Carleton UNIVERSITY | Faculty of Engineering and Design

DESIGN REPORT:

iPot

Zubaer Ahmed - 101001925
Edmond Chow - 100883365
Kevin Johnson - 101077070
Aaron Vuong - 101023182

GROUP: T11
SYSC3010

November 1st,2019

Contents

1.0 - Problem Statement

Growing plants is often seen as a very rewarding process. By giving care and nurture to your plants you are rewarded by seeing them bloom. Although this is quite gratifying it is a process that can take months of consistent and diligent care, which interferes with most people's busy lives. Many individuals have to worry about work, school, kids, and other unexpected events that could come up throughout their lives. Day to day activities are often not structured, as they are often changing to meet the goals and obstacles for ones day.

Perhaps a student has to study all day at the library or maybe a mother has to take care of a sick child. It is quite easy to forget about caring for ones plants when other responsibilities take priority. Even with enough time to water plants, being able to garden is a talent that must be earned through hard work, patience and dedicated learning [1].

Moreover, it has been shown that growing plants has many beneficial properties for mental health. As mentioned in this post online, gardening and growing plants of any sort has shown to promote brain health, has allowed individuals to stay more mindful and has promoted a sense of greater responsibility in life [2].

There are multiple factors that go into growing plants and accounting for each one while also navigating a busy life is too much for the average person. There are many systems that have already been manufactured which allow for gardening to be made easy. Based upon this idea, our product adds additional functionality on top of these models.

The objective of this project is to develop a system that enables novice gardeners to grow and sustain the plants of their own choice. During the setup process of the iPot, users will be able to specify specific criterion in order for the product to take care of their plant. Our objective will be achieved through the use of both software and hardware capabilities.

