**Embedded Linux**
**Self-Watering System Final Report**
Francesca Diamente, Hui Li, Mikaela Stiklickas

## I. Project description

System that keeps the soil at a constant level of moisture, as well as monitor the ambient temperature and humidity around the plants. Self watering the plant if the moisture of the condition is below the expected threshold. If the moisture is below threshold value, that plant gets watered for twenty seconds, if the moisture is above or equal to the threshold, the plant will not be watered. The data is displayed in a mobile friendly web interface.

## II. Project goals (overall)

Create a functioning self watering system and web display to show results.
Learn how to control sensors with code.
Learn how to work with raspberry pi's strictly with command line.
Learn about new hardware (relay, PCF8591 converter).
Learn how the pi's GPIO pins control hardware pieces.
Work efficiently and effectively in a group to develop a large project.
To efficiently use GitHub to help keep the most recent version of our code.

## III. Project implementation details

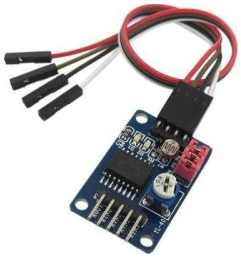
Components for the project are as follows, Raspberry Pi 3, Sqlite3 database hosted on the pi, Moisture sensor, an PCF8591 converter, DHT humidity sensor, Temperature probe, Relay (we used an opto 22) used on the motor to control current, the motor, Tubing attached to motor for water to flow easily to plant, and the Flask framework for web development (python framework).

Here are shopping links and pictures for each component:

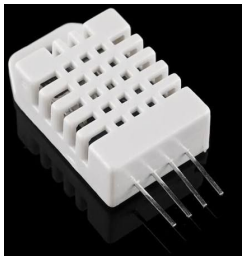

Opto 22 Relay link

Raspberry Pi 3 link

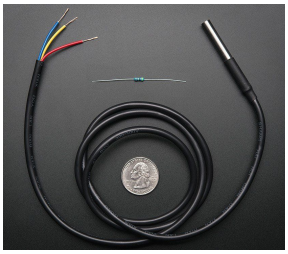

Tubing (we used fish tank tubing)



PCF8591 converter



DHT humidity sensor



Temperature probe

Brushless Motor Submersible Water Pump

Any 5 volt power source, that can be wired to a breadboard

The implementation of each component is described in section IV

## IV. Setup Requirements

Above are photos of our setup. All the hardware is within the basket that acts as a stand

for the plant. The motor is placed through the handle of the basket and in the water bucket. This is just a prototype so, the water supply is external. Ideas for future setup include, building a stand that will hide a water pack and all the hardware and wires.

If someone were to set this project up on their own Pi, they would need to install a few packages …

1. Firstly for the DHT22 sensor, the user needs to install the Adafruit's DHT library. To do this the user enters the following commands inside their project directory:

*apt-get update*
*sudo apt-get install build-essential python-dev*
*git clone https://github.com/adafruit/Adafruit_Python_DHT.git*
*sudo python setup.py install*

2. The user will next need to install the Flask framework. To do this simply type

*sudo apt-get install python3-flask*

3. The user needs to enable the SPI functions on the Raspberry Pi. These instructions are taken from the following link: http://www.instructables.com/id/Wiring-up-a-MCP3008-ADC-to-a-Raspberry-Pi-model-B-/. If this link somehow isn't available in the future the instructions are as follows…. Type: "*sudo raspi-config",* scroll down to Advanced Options and enable SPI. Now reboot. Then type in the command line:

*ls /dev/*

If *spidev0.0* and *spidev0.1* is not there you need to perform an update by doing:

*sudo apt-get -y update*
*sudo apt-get -y upgrade*
*sudo wget http://goo.gl/1BOfJ -O /usr/bin/rpi-update*
*sudo chmod +x /usr/bin/rpi-update*
sudo reboot

After you reboot to activate the update, try to run the command again:

*ls /dev/*

Now the *spidev0.0* and *spidev0.1* should be there, if not you might have made a mistake so retry updating your pi. Once spidev is there, run the command:

