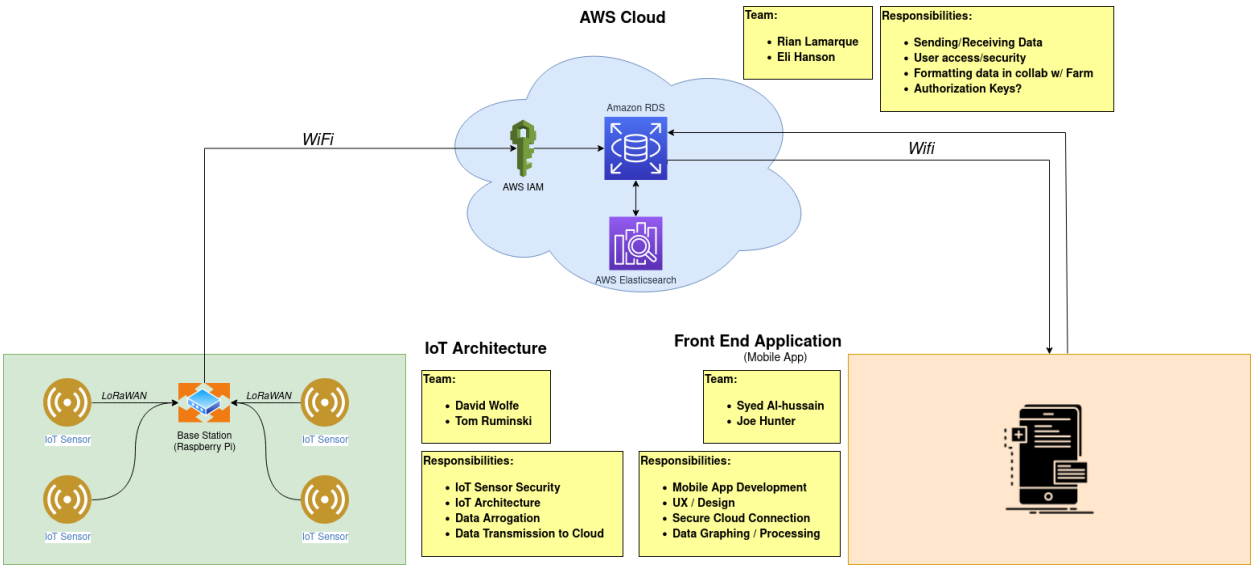
4.3 Proposed Design

4.3.1 Overview

Our project is focused on solving the problem of professional farmers and hobby gardeners needing to manually test, record, and chart IoT sensor data. The solution we will be working towards will include the farmers deploying IoT soil sensors and a raspberry pi base station to aggregate data. From the base station, data will be uploaded to Amazon Web Services for storage and processing. Users then access this data and information from our front end application. This front end app will give the user access to different charts and graphs to help the users track crop progress and health over a selected period of time. We will also allow specific users to run different intrusion detection algorithms on the data that has been collected. These algorithms will produce alerts for the user if there are any issues with the data collection, if there are any data values that are out of the ordinary, or if there has been an indicator of compromise from a cyber security perspective. Below is a more detailed breakdown of the individual components and technologies we plan on using for our first deployment.

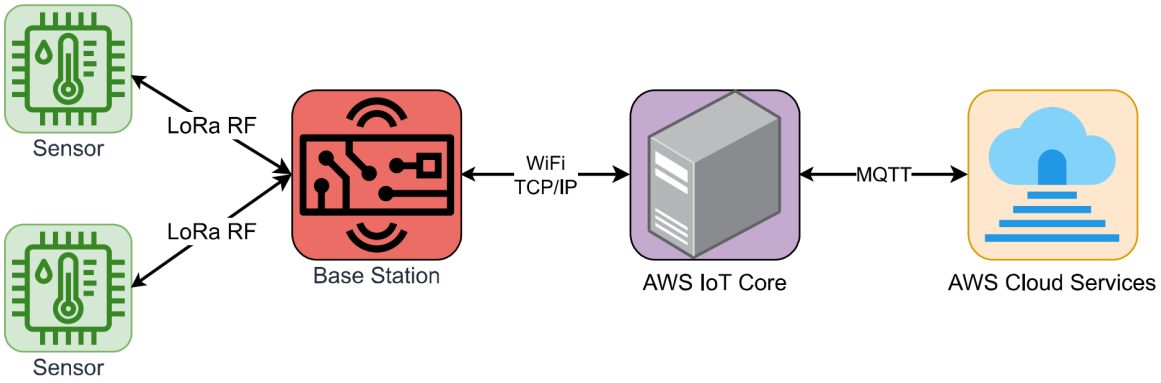
4.3.2 Detailed Design and Visual(s)

Our senior design system consists of three key components: IoT infrastructure, cloud storage & processing, and the front end application.



IoT Infrastructure

Starting with the IoT infrastructure, we have two sensor nodes connected to a base station in a star network topology. This base station will connect to a network server (AWS IoT Core) which will then interface with other AWS services utilized by our Cloud Development team. This system will be scalable by adding more base stations and nodes to the design.