2.0 - System Architecture

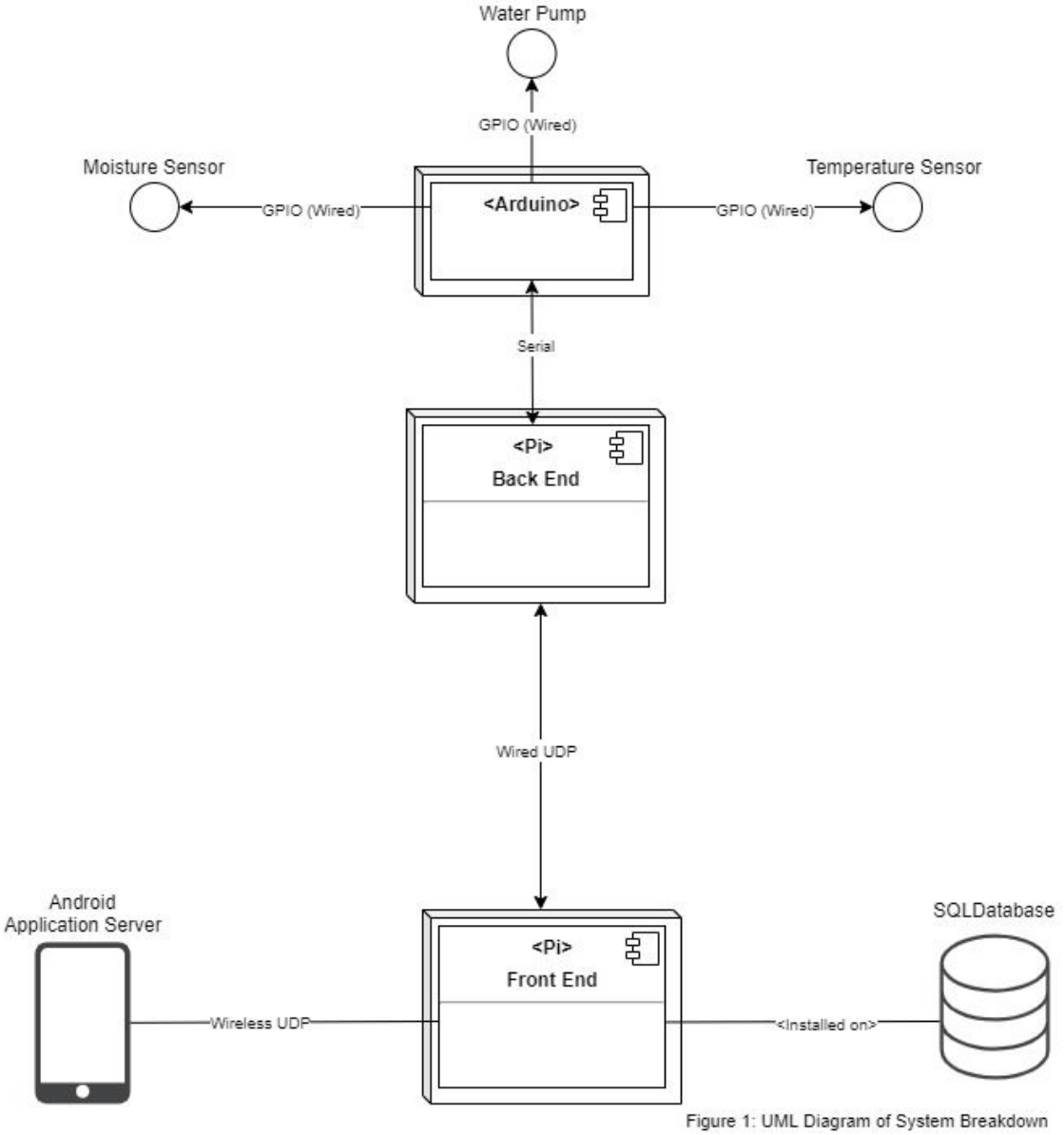

Figure 1: UML Diagram of System Breakdown

The deployment diagram shown above shows the system architecture. There is an RPi3 acting as the central back end of the system and an RPi3 acting as the front end. The Front End RPi consists of two modules; the Application Server and SQL Database. The Application Server will handle the communication between the user and the system through a wireless network, while the SQL Database will hold data from sensors as well as inputs from the user. On the App an interface will be provided from which the user will be able to send in various requests.

The PI which is labelled as the Back End will control the overarching communications going on between the database and Arduino, this is via wired UDP and wired GPIO respectively.

The previously mentioned database will be installed on the Front End RPi. The Arduino will handle the flow of data from the various sensors. As seen from Figure 1, the sensor data that circulates the system will be from the Moisture sensor and Temperature sensor. The sensors will provide various data to the system and will be connected to the Arduino via GPIO. The watering system that exists is connected similarly to the Arduino via GPIO. The watering system will be activated when certain user inputted benchmarks are reached by the system through data from the water sensor.

The database on the Front End RPi will have two tables. The first table will store the various records of sensor data that is captured from the system. At varying times during the runtime of the system, the sensor data will create records on the database table for moisture and temperature. The second table on the database will store the user inputted benchmarks for the moisture and temperature. These benchmarks will later be used in the system to determine various behaviours.

This project will be coded using Python for coding the communication protocol between the two Pis and the android app, while the database will be coded using SQLite. The choice to use python is due to the ease of use and the groups familiarity of use coding in a python environment.

3.0 - Hardware

3.1 - Functional Design

The system is essentially always active since the system requires that the environment of the iPot is always being monitored. The process begins at the mobile application from which the user is able to set the benchmark data points for their respective iPot as well as request data about the plant. The application will allow the user to obtain the most recent data about their plants environment.

Interaction with the iPot begins at the mobile application where a user will set up their own plant. During this installation step, the user sets the specific moisture and temperature that is most optimal for their plant, which will be saved onto the database with a respective pot identification id. Along with the set points of the sensor data, the user will also set the time interval for which the data will be sent each time. Once the user has inputted the set points for their own plant, their job is essentially over for the growth of their plant. With the understanding that the user has appropriately set the benchmark sensor values the system will consistently check with the outputted readings on the Arduino.

After the initial setup of the iPot is processed the user will have the ability to monitor the sensor readings through the mobile application. On the Arduino the sensors will send data through the system with respect to the initial time interval set. Depending on the readings that are captured on the sensors, a decision by the system will be made to initiate the water pump in order to water the plant.

3.2 - Circuit Design

The sensors that exist in the system are all connected to the Arduino. Specifically the temperature and moisture sensor are both wired up to the Arduino and will feed their data to the entire system through the Arduino. As can be seen in *Appendix A, Figure 9 and 10* ,the sensors will be wired as hardware components in a similar configuration.

3.2.1 - Temperature Sensor

The temperature sensor that will be used in our system is the *LilyPad Temperature Sensor.* This temperature sensor was obtained by the team through the toy box of hardware parts that were present in the labs. The temperature sensor will be used in order to capture the temperature of the iPot surroundings, hence it will be installed inside the plant pot itself. Below is a simple program that enables the *LilyPad Temperature Sensor* to start capturing readings for the temperature.

```
float temp;
void setup() {
Serial.begin(57600);
};

void loop () {
temp = analogRead(0)*5/1024.0;
temp = temp - 0.5;
temp = temp / 0.01;
Serial.println(temp);
delay(500);
};
```

The code above is run through the Arduino itself and allows an output of the temperature to the Arduino prompt in Celsius. As can be seen from the code, the sensor outputs a voltage which is to be converted into the appropriate temperature. The hardware components that is

present for the sensor is wired up to the Arduino according to the circuit diagram depicted in *Figure 2* below.

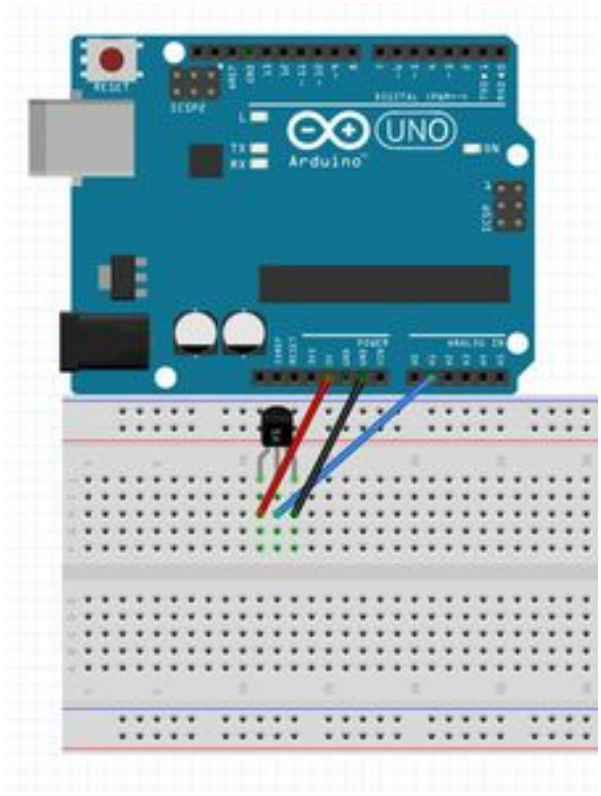

Figure 2: Circuit diagram displaying how the temperature sensor is wired up to the Arduino Uno [3]