*sudo nano /etc/modprobe.d/raspi-blacklist.conf*

Now there is two options, either you will see nothing in the blacklist file or you will see three lines of text. If you see nothing just type:

*blacklist i2c-bcm2708*

If you do see three lines of text delete the hash before the *blacklist i2c-bcm2708* sentence. After you're finished Unblacklisting do:

*sudo reboot*
*sudo apt-get install python-dev git*
*git clone git://github.com/doceme/py-spidev*
*cd py-spidev/*

*sudo python setup.py install*

Enabling the i2c on the Pi is required. Specifically, the user needs to add "*dtoverlay=w1-gpio*" to the file in the directory /boot/config.txt, and then do

*sudo reboot*
*sudo modprobe w1-therm*
*sudo modprobe w1-gpio*

4. In addition they need to set up the sqlite3 database on the pi. If sqlite3 package does not exist on the pi, do

*sudo apt-get install sqlite3*

to install the package. To set up the database enter the following commands:

*sqlite3 project_data.db;*

The terminal then goes into the sqlite3 shell, you should see the "sqlite>" appear on your command line. Next to create the table type exactly sqlite>

*CREATE TABLE Data(data_time text,temp1 double,humidity double,temp2 double,moisture double,pumpStatus int);*

The database is now set up.

5. One would also need to insert the value (starting with 28-xxxxxxxxxxxx) from their pi in the first line of code within the temperature function: open("/sys/bus/w1/devices/28-041692b69eff/w1_slave"). The user would go to the directory /sys/bus/w1/devices and then perform the "ls" command to get the number for their device. Simply go into the main.py function, under the temperature function and replace the value with your pi's unique number.

6. Installing python on the pi is as simple as

*sudo apt-get install python3*

7. Simply install the data analysis toolkit used with the command:

*sudo apt-get install python-pandas*

## V. Project Components How you broke up the project into components – describe each component

- Hardware - wire up all the hardware components

Moisture sensor - This device is inserted into the soil to gather data on its moisture reading. Sends out a 1 or 0 if the moisture threshold is below or above.

Humidity sensor - This device collects the humidity within the air, as well as the temperature of the air.

Temperature sensor - This device records the temperature of the soil.

Motor/Relay - The purpose of the motor is to be turned on dependent of the moisture threshold. This motor is what moves water through the tubing to the plant. The purpose of the relay is to give control over the current going towards the motor.

External power source - This device is used for the direct input of power that the

motor needs to utilize in order to have enough power to make water flow to the plant.

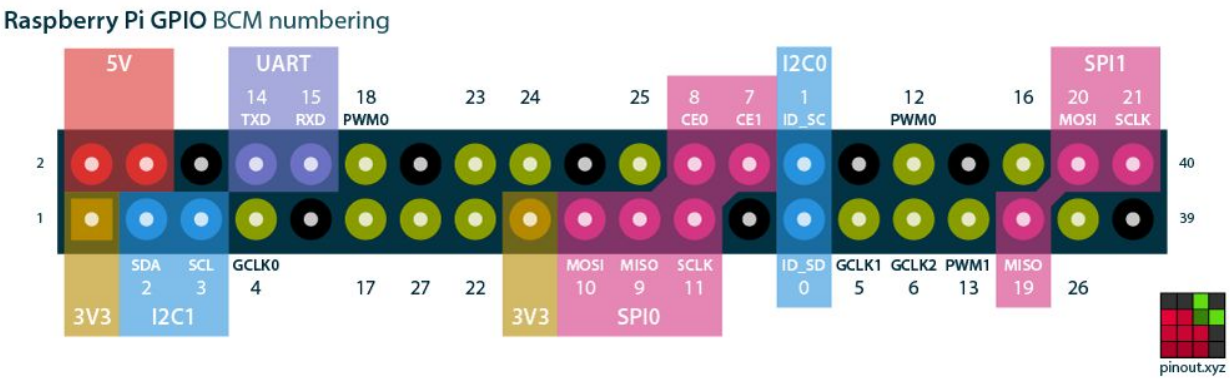
- Software
  - Main loop
  - Logging
  - Moisture function
  - Humidity function
  - Temperature function
  - Motor function
- Database setup
- Web
  - Flask setup
  - Chart using Charts.js
  - Table using html

