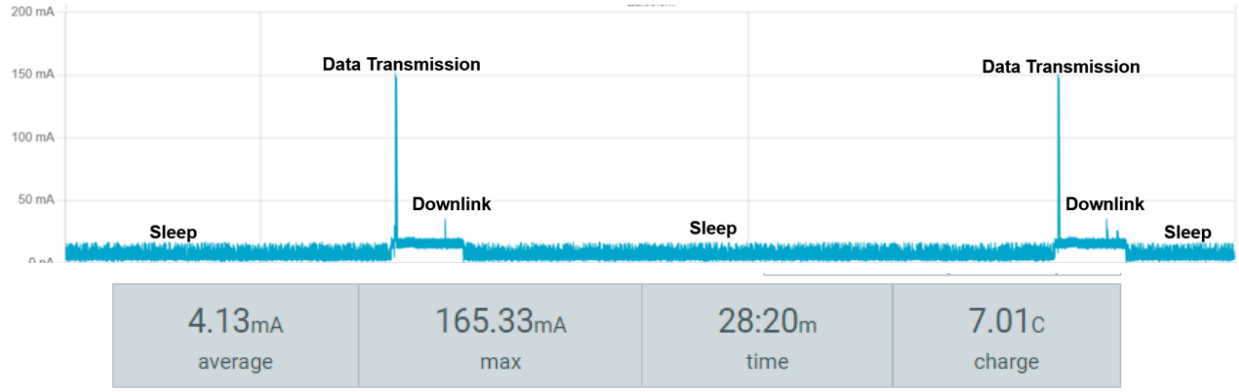


Figure 31. Nordic power profiler kit application showing an average current of 4.13 mA

$$Bat\ Life\ = \frac{(5200\ mAh * 0.80)}{4.13\ mA} = \frac{4160\ mAh}{4.13\ mA} = 1007.26\ hours\ =\ 41.96\ Days$$

Figure 32. Equation for battery life in hours and days showing our sensor node lasting 41.96 days with a 5200 mAh battery.

6. Ethics of the Engineering Profession and Our Project

The engineering design process should not be thought of as creating a solution to a problem. but rather solving the problem in the most efficient and ethical fashion possible. It is crucial to consider “future generations, and the promotion of healthy and sustainable life on this planet” [4]. More than ever we as engineering students must consider the footprint our project will leave behind. In the growing age of technology humankind has produced billions of IoT devices and billions more to come. Factors like safety, environmental, sustainability, social impact, privacy, and many more are all to be considered in the engineering design process.

Our project is combined on multiple parts that make up a system, such as the sensor nodes themselves, the gateway, the data base, and the graphical user interface. Each part of the system comes with its own unique set of ethical considerations. For the sensor nodes we shall consider environmental impact and sustainability. We shall choose parts that are manufactured by ethical companies and will be long lasting so that we minimize waste. For the gateway we must consider environmental impact, sustainability, and social impact. We have chosen a gateway based on the brand MikroTik because of the reliability and price, thus allowing a longer sustainability and limiting waste while keeping cost low so that our product can be more accessible. The database we choose shall be safely secure for privacy and must have vendor accountability. Security is of utmost importance in order to keep our customers’ data from being leaked or altered. Finally we shall consider security and social impact when designing our graphical user interface. Authentication and authorization shall be implemented to ensure the customer’s data is secure and safe as possible. We will also consider translation of our graphical user interface to other languages to provide a more accessible product. Solving the problem is our job as engineers, but our duty is to create a solution that is ethical.

It is important to ensure that the materials and components we used were sourced responsibly to minimize environmental impact. Our design prioritizes energy efficiency to extend battery life, therefore reducing electronic waste and promoting sustainability. Safety is another critical aspect, our PCB was designed and reviewed to help prevent hazards such as overheating or short circuits, protecting users and the environment. We have also considered the lifecycle of our product, including proper disposal and recycling processes, to ensure that our engineering solutions contribute positively to society and the planet.

References

[1] A. Lavric, A. I. Petrariu and V. Popa, "SigFox Communication Protocol: The New Era of IoT?," 2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI), Lisbon, Portugal, 2019, pp. 1-4, doi: 10.1109/ISSI47111.2019.9043727.