3.2.1 - Soil Moisture Sensor

The moisture sensor which is used in the system is a generic one that was obtained similarly to the temperature sensor. The toy box present in the labs had many soil moisture sensors which the team (T11) acquired. The soil moisture sensor will be stuck into the soil of the iPot such that it can obtain the most accurate data. Similarly to the temperature sensor a simple program which set up the moisture sensor to capture data is shown below.

```
#define SensorPin A0
float sensorValue = 0;
void setup() {
 Serial.begin(9600);
}
void loop() {
 for (int i = 0; i <= 100; i++)
 {
   sensorValue = sensorValue + analogRead(SensorPin);
   delay(1);
 }
 sensorValue = sensorValue/100.0;
```

```
 Serial.println(sensorValue);
 delay(30);
}
```

The code above is run through the Arduino itself and allows an output of the specific soil moisture around the sensor.As can be seen from the code, the program has a loop that takes 100 readings of the soil moisture and averages the values to gain an accurate approximation. The hardware component that is present for the soil moisture sensor is wired up to the Arduino according to the circuit diagram depicted in *Figure 3* below.

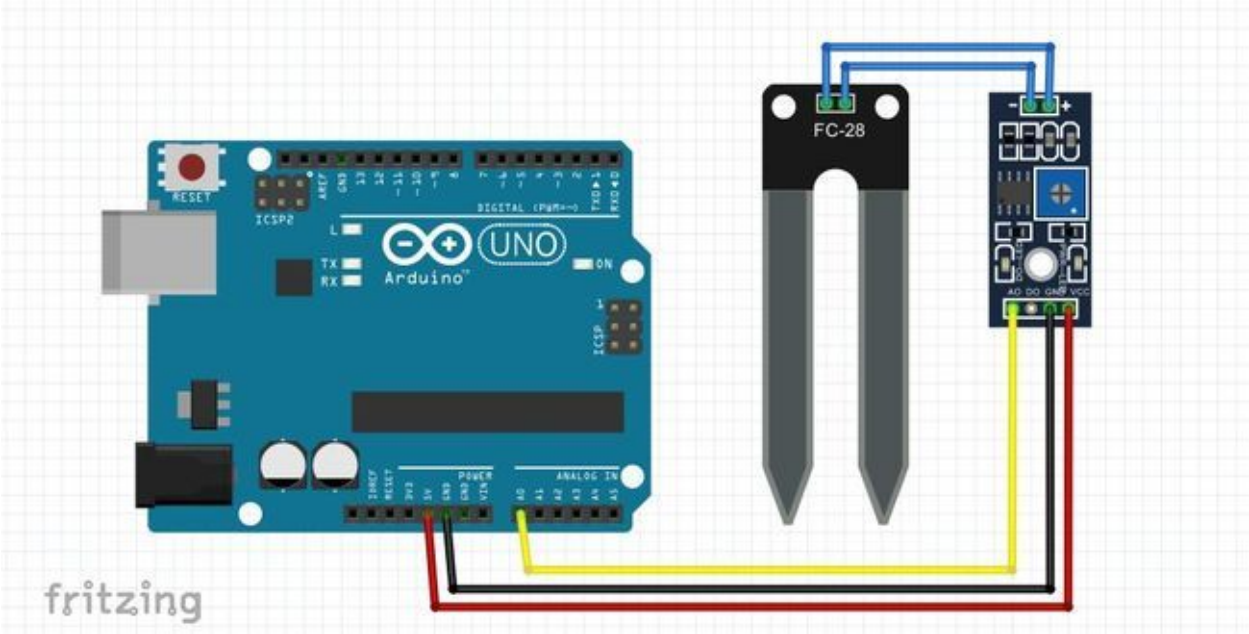

Figure 3: Circuit diagram displaying how the soil moisture sensor is wired up to the Arduino Uno [4]

3.2.3 - Water Pump

The actuator which will be used in the iPot system is the submersible water pump. The water pump is essentially a motor that is enabled based on a voltage of 5V being run through its input terminal. The water pump which will be used in this system is the *WayingTop* Submersible Water Pump. The hardware component that is present for the water pump is wired up to the Arduino according to the circuit diagram depicted in *Figure 4* below. Note this configuration was used purely to test the pump, a modified version of figure 4 will be used inorder to allow control of the pump through the GPIO pins of the Arduino.