**VI. For each component above, describe what work was done, including hardware and software used.**

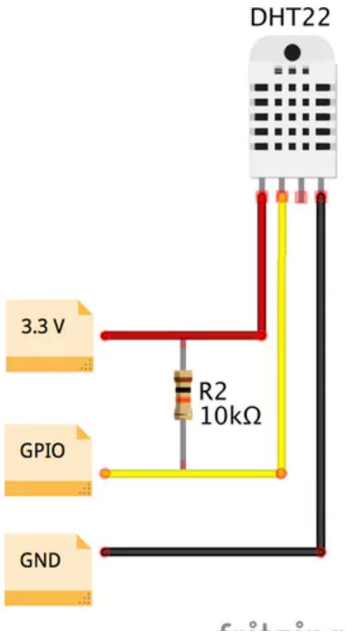
Here is a basic pseudo for our whole system

```
Main:
        Calls LogData()
        Sleeps(hours)
LogData:
        Calls each function and stores the return values
        Records all info in DB
TempSensor:
        Reads and returns value from temp sensor
HumiditySensor:
        Reads and returns values from humidity sensor
MoistureSensor:
        Reads and returns value from moisture sensor
WaterPump(moisture results passed):
        if(moisture results <= threshold)
                Supply power to motor
                Sleep(seconds)
                Remove power from motor
                Return 1
        Else
                Return 0
```

For the hardware here is a diagram of the GPIO pins on the raspberry pi 3

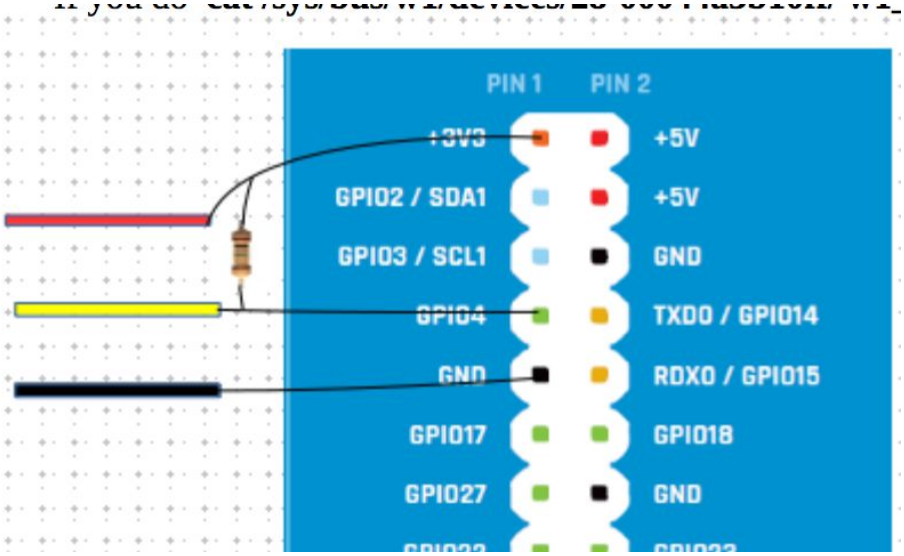



Humidity Sensor setup diagram:




The humidity GPIO was set to pin 22. Power and the GPIO require a 10k resistor. The humidity function can be found on github, but it just uses Adafruit's DHT read to get the humidity and temperature data.

