CSC2003 - Group 123
Smart Pot Report

Name	**Student ID**
Chia Cheng Yee Shirley	1901894
David Chua Chong Ren	1901870
Ng Jia Cheng	1901825
Tan Wei Jian	1901864
Puan Jin Yao Daren	1901822

Table of Contents

1. Introduction

Plant owners often do not know that their plants are suffering until the leaves have turned brown or fallen. Maintaining healthy plants can be time-consuming and if the plant owners missed a schedule for the plant or did not pay attention to them, the plant's growth may be largely affected. It is also hard to know if the spot, which the plant is placed, is best for the plant's growth.

1.1 Motivation

The motivation of this project was to explore the possibilities of using IT technologies to tackle some of the challenges faced when taking care of plants, especially when plant owners are exceptionally busy with their schedules and to assist them in monitoring and making better decisions with regards to caring for their plants.

1.2 Proposed Solution

The solution is a smart pot where temperature, humidity, moisture, and light data will be collected with the help of multiple input sensors for analysis. Based on the data analyzed, we are looking to predict growth behavior of the plant and relay the prediction to the user through an LCD display using the Pioneer600. In addition, the user can also view data which are pushed into ThingsBoard without interacting with the Pioneer600.

2. System Overview

2.1 System Architecture

The system architecture diagram is used to display the overall relationships between the components.

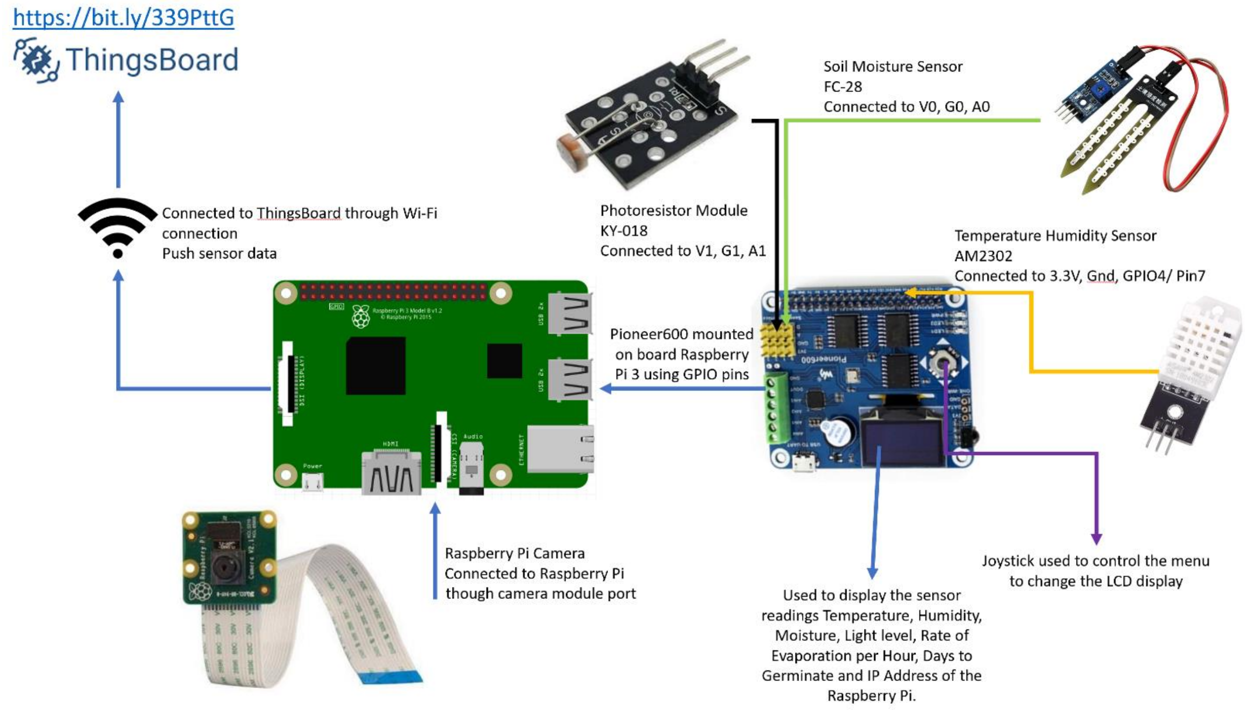

https://bit.ly/339PttG
ThingsBoard
Connected to ThingsBoard through Wi-Fi connection
Push sensor data
Photoresistor Module
KY-018
Connected to V1, G1, A1
Soil Moisture Sensor
FC-28
Connected to V0, G0, A0
Temperature Humidity Sensor
AM2302
Connected to 3.3V, Gnd, GPIO4/ Pin7
Pioneer600 mounted on board Raspberry Pi 3 using GPIO pins
Raspberry Pi Camera
Connected to Raspberry Pi though camera module port
Used to display the sensor readings Temperature, Humidity, Moisture, Light level, Rate of Evaporation per Hour, Days to Germinate and IP Address of the Raspberry Pi.
Joystick used to control the menu to change the LCD display

Figure 1: System Architecture

2.2 Hardware Requirements

The Raspberry PI 3 will be responsible for collecting the data from the various sensors connected to it through the Pioneer600 board and pushing the sensor data collected into ThingsBoard. Additionally, it also takes pictures of the plants using the camera which is connected to it. It also computes the machine learning algorithms for the necessary predictions.

Figure 2: Raspberry Pi 3

The Pioneer600 is an expansion board that supports the Raspberry Pi 3 which has an inbuilt OLED Display that will be used to display the temperature and humidity of the area, moisture level of the soil, light level of the area, rate of evaporation per hour, days to germinate, plant health status, last updated date and time. The Pioneer600 also has integrated ADCs and available libraries for the connectivity of sensors.

Figure 3: Pioneer600

AM2302 is a Temperature & Humidity sensor which will be used to measure the temperature and humidity of the surrounding area for the plant. Using the AM2302, the gathered temperature and humidity data will be used to make decisions on the plant's germination rate and help user make decision on whether the plant is suitable for that environment. It is connected to the GPIO Pin 4 on the pioneer600 or Pin No.7.

Figure 4: AM2302 Temperature & Humidity Sensor

KY-018 is a photoresistor module which will be used to detect the amount of light in the area. The amount of light detected will help users determine whether the light condition is optimal for the plant to germinate or grow. It is connected to the pioneer600 using the onboard sensor interface on Pin A1 which is connected to the PCF8591 8-bit AD/DA converter using the I2C interface.

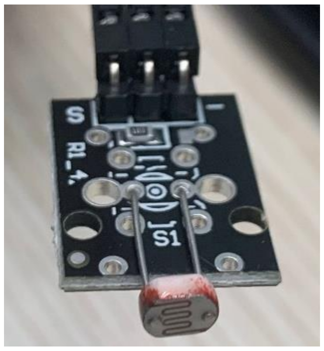


Figure 5: KY-018 Photoresistor Module (Light Dependent Resistor)

The soil moisture sensor will be used to measure the moisture level of the soil. The moisture level of the soil will determine whether there is sufficient water for the plant to grow and whether there is a need to water the plant. It is connected to the pioneer600 using the onboard sensor interface on Pin A0 which is connected to the PCF8591 8-bit AD/DA converter using the I2C interface.