[2] M. Bor, J. Vidler, and U. Roedig, LoRa for the Internet of Things, https://www.eprints.lancs.ac.uk/id/eprint/77615/1/MadCom2016_LoRa_MAC.pdf (accessed Oct. 1, 2023).

[3] M. Elsaadany, A. Ali and W. Hamouda, "Cellular LTE-A Technologies for the Future Internet-of-Things: Physical Layer Features and Challenges," in IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2544-2572, Fourthquarter 2017, doi: 10.1109/COMST.2017.2728013.

[4] S. C. University, “An ethical toolkit for engineering/design practice,” Markkula Center for Applied Ethics, https://www.scu.edu/ethics-in-technology-practice/ethical-toolkit/ (accessed Dec. 14, 2023).

[5] “Ranch Eyes,” Rancheyes.com, https://rancheyes.com/lander (accessed May 17, 2024).

[6] “Agriculture and livestock remote monitoring solutions,” Monnit, https://www.monnit.com/applications/agriculture-livestock-monitoring/ (accessed May 17, 2024).

[7] “Ranch Systems” Ranch Systems, https://www.ranchsystems.com/home/ (accessed May 17, 2024).

[8] “RAK3172 WisDuo Lpwan Module,” RAK Documentation Center, https://docs.rakwireless.com/Product-Categories/WisDuo/RAK3172-Module/Overview/ (accessed May 17, 2024).

[9] “Getting started with Seeed Studio xiao samd21: Seeed Studio Wiki,” Seeed Studio Wiki RSS, https://wiki.seeedstudio.com/Seeeduino-XIAO/ (accessed May 17, 2024).

[10] “Nordic Power Profiler kit,” Technical documentation, https://docs.nordicsemi.com/bundle/ug_ppk2/page/UG/ppk/PPK_user_guide_Intro.html (accessed May 17, 2024).

Appendix

The following is the code for the water tank node:

```
#include <RTCZero.h>
#include <ArduinoLowPower.h>

const int ultrasens = A2; //Orange Wire
const int batRead = A1;
float sensorValue;
float distance;
String distMeasure;
int waterLevel;
String ATwaterStatus;
String STRwaterLevel;

String receivedPayload;
String receivedResponse;

int payloadTime;
int delayTime;
int distanceINT;
int voltageINT;
String TTNdist;
String TTNvoltage;
String STRvoltage;
String payload;

void setup() {
  pinMode(11, OUTPUT);
  digitalWrite(11,HIGH);
  pinMode(12, OUTPUT);
  digitalWrite(12,HIGH);
  pinMode(13, OUTPUT);
  digitalWrite(13,HIGH);
  analogReadResolution(12);
  Serial.begin(115200);
  Serial1.begin(115200);
  delay(1000);
  pinMode(A1, INPUT);
  Serial1.println("AT+JOIN=1:0:10:8");
  delay(5000);
  Serial1.println("AT+LPM=1");
  payload = 8
  ; //delay time (900 is 15 minutes)

}

void loop() {
    //--------------WATER LEVEL MEASURMENT-------------//
////////////////////////////////////////////////////////////////
  sensorValue = analogRead(ultrasens);
  delay(50);
  float distance =(2.227922e-7 * pow(sensorValue, 2)) + (0.150025 * sensorValue) -31.45686;
  distanceINT = round(distance);
  STRwaterLevel = String(distanceINT);
  if (distanceINT < 100){
    TTNdist = "FF" + STRwaterLevel;
  }
  else {
    TTNdist = "F" + STRwaterLevel;
  }
  delay(100);
////////////////////////////////////////////////////////////////

```

```
    //--------------BATTERY VOLTAGE-------------//
////////////////////////////////////////////////////////////////
  float voltage1 = analogRead(batRead) * (3.3 / 4096.0) * 2.0145 *100;
  delay(100);
  voltageINT = round(voltage1);
  STRvoltage = String(voltageINT);
  TTNvoltage = "F" + STRvoltage;
  delay(100);
////////////////////////////////////////////////////////////////

  ATwaterStatus = "AT+SEND=1:" + TTNdist + TTNvoltage;
  Serial1.println(ATwaterStatus);
  delay(1000);

////////////////// CHECK FOR PAYLOAD //////////////////////////////////
  while (Serial1.available() == 0) {}
  receivedResponse = Serial1.readString();
  Serial.println(receivedResponse);

  int payloadStartIndex = receivedResponse.indexOf(":ff") + 1; // Adjust index to start after ':'
  int payloadEndIndex = receivedResponse.indexOf("ffff", payloadStartIndex);

  if (payloadStartIndex != -1 && payloadEndIndex != -1) {
    payload = receivedResponse.substring(payloadStartIndex + 2, payloadEndIndex);
  } else {
    Serial.println("no downlink");
  }

  payloadTime = payload.toInt();
  delayTime = payloadTime * 1000; //Convert payload to milliseconds
  Serial.println(delayTime);
  delay(5000);

////////////////////////////////////////////////////////////////////
  //delay(delayTime);
  LowPower.deepSleep(delayTime);

}

void dummy(){

}
```

