Spring 2022

A Low-cost, Long-range, and Solar-based IoT Soil Quality Monitor

Salvador Garcia

Trina Nguyen

Julian Wong

Santa Clara University

Department of Computer Science and Engineering

Department of Electrical and Computer Engineering

Date: June 9, 2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Salvador Garcia, Trina Nguyen, Julian Wong

ENTITLED

A Low-cost, Long-range, and Solar-based IoT Soil Quality Monitor

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

and

BACHELOR OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

DocuSigned by:

Behnam Dezfouli

6ED175D6B91A4FF...

Thesis Advisor
Dr. Behnam Dezfouli

[illegible]

Thesis Advisor
Dr. Sarah Wilson

Chairman of COEN Department
Dr. Nam Ling

Chairman of ECEN Department
Dr. Shoba Krishnan

A Low-cost, Long-range, and Solar-based IoT Soil Quality Monitor

By

Salvador Garcia, Trina Nguyen, Julian Wong

Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Science
in Computer Science and Engineering
and Electrical and Computer Engineering
in the School of Engineering at
Santa Clara University,

Spring 2022

Santa Clara, California

Acknowledgments

We would like to thank our advisors Dr. Behnam Dezfouli and Dr. Sarah Wilson for all of their help throughout the past year on this project. We would also like to thank our friends, family, and professors for all of their support during our time spent at SCU.

A Low-cost, Long-range, and Solar-based IoT Soil Quality Monitor

Salvador Garcia, Trina Nguyen, Julian Wong

Department of Computer Science and Engineering
Department of Electrical and Computer Engineering
Santa Clara University
Santa Clara, California

June 9, 2022

ABSTRACT

The project objective is to create a low-cost, long-range, and solar-based IoT soil quality monitoring system. The system must transmit packages of data gathered from separate nodes, consisting of two different types of sensors, to a centralized gateway receiver to be displayed to the user in an elegant and readable manner. The end goal of the project is to supplement produce grown by large agricultural bodies around the United States without the misuse of water resources. This report presents the need for this system, details the components of the system, and the rationale behind design choices. It serves as a comprehensive guide to all the work that has been completed, provides an outlook for future iterations, and demonstrates the viability of LoRa communication for low power packet sending in a rural environment.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

IoT	Internet of Things
LDO	Linear Dropout Regulator
LoRa	Long Range Radio
MPPT	Maximum Power Point Tracking

Chapter 1
Introduction

California is the United States of America's biggest agricultural contributor, despite having only 4% of the nation's farms [9]. However, the state is no stranger to drought, with over 50% of the state in the state of "severe drought," meaning that the "Water is inadequate for agriculture, wildlife, and urban needs" [10]. Because of this, it becomes extremely important to have a system to help out these farmers in measuring soil quality data in order to put their resources to efficient use. Our project objective is to create a sensor system capable of using LoRa communication to transmit soil quality variables over long distances to a home network without the reliance on Wi-Fi or high power consumption communications. Our end goal is to develop a fully functioning system that enables farmers and other owners of large areas of land on which crops grow to use less water, and only use water when it is required.

Furthermore, with targeted soil quality measurements, an agricultural land owner will be able to easily read the health of their soil and take action as soon as possible. This report goes over the details and rationale behind our system's design, as well as the testing and evaluation of the system. Additionally, this report will include guides and discussions on the work completed, and future work to be completed so that the next iteration of this project can improve the viability of the system in a real world scenario.

Chapter 2

Related Work

In this chapter, we overview relevant smart garden systems and justify the importance of our garden sensor system. Also, we summarize relevant wireless communication protocols, and existing low consumption IoT solutions based on WSNs and LoraWAN technology. Afterwards, we present the goals and objectives for our project. Then we go over types of energy harvesting systems, and the research behind the type of sensors we looked for. Finally, we discuss the challenges and shortcomings of our project.

2.1 Analyzing Existing Solutions

Currently on the market, there are several smart home irrigation controllers that give the consumer a variety of tools and a number of watering zones to monitor their plants. A list of systems researched are listed in Figure 2.1 below.

All of these controllers, with the exception of the ESP4ME3, boast of being able to adjust its watering schedule automatically, relying on Wi-Fi to receive weather reports and predictions to lessen or completely stop the sprinkler system [15]. Controllers like the Rachio 3, RainMachine Touch HD, Blossom 7, and Spark-8 also come with free mobile apps that allow the consumer to monitor and adjust the timing of their watering schedules [12,13,16,17]. The Rachio 3 in particular has a "Smart Cycle" function, which makes watering periods shorter and more frequent to

Name	Manufacturer	Installation Type	Price for: ≤ 8 zones	Price for: 12 zones	Price for: 16+ zones
Rachio 3	Rachio	Indoor*	$199.99	N/A	$249.99
RainMachine Touch HD	RainMachine	Indoor*	N/A	$259.00	$269.00
B-Hyve XR	Orbit	Outdoor	$149.99	N/A	$179.99
ESP4ME3**	Rainbird	Outdoor	N/A	N/A	$209.00
Blossom 7 Zone Wi-Fi Smart Water Controller	Blossom	Indoor	$52.10	N/A	N/A
Spark-8	Netro	Indoor	$179.00	N/A	$199.00

**offers outdoor installation with separate purchase*

***does not come with wifi compatibility upon purchase*

Table 2.1: Existing solutions and their prices

prevent runoff [12]. Many of these devices also have a very sleek and modern look, and have a touchscreen right on the micro-controller. The main benefits to these types of systems are the ease of installation, the aforementioned mobile apps, and compatibility with other smart home devices such as the Google Home and Amazon Alexa that allows the consumer to have a more hands free approach to caring for their plants.

However, these benefits are also its biggest downsides. First of all, the consumer must have an established irrigation system built into their home to take advantage of these controllers. Being reliant on Wi-Fi also prevents the system from being expandable beyond the size of an upper middle class home, and its application for more rural areas with larger amounts of farmland is near impossible. It may also lead to issues in watering schedules if connection to the controller is disrupted. In the case of the Blossom 7, though much cheaper than its competitors, suffers from unreliability due to its Wi-Fi functions, as many consumers report having their watering schedules completely canceled without any notice if their home network, or the company website goes down [16].

The market solutions also do not cover the specific niche of having any sensors in the ground to monitor health and moisture of the microbes in the soil. While the Rachio 3 does its best of calculating the moisture content through taking in the degree of field incline, the type of soil, and how much water is used, it cannot measure the temperature and salinity of the soil, both of which are major factors in ensuring crop health [12]. Though not designed for long range wireless systems, such solutions to this problem did exist, such as the now defunct Edyn Garden Sensor. Similar to the irrigation controllers, the Edyn featured Wi-Fi connectivity and a mobile app to track moisture levels, temperature, light intensity, humidity, and soil nutrition. However, it is plagued by the same issues brought up with the irrigation controllers, as well as having a short term battery life [27]. Other similar products, such as the Oso Technologies PlantLink and Parrot Flower Power either only tracks moisture levels, or has a high price tag per unit [28, 29]. Therefore, there are many areas of improvement to go into, such as creating a low cost wireless modular system that relies on renewable solar energy to power itself.

2.2 Project Objectives and Goals

Our team's goals for this project are based off concerns for potential consumers, which are rural farmers, and concerns for the environment in which the system would be placed. Our main objective is to create a scalable sensor system that does not rely on Wi-Fi, that could measure different soil health variables and transmit that data to a central gateway, which would send that information to a web page that could display that data in readable terms. The team is comprised of two sub-teams, computer science and electrical, with the different objectives for each team to be done over the course of the year. The computer science team focused on interfacing and programming both the boards and the central gateway, as well as handle the communication protocols. The electrical team focused on the hardware and circuitry, and worked on the power consumption and collection for the system.

2.3 Sensor Type

The original design of our project wanted to implement three different types of sensors, each pertaining to a different component of soil health to monitor. We wanted our system to be able to measure moisture, temperature, and pH. Finding sensors for the first two criteria is relatively easy. There were many cheap and simple solutions with added features that fit well with the objectives of our system. The model we ended up choosing for the moisture sensor is the non-corrosive sensor provided by Seeed: the Grove Capacitance Soil Moisture Sensor [19]. Unlike other popular moisture sensors on the market, which use resistivity to measure moisture levels, this unit uses a capacitive sensor. This difference in measurement technique does not require direct exposure of the metal electrodes like its resistive peers, which mitigates any corrosion caused by the sensor and lengthens the longevity of the sensor. Similarly, the temperature sensor chosen, the DS18B20 [18], features a non-corrosive, waterproof covering for the sensor and the wires attached, while boasting a 0.5C accuracy in its readings. The long cable also allows deployment of the sensor from up to five meters from the microcontroller itself, allowing farmers to place the temperature sensor to their convenience if the situation calls for it.

However, the issues arose when researching pH sensors. Most pH sensors are for single time use, and are usually submerged in some type of liquid. Of the pH soil sensors researched, they either consumed more power than our board can output, or they were not meant for long term deployment in soil. Moreover, those sensors needed to be re-calibrated frequently, and low maintenance is one of the main objectives this project is aiming to achieve.

The solution to this problem is to search for electrical conductivity (EC) sensors instead of pH sensors, as they were deemed to be more practical for the scope of our project. Electrical conductivity in soil is an indication of its salt concentration, similar to pH calculation. One of the positives in this new direction of research is that many of the EC sensors found were 3-in-1 sensors that also included moisture and temperature probes. However, almost all of the models were not readily available, as many of them either needed us to ask for a quote, or were only research prototypes. The one we decided on is the Industrial grade Soil Moisture, Temperature, and EC Sensor MODBUS-RTU RS485 sold by Seeed, which allowed for long time deployment in solids and liquids, stated to be anti-corrosive, and is readily available for consumer purchase. It also came with a five meter long cable, similar to the temperature sensor we were looking at, which allowed for the probe to be deployed away from the Arduino microcontroller. Cost wise, it is the same, if not cheaper than, the price of the three previous sensors totaled. However, once the actual sensor came in, we learned drew much more power at a higher voltage than our board could output, despite being advertised as requiring a minimum of a 3.6V power supply. Due to this incompatibility, our team decided to reduce the sensor number to two: the temperature and moisture sensors.