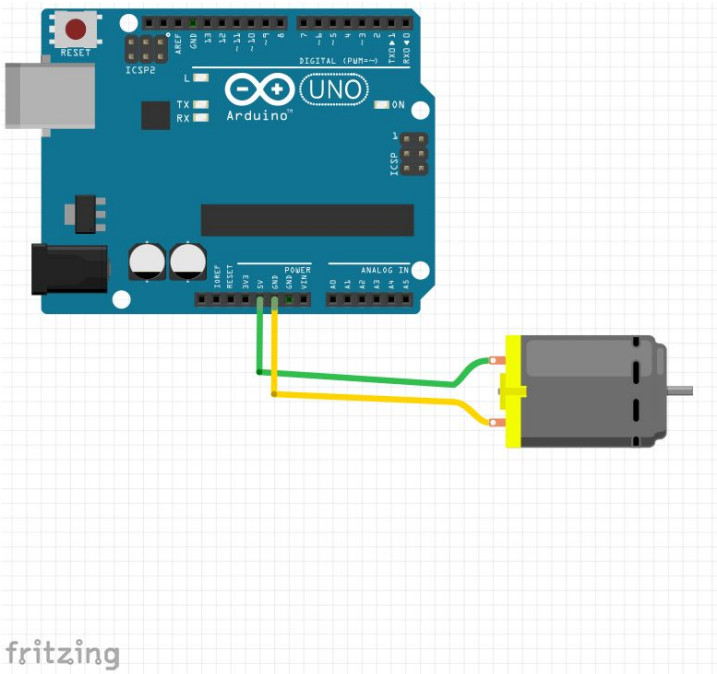

Figure 4: Circuit diagram displaying how the submersible water pump is wired up to the Arduino Uno for testing purposes.

4.0 - Communications Protocol

4.1 System Communication Sequence Diagrams

In a system that is built with various software and hardware components, there are many considerations that need when observing the runtime of the various programs. There are various cases in the system where components communicate across in order process an activity, to model these communications sequence diagrams have been used.

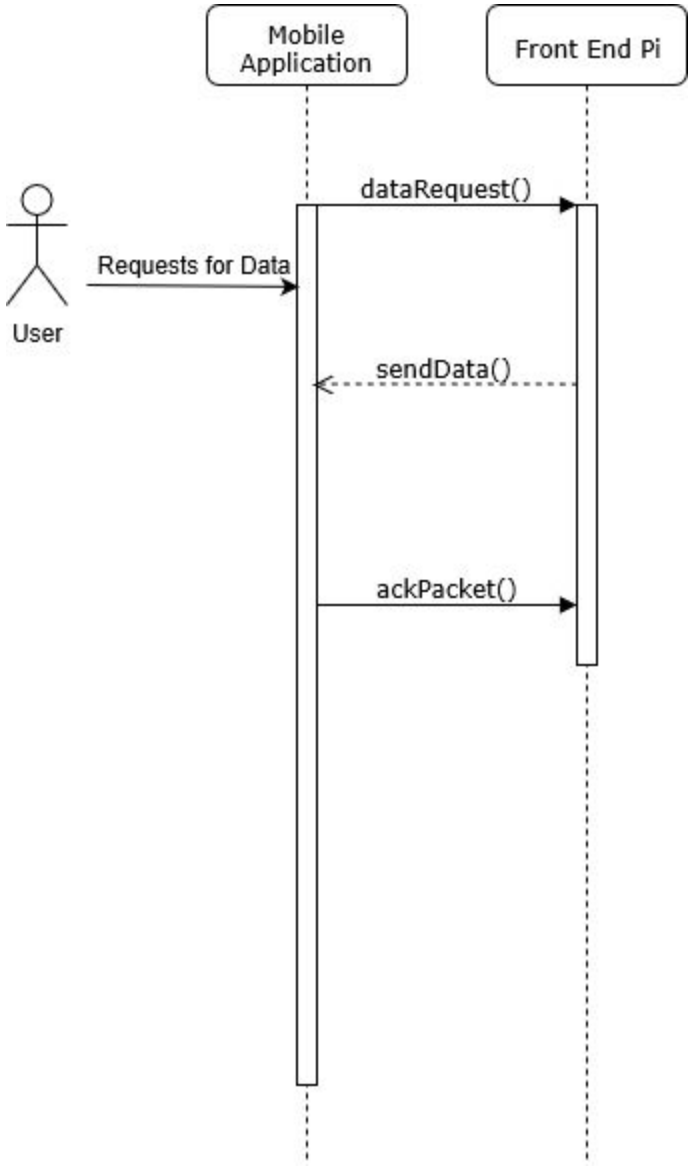

Figure 5: Sequence Diagram showing the application asking for data without errors

The figure above shows the simple process of a data request where there are no errors in the sending and receiving of the UDP packet. This event takes place when the user requests data to be displayed to them via the application. A data request will be sent to the Front End Pi in order to gather the appropriate data to display to the user. The event is terminated with an acknowledgement packet to notify the Front End Pi that the data has successfully been received by the user (Mobile Application).