The following is the code for the gate node:

```
#include <RTCZero.h>
#include <ArduinoLowPower.h>

const int hallEffectPin = D0;
int hallEffectStatus;
String gateStatus;
String ATgateStatus;

const int batRead = A1;
int level;
int batteryLevel;
String payload;
String receivedPayload;
String receivedResponse;
int payloadTime;
int delayTime;

int voltageINT;
String TTNdist;
String TTNvoltage;
String STRvoltage;

void checkStatus(){
    hallEffectStatus = digitalRead(hallEffectPin);
}

void setup() {
    pinMode(hallEffectPin, INPUT);
    pinMode(A1, INPUT);
    analogReadResolution(12);
    pinMode(11, OUTPUT);
    digitalWrite(11,HIGH);
    pinMode(12, OUTPUT);
    digitalWrite(12,HIGH);
    pinMode(13, OUTPUT);
    digitalWrite(13,HIGH);
    Serial.begin(115200);
    Serial1.begin(115200);
    delay(1000);
    Serial1.println("AT+JOIN=1:0:10:8");
    delay(5000);
    Serial1.println("AT+LPM=1");
    LowPower.attachInterruptWakeup(digitalPinToInterrupt(hallEffectPin), checkStatus, CHANGE);
    payload = 900; //delay time (900 is 15 minutes)
}

void loop() {
    hallEffectStatus = digitalRead(hallEffectPin);
    delay(500);
    //--------------GATE STATUS PARAMETERS-------------//
////////////////////////////////////////////////////////////////////
    if (hallEffectStatus == 1){gateStatus = 10;} //Gate is Open
    if (hallEffectStatus == 0){gateStatus = 11;} //Gate is Closed
    delay(500);
////////////////////////////////////////////////////////////////////

    //--------------BATTERY VOLTAGE-------------//
////////////////////////////////////////////////////////////////////
    float voltage1 = analogRead(batRead) * (3.3 / 4096.0) * 2.0145 *100;
    delay(100);
    voltageINT = round(voltage1);
    STRvoltage = String(voltageINT);
    TTNvoltage = "F" + STRvoltage;
    delay(100);
////////////////////////////////////////////////////////////////////
```

```
66
67      ATgateStatus = "AT+SEND=1:" + gateStatus + TTNvoltage;
68      //ATgateStatus = "AT+SEND=1:" + gateStatus;
69      Serial1.println(ATgateStatus);
70      //Serial1.flush();
71      //Serial.println(ATgateStatus);
72      delay(1000);
73
74  ////////////////// CHECK FOR PAYLOAD //////////////////////////////////////
75      while (Serial1.available() == 0) {}
76      receivedResponse = Serial1.readString();
77      //Serial.println(receivedResponse);
78
79      int payloadStartIndex = receivedResponse.indexOf(":ff") + 1; // Adjust index to start after ':'
80      int payloadEndIndex = receivedResponse.indexOf("ffff", payloadStartIndex);
81
82      if (payloadStartIndex != -1 && payloadEndIndex != -1) {
83        payload = receivedResponse.substring(payloadStartIndex + 2, payloadEndIndex);
84      } else {
85        //Serial.println("no downlink");
86      }
87      payloadTime = payload.toInt();
88      delayTime = payloadTime * 1000; //Convert payload to milliseconds
89      //Serial.println(delayTime);
90      delay(5000);
91  ////////////////////////////////////////////////////////////////////////
92
93      //Serial.println("START LONG DELAY");
94      //delay(delayTime);
95
96      LowPower.deepSleep(delayTime);
97  }
98
99  void dummy(){
100
101 }
```