2.4 Challenges and Shortcomings

One of the main challenges faced while starting this project is dealing with the hardware supply chain. While many of the simpler parts we needed, such as boards and wires, were readily available and shipped within the week, the sensors took about a month to arrive from overseas, and there is only one available at the time of purchasing. The module we wanted to use for our communication protocol is also plagued with shipping issues, to the point that we would not be able to receive it in time for the project, and had to pivot to a different model halfway through. Therefore, we have to treat a lot of these pieces carefully, as well as glean as much information as we can from them, since we have a limited supply, budget, and time to understand it.

Another challenge is the ongoing global pandemic due to COVID-19, which hampers not only the shipping times for our design components, but also the meeting times between group members. Though the risk of contraction is low, due in part to the weekly testing and mandatory vaccination regulations of the school, new variants such as the Omnicron variant still has a high risk in postponing the project development in case the school decides to go fully back to online classes as it did back in 2020. A precaution that is already in place is keeping the mask mandate whenever the group meets in person, and to hold virtual meetings whenever the former is not possible.

Lastly, the final challenge is learning soldering and gaining access to the Maker Labs on campus. Soldering comes with its own set of risk factors, such as handling burns, inhalation of toxic fumes, skin contact with toxic materials, and touching sensitive areas such as eyes and mouth after handling molten solder. The Maker Labs also require taking an online course and an in person tour before being allowed access. Before the start of the project, only one member knew how to solder safely, and no one else had Maker Lab access before. Therefore, the group needs to find time to fully learn and master these two skills so that we can solder parts together and access the 3D printer to build a weatherproof hood for our devices.

Chapter 3

System Architecture

In this chapter, we go over the parts of the system we designed. First, we will go over the rationale behind choosing LoRa, the Raspberry Pi gateway, the Arduino boards, and the details behind the sensors bought. Then, we will go over the power system behind our project, and the solar panels used. Lastly, we will explain our usage of PCBs and how we went about the sleep mode for the Arduino boards.

3.1 LoRa Communication

LoRa is the main type of technology that allows us to address the majority of the needs of our project. It is a long range, low power modulation technique that creates a wireless platform for IoT devices.

In most IoT applications, the most optimal mode of network communication is through Wi-Fi, since it allows for more data to be sent at a faster rate with low transmission costs. However, the intended consumer for our project are mid-sized farmers, whose access to Wi-Fi is limited across the whole range of the farm. This would require additional hardware to be installed to collect and transmit data through Wi-Fi, and this goes against our goal to keep our system low cost and low maintenance. Other solutions researched were Ethernet, Bluetooth, and LoRa. Ethernet cables are a good substitution for Wi-Fi, since they can transmit data at a faster rate than Wi-Fi and ensure a secure and clear connection due to being

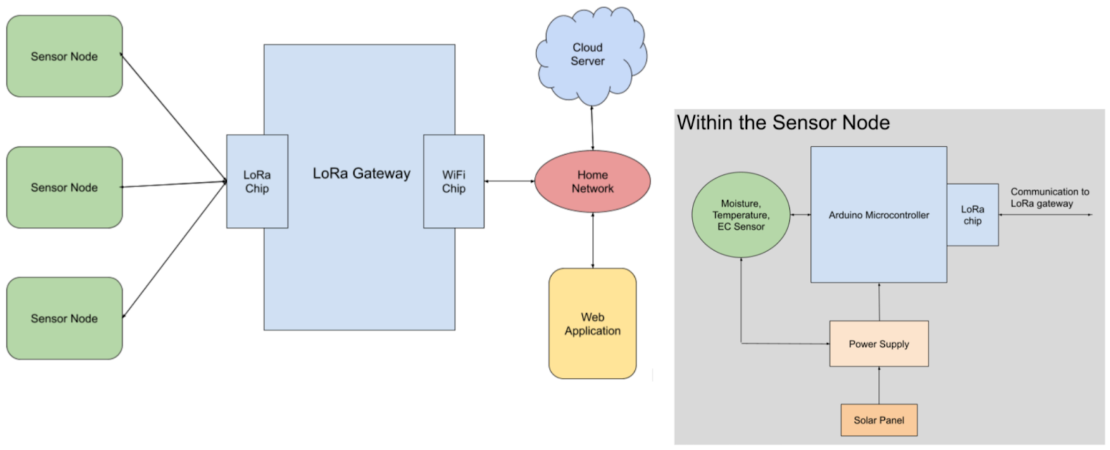


Fig. 3.1: Block Diagram of System Architecture Using LoRa

a wired, physical connection. However, this is very impractical for our project, as Ethernet cables can only run up to 100 meters, and requires a physical wired connection, while the range of our system needs to go way beyond that distance.

Bluetooth is a short-range, wireless network used mainly for transmitting data between fixed and mobile devices. It utilizes either hub-and-spoke model, where each host is connected to a central hub, or a mesh system, where data is shared through different nodes to efficiently transmit data. However, Bluetooth's short range and makes it difficult to justify using it for our system, despite its low power consumption.

LoRa provides not only a long range, wireless solution, but also ensures low power consumption at a low cost due to its frequency shift keying modulation technique. It allows for one gateway module to accommodate up to 1000 end-node devices, making it ideal for our project, which aims to add more end-node sensor devices to cover large amounts of farmland. For larger and broader IoT implementations the LoRaWAN communication protocol and network architecture can be used on top of the physical LoRa hardware to deploy a WAN (Wide Area Network), also capable of incorporating node authentication and data encryption for security [11].

Figure 3.1 is an example block diagram we created to demonstrate our system architecture using LoRa. We elaborate on the components of this block diagram in the following sections.

3.2 Raspberry Pi Gateway

With one of the primary motivations of the project being the construction of an affordable system easily available to farmers, building a custom LoRa gateway from cheaper components than existing fully assembled models is a straightforward first step. Many industrial grade outdoor gateways require a basis to basis price quote, but popular list models such as RAK's RAK7289 priced at $372, Seeed Studio's SenseCAP M1 priced at $519, and Cisco's IXM-LoRaWAN-CPF priced at $564 set a price range of several hundred dollars [22,23,24]. In comparison, building a packet forwarding gateway from a Raspberry Pi is a significantly cheaper, but still well documented route, largely due to the Raspberry Pi's native Wi-Fi capabilities.

After choosing to use a Raspberry Pi to power the packet forwarder, a LoRa hardware module needs to be attached in order to allow the Raspberry Pi to receive LoRa packets. To this end, a LoRa chip developed by Semtech is needed. To further reduce costs, we could have purchased an isolated LoRa chip and connected it to the GPIO pins on the Raspberry Pi with our own lead wires, but we chose to purchase a pre-built LoRa Pi Hat due to our inexperience with IoT development. Ideally we would have purchased a single channel LoRa module such as the Seed WM1302 LoRaWAN module, but due to supply chain constraints we purchased a multichannel LoRa module, the RAK2245. As the capability to receive packets on multiple channels and spreading factors at once is unnecessary for our purposes, the RAK2245 is an unfortunate over investment.

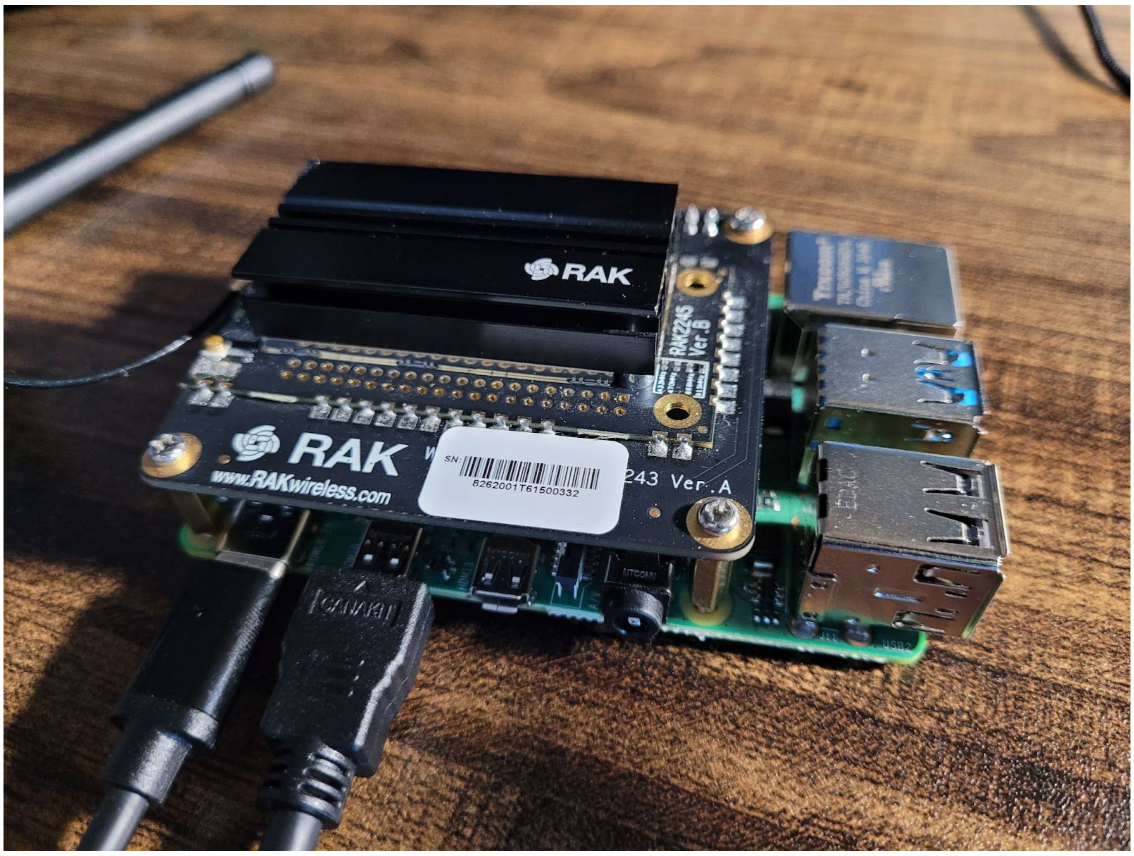


Fig. 3.2: Raspberry Pi with RAK2245 Pi Hat Mounted

Figure 3.2 below shows our Raspberry Pi, complete with a RAK2245 Pi Hat mounted on top to allow LoRa packet reception.