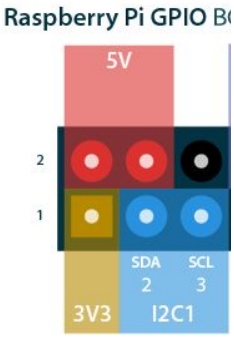
Temperature sensor setup:

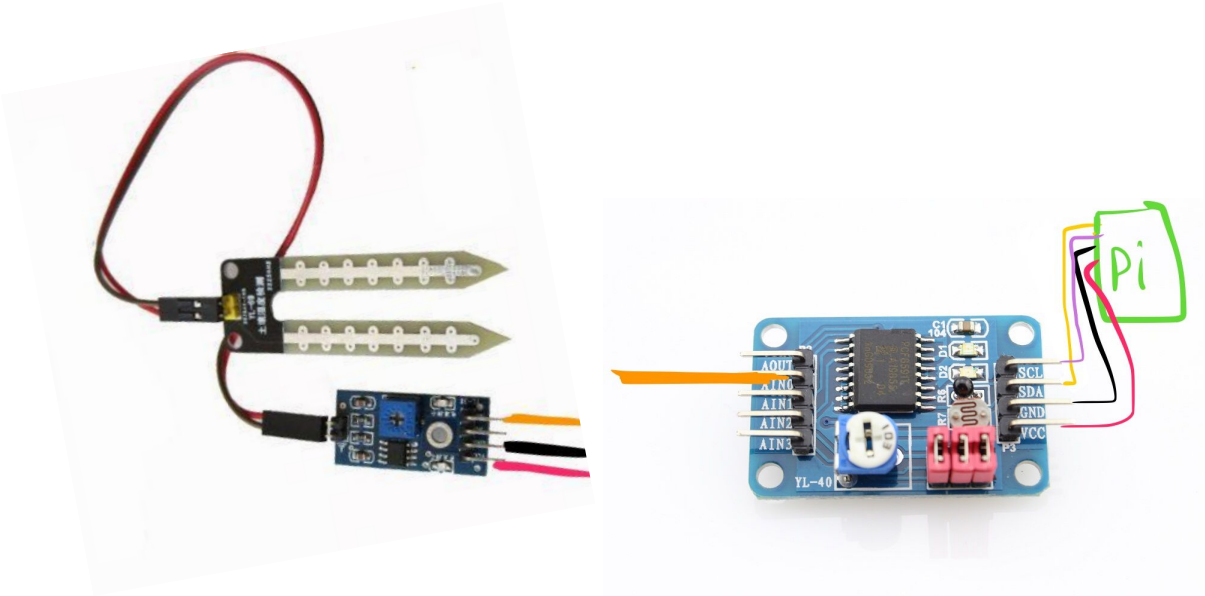



The GPIO pin for the temperature probe is pin 4. The power and GPIO require a 4.7k resistor. The temperature function can be found on github.

Moisture setup diagram:

We used an ADC converter specifically the Pcf8591, to convert the digital data to analog. This allowed us to get a numerical value from the moisture sensor instead of a 0 or 1. Our code did not require us to assign a GPIO to the moisture. Instead we connected the converter to the two I2C GPIO pins. These are the two blue pins numbers 2 and 3. We used these channels to get a digital reading of the moisture from our sensor. The SCL line on the converter (purple wire in diagram below) goes to pin number 3, and SDA (yellow wire on diagram below) goes to pin number 2. On the left hand side of the converter, the 2nd pin (AIN0) connects to the GPIO out of the moisture sensor (the 4th pin) (orange wires in diagram below)

