IrrEEgation: Low Level Design

By: Tyler Dale, Nicolas Garcia, Ellen Halverson,

Kenneth Harkenrider, and Trenton Kuta

2 Table of Contents

Detailed System Requirements 2

Detailed Project Description 5

1 Detailed System Requirements

The current system for flood irrigation is ineffective, time-consuming for the farmers, and can waste precious natural resources which are already scarce to the region. Currently a flood irrigation farmer must walk out to his or her field at regular intervals for up to 36 hours. Each time he or she monitors the field, he or she must determine how far the water has traveled. Once it reaches a certain distance the farmer is responsible for closing the valve manually. This entire process is time-consuming and ineffective. In order to address each of those issues and make the process more efficient, the previously necessary manual checks done by the farmer will be replaced with a system of sensing pylons. This allows the farmer to monitor the waters levels from a remote location.

In order to solve this problem, the pylon system has many specific requirements in order to make this process as efficient as possible. In order to create a more user friendly system for the farmer, the finalized pylon system will integrate the following components consist of a central hub, the pylons, the motorized valve, and a user interface seamlessly.

First, the central hub is responsible for setting up the communication framework for all the pylons, the user interface, and the valve. It will be continuously reading and writing data from the pylons, user interface and motor, therefore it will need a large amount of processing power. To accomplish this, the central hub will be plugged into an outlet at the farmer's house. In order to effectively communicate with each of the pylons and valve, it will have to setup a wifi network and MQTT server with sufficient range to cover the entire field which can be up to 100 yards. Also it needs to have an interface which provides the user with necessary information and allows the user to control the system manually. This information includes the status of the

pylons, the status of the water in the field, the status of the valve. Also it has the ability to change the valves position from open to close with the touch of a button.

The pylons will be placed periodically throughout the field. In order to receive power in the middle of the field, each pylon will have a solar panel and battery system which will power the pylon. It is important that each pylon is able to easily connect to the wifi network and establish a connection with the MQTT server. The pylons need to be adaptable, therefore they can be reset and moved around the field easily. In order to determine the presence of water we will use a resistive based water sensor. The pylon needs to be able to publish the status of the water sensor to the MQTT server. This component of the pylon needs to be easily accessible because it will face the most wear and tear and therefore it will have to be able to be replaced easily. The entire processing system of the pylon will be placed in a water tight container in order to protect it from the elements. This container needs to be attached to a stake which allows the farmer to easily place the pylon but also provides a sturdy base for the pylon to protect it from the harsher conditions of the elements

Third, the motor must be able to remotely open and close a valve. The valve is a large metal panel which is raised or lowered in order to seal a water pipe. Therefore, this large metal panel needs to be raised or lowered electronically via a linear actuator controlled by information given to it over the MQTT server. The status of the valve must be reported back to the central hub.

Finally, in order for the user to easily control the pylon system he or she will have access to a phone app. The app needs to be easy to understand as well as instructive. Therefore, the app will display all the pertinent information in a clear and organized fashion which removes all ambiguity. It should display information such as the status of the valve, the location of the water,

the status of the pylons, potential errors or problems in the system. Also the user should have the ability to open or close the valve directly from the app.

4.1 System theory of operation

This product combines several components connected wirelessly to sense and react to sense and react to water touching pylons. Essentially, the system involves a user interface, an unlimited number of valves, and an unlimited number of pylons, all of which are able to read from and write to a centralized hub (see Figure 1). The operation begins and ends with the valves. When the user wants to begin irrigating his or her crops, he or she can remotely open any valve using either the app or the LCD touch screen on the hub. Once the valve is opened, water will begin flowing throughout the field. When a pylon gets wet, it will relay this information to the central hub. The farmer will be able to monitor the status of all available pylons throughout the irrigation process and remotely close the valve at any time. Alternatively, the farmer can elect to have any valve close automatically if a certain pylon gets wet. Overall, this allows the entire flood irrigation process to be done without the farmer taking a single step into his or her field, making the entire event dramatically easier.