Devices

We have purchased two SenseCAP S2104 sensors which collect soil temperature and moisture data. These sensors are LoRaWAN enabled and will broadcast data to the base station. The base station is a raspberry pi with a LoRaWAN receiver connected to it. The raspberry pi is WiFi enabled and will connect with AWS IoT Core via TCP/IP. AWS IoT Core is responsible for handling uplinks and downlinks from the sensors to the AWS Services the cloud team is using.

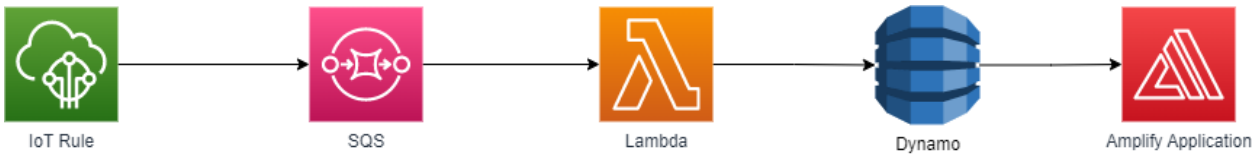
Included to the right is a sample of the data format being sent from the sensors to our base station. This data is then validated using the CRC-16/Kermit algorithm to ensure no data was corrupted between the sensors and base station.

01 0610 245E0000 01 0710 BCB10000 A3D9

Part	Value	Raw Data	Description
1	Soil Temperature	01 0610 245E0000	01 is the channel number. 0610 is 0x1006 (little-endian byte order), which is the measurement ID for soil temperature. 245E0000 is actually 0x00005E24, whose equivalent decimal value is 24100. Divide it by 1000, and you will get the actual measurement value for soil temperature as 24.1℃.
2	Soil Moisture	01 0710 BCB10000	01 is the channel number. 0710 is 0x1007 (little-endian byte order), which is the measurement ID for soil moisture. BCB10000 is actually 0x0000B1BC, whose equivalent decimal value is 45500. Divide it by 1000, and you will get the actual measurement value for soil moisture as 45.5%RH.
3	CRC	A3D9	The CRC verification part.

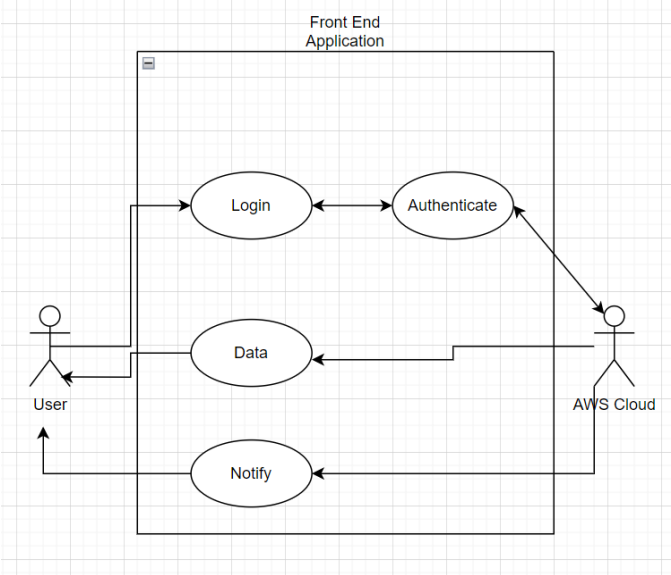
Cloud Storage & Processing

Currently, the cloud storage option uses dynamodb with the primary key of the farm name. The data travels from the base station to the IoT Rule, which passes the message through a Simple Queue Service, reaching a Lambda function to break down the message. The Lambda function breaks down the message per hex values and converts them to decimal values that are then pushed into the database. In the future, this process may include the AWS ELK machine learning stack to ensure the data is validated.



Front End Application

For our application we decided to use a Flutter application to build our application layer. After researching multiple different ways to develop, including making a decision matrix, we came to the conclusion that Flutter would be our best choice for development. To connect our app to Amazon Web Services we used AWS Amplify, a backend service that connects our frontend Flutter app to AWS. To connect to our backend data we will use REST api that will allow for easy and efficient communication. We will use multiple different Widget libraries to build our application, including AppBar, BottomNavigationBar, and Containers. These libraries will allow us to build a quality application that looks modern and performs efficiently. We will display information about our sensor on multiple different pages that we will design for our users. These pages will be pre-designed using Figma.



4.3.3 Functionality

Initial Setup

Our end users would purchase our solution as a kit with a basestaion and select sensors included. They would then place these sensors in different positions around their farm, garden, or home. Users will then download our front end application to initialize and position the sensors for monitoring. Users will be able to name all of the sensors they have placed and then mark their positions on a top down view of their land. From here users will have different options for how often they want updates and what data they want to see.

Daily Use

After the initial setup users will simply have an application on their phone, tablet, or computer that allows them to monitor the sensor's reading at any time of day from anywhere. Users will open the mobile application and be greeted with a list of the different sensors and locations they are monitoring. From there they will be able to tap into any specific sensor and get live readings of the sensor values as well as a graphical representation of the sensor readings from the last week. This time scale for historical data will be customizable by the user.