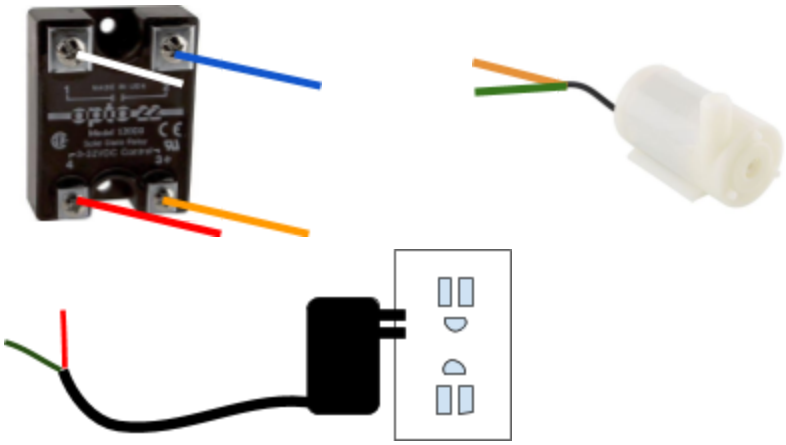





The moisture function executes by reading the value from the i2c bus 50 times and creates an average read to return. This is done to get a more accurate reading of the moisture within the soil. The actual code can be found on github.

Relay, Motor and external power source:

Relays are usually operated through a small current that in turn acts as a binary 1 and 0 switch. The purpose of these relay switch system, also considered to be an electromagnetic switch, is so that the user now has the ability to turn on (1) or off (0) a much larger electric current. Since our water pump motor only has inputs of power and ground, we really needed to use this relay to be able to control when we wanted to motor to be on or off. Without this relay, we were not able to have full control over the water pump motor, which in turn, created more problems on our circuitry than planned.



The power wire of the external power source, and the power line of the hardware you are trying to control (motor) go to lines 4 and 3 on the relay respectively. The switch aspect of the relay is controlled by the GPIO pin. The GPIO pin 27 is hooked up to line number 3 of the relay. The last line of the relay numbered 1 is hooked up to ground. The motor function pseudo is:

```
startMotor(moisture_percent):
    if(moisture_percent >= needed_moisture):
        Supply power
        time.sleep(25)
        Remove power
        return "1"

    elif(moisture_percent <= needed_moisture):
        Remove power
        return "0"
```

Log Data Function: The log data function will call each sensor function and place the value that they return into the database in a single query. The pseudo is:

```
logData():
    Connect to db
      try:
        [t,F]=readTemp()
        [h, t2] = readHumidity()
        [m] = readMoisture()
        [w] = startMotor(m)
        (‘insert into Data values(?,?,?,?,?,?)', (t,F,h,t2,m,w))
```

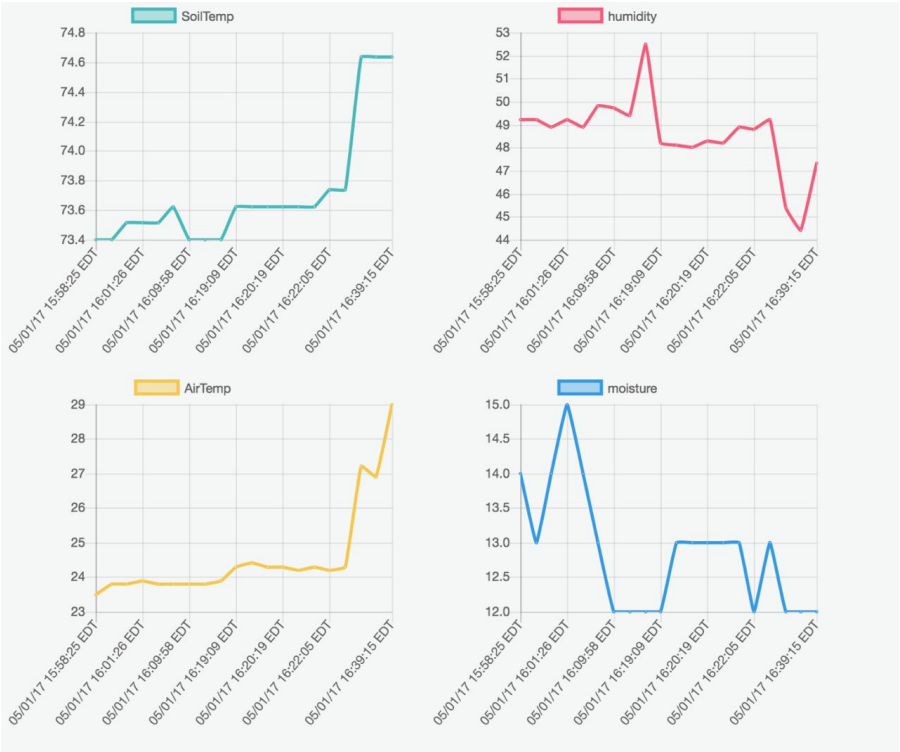
Main loop

The main loop is simple and just calls the logData function and then sleeps for a desired amount of time, and repeats. We have our sleep for 2 hours at a time.