3.3 Arduino Nodes

The Arduino boards used to power and interface the sensor system are the MKR WAN 1310 model. This board comes with the LoRa chip pre-installed onto its hardware, so it could be capable of LoRa communication without any additional hardware added to it. This allowed our system to be much cheaper, and more efficient to use, as the alternative would've been to put an Arduino board and a separate LoRa module to attach to it.

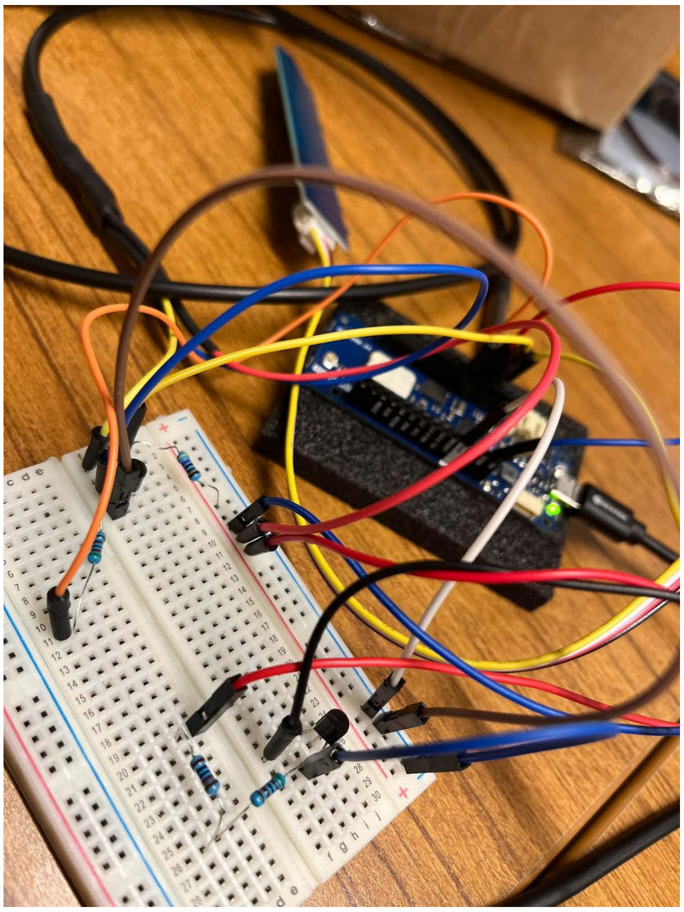


Fig. 3.3: Early Arduino Build Using a Breadboard

Figure 3.3 shows an early Arduino build using a breadboard to connect the temperature and moisture sensors to the Arduino.

3.4 Sensors

Choosing sensors that are compatible with the rest of our hardware and that can be integrated within the constraints of our specifications is rather difficult, but as mentioned in a previous section, we ultimately settled with two sensors: the DS18B20 1-wire temperature sensor and the Grove Capacitive Soil Moisture Sensor.

The DS18B20 1-wire temperature sensor comes in two form factors [19]. One

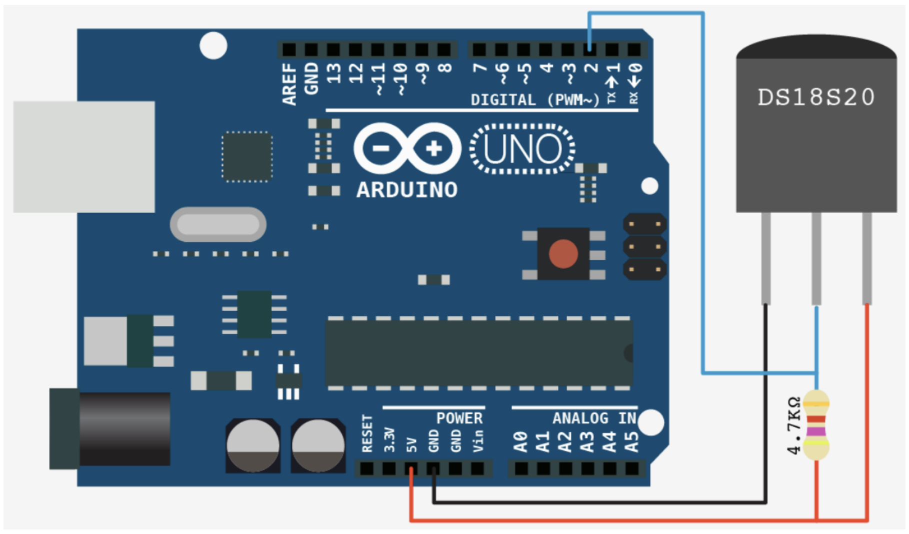


Fig. 3.4: Schematic of the DS18B20 with the pull-up resistor

is the IC package and the other comes with a waterproof probe styling which is much more useful for our purpose. Apart from the packaging of the sensor being perfect for our project, it can measure temperatures from -55 degrees Celsius to 125 degrees Celsius with an accuracy of plus or minus half a degree Celsius. Apart from its use, the sensor runs safely with a power supply of 3.3V to 5.5V and a power consumption of 1mA maximum, which means we can easily integrate it into our 3.3V system. The difficulty with this sensor is that none of our team members had prior experience with the 1-wire communication protocol. In order to overcome this, our team is able to use tutorials released by Maxim Integrated and OneWire libraries released by the manufacturer of the sensor and Arduino's official OneWire library. As per the manufacturer,Äôs directions, in order to integrate the sensor properly in hardware we needed to implement a 4.7 kOhm pull-up resistor as seen in Fig 3.1. This will effectively mitigate any noise present in the output signal of the DS18S20 temperature sensor.

The chosen soil moisture sensor selected for the capacitive nature of the sensor and its corrosion resistive properties. Contaminating the soil with metal corrosion is extremely counterproductive to the goals of our system, so its non corrosion measurement method is a big factor in choosing it. Additionally, this sensor can take in either 3.3V or 5V, allowing us to easily integrate it into our system with our 3.3V circuit voltage [21]. Integrating this soil moisture sensor via software appears to be far easier than the OneWire temperature sensor because it does not use this obscure communication technology, but we certainly run into issues transforming our analog output signal into readable data. This will be discussed in the evaluation section of the range and data accuracy performance section of our evaluation chapter.

3.5 Power Criteria

The system requirement for our project to be considered successful, per our advisors' specifications, is a maximum quiescent current of 50 micro-amps. Quiescent current is the amount of current our system as a whole draws from the power system when it is in ”sleep mode,” or when it is actively gathering or transmitting sensor data. The advertised quiescent current minimum on Arduino‚Äôs website for our board, the MKRWAN 1310, is 104 micro-amps; because this is much higher than our specified 50 micro-amps, our team would need to make hardware modifications to the MKRWAN 1310 boards [21].

Our boards generally take in a 5V regulated input, but the on-board circuit voltage is 3.3V [21]. This issue became apparent once we started to disconnect the USB input from our nodes for a much-needed power consumption drop. The only other way to power the board without the input is via the VCC pin on the board, which bypasses the voltage regulator on the board. This effectively meant that we were attempting to power a 3.3V circuit with a 5V regulated power supply. This issue is fixed by applying an 3.3V linear dropout regulator (LDO) at the output of the solar charge controller. This chip is the same one as the low quiescent current chip we originally replaced on-board the Arduino to cut down on power consumption while in sleep mode.

3.6 Power System

The main part of the power system we used to power our nodes is the Solar Power Manager 5V v1.1 board. Not only is this board relatively cheap, and readily available for off-the-shelf purchase via Amazon, but it has all of the features our team is looking for. This module features inputs from both a 3.7V nominal lithium battery and simple 5V nominal solar panel, but ensures we are powering our system with the solar panels as often as possible via an integrated Maximum Power Point Tracking (MPPT) feature [25]. This feature is meant to take the I-V characteristic curve of a solar panel into consideration when charging the battery and supplying power to the DC load that is our Arduino board. The output of the photovoltaic module (the solar panel) will fluctuate depending on the light intensity stimulating the photovoltaic; the variable current will be matched with an optimized voltage via the MPPT feature in order to ensure maximum power from the solar panel. As seen in Fig. 3.2, the power manager has the ability to draw power from the lithium battery as well as charge it, and with this feature comes the need for proper safety precautions. These include over charge, over discharge, over current, short circuit, over heat protection, and a few reverse connection protections [25]. These features are crucial to keep the battery, solar panels, and board working safely and properly.