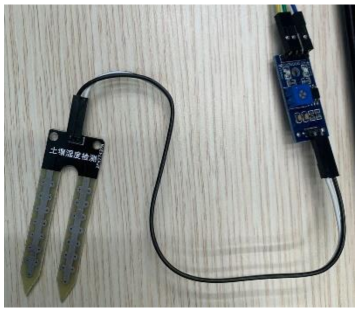


Figure 6: FC-28 Moisture Sensor

A Raspberry PI Camera V2 will be used to capture images of the leaves to determine the behavior of the plant based on color detection using machine learning algorithm. It is connected to the Raspberry Pi through the camera module port on the board itself using the Serial Camera Control Bus (SCCB).

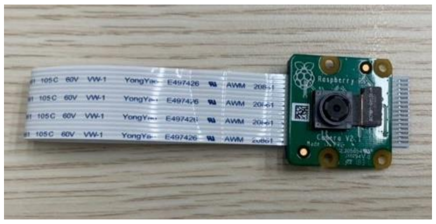


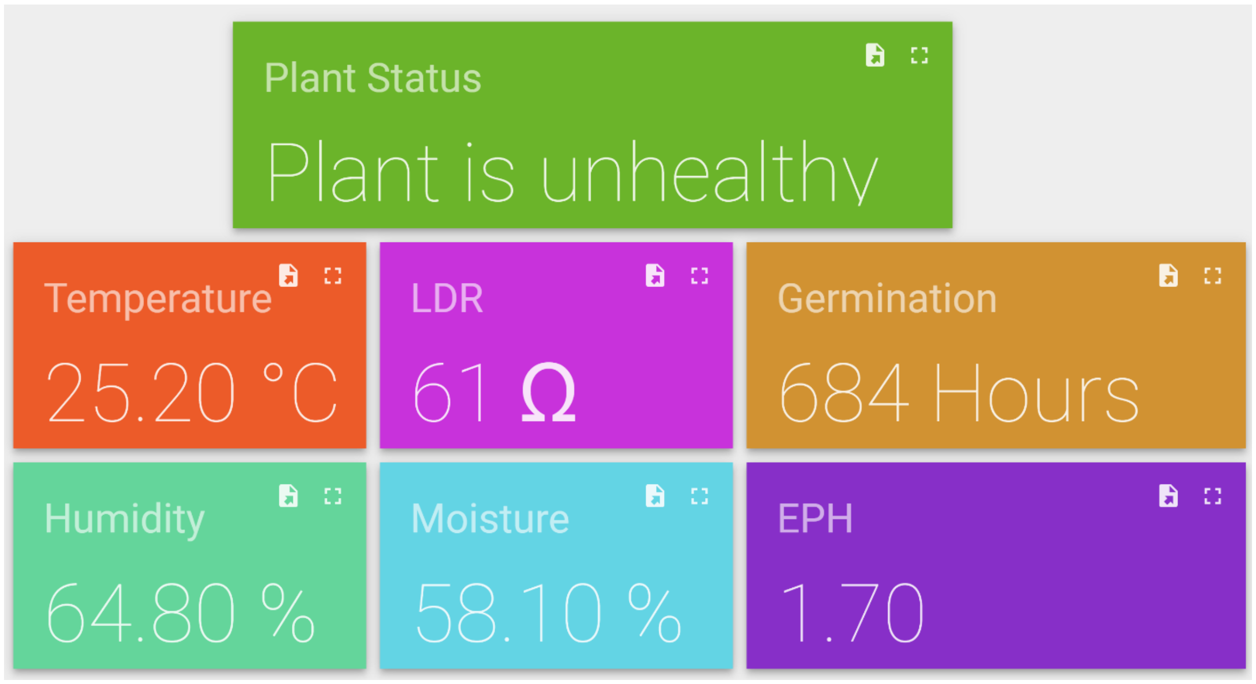
Figure 7: Raspberry PI Camera V2

2.3 Software Requirements

Raspberry PI Operating System, known as Raspbian, will be installed, and used on the Raspberry PI 3. The integration and execution of codes will be done on the Raspbian OS. There are additional libraries that are being used for the code to function properly. These libraries include:

1. Paho MQTT Library
   - Allows the Raspberry Pi to post the sensor data to the ThingsBoard dashboard.
2. Adafruit_DHT Library
   - Allows the Raspberry Pi to communicate the AM2302, Temperature & Humidity Sensor.
3. JSON Library
   - Encoding and decoding JSON data.
4. SMBus Library
   - It allows SMBus access through the I2C /dev interface on Linux hosts.
5. Time Library
   - Provides the functions for working with time in the python code.
6. DateTime Library
   - To get current date and time
7. Pandas Library
   - Open-source data analysis and manipulation tool.
8. Sklearn Library
   - Predictive data analysis.
9. PiCamera Library
   - For the camera to work with the Raspberry Pi.
10. OpenCV Library
    - Used for the purpose of computer vision.
11. Numpy
    - Used for working with arrays and linear algebra.
12. threading Library
    - Used for timer interrupts
13. ctypes Library
    - Used for calling C files in python

ThingsBoard is used to display the moisture level of the soil, temperature and humidity of the surroundings, amount of light, germination rate of the seed and the evaporation rate of the soil. This data is then calculated and will display the status of the plant.



Plant Status

Plant is unhealthy

Temperature

25.20 °C

LDR

61 Ω

Germination

684 Hours

Humidity

64.80 %

Moisture

58.10 %

EPH

1.70

Figure 8: ThingsBoard Dashboard

2.4 Initial Data Collection

Before building a dataset to use for Machine Learning algorithms training, we first built a simple prototype using the NodeMCU board, which can collect and push data via MQTT to ThingSpeak. It is also another dashboarding service that allows posting and displaying of data in graphical formats. The data is then collected and exported for the use of training the Machine Learning algorithms.

The NodeMCU code base is in C++ using the Arduino IDE. Sensors used are, DHT22 Temperature & Humidity sensor and the FC-28 Soil Moisture sensor. Additional libraries used are ThingSpeak.h, ESP8266Wifi.h and DHT.h.