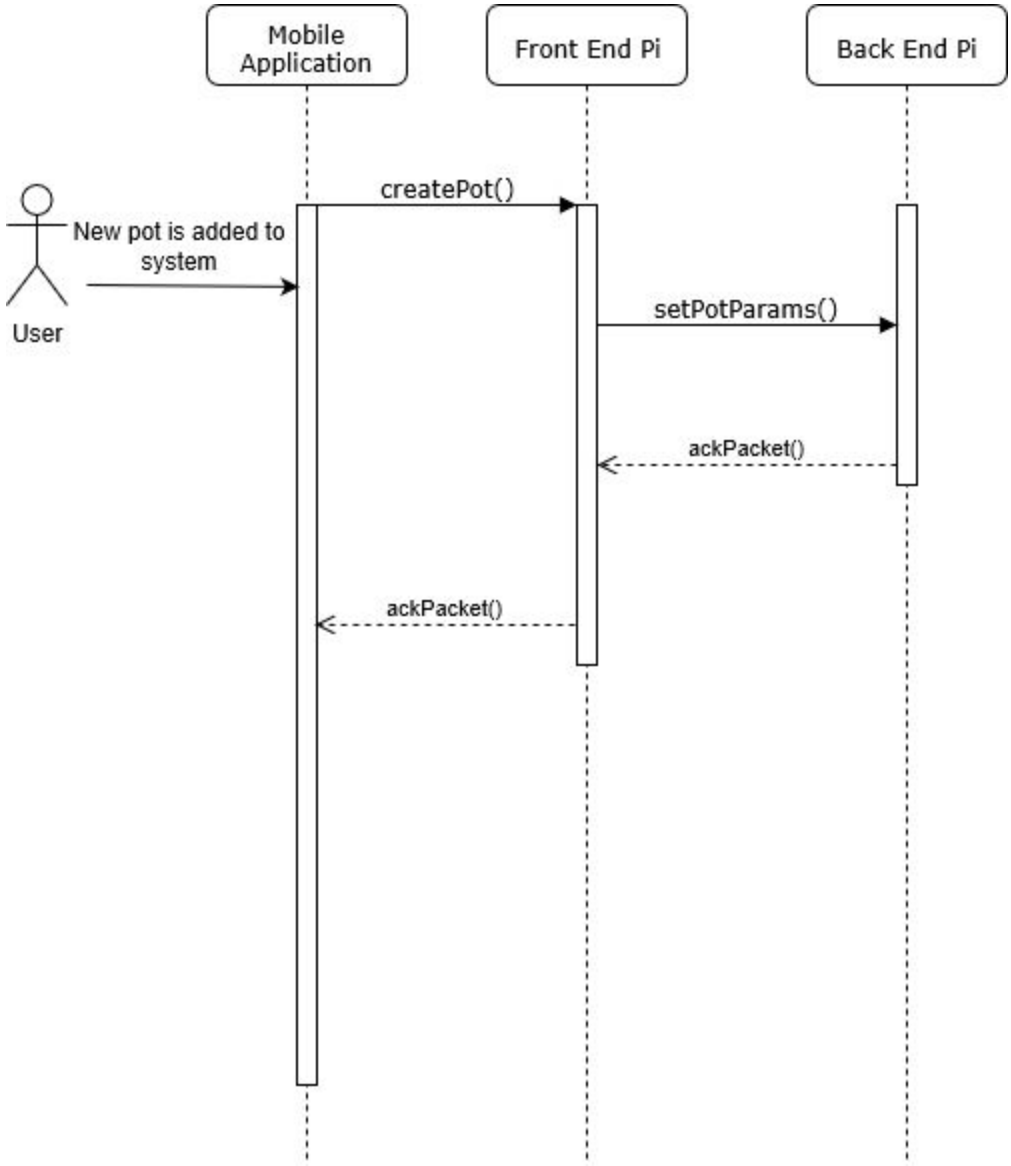

Figure 6: Sequence Diagram showing the creation of a new pot

The figure above shows the process of adding a new pot where there are no errors in the sending and receiving of the UDP packet. When the a new pot is created the set points for the sensors are also sent. As seen in the diagram, the communications between components are acknowledged by dedicated packets. Acknowledgements packets will be encoded specifically such that the receiver will know which type of packet is arriving (See *4.2 - Communication Protocol Table*).

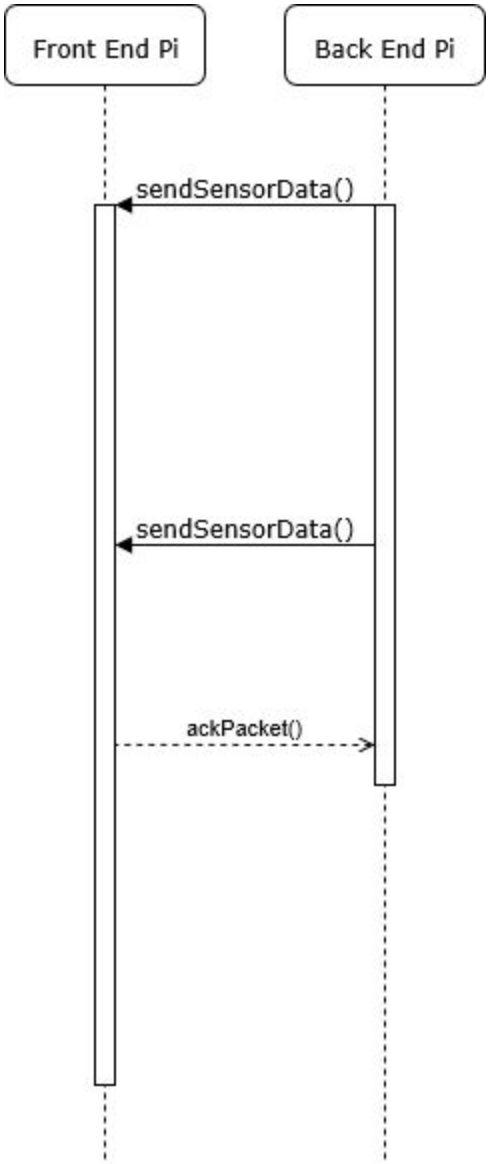

Figure 7: Sequence Diagram showing the periodic transfer of data from the sensors with an error in receiving the first packet