4.2 System Block diagram

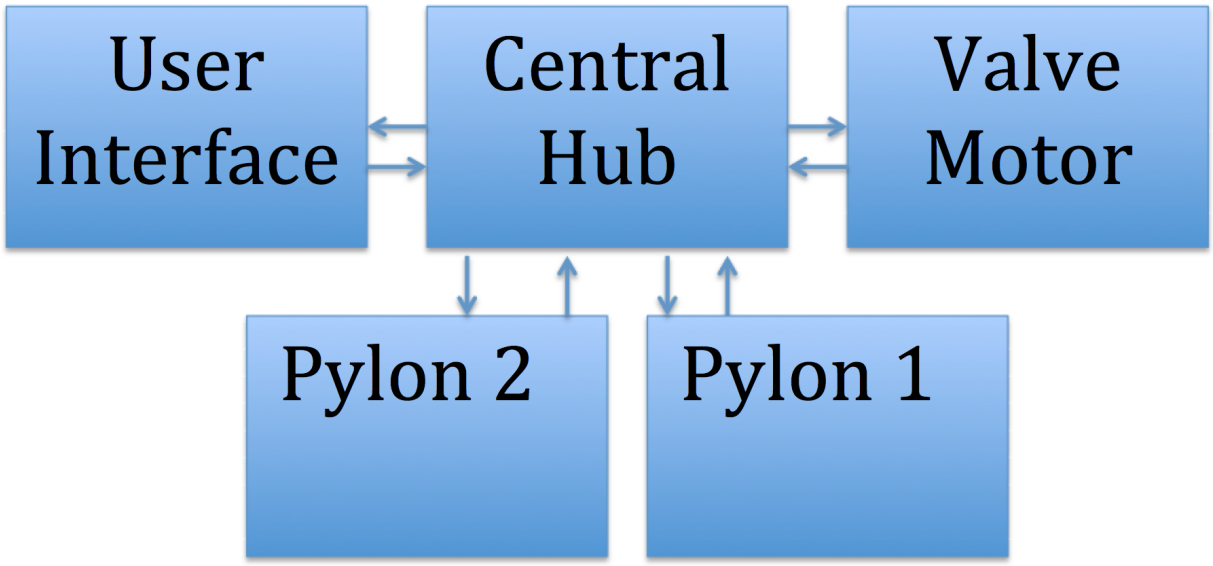

Figure 1. Block diagram showing connections between each subsystem.