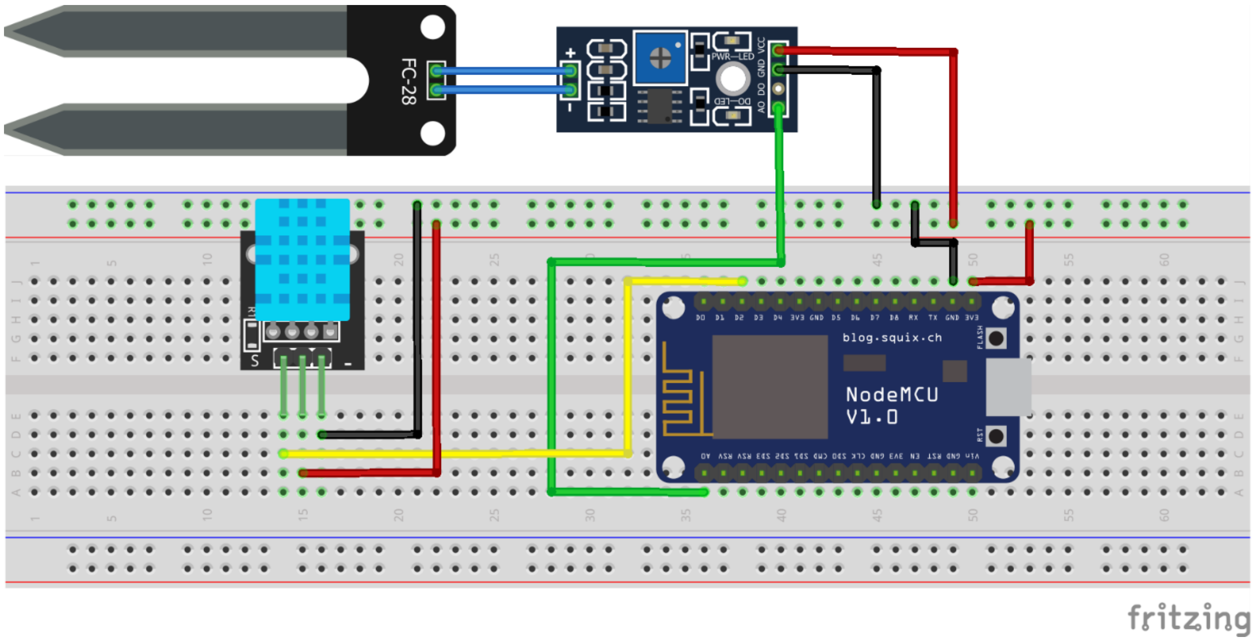


Figure 9: NodeMCU, DHT11, FC-28

2.5 Machine Learning Algorithm

Supervised learning will be used in the project solution. This includes multiple linear regressions and decision trees.

Multiple linear regressions will be used to predict the number of hours taken for plant to germinate. There will be a *get_Germination* function which will take in the training data from a csv file. The data is then compared to the sensor data that is being collected by the Raspberry Pi by inserting the data into a graph, from there the machine learning algorithm will predict the required hours remaining to germination. The predicted result will then be dumped into a JSON type format and pushed to ThingsBoard. The result will also be appended to an array and return to the Pioneer600 to be displayed on the onboard OLED screen.

Multiple linear regressions will also be used to predict the number of hours before the plant needs to be watered. A *get_EPH* function will be integrated to determine the evaporation per hour. The evaporation per hour will be pushed to ThingsBoard using JSON and appended to the array for Pioneer600 OLED screen for users to view.

Decision trees will be used to determine the health status of the plant based on the color of the leaf. There will be 3 functions working together to determine the health status of the plant – *take_picture*, *get_Brown* and *get_Green*. The *take_picture* function will take a picture of the plant and the image will be passed to the color detection algorithm. The get_Brown function will process the image for any brown pixels and if the number of brown pixels hits the threshold, it will inform the user that the plant is unhealthy. Concurrently, the get_Green function will process the image for any green pixels and if the number of green pixels hits the threshold, it will inform the user that the plant is healthy.