The figure above shows the process when the Back End Pi is sending data to the Front End Pi to be logged into the database but there is an error in the sending and receiving of the UDP packet. If there is an error that occurs when a SensorData packet is being sent, the packet will be resent by the source device if no acknowledgment is given in 10 seconds.

4.2 - Communication Protocol Table

Sender	Receiver	Message	Format
App	Front End PI	dataRequest	{int: 02, string: PotName}

Database	App	sendData	{int: 03 | 04, float: waterSense, float: tempSense, Date time}
App	Front End	createPot	{int: 01, Float: waterSet, Float tempSet, Float intervalTime, String PotName}
Front End	Back End	setPotParams	{int: 01, float: waterSet, Float tempSet, Float intervalTime, String PotName}
Back End	Front End	sendSensorData	{int: 05,String name, Float waterSense, Float tempSense, Float lightSense, Date time}
Front End	Back End	connectCheck	{int 09}
Any	Any	ackPacket	{int: 06| 07|08|09}

Table 1: Protocol table displaying the format of data in the system

Table 1 above shows the different types of data that is present in the iPot system. Along with the various data types there is also a clear breakdown on the specific format with which the data will be sent. These specifications will allow the logic in the various parts of the system (software/hardware) to know where data is being communicated to/ from as well as what to do with the data packets themselves. The data packets will be sent using the JSON format due to its simplicity in packing and unpacking the packets.

4.3 - Integer Packet Type Table

Integer	Packet Type
1	Indicates that the packet is designed to send configuration data from the app to both the database and the backend PI for a Pot
2	This is a packet from the app to the backend designating a pot it would like information on
3	This is a packet from the front-end PI to the database sent in response to receiving an Integer 2 packet. This code designates that it is not a complete transmission of data and more packets will be coming.
4	This is a packet from the front-end PI to the database sent in response to receiving an Integer 2 packet. This code states that all sensor data for the pots have been sent and that there will be no more packets incoming.
5	Code is used when the back end PI is sending sensor information to the database on the frontend PI. These packets will be sent based on the time sent in the established time interval set in pot creation.
6	This int is part of the error handling code, if an Int 6 acknowledgement packet is received by the sender, then the sender will assume the packet was sent correctly
7	This int is part of the error handling code, if an Int 7 acknowledgement packet is received by the

	sender, it means that there was a data mismatch in the packet (IE there is more or less data in the packet than what is expected by the system)
8	This int is part of the error handling code, if an Int 8 acknowledgement packet is received by the sender, it means that the data in the packet contains an unexpected or unrealistic value (IE: if the backend PI sends an int 5 packet that has a temperature value that is greater than the temperature on the sun)
9	This packet is sent if there has been no connections between the front-end and the back-end PIs for a length of time, this packet is sent to verify that there is still a connection between the front and back end Pis

Table 2: Table showing the meaning of the integer identification codes on the packets

Table 2 above displays the encoding behind the various integers that are used in the system. As can be seen from section *4.2 - Communication Protocol Table* , many of the packets have integer values which are encoded in the formating. Each integer in the formatting of the packets has a respective encoding in the system.

All packets will start by having an integer value attached to the front of the packet. These integer values will allow the system to distinguish what type of data packet is being sent. For example, if the front end PI receives a packet from the back end PI that has an integer of 5 at the start, the front end PI knows that it has received a sensor log packet and will extract the data based on communications protocol table above .

As a way to handle errors, after any packet is received by a part of the system, there will be a return packet with an int value indicating the status of the packet. So for example, a return packet with an integer of 6 means that no errors were found with the packet. An integer value of 7 would mean that the data in the packet doesn't match what is listed by its integer value. So for example if the app received an int 5 sensor data packet without any actual data in it, the app will acknowledge the packet with an int 7 acknowledgement packet.

5.0 - Software

5.1 - Database Management

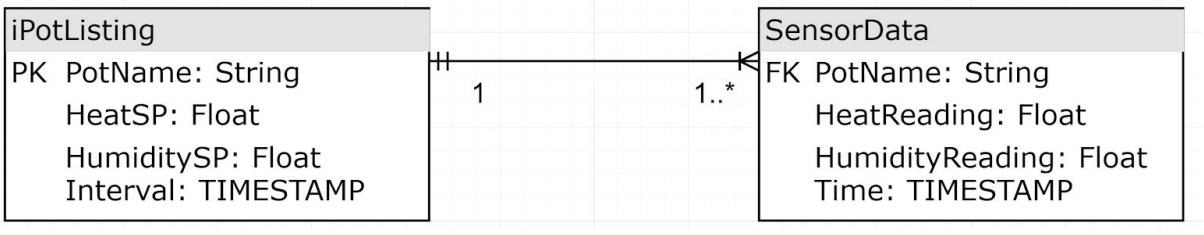

Figure 8: Database Schema showing the layout and design of the database.