Web:

The mobile friendly web is hosted on the raspberry pi. To access the web enter the following in the browser http://{your pi’s ip}:8088/list. The information you will see in the chart is all the data within the database. The chart is a graph of the moisture over time so the user gets a visual feel for the history of their plant. Using charts.js we were able to create the chart. Charts.js is a javascript library that is used to create animated, interactive graphs to include on web pages.

Here are pictures from the web display:




| Time stamp | Soil Temp | Humidity | Air Temp | Moisture | Pump Status |
|---|---|---|---|---|---|
| 05/01/17 15:58:25 EDT | 73.40 | 49.20 | 23.50 | 14.0% | 1 |
| 05/01/17 15:58:36 EDT | 73.40 | 49.20 | 23.80 | 13.0% | 0 |
| 05/01/17 16:01:12 EDT | 73.51 | 48.90 | 23.80 | 14.0% | 1 |
| 05/01/17 16:01:26 EDT | 73.51 | 49.20 | 23.90 | 15.0% | 1 |
| 05/01/17 16:02:48 EDT | 73.51 | 48.90 | 23.80 | 14.0% | 1 |
| 05/01/17 16:05:02 EDT | 73.62 | 49.80 | 23.80 | 13.0% | 0 |
| 05/01/17 16:09:58 EDT | 73.40 | 49.70 | 23.80 | 12.0% | 0 |
| 05/01/17 16:10:10 EDT | 73.40 | 49.40 | 23.80 | 12.0% | 0 |
| 05/01/17 16:10:21 EDT | 73.40 | 52.50 | 23.90 | 12.0% | 0 |
| 05/01/17 16:19:09 EDT | 73.62 | 48.20 | 24.30 | 12.0% | 0 |
| 05/01/17 16:19:23 EDT | 73.62 | 48.10 | 24.40 | 13.0% | 0 |
| 05/01/17 16:20:07 EDT | 73.62 | 48.00 | 24.30 | 13.0% | 0 |
| 05/01/17 16:20:19 EDT | 73.62 | 48.30 | 24.30 | 13.0% | 0 |
| 05/01/17 16:21:42 EDT | 73.62 | 48.20 | 24.20 | 13.0% | 0 |
| 05/01/17 16:21:54 EDT | 73.62 | 48.90 | 24.30 | 13.0% | 0 |

## VII. Summarize your project's “good” and “bad” aspects, “easy” and “tough” aspects, from the group's point of view. List what was achieved and what was not achieved.

Easy:

-Temperature sensor - easy to set up and do the coding since we have already done it before in a previous assignment.

-Water Pump software - Was similar to that of the blinking LED code we did on an earlier assignment. This is simple enough since we just need to supply power to make the pump run. The only two difficult parts about the water pump code was determining an appropriate threshold. The other hard aspect was realizing that we needed a relay to control the current.

-Hardware Setup for all Sensor - Most of these like the temperature, and humidity were generic set ups that either had pins going to ground and power, or just power and a GPIO pin. These types of wiring diagrams could be found on data sheets found through research.

Hard:

-Moisture sensor- Needed to use a PCF8591 converter to get a digital signal. Before we could just get a 1 or 0 to tell us if there was moisture present in the soil. This however is not as precise as using a digital signal. We can set a more exact threshold, which can be useful for more delicate types of plants. The setup of this converter was not hard, rather the coding aspect was challenging. To use this converter we needed to use an i2C bus. The way we were reading the code would display 4 results from the 4 different channels, which took some time to figure out which channel we needed to be reading. With a slight manipulation of the code we were able to get the desired moisture reading. After performing a calculation we got a more sensible result, 2 digit number that reads like a percentage to the user.