3. Flow Chart

3.1 General System Flow

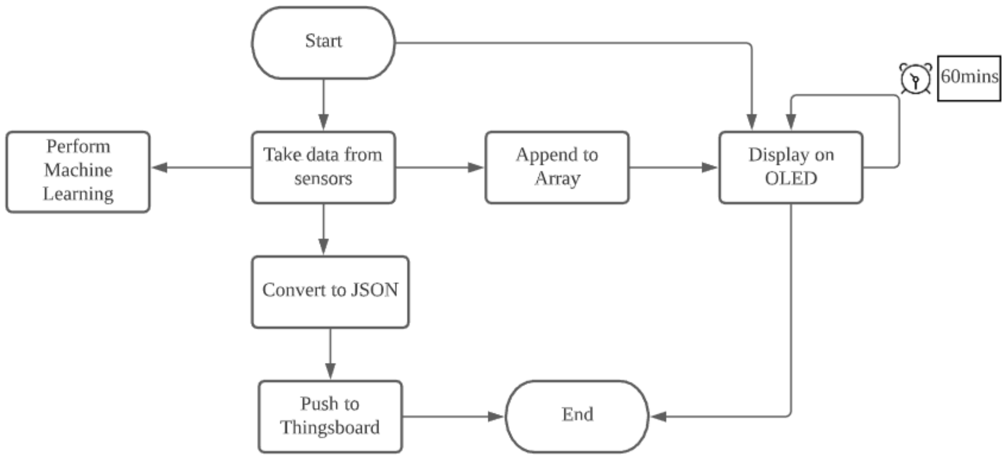


Figure 10 : General System Flowchart

3.2 EPH System Flow

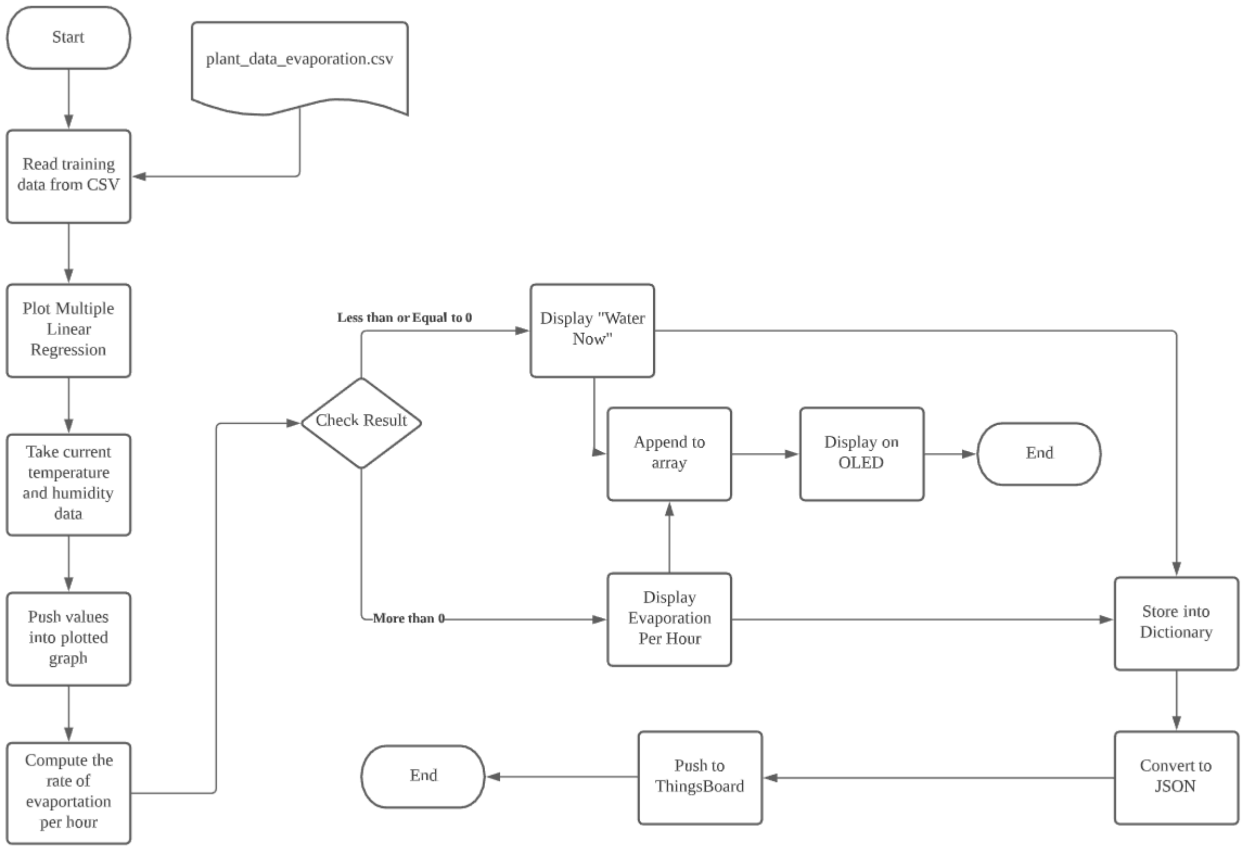

Start
plant_data_evaporation.csv
Read training data from CSV
Plot Multiple Linear Regression
Take current temperature and humidity data
Push values into plotted graph
Compute the rate of evaportation per hour
Check Result
Less than or Equal to 0
Display "Water Now"
More than 0
Append to array
Display on OLED
End
Display Evaporation Per Hour
Store into Dictionary
Convert to JSON
Push to ThingsBoard
End

Figure 11: EPH FlowChart

3.3 Plant Health Status System Flow

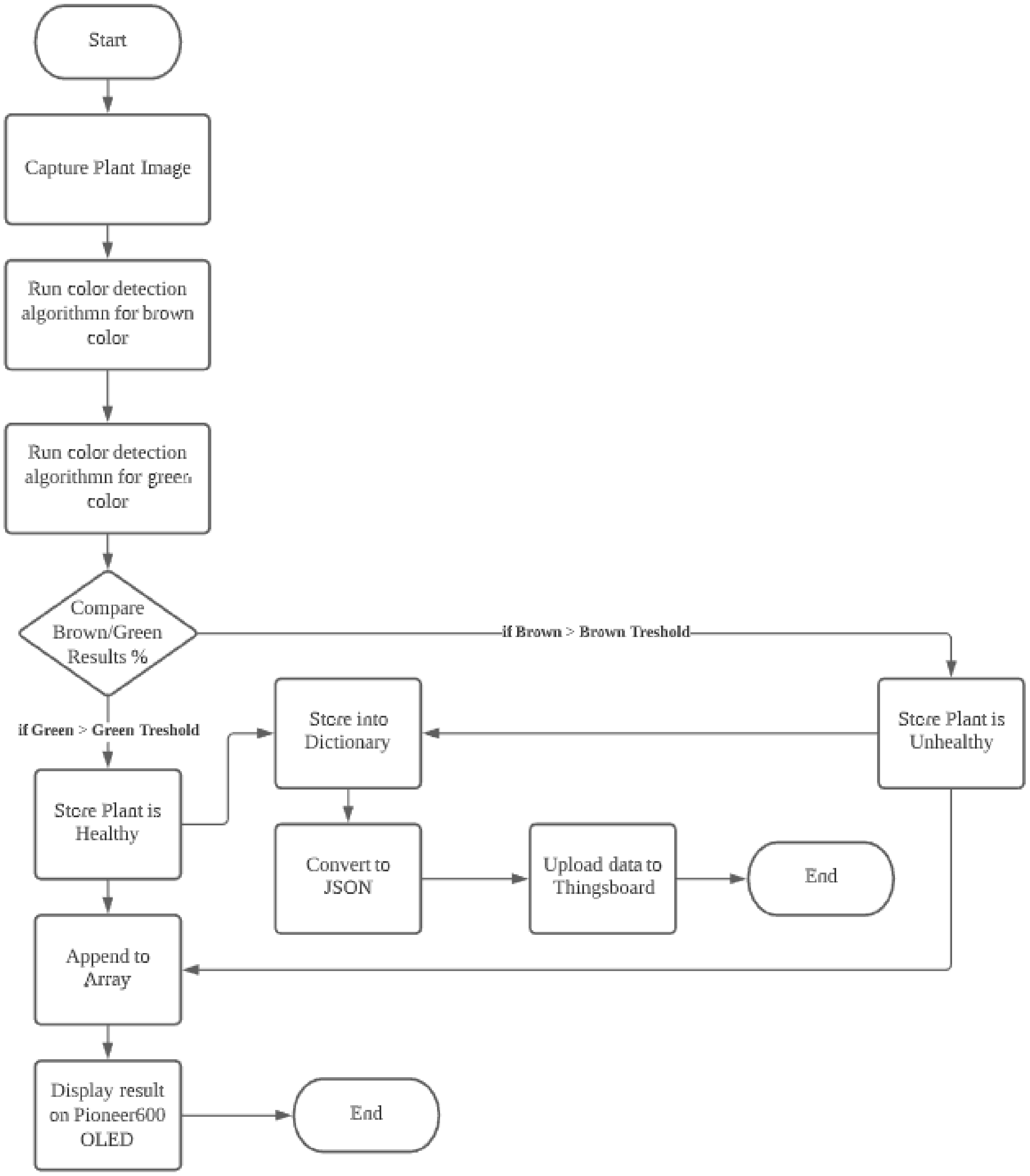
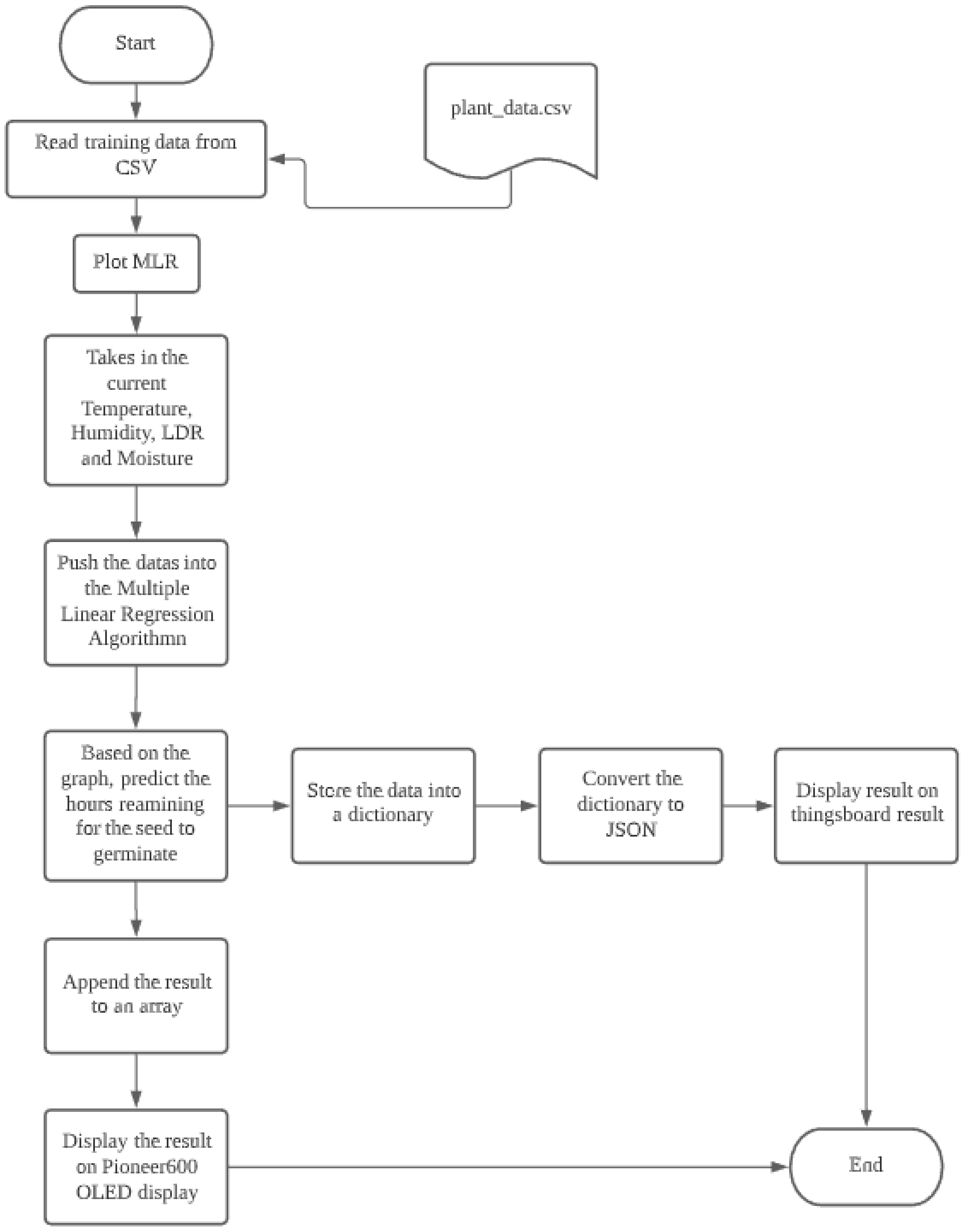