For testing, we power the board using the USB 5V, and this made it very easy to reprogram the board and test software quickly. Once the team started to progress

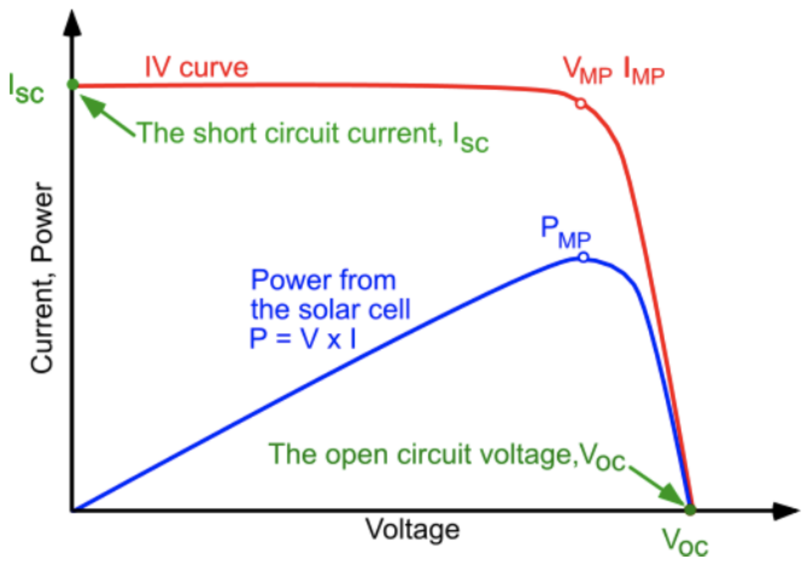


Fig. 3.5: Current Power vs. Voltage

into the testing stage of the systems' power consumption portion of this project, we soon realized that the USB connection can actually cause lots of current to be drawn even while the system is in low power 'sleep mode.' A simple solution is to use the 3.3V LDO we had modified the Arduino's with directly from one of the 5V output pins on the solar power manager board into the Arduino's VCC pin. In order to do this properly, the electrical engineering sub-team is able to come up with a small PCB board meant to test IC's with the same/similar packaging as the LDO used.

3.7 Solar Panels

When looking for solar panels that were compatible with our system, we took into account a few things: size, nominal voltage, and ease of use. The search did not take long, as we were able to find 5V nominal solar panels that were 130x150 (mm2) in area. These solar panels are relatively small, only capable of outputting a

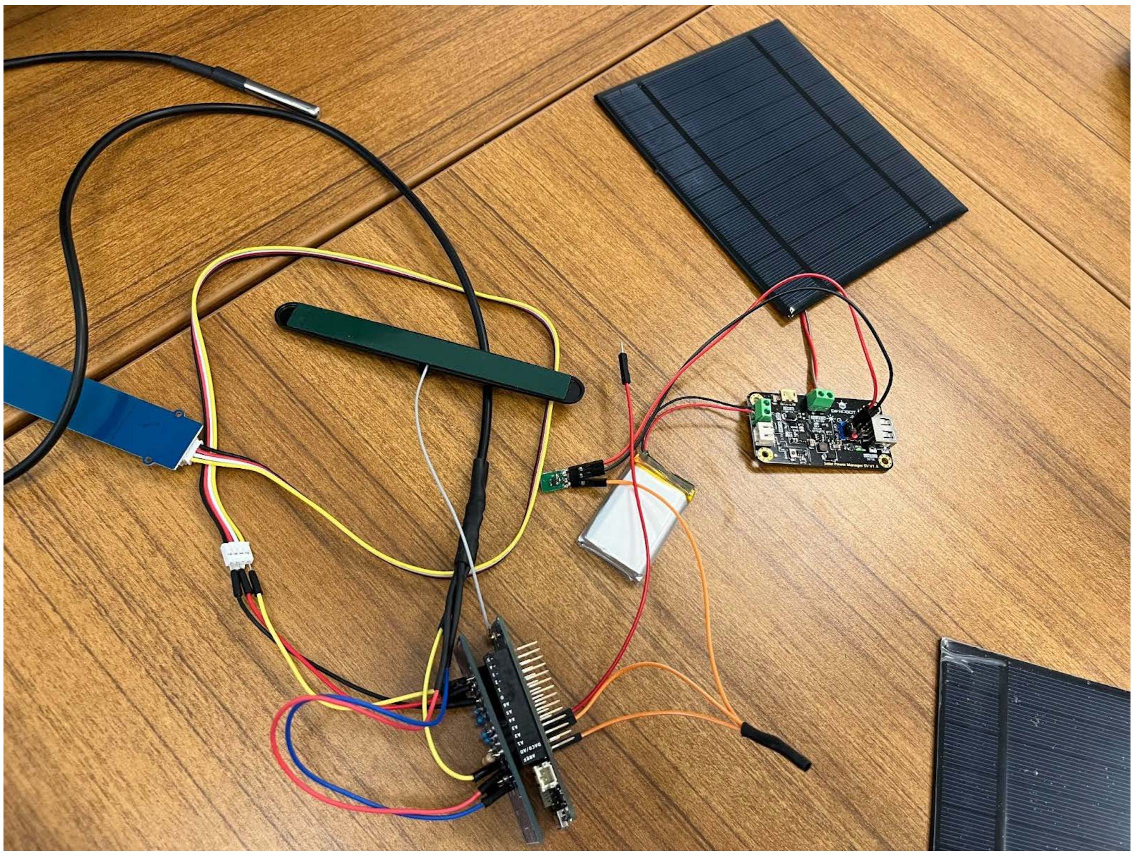


Fig. 3.6: Arduino Complete Build with Solar Panels Attached

around 2.5 W, but this is more than enough to power a system drawing micro-amp levels at 3.3V most of the time. While there were concerns of what will happen to the system in sleep mode, we will only spike to a maximum current draw of 27 mA while the system is operational which comes to a total of 89.1 mW (or 0.089 W).

The benefit behind using this size solar panel is that it will produce more than enough power to run our system without help of the battery even during sub-optimal lighting conditions such as cloudy or rainy days. This will also drastically increase the system's longevity because the lithium battery will go through less discharge and charge cycles over time. These cycles are what causes batteries to degrade over time, and thus by limiting these cycles, our system will last longer.

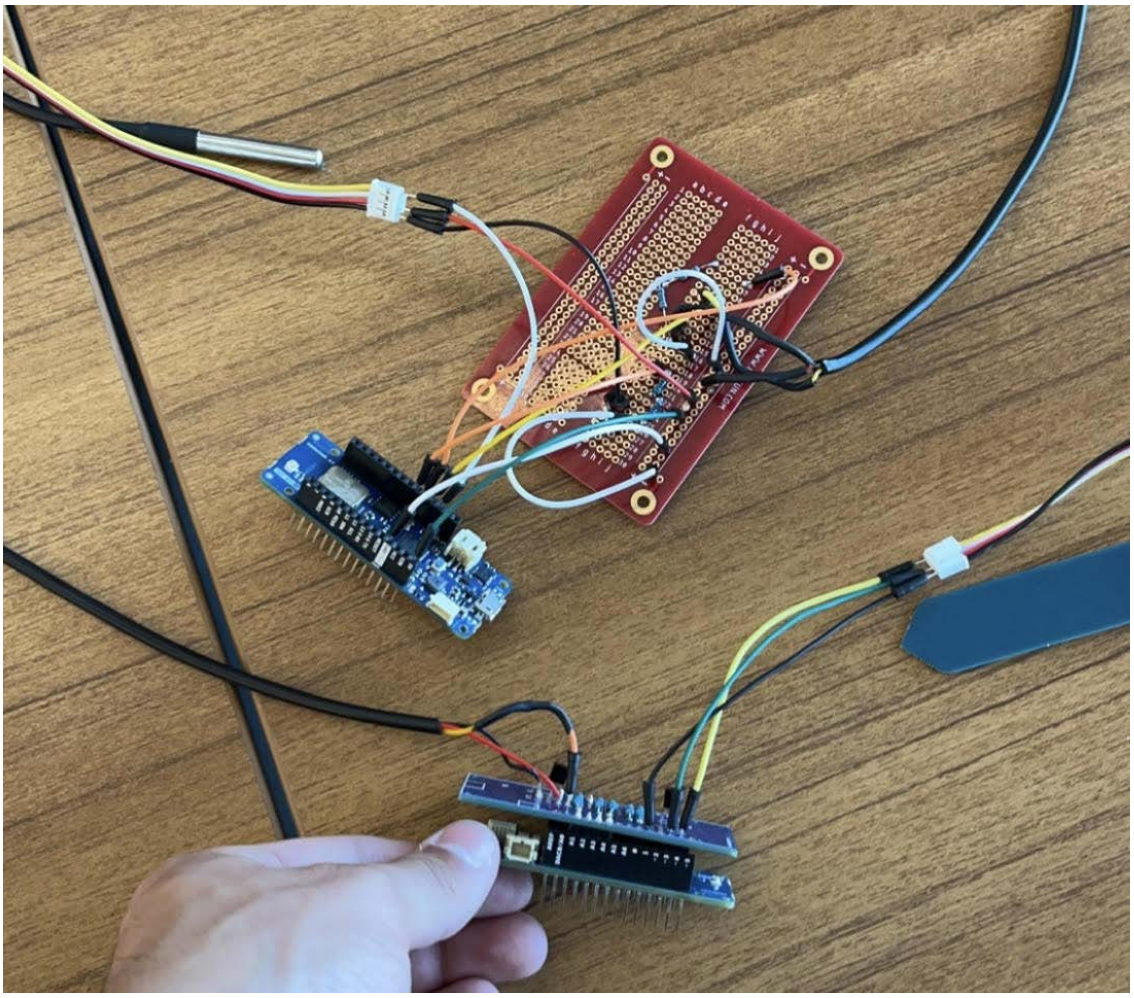


Fig. 3.7: Arduino Breadboard vs PCB Build Comparison

3.8 PCBs

Prototypes can only get a product so far, so once we were able to finalize the overall system schematic using breadboards, our team is able to design a PCB capable of integrating all of our components safely onto the Arduino system. For ease of assembly, our custom PCB is designed in an Arduino shield format capable of inserting directly onto the MKRWAN 1310 board using mount headers [21]. Apart from comfortably implementing all of our sensors and necessary microcontroller pins, our PCB shield also implemented transistors to be used as switches that disconnect power from the sensors. This creates an open circuit between the GND pin of each sensor and the ground on the MKRWAN 1310 whenever the base of the transistor is not getting an input current from the microcontroller's general purpose output pin; this creation of an open circuit is incredibly useful for lowering power consumption while in 'sleep mode' because no current will leak through the sensor. This is a serious consideration because the VCC pins of the sensors are directly connected to the power pin of the Arduino board, and therefore would constantly draw current otherwise.