-Water Pump- The issue with the water pump wasn't within the software, but with the hardware. The software for the water pump was very simple, and was similar to the blinking led code we had written before. The only other thing to keep in mind about the water pump code, is that the pump needs to have power for a certain period of time. The water pressure was kind of low, so to account for that we supplied power to the motor for 30 seconds., This allowed enough water to flow and water the plant, without over watering it. Before we realized we needed to use a relay, our system seemed to have a bug. Whenever we would run our code, the other sensors would get blown out by the external power source. After much research, we realized our bug could easily be fixed with a relay. Since the water pump had wires only going towards ground and power, we had to figure out how to control the on (1) and off (0) of this device. Using the relay allowed us to use an electromagnetic switch to control the larger current that goes into the motor.The power wire of the external power source, and the power line of the hardware you are trying to control (motor) go to lines 4 and 3 on the relay respectively. The switch aspect of the relay is controlled by the GPIO pin. The GPIO pin 27 is hooked up to line number 3 of the relay. The last line of the relay numbered 1 is hooked up to ground.

-Flask- Charting data using charts.js had a few issues our first time around, specifically getting the data to the chart. Flask framework is basically a set of tools and libraries that make it easier to build web applications in Python. However, we have never used this

before so there was a learning curve. Following tutorials online ultimately lead us to the successful web design.

## VIII. List group members, and each member's contributions.

Hui Li: All of the Humidity function and made humidity able to be called without user having to input the pin number in command line. Moisture function research and debug, also came up with the computation/conversion to a 2 digit value. The entire Flask and web setup. Got the web charts working with chart.js. Set up the database table.

Francesca Diamente: Moisture function research and debug, and made the function return an average read. Added C to F conversion in humidity function. All hardware setup. Physical design setup (including tubing). System's software design/layout. The LogData() function. Had each individual function work together/return appropriately. Web/Flask research. Motor function (determined appropriate threshold, and made the pump for a period of time). Fixed the current issue to have the motor pump & relay working together (got relay/wired it). Wrote the iterm progress report and the Final Report and Presentation.

Mikaela Stiklickas: The temperature function. The initial Motor function. Defined sensor variables. Soldered the temperature probe and the motor wires. Helped transfer hardware setup to smaller more compact design. Helped find the relay in which we got the water pump motor problem solved. Helped with the Final Report. Researcher. Helped write the Final Report and Presentation.

## IX. List of References/Resources


"Chart.js." *Chart.js* | *Documentation*. N.p., n.d. Web. 12 May 2017.

"Devin Ellis." *Devin Ellis* | *DHT11/22 Temperature Sensors Using the WebIOPi REST API*. N.p., n.d. Web. 5 May 2017. https://www.devin.cl/blog/dht11-webiopi/

"How Do Relays Work?" *Explain That Stuff*. N.p., 23 Apr. 2017. Web. 10 May 2017. http://www.explainthatstuff.com/howrelayswork.html

Seempie. "Analog Sensor Input Raspberry Pi Using a MCP3008: Wiring/installing/basic Program." *Instructables.com*. Instructables, 12 May 2016. Web. 5 May 2017.

http://www.instructables.com/id/Wiring-up-a-MCP3008-ADC-to-a-Raspberry-Pi-model-B-/

*Solid-State Relays*. Digital image. *Explain That Stuff*. Opto22, 1 Jan. 2006. Web. 5 May 2017.https://www.opto22.com/documents/0859_Solid_State_Relays_data_sheet.pdf

*SUNY New Paltz. Week 6. In CPS342: Embedded Linux: Spring 2017 [RaspPi_setting_up_temp_sensor]. Retrieved from https://bbnewpaltz.sln.suny.edu/bbcswebdav/pid-2139070-dt-content-rid-3815447_2/courses/spring17_CPS342_01/RaspPi_setting_up_temp_sensor.pdf*