Start
Capture Plant Image
Run color detection algorithmn for brown color
Run color detection algorithmn for green color
Compare Brown/Green Results %
if Brown > Brown Treshold
Store Plant is Unhealthy
if Green > Green Treshold
Store Plant is Healthy
Store into Dictionary
Convert to JSON
Upload data to Thingsboard
End
Append to Array
Display result on Pioneer600 OLED
End

Figure 12: Plant Health Status Flowchart

3.4 Germination Rate System Flow



Start
plant_data.csv
Read training data from CSV
Plot MLR
Takes in the current Temperature, Humidity, LDR and Moisture
Push the datas into the Multiple Linear Regression Algorithmn
Based on the graph, predict the hours reamining for the seed to germinate
Store the data into a dictionary
Convert the dictionary to JSON
Display result on thingsboard result
Append the result to an array
Display the result on Pioneer600 OLED display
End

Figure 13: Plant Germination Rate Flowchart

4. System Testing

4.1 Sensor Testing

Component Name	Purpose of Test	Test Method	Expected Results
Raspberry Pi 3 Model B+	Boot Successful	Switch on power and wait for it to load.	Boot Successful depicted by Built-In Green LED blinking on Pi
	Connected to Internet	Turn on mobile hotspot or router and wait for pi to auto connect.	Check the connected devices on your mobile device or router and ensure that there is a raspberry pi device connected to it.
Pioneer600	Working OLED navigation with joystick	1. Display "Last Updated" screen on start (Menu 1) 2. Joystick Up to enter Submenu of Menu 1. 3. Joystick Right to enter Menu 2, Readings. 4. Joystick Up to enter Submenus of Menu 2. 5. JoyStick Right to enter Menu 3. 6. JoyStick Right to enter Menu 4. 7. JoyStick Up to enter Submenus of Menu 4.	1. Arrive at Last Updated OLED display. 2. Arrive at Status and Live timing of the day. 3. Arrive at Temperature reading display. 4. Arrive at other Reading Page. 4.1 Arrive at Moisture reading display. 4.2 Arrive at LDR reading display. 4.3 Arrive at Humidity reading display. 5. Arrive at Close App page. 6. Arrive at Evaporation per hour ML page. 7. Arrive at other ML Page. 7.1 Arrive at Plant Status ML page. 7.2 Arrive at Germination hours ML page.
AM2302 (Temperature & Humidity)	Working AM2302	Connect AM2302 to Pioneer600 – 3.3V, GND, GPIO4/PIN7. Run the AM2302 code. Place AM2302 into a palm and cover it.	Value should increase.

KY-018 (Photoresistor)	Working KY-018	Connect KY-018 to Pioneer600 – V1, G1, A1. Run the KY-018 code. Place KY-018 into a palm and cover it.	Value should decrease.
FC-28 (Soil Moisture)	Working FC-28	Connect FC-28 to Pioneer600 – V0, G0, A0. Run the FC-28 code and check if the data is collected. Place FC-28 into a cup of water.	Value should increase.
Raspberry Pi Camera V2	Working Camera	Connect Camera to Raspberry Pi through camera module port. Run the *rapistill –o Desktop/image.jpg* command on the terminal.	An image.jpg file should be seen on the desktop.

4.2 Color Detection

The color detection function is used to determine the health status of the plant based on the color of the leaves: green or brown. 2 sample leaves set were prepared to test the color detection function.