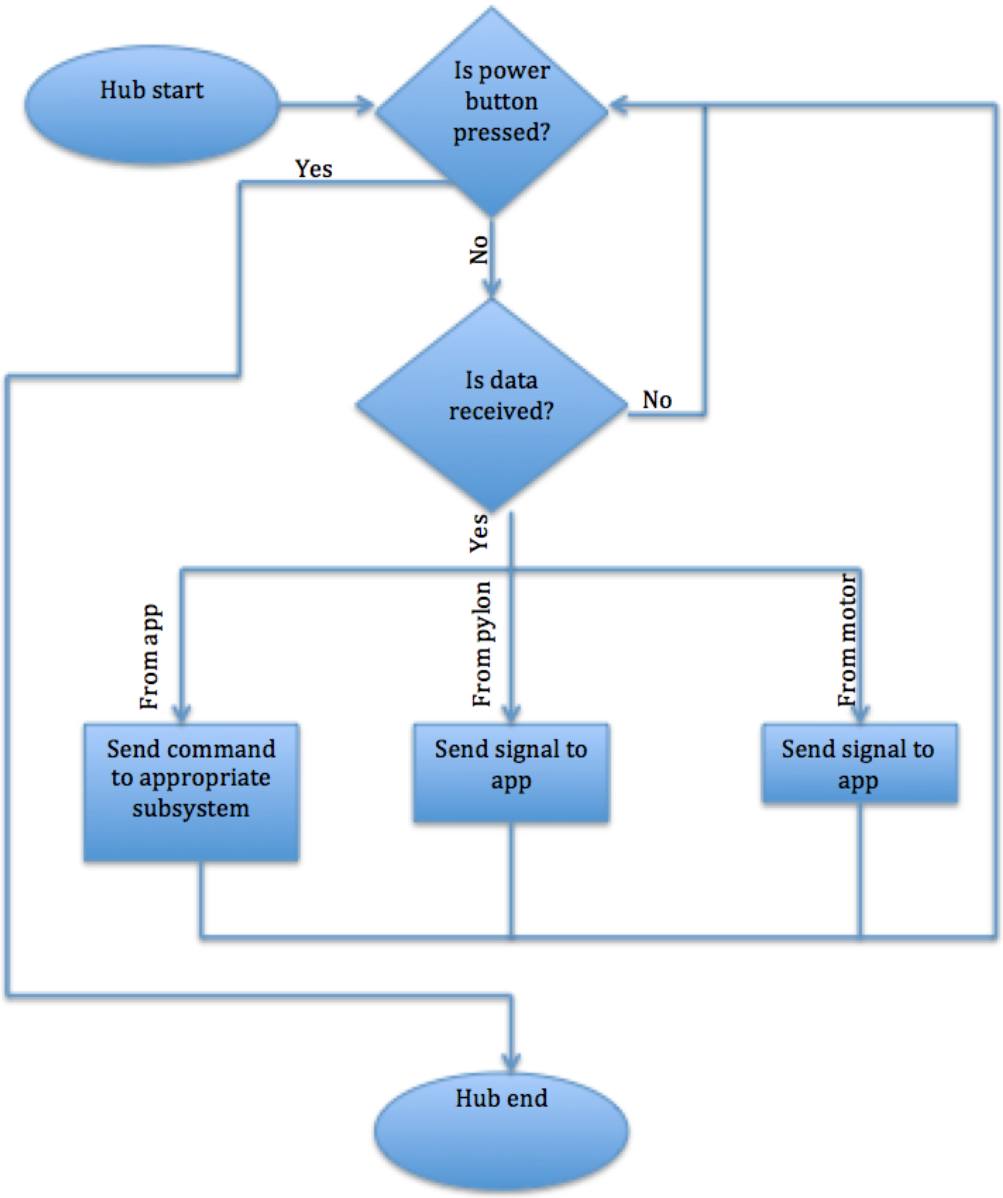

Figure 2. Block diagram for the central hub.