Figure 3.5 above shows the completed Arduino build complete with solar panel, solar power manager, battery, and temperature and moisture sensors. Figure 3.6 below compares a previous Arduino breadboard build with the the finalized Arduino PCB build with soldered connections and less clutter.

3.9 Sleep Mode

While our soil quality measurement system is not actively gathering data and sending it via LoRa communication, it is in a state called 'sleep mode,' a state where lots of system functionalities are shut off to avoid consuming power. With an overall goal of sub 50 micro-amperes while in the sleep mode state, we needed to take into consideration many aspects of the microcontroller: how long we can be in true sleep mode, what available libraries there are, and how we can interface existing code to fit our needs [21].

The first thing to note about our sleep mode is that we used an internal CPU timer to count up to the maximum count of eight seconds that wakes the CPU up after those 8 seconds have passed. This 8 second limitation is due to the limited memory used for the counter when keeping track of how long the CPU has been in low-power mode for. To get around this issue, our team used a simple for-loop, allowing the microcontroller to shut off all functions that were not necessary for 8 seconds, wake up for a fraction of a millisecond, increase a variable until the desired amount of time is reached, and then go back to sleep if said variable has not reached the end of the loop specification. Once the loop is finished, the code allows the microcontroller to fully wake up to gather and transmit data via LoRa communication. Cnce this transmission is done, the while loop that drives the microcontroller will trigger the sleep mode for-loop once again.

Luckily for our team, Arduino itself has its official ArduinoLowPower library open sourced. Using this library, our team is able to get a solid sleep mode function that detaches the USB connection in order to stop any current leaks this may have caused, shuts off power to all unused timers, adc's, and serial communication ports that are not needed to wake the board up, and then fully wakes up after the specified time is up.

The current draw of our system does spike for a small fraction of a second to increment the variable keeping track of sleep time, but that can be disregarded as negligible given that for a full 8 seconds, we get a low 26 micro-amps of current being drawn. Because of this low current, we were able to estimate the life of our system to be 4.83 years using DigiKey's estimation tool [8]. The general equation for battery life is battery capacity over load current, but this equation does not take our solar system into account. Therefore we can predict that our system's battery lifetime will be much closer to three times DigiKey's predicted amount.

Figure 3.5 below shows the final Arduino current measurement achieved by putting the Arduino board into deep sleep.

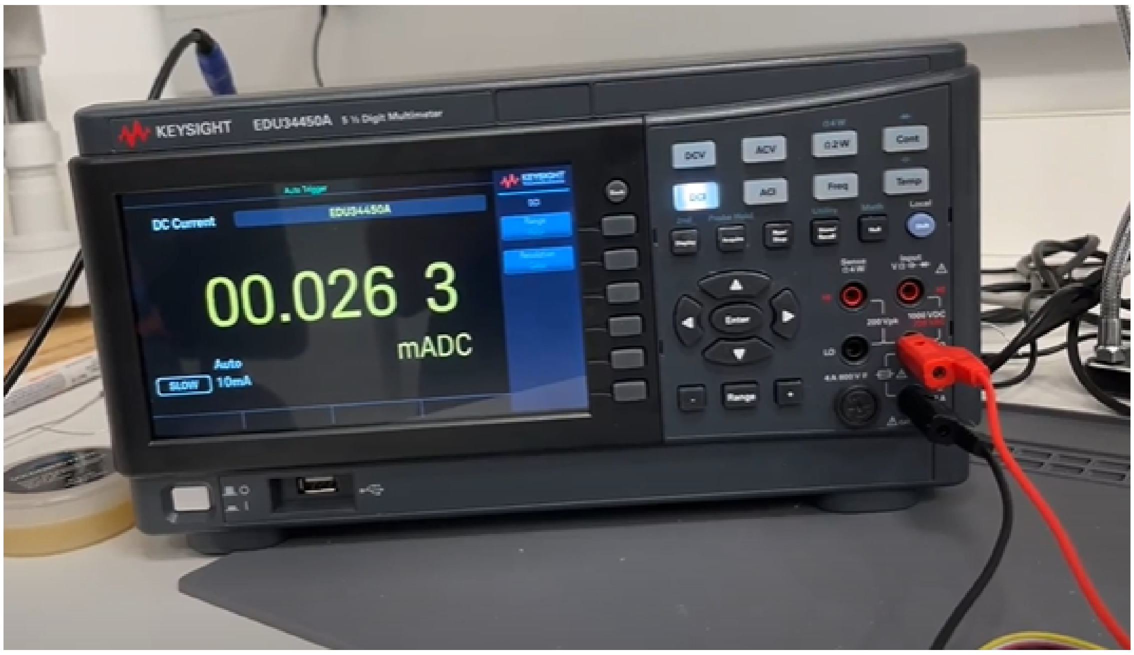


Fig. 3.8: Arduino Sleep Current Measured on Digital Multimeter

Chapter 4
Evaluation

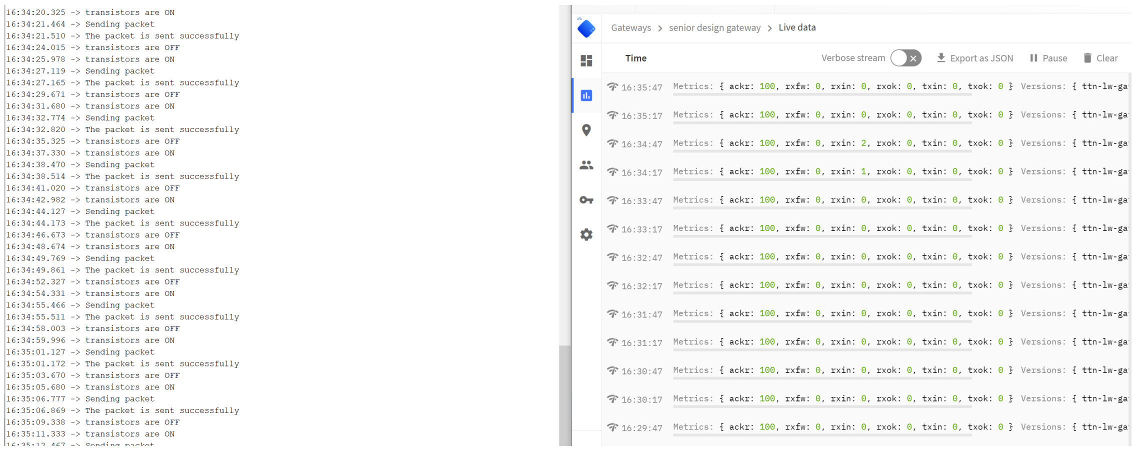
In this chapter we discuss our system evaluation, from sensor and current measurements to packet transfer between components.

4.1 Verifying the Complete Data Path

As the planned function of our system is to convey sensor data collected from end nodes in the field to a gateway module back on the farm, the essential test of our system is to verify the completed communication between the Arduino end nodes and our LoRa gateway build.

As referenced earlier, each node consisted of an Arduino MRKWAN 1310 with its included antenna, a Grove Capacitance Moisture sensor, a DS18B20 Temperature sensor, and a custom PCB shield. Our testing involved three of these end nodes, running separate code for sending and receiving peer-to-peer packets using their integrated LoRa modules. For testing purposes, these end nodes were connected to external computers in order to compare moisture and temperature readings and confirm packets were being sent and received.