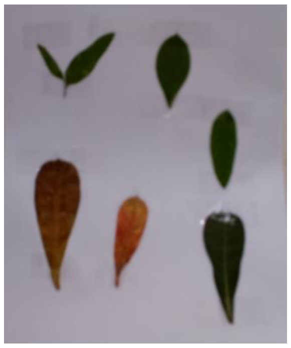


Figure 14: Sample Set 1

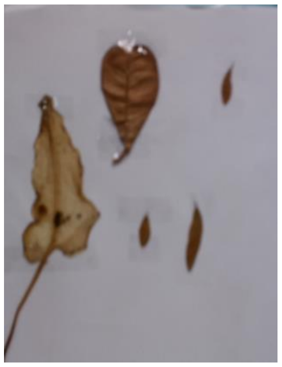


Figure 15: Sample Set 2

Finding the Right HSV Values

Using the image of Sample Set 1 (Figure 10), we can extract out the RGB pixel value of the green and brown leaves. We will use the green and brown leaves circled here to find out the RGB value of the color green and brown in this image which works out to be around rgb(24, 33, 4) for green and rgb(69, 27, 1) for brown.

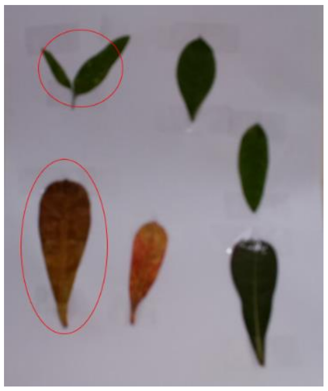
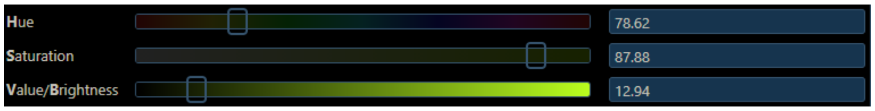


Figure 16: Sample Set 1 with Circled Green and Brown Leaf

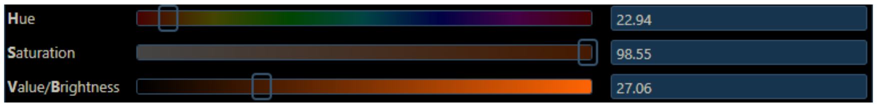
With the help of online tool http://colorizer.org/ , we key int the RGB values for green to get the HSV values. The HSV value of green works out to be around hsv(79, 88, 13).



Hue 78.62
Saturation 87.88
Value/Brightness 12.94

Figure 17: HSV Value of Circled Green Leaf

The HSV value for brown works out to be around hsv(23, 99, 27). The color brown is particular challenging to find the range we want as can be seen from the Hue spectrum the far right end of the Hue spectrum seems like a reddish kind of brown. We are unable to cover that end of the brown spectrum without the algorithm detecting every other color in between hence we settled on the left end of the Hue spectrum after testing with several brown leaves.



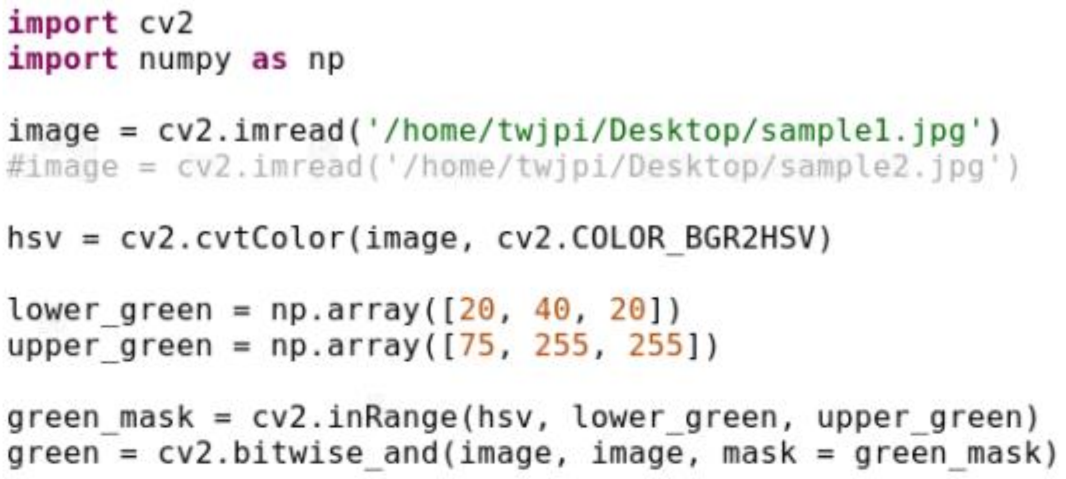
Hue 22.94
Saturation 98.55
Value/Brightness 27.06

Figure 18: HSV Value of Circled Brown Leaf

Now that we have the HSV value for green and brown we can use them as a middle point to then slowly fine tune the value to cover the spectrum of green and brown we want. The values would first need to be converted to the range that OpenCV accepts which is H[0,179], S[0,255], V[0,255] as the HSV range http://colorizer.org/ provides is H[0,360], S[0,100], V[0,100].

Color Green

The experiment is performed on the images taken by the Raspberry Pi Camera of Sample set 1 and 2. The images are then used with the color detection algorithm to detect green. The algorithm is accomplished with the help of python libraries *cv2* and *numpy*.



```
import cv2
import numpy as np

image = cv2.imread('/home/twjpi/Desktop/sample1.jpg')
#image = cv2.imread('/home/twjpi/Desktop/sample2.jpg')

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

lower_green = np.array([20, 40, 20])
upper_green = np.array([75, 255, 255])

green_mask = cv2.inRange(hsv, lower_green, upper_green)
green = cv2.bitwise_and(image, image, mask = green_mask)
```

Figure 19: Code Snippet of Detecting Green

Firstly, read the image captured from the Raspberry Pi camera for this experiment. Then, use the *cv2.cvtcolor* function to convert the RGB value of the image to HSV for better color segmentation result.

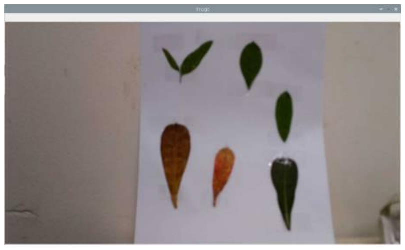


Figure 20: Image of Sample Set 1

Afterwards, set the lower green (*lower_green)* and upper green (*upper_green)* range of the HSV value to the desired spectrum of green for the camera to recognize.

With the lower green and upper green range, create a green mask (*green_mask*) with *cv2.inRange* function to perform segmentation.

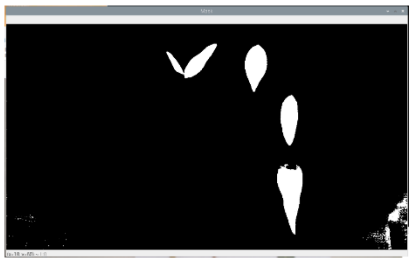


Figure 21: Green Mask Segmentation Result on Sample Set 1

Based on the result, we can infer that the recognition done is fairly accurate. It is able to recognize the four green leaves from Sample Set 1.