4.3.4 Areas of Concern and Development

The current design satisfies physical requirements by keeping a small profile in sensor distribution. Sensors can be sparsely placed due to the use of a LoRa RF, star topology rather than a mesh network. The sensors we purchased are also weather and temperature resistant with a rating between -40 and 185 degrees fahrenheit. This will be sufficient for a majority of climates, including Iowa.

Our primary concern for our user and client needs is security. Security is very important to both our client and our user. We know we can meet the main functionality of our product by delivering data from our sensors to user devices. Securing the data being transferred will be a little trickier. We must secure our data on all three levels of our product (Hardware, Cloud, User Interface), each of which will require unique security solutions.

IoT Devices

Currently we use CRC16/Kermit to ensure data integrity between our IoT sensors and the base station. We also employ asymmetric encryption between our base station and AWS to ensure data confidentiality when transmitting across the internet. Each sensor has a unique key and EUI that can be registered to the base station to help verify sensors. These sensor nodes are simple to install and farmers would not need a technician to add more sensors to their farm.

Cloud

To mitigate cloud security concerns we plan to implement AWS Identity and Access Management policies so that users will not have unlimited privileges and are able to see other users data. Also, we plan to use AWS Cognito to handle user authentication, this will help to make sure that only authorized users are able to access our cloud resources.

Front End

The front end will employ proper data sanitization to ensure users security. On top of only allowing a strict set of characters to be entered for any user input field user password and accounts details will be stored securely in AWS. This includes hashing passwords and transmitting all data over security, encrypted channels.

4.4 Technology Considerations

We have made key technology choices within each of our subteams. This includes the protocol stack between gateways and sensors; our cloud service; and front end application framework.

LoRA RF and LoRaWAN

At the perception level, Sensor-Base Station communication is performed through LoRaWAN and LoRa RF. The link layer and physical protocol have a key advantage over most other low power transfer options: range. Bluetooth Low Energy (BLE) and Zigbee are both popularly used in IoT applications for their low power usage. However, these protocol stacks are typically used in a mesh network configuration where long signal range can be overkill. Limited to a maximum of 100 meters, BLE and Zigbee are not suited for the sparse dispersion of sensor nodes on a moderately large farm. Enter LoRa RF. LoRA RF has similar low-power usage at a significantly longer range of 1km. LoRaWAN is the accompanying link layer to LoRa RF and our purchased sensor node and transceiver (for the base station) are already LoRaWAN enabled. In addition, LoRa devices cost about the same BLE or Zigbee making it financially viable for our project. In conclusion, LoRa enabled devices were chosen for their low-power usage, long range, and affordability.

AWS

There were three main options we thought about when choosing our cloud platform, AWS, Google Cloud, and Azure. We chose to go with AWS for a number of reasons. One of them was the fact that one of the members of the group is very familiar with AWS. Also, our client already had an AWS account so it would be simple to give us access to the platform. Our client also

mentioned the idea of using machine learning to implement an intrusion detection system and AWS has a service that supports that functionality called ELK. We briefly entertained the idea of using other cloud platforms but it quickly became clear that AWS was the best fit for our project due to previous experience and the services it offers.

Flutter

When first considering what we wanted to create for farmers, we considered multiple different options. At the start we debated whether we wanted a web application or a mobile application. After talking to farmers about their wants we decided that we would go with a mobile application. Once we decided on a mobile application, we had to figure out how we would create our mobile application. We looked at mobile frameworks that were compatible with AWS amplify. And after making a decision matrix we ended up going with a Flutter mobile application. We chose Flutter because we can simultaneously develop an Android and IOS mobile application.

4.5 Design Analysis

IoT Infrastructure

So far we've ordered parts for our IoT infrastructure and created a mock base station to feed simulated data to the cloud. Inside of this mock base station we've also added a function for calculating and validating the CRC values (using the CRC-16/KERMIT algorithm) associated with each packet of data received from the IoT sensors. This will allow us to ensure no data was modified between the sensor and the base station.

Cloud

In the cloud we've implemented an AWS Lambda function to take in that simulated data from the base station and parse it out into the proper json format according to the data sheets for our sensors. We've also set up a simple DynamoDB for storing our sensor reading, this may not be our final design. Additionally, we have another Lambda function that will pull data from our database to send to the user. However we have not been able to connect it to our frontend app yet.

Front End

Front end development has been focused on establishing a connection between the app and AWS. So far we have a rough outline of one screen of our application and a location to display data once it's retrieved.

Overview

One issue we've run into that has not yet been fully fleshed out is how we plan on storing data in AWS. As the type of data we will be storing and what format it will be in is still up in the air it's hard to commit to one specific mode of storage. Currently we are building out a DynamoDB table. However we've also considered using AWS's RDS service.

We have not yet received our parts order for the IoT devices and have been unable to do any implementation of them as a result. In the interim, we have created a mock base station to simulate upload data to the AWS cloud in order to work on round trip communication.