Our LoRa gateway build consists of a Raspberry Pi 4 and a RAK2245 Pi Hat with its included antenna, running on RAK developer firmware with packet forwarding software. For testing purposes, this build is connected to a monitor and keyboard.



```
16:34:20.325 -> transistors are ON
16:34:21.464 -> Sending packet
16:34:21.510 -> The packet is sent successfully
16:34:24.015 -> transistors are OFF
16:34:25.978 -> transistors are ON
16:34:27.119 -> Sending packet
16:34:27.165 -> The packet is sent successfully
16:34:29.671 -> transistors are OFF
16:34:31.680 -> transistors are ON
16:34:32.774 -> Sending packet
16:34:32.820 -> The packet is sent successfully
16:34:35.325 -> transistors are OFF
16:34:37.330 -> transistors are ON
16:34:38.470 -> Sending packet
16:34:38.514 -> The packet is sent successfully
16:34:41.020 -> transistors are OFF
16:34:42.982 -> transistors are ON
16:34:44.127 -> Sending packet
16:34:44.173 -> The packet is sent successfully
16:34:46.673 -> transistors are OFF
16:34:48.674 -> transistors are ON
16:34:49.769 -> Sending packet
16:34:49.861 -> The packet is sent successfully
16:34:52.327 -> transistors are OFF
16:34:54.331 -> transistors are ON
16:34:55.466 -> Sending packet
16:34:55.511 -> The packet is sent successfully
16:34:58.003 -> transistors are OFF
16:34:59.996 -> transistors are ON
16:35:01.127 -> Sending packet
16:35:01.172 -> The packet is sent successfully
16:35:03.670 -> transistors are OFF
16:35:05.680 -> transistors are ON
16:35:06.777 -> Sending packet
16:35:06.869 -> The packet is sent successfully
16:35:09.338 -> transistors are OFF
16:35:11.333 -> transistors are ON
16:35:12.467 -> Sending packet
```

```
16:35:47  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:35:17  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:34:47  Metrics: { ackr: 100, rxfw: 0, rxin: 2, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:34:17  Metrics: { ackr: 100, rxfw: 0, rxin: 1, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:33:47  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:33:17  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:32:47  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:32:17  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:31:47  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:31:17  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:30:47  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:30:17  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
16:29:47  Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 }  Versions: { ttn-lw-ga
```

Fig. 4.1: Example of packets being sent but not received

Our initial test is to verify the communication between the end nodes and our LoRa gateway build. For this test, we register our LoRa gateway to The Things Network and run the developer packet forwarding software on the Raspberry Pi 4 LoRa gateway build. On the Arduino end node, we use the RadioHead Packet Radio Arduino library to set the frequency and spread factor for the Arduino LoRa module to values that the LoRa gateway module can receive. However, during this phase, we experienced several technical issues with receiving and processing the LoRa packets forwarded to The Things Network. While our packets were successfully being forwarded to The Things Network, they were showing up with inconsistent amounts of information, often missing the entire raw data segment of the payload (Fig. 4.1). This missing data coupled with inconsistencies in the amount of time it took for the payload to be registered on The Things Network,Äôs live data stream made it difficult to connect which packets being sent were the ones being received.

Following our issues testing the complete communication path from the end nodes to the data being displayed, we split up the transfer process into segments to evaluate the success of each segment.

Our main overarching test and goal is still as follows: to transfer data from the end node integrated sensors to be displayed graphically through ThingSpeak, hosted on the ThingSpeak website. This involves the data first being collected by the sensors. Next it is sent in a LoRa packet by the Arduino microcontroller to the LoRa gateway build [21]. Finally, it is forwarded to The Things Network servers where the packet is published to ThingSpeak and decoded by an integrated API payload decoder. For clarity and evaluation, we split this data transfer process into four stages: data collection and packet sending from each end node, packet reception by the LoRa gateway, packet forwarding to The Things Network, and publishing to ThingSpeak.

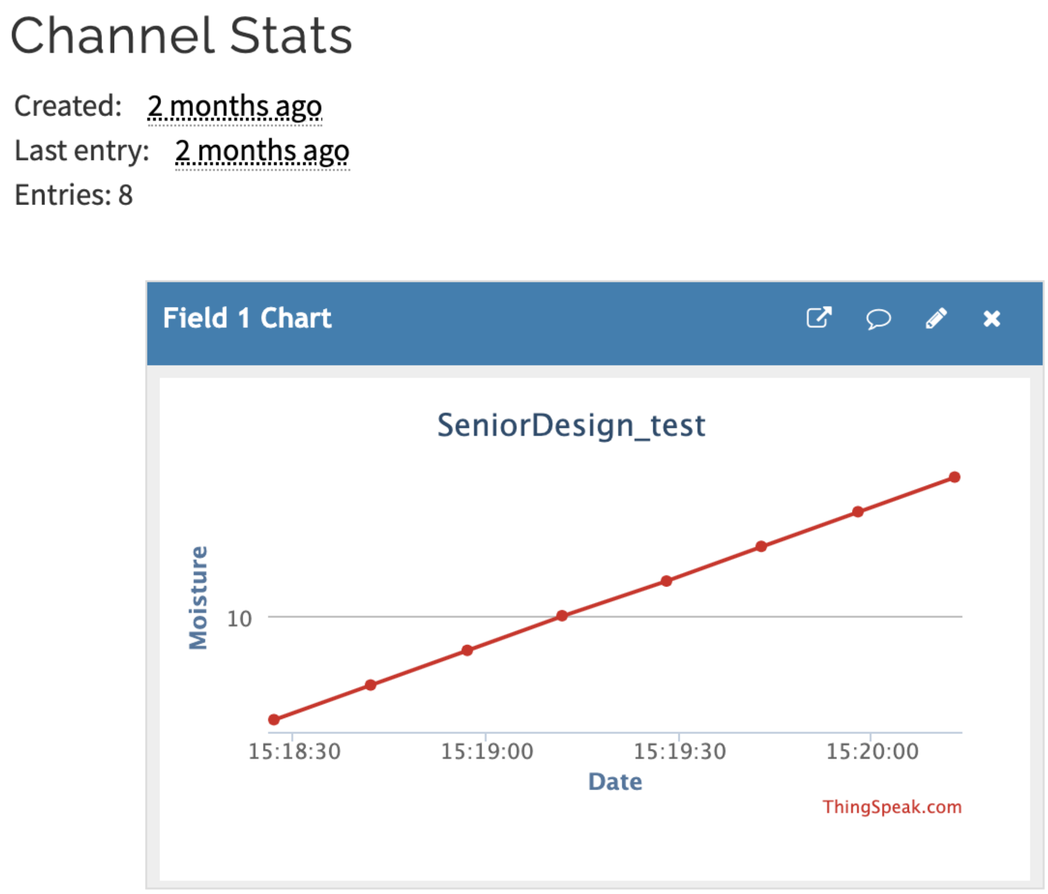
4.2 Verifying Peer-to-Peer LoRa Transmissions

Our first evaluation of our system is verifying the peer-to-peer communication between our individual end device nodes.

With a peer-to-peer system between Arduino end nodes, we are able to send LoRa packets with a perfect reception rate, although only tested over short distances. These LoRa packets are also accurate in conveying the measured moisture and temperature readings between end nodes. While we did not plan to utilize peer-to-peer communication between end nodes in our final model, this evaluation step is important in confirming that the end nodes can send the necessary data both accurately and consistently in the manner that we want using their integrated LoRa modules.

4.3 Verifying ThingSpeak Publishing

Next, we test the capability of our gateway to connect to ThingSpeak in order to display our data on the ThingSpeak data analytics platform. As the gateway is not


Channel Stats

Created: 2 months ago

Last entry: 2 months ago

Entries: 8

Fig. 4.2: ThingSpeak Graph Published with Dummy Data Payload

able to consistently receive packets from the end nodes, we test this component of the system by creating a short program to send dummy temperature and moisture data to ThingSpeak using RESTful API and displayed that data in graphical form as shown in figure 4.1.

These two evaluations isolate the issues of our communication system down to the inconsistency of the LoRa gateway in receiving LoRa packets from the end nodes. We speculate that this inconsistency is due to differences in sending and receiving frequency and spreading factor, with possible solutions addressed in the future work section.

4.4 Current Measurements

All current measurements were done using the multi-meters available in SCU's soldering lab. The equipment there is capable of accurately measuring in the micro-amp range, and are fairly new pieces of equipment with little risk of malfunction.

Initially all of our measurements were done via the USB port from a laptop to the board by splicing the V+ wire in the USB cable and putting an ammeter in series with both ends. This led to the team discovering that we needed to power our boards through the VCC pin rather than the USB input due to the high power consumption associated with using the USB connector.

After this discovery, it is much easier to measure power consumption because we can safely connect the ammeter in series with the output pin of the LDO on our power supply and the VCC pin of the Arduino board, or alternatively we can connect the ammeter to the ground of the power supply and the ground of the Arduino board to collect all of the current being drawn throughout the system.

4.5 Data Accuracy and Range Performance

The sensor data collected is always very accurate for the temperature sensor, but we run into issues using the capacitive moisture sensor due to the nature of its analog signal output. Data gathering periods are all relatively short, with each session being one hour long, but our data gathering periods are stretched out throughout the spring quarter.

In room temperature soil inside of Santa Clara University's Sobrato Center of Discovery and Innovation (SCDI), the data received from the transmitter was often within the range between 73.8 degrees Fahrenheit and 73.96 degrees Fahrenheit.

Luckily, SCDI has reliable climate control, and so we can verify the accuracy of our temperature data.

Given the nature of the capacitive sensor, the analog signal it produces is inversely proportional to the actual moisture level of the soil enveloping the sensor. This is relatively easy to take into account, but there is a complication with using this sensor. The data sheet for the NE555 chip shows a non-linear IV curve under every condition, and depending on temperature as well. We chose to approximate these curves as linear for simplicity; however, when obtaining actual data, this causes lots of inaccuracy. The data gathered is displayed as a percentage of moisture level, and when completely submerged in water, the sensor reads 100%; this allows us to be sure that any reading below unity is, at the very least, somewhat accurate.

Despite our antenna being a cheap 5 dollar part, at the time of writing this thesis, we discovered this component may be unusable for our design goals. When testing the system within SCDI, one end of the fourth floor to the other, we are able to get packages from transmitter to receiver; however, upon further researching our antenna, we find a complication. According to the data sheet of the X000016, the antenna provided along with the Arduino board, it has a dBi, or decibel relative to isotrope, of less than 1 dBi [20]. What this means is that the antenna is capable of 360 degree radiation when transmitting, but the range is very short relative to what LoRa systems are generally capable of. We have yet to test this range issue, so we leave this to other teams who decide to pick up the mantle sometime in the future.

4.6 Ethical Considerations

The main ethical concern with our design is the risk of metal corrosion into the soil. If this is a major problem with our design, then it takes away the whole point of a system meant to help monitor soil health. The precautions taken with our design is to create a casing for the probes to house the wires away from ground contact. The sensors used for this project were also chosen with anti-corrosion materials in mind, to not only mitigate the probe corrosion, but also to lower the maintenance costs of the overall system [18,19]. Due to these considerations, our system is extremely resilient to corrosion and will most definitely score highly in the ethics department.