Now, with the green mask, use bitwise AND operation *cv2.bitwise_and* to extract the green part of the image for a better visualization of the green leaves.

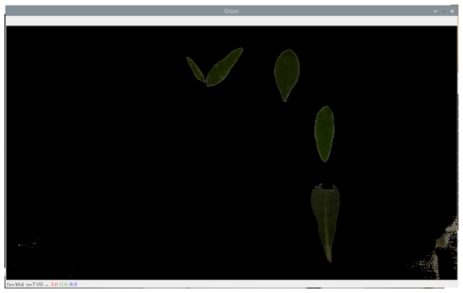


Figure 22: Extraction Result on Sample Set 1

Then, we repeat the same experiment, but this time on Sample Set 2. Similarly, read the image of Sample Set 2 captured from the Raspberry Pi Camera and covert the RGB value to HSV with the *cv2.cvtColor* function.

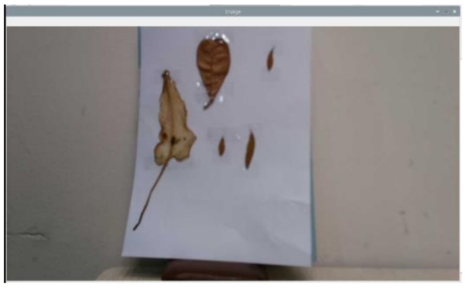


Figure 23: Image of Sample Set 2

Using the same green mask from before, based on the result, we can see that no pixels belonging to the leaves are segmented for Sample Set 2 though there are some noises that were segmented which does not belong to the leaves.

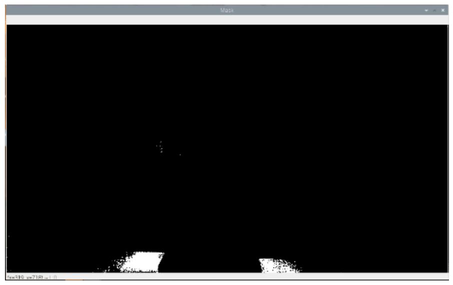


Figure 24: Green Mask Segmentation Result on Sample Set 2

Like Figure 17, the bitwise AND operation extracts none of the leaves from the image.

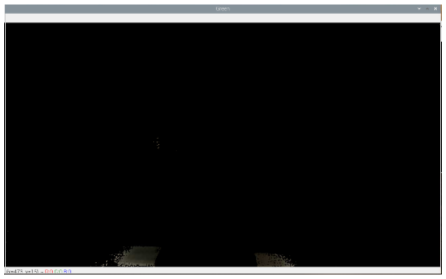
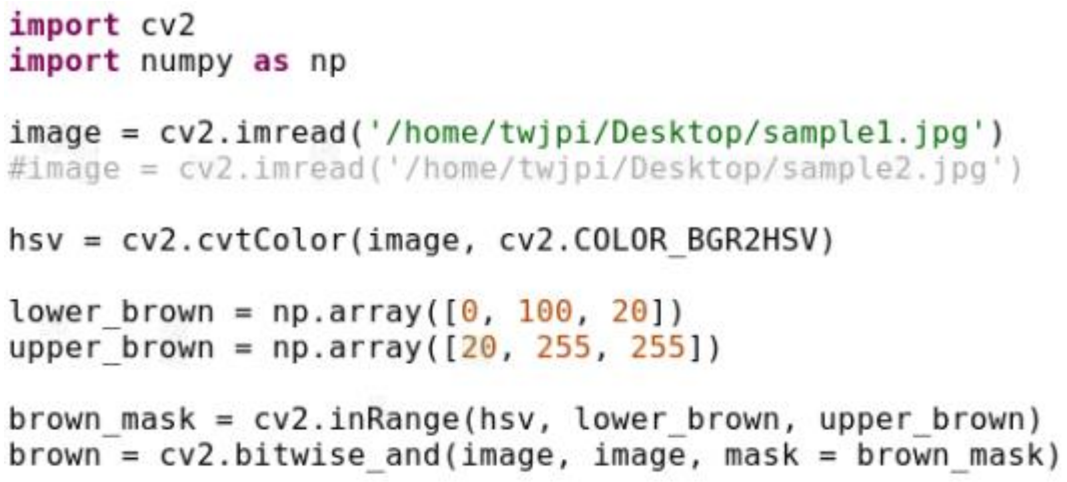


Figure 25: Extraction Result on Sample Set 2

Color Brown

The experiment is repeated on the image taken by the Raspberry Pi Camera of Sample set 1 and 2. The images are then used with the color detection algorithm modified for detecting brown.



```
import cv2
import numpy as np

image = cv2.imread('/home/twjpi/Desktop/sample1.jpg')
#image = cv2.imread('/home/twjpi/Desktop/sample2.jpg')

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

lower_brown = np.array([0, 100, 20])
upper_brown = np.array([20, 255, 255])

brown_mask = cv2.inRange(hsv, lower_brown, upper_brown)
brown = cv2.bitwise_and(image, image, mask = brown_mask)
```

Figure 26: Code Snippet of Detecting Brown

Likewise, in Figure 12, the algorithm is similar but the main difference here is that the lower brown range and upper brown range has been adjusted to detect the spectrum of brown which we are interested in. After which, the RGB value is again converted to HSV with the *cv2.cvtColor* method.

With the lower brown and upper brown, create the *brown_mask* with cv2.inRange function for segmentation.

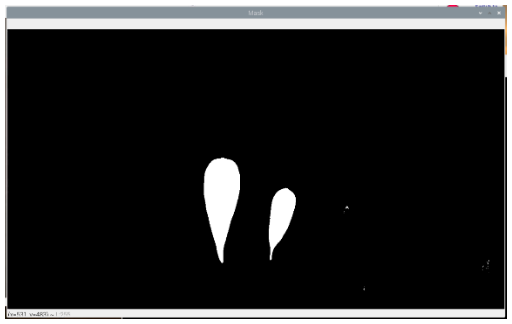


Figure 27: Brown Mask Segmentation Result on Sample Set 1

Based on the result, we observed that the result is also fairly accurate as it managed to point out the two brown leaves from sample set 1.

With the mask, perform the bitwise AND operation with *cv2.bitwise_and* to extract the brown section of the image to visualize the brown leaves.

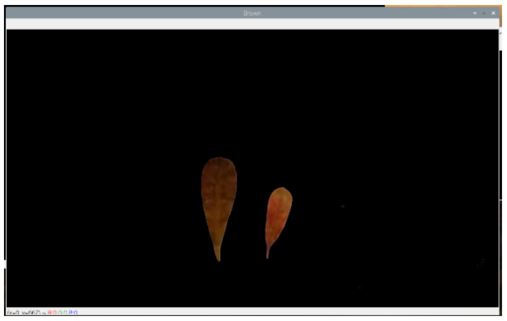


Figure 28: Extraction Result on Sample Set 1

Repeat the experiment again with Sample Set 2 (Figure 16). After converting the image RGB value to HSV, proceed to perform segmentation with the brown mask created from before.