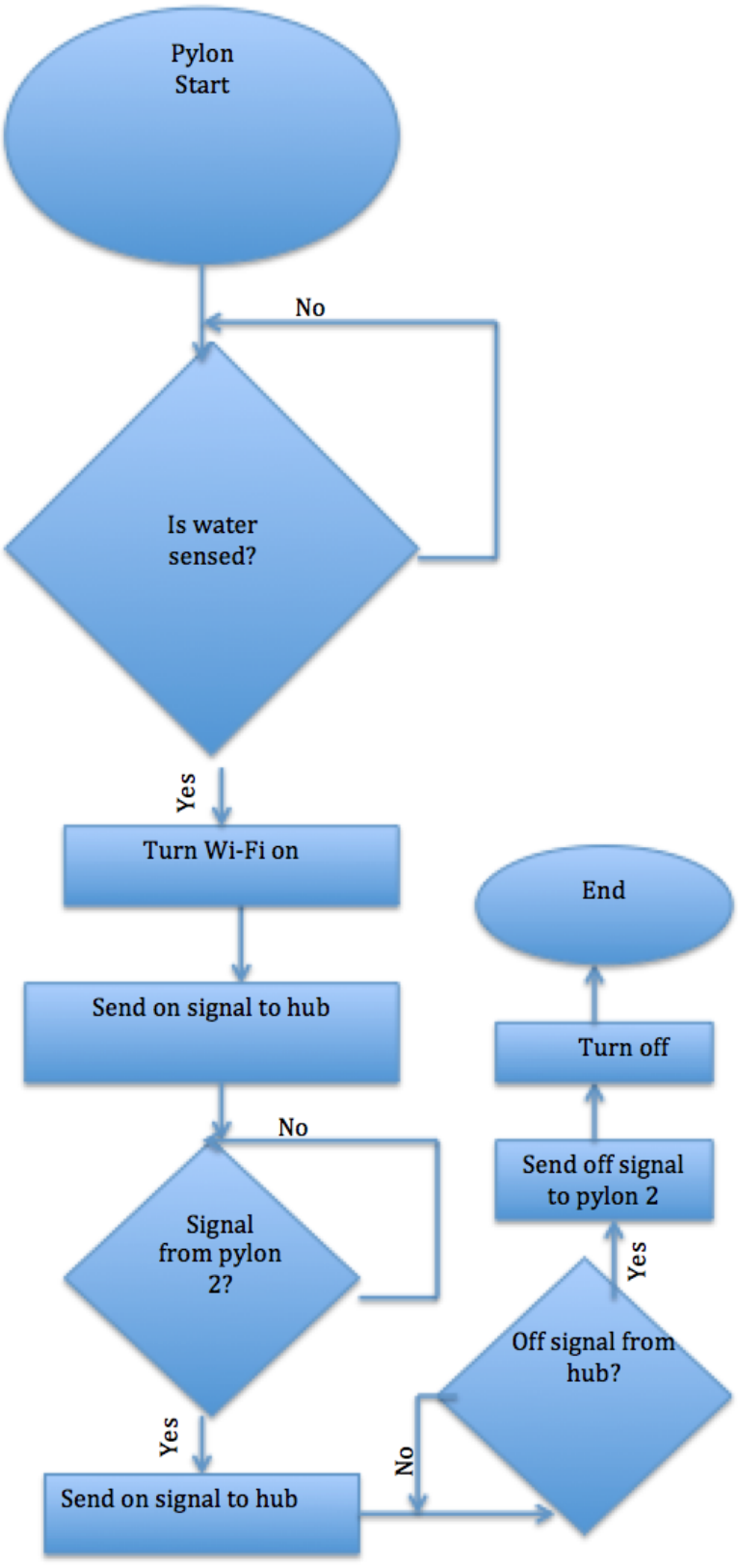

Figure 3. Block diagram for the pylons.

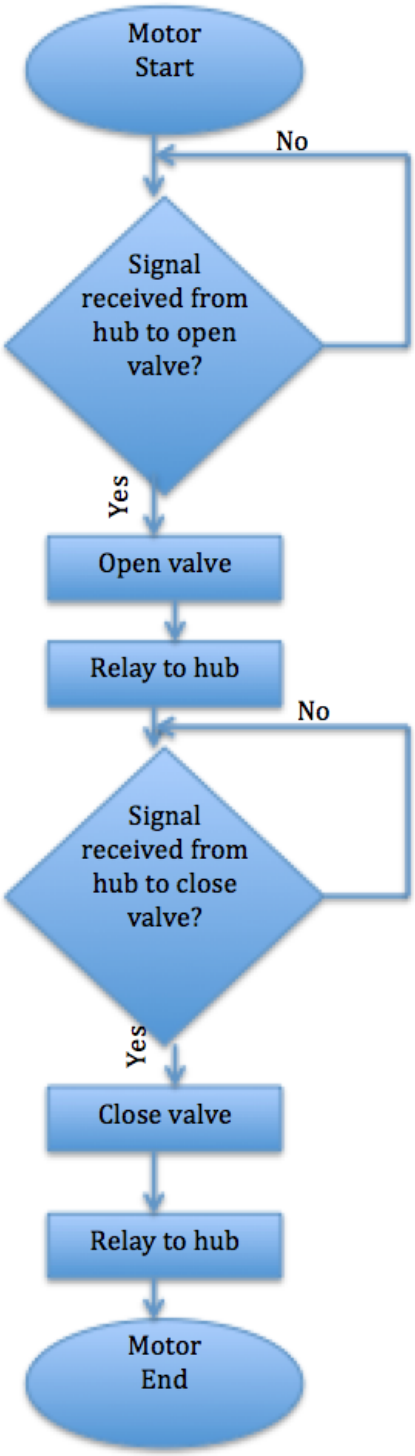

Figure 4. Block diagram for the motor.