A positive ethical gain from this project is the improvement in the lives of the user, as well as the environment. Our target consumers are local mid to large-sized farmers, and by creating a wireless long range system, our target demographic will have access to dire soil quality information even in rural areas; this system allows them to pinpoint problem areas of their crops without using too much time and resources. Additionally, by using a hybrid power delivery system that incorporates solar panels, it reduces the amount of battery degradation in the system. While lithium based batteries are relatively environmentally friendly as opposed to lead-acid batteries, they tend to end up in landfills where they leak their hazardous components into the soil and groundwater. Our hybrid power delivery system ensures that our batteries will not need to be replaced for over a decade; this combined with an educated user properly disposing of their batteries will nearly eliminate the ethical risk of using a lithium battery to power our system.

The hope here is that by providing the consumer readable, real-time, information on their crop health and management, it will encourage them to be more conscientious about the environment, especially as California's agricultural department uses a lot of water for their food exports. On a more practical level, it will also save the consumer money in the long run.

Chapter 5
Future Work

In this chapter we present potential future works to extend the proposed system. We will list potential future improvements for this project, based on importance.

The first step that we would like to improve on is the overall costs of the system. Many of the components we have in our final design were not the ones we had originally sought out to buy. This is mostly due to supply chain issues, and lack of time to be able to have the parts we wanted shipped to us. Because of this, we had to go for more expensive modules had more hardware outfitted to it than we needed, such as the usage of the RAK2245 as opposed to a cheaper single channel module as our gateway module. While the extra hardware may be helpful if this project goes into a larger scale, the scope of our work and what we were aiming for is much smaller than that, and thus it raised the costs of our system higher than we wanted it to be.

The next potential future improvement would be to improving the node itself. This includes its physical components and the communication protocol itself. Since our project is a proof of concept, a lot of our testing happened in ideal conditions, in such a way that the hardware is never in any real danger of damage. In future iterations, we would like to design some sort of weatherproof housing for the nodes, so that it could actually be deployed in fields long term without getting damage from the elements. We would also like to improve on the communication protocol, with the obvious being that it would be fixed so that the entire system could be up and running. However, how we designed it is so that every node is sending their data directly to the central gateway. Future iterations may look into going into creating a mesh network, so that it could cover dead zones in the area and have last connection failures.

Finally, there could be improvements on the way soil quality data is displayed on the web application. In its current state, the information gathered from the sensors are sent to the gateway, which publishes that data onto ThingSpeak's website which uses MATLAB to configure it into a readable graph. Despite the ease in publication, we would like to be able to create a dedicated web server and database to host this information instead. It would allow the user more privacy to their sensor data, as well as give the team more freedom in how the data is stored and displayed.

Chapter 6
Conclusion

California is one of the country's largest provider of food exports, making its agricultural department very important to the livelihoods of many people in the area. However, with the frequent droughts plaguing the state in the last couple of decades, there becomes a large concern over how water is distributed and used, especially as around 70% of the world's water goes into agriculture, but 40% of that water is lost to poor irrigation systems [7]. Therefore, creating a system that would help farmers keep track of water management, as well as the health of their soil, while being affordable and easy to use, will help generate healthier and efficient habits in their day to day routine to combat these water loss numbers.

In order to create our proposed system in a way that is different from commercial systems, we gathered information on low power parts and wireless and non Wi-Fi reliant communication protocols. In doing so, we found hardware components that fit the low power threshold that we discussed during the planning stages of the project, and decided on basing the communication protocols around LoRa, which is free and allows us to send data over long distance with a low power consumption and without Wi-Fi.

Many lessons were learned during the design process, especially in terms of being flexible and communicative. Many issues with parts or software came in later iterative stages of our project, and the design of the system frequently had to change as new information and parts were discovered and tested. Having an open channel of communication between ourselves became very important as a result, since many parts overlapped and affected the work between the two sub-teams.

Despite many issues that cropped up during the project timeline, such as loss of a member, supply chain issues, and the global pandemic, the team is still able to design and test a majority of the system we had planned at the beginning of the project. Although we were unable to have a fully working system by the end of the year, many of the parts we did finish can work independently of each other, which bodes well for future iterations of the project. We hope that our work and research will help future teams looking to continue this project, as it will help a glowing environmental issue that is a very pressing and local problem, and that it will help inspire others to not only create similar systems, but to be more aware of the earth and natural resources around them that help us survive.

Chapter 7

References

[1] D. Ilie-Ablachim, G. C. PfÉtru, I. Florea and D. Rosner, ”Monitoring device for culture substrate growth parameters for precision agriculture: Acronym: MoniSen,” 2016 15th RoEduNet Conference: Networking in Education and Research, 2016, pp. 1-7, doi: 10.1109/RoEduNet.2016.7753237.

[2] H. Fitriawan, M. Susanto, A. S. Arifin, D. Mausa and A. Trisanto, ”ZigBee based wireless sensor networks and performance analysis in various environments,” 2017 15th International Conference on Quality in Research (QiR) : International Symposium on Electrical and Computer Engineering, 2017, pp. 272-275, doi: 10.1109/QIR.2017.8168495.

[3] Sakthipriya, N. ”An Effective Method for Crop Monitoring Using Wireless Sensor Network.” Middle-East Journal of Scientific Research, vol. 20, no. 9, 20 Sept. 2014, pp. 1127,Äì1132., https://doi.org/10.5829/idosi.mejsr.2014.20.09.114152.

[4] Abbas, et al. ”How to Measure USB Current and Voltage [Practical Solution].” Yaman Electronics, 25 Oct. 2021, https://www.yamanelectronics.com/measure-usb-voltage-current/.

[5] Groot, Tai, and Benham Dezfouli. ”IEEE GLOBAL HUMANITARIAN TECHNOLOGY CONFERENCE (GHTC), 2020.” Flomosys: A Flood Monitoring System.

[6] Petrariu, Adrian I., et al. ”Hybrid Power Management System for LORA Communication Using Renewable Energy.” IEEE Internet of Things Journal, vol. 8, no. 10, 2021, pp. 8423,Äì8436., https://doi.org/10.1109/jiot.2020.3046324.

[7] Balsom, Paul. ”Water Usage in the Agricultural Industry.” High Tide Technologies, 29 Sept. 2020, https://htt.io/water-usage-in-the-agricultural-industry/.

[8] ”Battery Life Calculator.” Battery Life Calculator | DigiKey Electronics, https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life.

[9] Moyer, Karyn. ”California Is Largest Food Producer in the U.S.” AgHires Blog, https://blog.aghires.com/california-largest-food-producer-u-s/.

[10] ”Current U.S. Drought Monitor Conditions for California.” Drought.gov, https://www.drought.gov/states/california.

[11] Froehlich, Andrew. ”What Is the Difference between Lora and Lorawan?” SearchNetworking, TechTarget, 4 Mar. 2020, https://www.techtarget.com/searchnet working/answer/What-is-the-difference-between-LoRa-and-LoRaWAN.

[12] ”Rachio 3 Smart Sprinkler Controller.” Rachio, https://rachio.com/rachio-3/.

[13] ”Rainmachine Touch HD - Forecast Smart Wi-Fi Irrigation Controller.” RainMachine Touch HD - Forecast Smart Wi-Fi Irrigation Controllers, https://www.rainmachine.com/products/rainmachine-touch-hd.html.

[14] ”B-Hyve XR Smart Indoor/Outdoor Sprinkler Timer.” OrbitOnline, https://w ww.orbitonline.com/products/b-hyve-xr-smart-indoor-outdoor-sprinkler-timer.

[15] ”ESP4ME3 - Indoor/Outdoor 120V Irrigation Controller (LNK WIFI Compatible).” Rain Bird, https://store.rainbird.com/esp4me3-indoor-outdoor- 120v-irrigation-controller-lnk-wifi-compatible.html.

[16] ”Blossom 7 Zone Smart Irrigation Controller, Wi-Fi Enabled with Real Time Weather Optimization 0080-AWICD.” The Home Depot, https://www.home depot.com/p/Blossom-7-Zone-Smart-Irrigation-Controller-WI-FI-Enabled-with-Real-Time-Weather-Optimization-0080-AWICD/301929169.

[17] ”Netro - Spark-8, Smart Watering Controller.” Netro, https://www.netro home.com/en/shop/products/35.

[18] Industries, A. Waterproof 1-wire DS18B20 compatible digital temperature

sensor. adafruit industries blog RSS. https://www.adafruit.com/product/381.

[19] Zuo, B. Grove - capacitive moisture sensor (corrosion-resistant). seeedstudio. https://wiki.seeedstudio.com/Grove-Capacitive_ Moisture_Sensor-Corrosion-Resistant/

[20] Google. Adobe Acrobat: PDF Edit, convert, sign tools. Google. https://chrome.google.com/webstore/detail/adobe-acrobat-pdf-edit-co

/efaidnbmnnnibpcajpcglclefindmkaj?hl=en-GB

[21] Arduino Mkr Wan 1310. Arduino Online Shop. https://store-usa.arduino.cc /products/arduino-mkr-wan-1310

[22] "Wisgate Edge Pro: Rak7289: Gateway for Lorawan." RAKwireless Store, https://store.rakwireless.com/products/wisgate-edge-pro-rak7289?variant=

40743451984070.

[23] "SENSECAP M1 Lorawan Indoor Gateway - US915." Seeed Studio, 18 Apr. 2022, https://www.seeedstudio.com/SenseCAP-M1-LoRaWAN-Indoor-Gateway-US915-p-5023.html.