The database will be built around two tables. The first table will be a listing of pots connected to the database and they will be indexed by the name of the pots with each database not allowing duplicate names.This table will contain columns for: the name of the pot, the temperature and humidity setpoints of the pot, and the time interval the user set for taking snapshots of the sensor data.

The second table will store the sensor data from the pots. The columns for this table will be: the name of the pot, the humidity and the temperature recorded by the sensor during the snapshot, and the time the snapshot was taken at. When the app requests data from database for a specific pot, the database will look at this second table and then send the data entries that are relevant from their to the app.

5.2 - Mobile Phone App

The phone app will be responsible for three things in the operation of the iPot. The first is the app will be used to set up the profiles for the new pot. To do that, the user will enter: the name for the pot that will be used in the database, the setpoints for temperature and humidity, and then how often they want the Backend PI to take a snapshot of the temperature and the humidity. The name of the pot should be unique and if user enters a name that has already been entered into the database then the app will report an error. At the end the app will notify if the creation of the pot was successful or unsuccessful.

The second function will be to update any of the variables attached to the pot other than the name (IE setpoints and time interval). To do this the user has to enter the name of the pot they want to edit. From there, the user will then enter what they want the new setpoints and time intervals are.

The third and final use of the app is to allow the user to view the logged sensor data of the pot in a graphical format. The user will enter the name of the pot that they want to view the data, then the database will send the logged sensor data of the pot to the app. From there the app will display that data graphically to allow the user to see the details of the condition of the pot.

6.0 - Errors

6.1 - Error Handling

The standard that will be used during the communication of data within the system is Transmission Control Protocol (TCP). As specified by the protocol itself, much of the need for error handling is reduced. Although TCP communication is slower, it offers reliability, which means there are no dropped packets or packets that are out of order. However, in order to handle

the errors that would still exist within the iPot infrastructure, we have various cases that have been established.

If a part of the system receives an acknowledgement packet that isn't int 6, or there is no acknowledgment packet after 10 seconds, the system will try to resend the packet that failed. This process will be repeated 3 times and if the system does not receive an int 6 packet by that point the system will display an error message and cancel the operation.

Another error the system will check for is if there is a loss of connection between back and front end PIs. If the front end PI has received no packets from the backend PI after a length of time greater than the longest time interval set in the database the frontend Pi will automatically send an int 9 packet to the backend to check if it is still connected. If the frontend Pi receives an int 6 acknowledgement packet, then operation resumes as normal. If not then the front end PI will display an error indicating that connection with the backend PI has has been lost and alert people using the app.