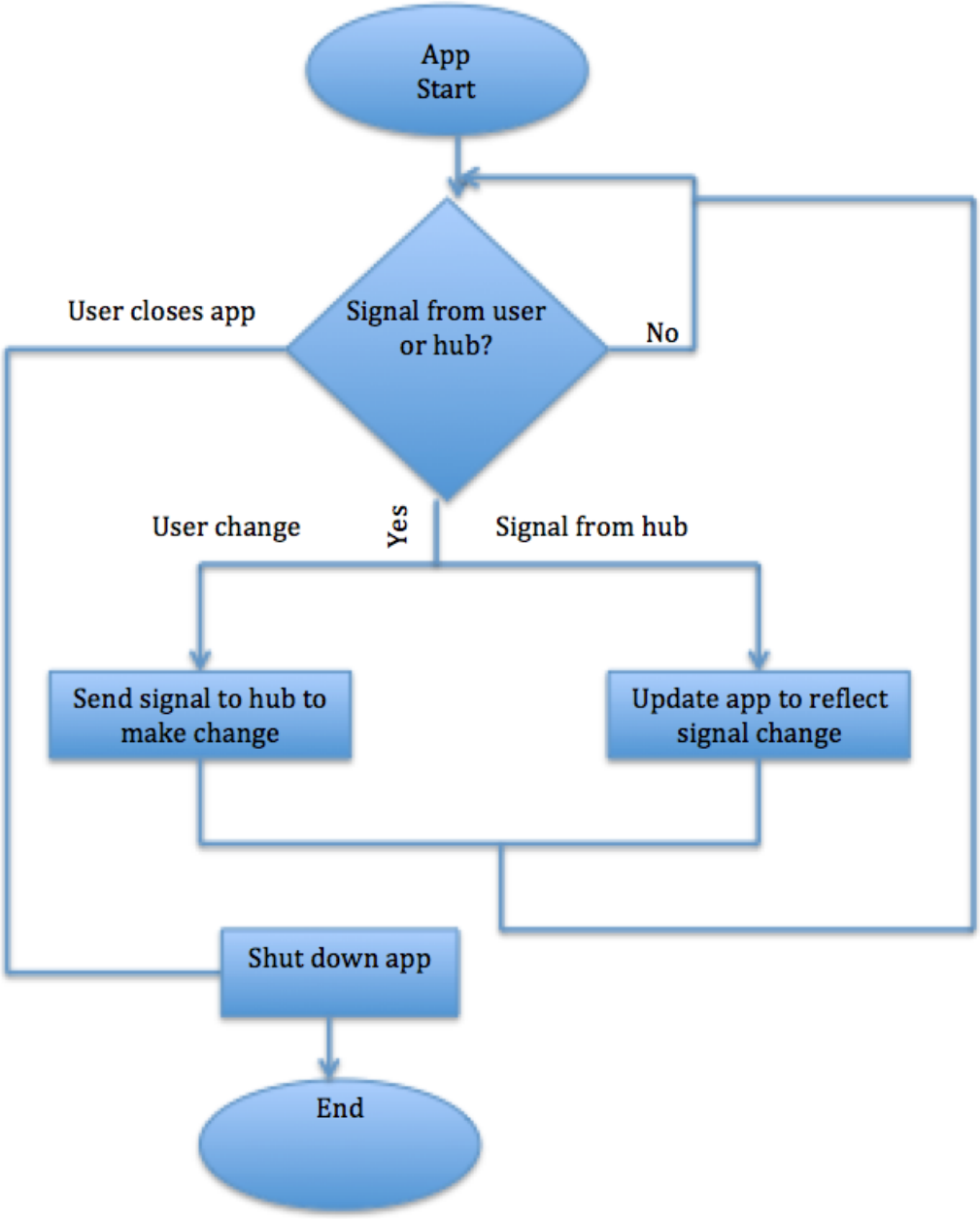

Figure 5. Block diagram for the user interface.

4.3 Pylon/Valve Subsystems and General Communications

The sensor pylon must be able to accurately sense the presence of a thin film of water and wirelessly alert the hub subsystem in that event. However, a slew of smaller requirements must be addressed for the above protocol to function properly.

First, a wireless communications protocol must be implemented; we chose MQTT. As stated elsewhere in this document, MQTT is ideal because of the abundance of resources (found both in online forums and IDE libraries) and its relative ease of implementation. Our MQTT code is enveloped in our general C code for the pylon microcontroller, programmed from the Arduino IDE. The Arduino IDE is employed for convenience - MQTT and microcontroller code are free, readily available, and easily uploaded into a microcontroller.

The wireless communications system requires a range larger than that typically used for WiFi systems, on the order of 200-500 meters. For this reason, the pylon has an external 3dB antenna for boosting receive and transmit gain.

The microcontroller on the pylon is an ESP8266, so chosen because of its WiFi capabilities. The chip itself doesn't need to perform any heavy computations and is primarily the medium of relaying moisture measurements. A desirable aspect of the ESP8266 is its ability to enter deep sleep. This is also a system requirement, as the pylon will go long stretches of time (in between irrigation events) where measurements will not be needed.

A critical constraint on the microcontroller is non volatile memory. Whenever the pylon activates or wakes from sleep, it will need to know the WiFi credentials of the hub. In general, these will be saved into the ESP8266's EEPROM from the last sync operation. If the pylon does not have these details saved, it becomes a wifi access point and allows the hub to connect and transmit its credentials. These details are written and saved into the EEPROM for future

reference, and can be erased in the event of another sync. A push button is included on the pylon to allow manual initiation of a sync and an LED is included to indicate sync progress.