[24] "CISCO IXM Gateway Common Packet Forwarder - License - IXM-Lorawan-CPF." Hummingbird Networks, https://www.hummingbirdnetworks.com/cisco-ixm-gateway-common-packet-forwarder-license-ixm-lorawan-cpf.

[25] DFROBOT 900ma MPPT Solar Panel Controller - Amazon.com. https://www.amazon.com/DFROBOT-900mA-Solar-Panel-Controller/dp/B07MML4YJV

[26] NCP161ASN330T1G Onsemi: Mouser. Mouser Electronics. https://www.mouser.com/ProductDetail/onsemi/NCP161ASN330T1G?qs=vLWxofP3

U2ze5zA%252BHmAd3w%3D%3D

[27] Gebhart, Andrew. "Edyn Garden Sensor Review: Paradise Lost: The Edyn Garden Sensor Falls Well Short of Perfection." CNET, https://www.cnet.com/reviews/edyn-garden-sensor-review/.

[28] Gebhart, Andrew. "OSO Technologies PlantLink Review: This Simple Sensor Will Give Your Plant a Voice." CNET, https://www.cnet.com/reviews/oso-technologies-plantlink-review/.

[29] Gebhart, Andrew. "Parrot Flower Power Review: This Connected Garden Sensor Has Trouble with Communication." CNET, https://www.cnet.com/reviews/parrot-flower-power-review/.

Chapter 8
Project Procedures

This section lists off the procedures followed to create this project, so that future iterations can follow and improve upon them.

8.1 Guide to Connecting Gateway and Nodes to The Things Network

1. Attaching the Raspberry Pi to the Pi Hat
    (a) Align Raspberry Pi GPIO pins with the Raspberry Pi connector on the RAK2245 Pi Hat and press together
    (b) Tighten screws on Pi Hat corners to secure Pi Hat to Pi connection
    (c) Attach antenna to LoRa pin on Pi Hat
2. Installing firmware with RAK2245 specific packet forwarding software
    (a) Connect micro SD card to a computer with internet access
    (b) Install micro SD imaging software such as https://www.balena.io/etcher/ onto computer
    (c) Download RAK2245 firmware from the RAK documentation center https://docs.rakwireless.com/Product-Categories/WisLink/RAK2245-Pi-HAT/Overview/product-description

            i. Other RAK model firmware can be found at
               https://github.com/RAKWireless/rak_common_for_gateway

   (d) Flash OS image to micro SD card: Select Image -> Select Drive -> Flash

   (e) Disconnect micro SD card from the computer and plug into micro SD card slot on the Raspberry Pi

3. Configure the LoRa gateway to connect to The Things Network

   (a) Configure the LoRa gateway to connect to a Wi-Fi network

   (b) Power on the Raspberry Pi. If the Raspberry Pi asks for a username and password, the defaults are "pi" and "raspberry" respectively

   (c) Enter "sudo gateway-config" to access the Configuration Options menu

   (d) Set gateway to Client Mode, Add New SSID for Client, and set Wi-Fi Country

   (e) The gateway will work in Wi-Fi Client mode after rebooting

4. Configure the LoRa gateway to connect to The Things Network

   (a) Type "gateway-version" on the Raspberry Pi and record the gateway EUI

   (b) Create a The Things Network account at https://www.thethingsnetwork.org/

   (c) Click your username in the top right corner and select "Console" from the dropdown menu -> Select your The Things Network Cluster based on region -> Select "Gateways" -> Click "Add gateway"

   (d) Enter information for your gateway including the recorded gateway EUI

   (e) On the Raspberry Pi type "sudo gateway-config" -> Select Edit packet-forwarder config -> Replace Gateway Server address with the address listed on The Things Network

8.2 Guide to Modifying the MKR WAN 1310

In this section, we will discuss how to be able to achieve a sleep more current consumption level below what Arduino themselves had officially specified as the lowest current draw possible for the MKRWAN 1310 board [21].

The first modification done is removing the on-board LEDs, especially the power LED that is always drawing current while the board is ON. These can be removed using a heat gun or a soldering iron. This part is relatively simple; make sure you do not melt away the plastic headers on board the Arduino board. This modification is okay to do because the LEDs are not in series with anything essential to this project. Next, we want to find the linear dropout regulator on board the Arduino. It is really easy to spot, look for the IC with the SOT-23-5 packaging, it will be an SMT style IC. We definitely destroyed a few boards when modifying them, so be sure to contact a professional or be super careful not to burn out a PCB trace when attempting this yourself. The LDO already on board the Arduino is the AP2112K-3.3TRG1 and it has a quiescent current of 55 micro-amps. This is a huge problem because our overall goal already is being sub-50 micro-amps while in sleep mode. Due to the semiconductor shortage, this specific IC may be unavailable, but the things to look out for are listed here: packaging has to match, 3.3V output voltage, and a lower quiescent current. Our team is able to find a lower quiescent current LDO IC on Mouser.com by onsemi: NCP161ASN330T1G [26]. This IC has a quiescent current of 18 micro-amps, well below the original 55 micro-amps.

The last thing that needed to be modified is cutting the solder jumper on the back of the board. This is essential to create a physical barrier, or open circuit, between the 5V input USB and the rest of the circuit, which will ensure the leaky USB connector will not dissipate current. The only downside to this is that the 5V pin on the Arduino board will no longer be functional, and a quick fix to this is to route the PCB shield to either take in the external 3.3V input via a jumper cable to power the sensors on the sensor node. That would have been a great solution, but our team did not have enough time at the end of our project,Äôs timeline to redesign the custom PCB shield to incorporate this, so what we ended up doing is connecting the regulated 3.3V output onto the 5V pin on the Arduino.

1. Remove unnecessary LEDs
2. Identify and remove original linear dropout regulator
3. Search for a proper replacement for the LDO with a lower quiescent current. This step may be difficult due to semiconductor shortages
4. Place, or find a professional to help you place, the SMD/SMT LDO IC onto the board safely.
5. Cut the solder jumper on the backside of the arduino board.

Chapter 9
Appendices

9.1 Appendix A: Software Code for the Nodes

```
    //for lora
//include <Console.h>
include <SPI.h>
include <RH_RF95.h>
include <LoRa.h>


    //for temp sensor and low power mode
include <ArduinoLowPower.h>
include <OneWire.h>
include <DallasTemperature.h>


    //library addendum to dsostrf.h
include <stdlib.h>
include <avr/dtostrf.h>


    //Variables
//LoRaModem LoRa;
```

```
RH_RF95 rf95;

    int sensorPin = A1;
int sensorValue = 0.0;
int transistor1 = A3;
int transistor2 = A4;
boolean on_off = HIGH;
int oneWireBus = A2;
char *node_id;
//ends all function calls
char temp_end[8] = ””;
char ms_end[8] = ””;
//creates bits for data to be sent
uint8_t datasend[144]; //4 variables with 36 bits each
float frequency = 915.0; //freq set to the US one

    OneWire oneWire(oneWireBus);
DallasTemperature sensors(oneWire);

    define sensorMin 0
define sensorMax 1023
define valueMin 0
define valueMax 100

    float temp = 0.0;
```

void setup()
Serial.begin(9600);
pinMode(sensorPin,INPUT);
pinMode(transistor1,OUTPUT);
digitalWrite(transistor1,LOW);

pinMode(LED$_BUILTIN, OUTPUT$);
$digitalWrite(LED_BUILTIN, LOW$);
$sensors.begin()$;

if(!rf95.init())
Serial.println(”Initialization Failed”);
rf95.setFrequency(frequency);
rf95.setTxPower(13);
// rf95.setSyncWord(0x34);

if (!LoRa.begin(915E6))
Serial.println(”Starting LoRa failed!”);
while (1);

float readTemp()
float temp = 0.0;

```
int oneWireBus = A2;
OneWire oneWire(oneWireBus);
DallasTemperature sensors(oneWire);
pinMode(LED_BUILTIN, OUTPUT);
digitalWrite(LED_BUILTIN, LOW);

    //sensors.begin();

    sensors.requestTemperatures();
temp = sensors.getTempCByIndex(0);
temp = sensors.getTempFByIndex(0);
return temp;

    float readMoisture()
int sensorPin = A0;
int sensorValue = 0.0;
sensorValue = analogRead(sensorPin);
// Convert it to a percentage
sensorValue = map(sensorValue, sensorMin, sensorMax, valueMin, valueMax);
return sensorValue;

// Prepares the data in a packet to be sent
void writeData()
char data[144] = ""; // Prepares data by setting all bits to NULL/0
// Sets the node ID
```

for(int i = 0; i < 144; i++)

data[i] = node$_id[i]$;

$//Callsinthedata$

$floattemp = readTemp()$;

$floatms = readMoisture()$;

// Stores the data by adding it onto a string

dtostrf(temp, 0, 1, temp$_end)$;

$dtostrf(ms, 0, 1, ms_end)$;

// Adds data until it hits the specified end and allocating specific fields

strcat(data,"field1=");

strcat(data,temp$_end)$;

$strcat(data, "field2 = ")$;

$strcat(data, ms_end)$;

$strcpy((char*)datasend, data)$;

void sendData()

// Creates packet in order to send data

LoRa.beginPacket();

LoRa.print((char *)datasend);

LoRa.endPacket();

Serial.println("The packet is sent successfully");

delay(360000); //delays six minutes before next sending

//rf95.send(datasend, sizeof(datasend));
//rf95.waitPacketSent();
//delay(400);

void loop()

on$_{o}ff = !on_{o}ff;$
$digitalWrite(transistor1, on_{o}ff);$
$digitalWrite(transistor2, on_{o}ff);$

if(on$_{o}ff == HIGH)$
$Serial.println("transistorsareON"); readTemp(); readMoisture();$

writeData();
sendData();

else

Serial.println("transistors are OFF");

delay(2000);