Appendix A

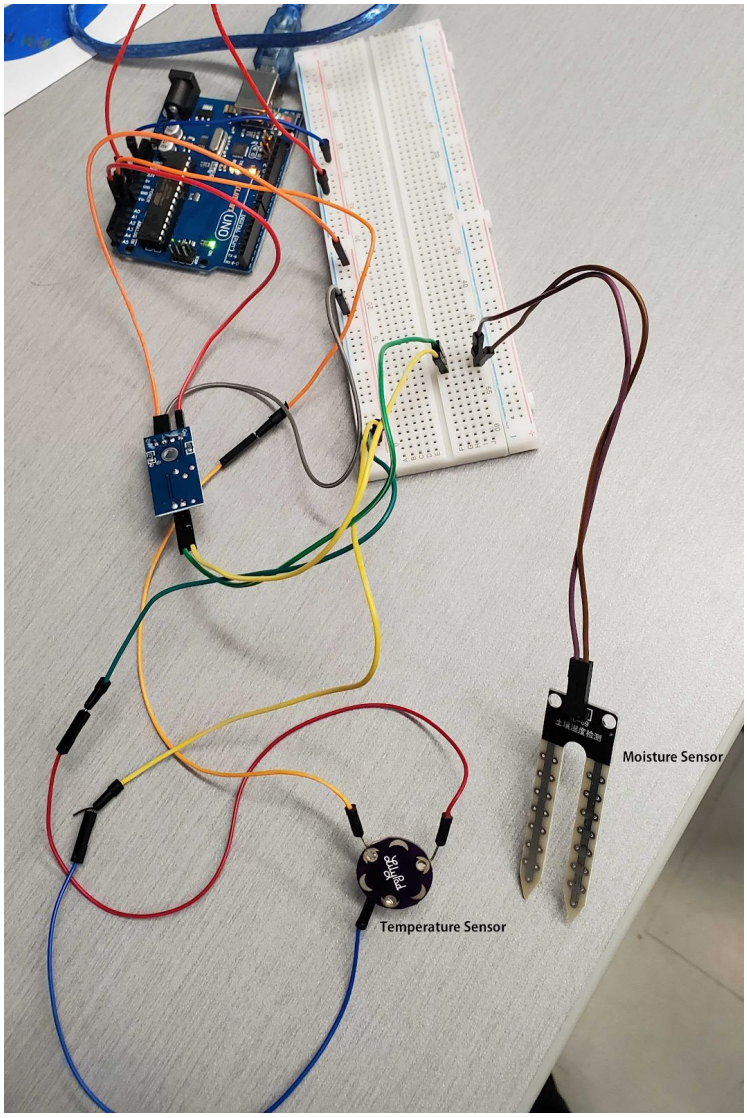

Figure 9: Image showing the sensors wired up to the Arduino

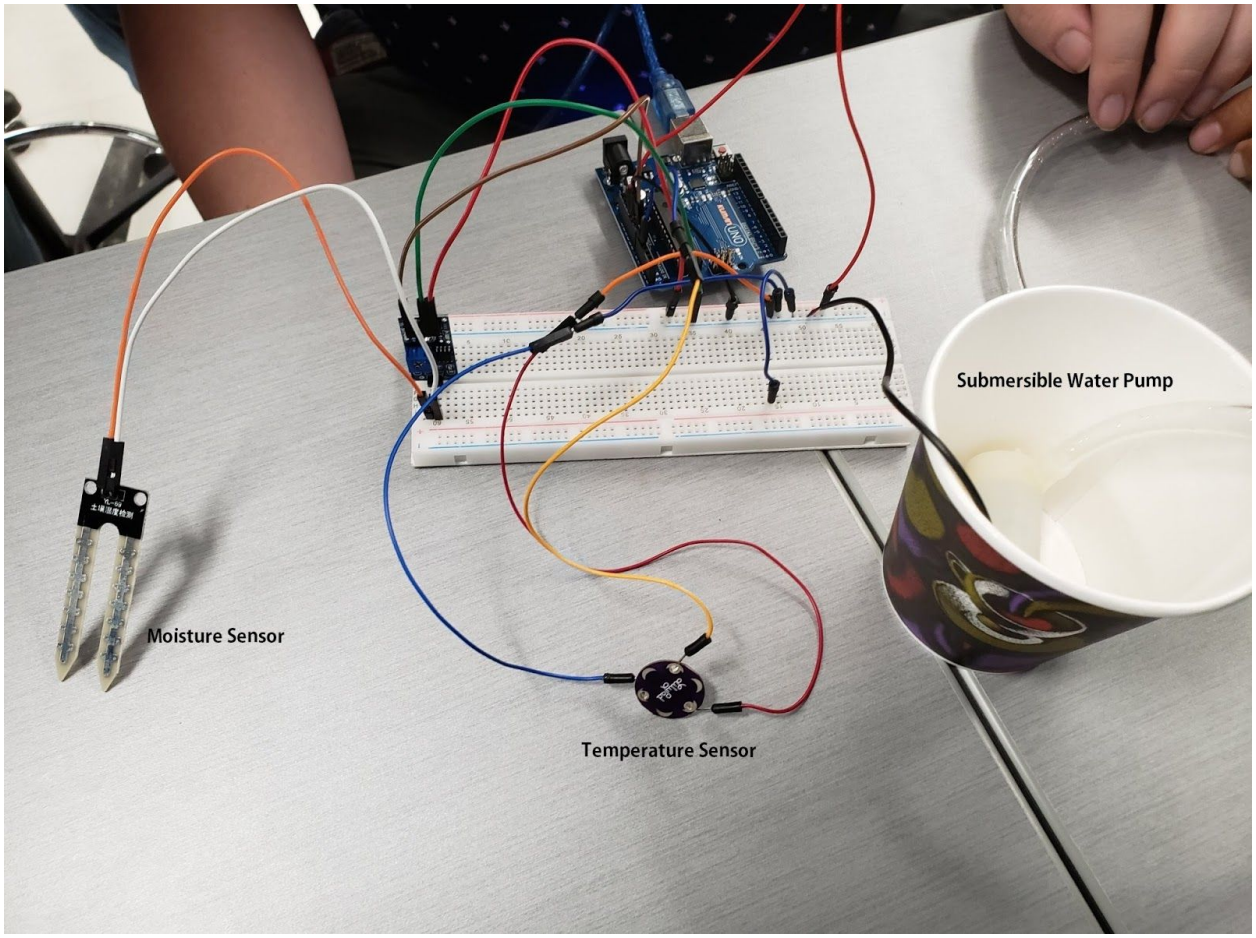

Figure 10: Image showing the sensors and water pump wired up to the Arduino

Appendix B

1. A. Overwatering, "Are You Sure that Plant Needs Water? 5 Signs of Overwatering | Teleflora Blog", Teleflora Blog, 2019. [Online]. Available: https://www.teleflora.com/blog/are-you-sure-that-plant-needs-water-5-signs-of-overwatering/. [Accessed: 19- Sep- 2019].
2. "The Connection Between Growing Plants and Mental Health," *Urban Cultivator*, 10-Sep-2018. [Online]. Available: http://www.urbancultivator.net/connection-growing-plants-mental-health/. [Accessed: 26-Oct-2019].
3. Ada, Lady. "TMP36 Temperature Sensor." Adafruit Learning System, 10-Sep-2018. [Online].Available: https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor. [Accessed: 26-Oct-2019].
4. "How to Use the Soil Hygrometer Module Arduino Tutorial." Ardumotive Arduino Greek Playground, 10-Sep-2018. [Online]. Available: https://www.ardumotive.com/soil-hygrometer-module-en.html. [Accessed: 26-Oct-2019].