The microcontroller interfaces with a Sparkfun moisture meter via an ADC pin. This relatively simple sensor is used because it is cheap and there are no minimum resolution constraints; the sensor only needs to determine if water is present, not how much water is present.

The pylon wirelessly updates the hub with frequency that depends on whether or not the pylon is in sleep or awake mode. In both modes, the pylon undergoes an amount of sleep before running an active cycle; the sleep and awake modes only dictate the length of that time. In practice, a wake/sleep command from the hub determines how long the pylon will sleep before undertaking the next cycle. This means that the pylon's activity is regulated on a cycle by cycle basis from the hub. Each of these cycle consists of the pylon waking up, reading its non volatile memory, connecting WiFi/MQTT, reading the moisture sensor, transmitting the moisture sensor data, and finally listening for commands from the hub before finally sleeping again.

The pylon runs on a 3.7 V LiPO battery, charged by a 12 V, 1.5 W photovoltaic cell. Charging for the cell is taken care of by a Linear Technology LT3652 chip, which ensures that overcharging does not occur. The moisture sensor and wireless communications do not drain the battery quickly, so energy level is of low concern on the pylon.

At the time of this writing, the valve subsystem has not been constructed as we are waiting on the materials to control the valve. However, its constraints and functionality are almost identical to that of the pylon with only a few differences. The valve will not be an active publishing device on the MQTT network; rather, it will only wait for commands from the hub to open or close. The other primary difference is the hardware on board. The valve subsystem will

have a large linear actuator controlled by the ESP8266 via an external relay system. This relay system will also allow the valve system to read opening/closing progress and update the hub.

4.4 Hub

The Hub is the MQTT broker and the single unifying subsystem; it communicates with the app, pylons, and valve and relays messages between them (per the Publish/Subscribe format). As such, it is currently a Raspberry Pi 3 and is the single most intelligent subsystem.

After the initial setup of the Raspian OS, the Mosquitto MQTT broker and client were installed on the Raspberry Pi in order to host and publish and subscribe using MQTT. The team tested the device's MQTT capabilities using an MQTT simulator, MQTT.fx, and was successfully able to publish and subscribe between multiple devices over the new broker. The Hub also consists of a Wi-Fi router fitted with three 9 dBi antennas.

Essentially, the hub holds two lists at all times. The first list contains details on every pylon connected to the hub and includes qualities such as their names, connection statuses, and whether they are wet or dry. The second list contains details on every valve connected to the hub and notes details similar to those of the pylons. The hub then runs in an infinite loop over MQTT, during which time it automatically updates the statuses of all valves and pylons approximately every thirty seconds. This system is entirely autonomous, leaving a reliable system that the farmer does not need to monitor or worry about. That being said, the user can monitor and command the hub at will using the app.

4.5 Mobile Application

The mobile application must be able to interface with the user and the central hub to provide system control and information to the user. Fundamentally, the user will be able to open and close the valve as well as view the valve and pylons statuses.

The single most important requirement for the application is wireless communication with the central hub, which we determined could be accomplished through the use of MQTT and would allow for easy integration with the other subsystems. Various MQTT packages are available for both iOS and Android environments, however, we decided to develop the application with the latter because the team member responsible owns an Android device. For reference purposes, Android Studio was the IDE used and the code was developed in Java because they are the most commonly found resources for Android app development. The MQTT client used in the app is the Paho Android Service from Eclipse, which supports the newest versions of MQTT protocol and is supported by Android Studio. In order to become familiar with and implement the client in our app, we downloaded a sample app available from Eclipse on GitHub (https://github.com/bytehala/android-mqtt-quickstart). The sample app is capable of new client creation and MQTT publication and subscription, which provides the basic functionality necessary for our application. For this reason, the sample app was implemented as the base code and we further developed the user interface to improve the simplicity and intuitiveness for our end user, the farmer.

To test the functionality of the application, we set up a Raspberry Pi 3 as an MQTT broker to simulate the Hub subsystem and broadcasted the WiFi signal with a limited-security router. Using the MQTT client simulator, MQTT.fx, we were successfully able to subscribe and publish from the mobile app client to the simulator client and vice versa.

4.6 Valve System

Waiting on relays to finish modeling the valve system. Currently able to show functioning of linear actuator.