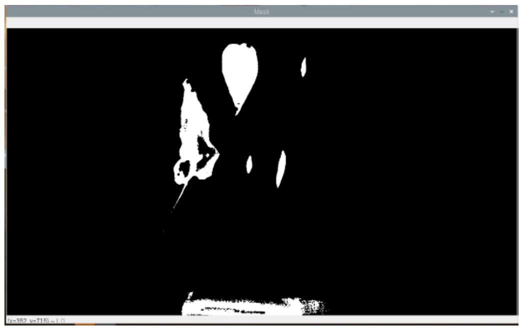


Figure 29: Brown Mask Segmentation Result on Sample Set 2

The result here is also fairly accurate, it is able to detect most of the brown leaves on Sample Set 2.

Proceed with the extraction process with bitwise AND operation for visualization of the leaves.

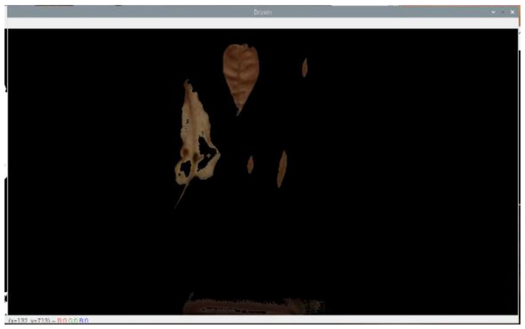
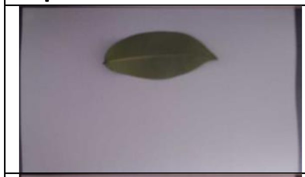
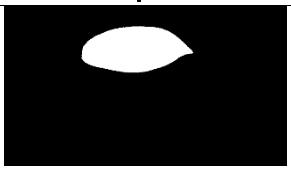
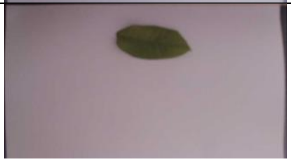
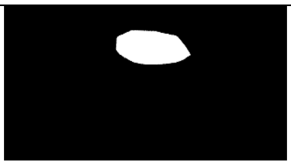
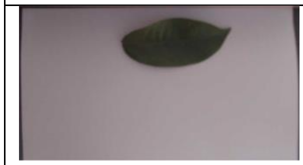
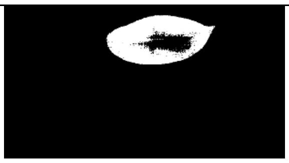
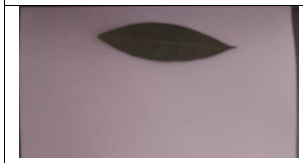
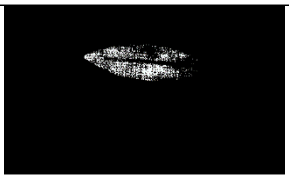
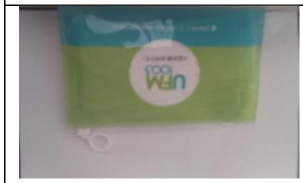
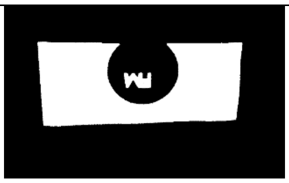
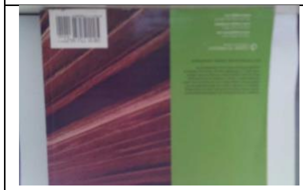
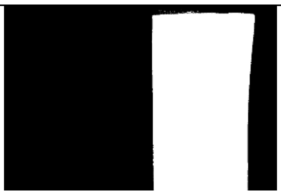


Figure 30: Extraction Result on Sample Set 2

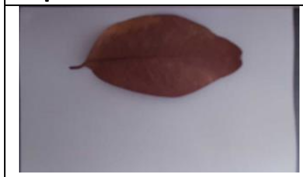
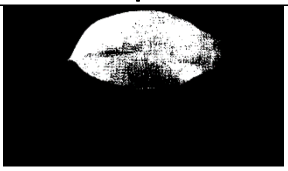
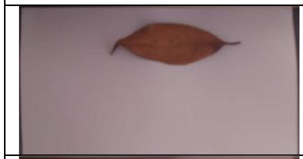
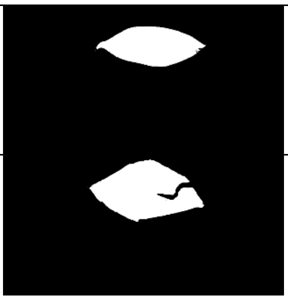
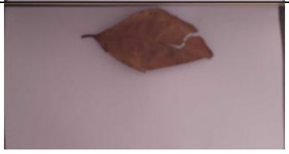
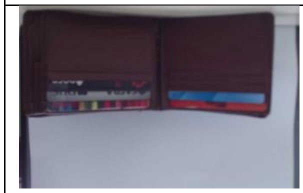
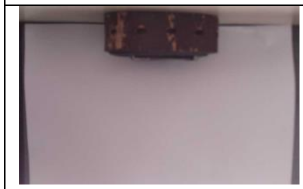
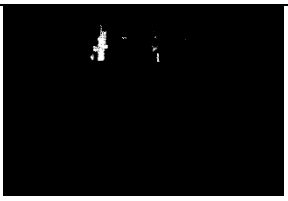
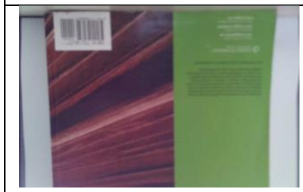
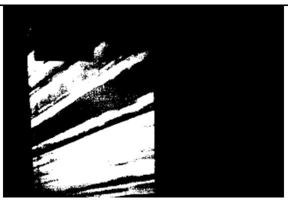
Blackbox testing

A detailed black box testing was performed on the color detection for brown and green. The result is illustrated in the table below.

Detect Color Green Function

Input	Expected Output	Actual Output	Remarks
	A mask with the green leaf segmented should be seen.		Result here is accurate as the entire leaf outline and inner portion is segmented
	A mask with the green leaf segmented should be seen.		Result here is accurate as the entire leaf outline and inner portion is segmented
	A mask with the green leaf segmented should be seen.		Result here is fairly accurate, the leaf outline is segmented though there are inner portion of the leaf which was not segmented.
	A mask with the green leaf segmented should be seen		For this result, the entire leaf is not segmented properly with many noises, though the shape of the leaf still can be made out.
	A mask with the green portion segmented should be seen		Result here is accurate as it is able to segment out the green section of the pouch including the alphabets that are in green.
	A mask with the green side of the book segmented should be seen		Result here is accurate but notice that it did not segment out he white and black words on the green side of the textbook

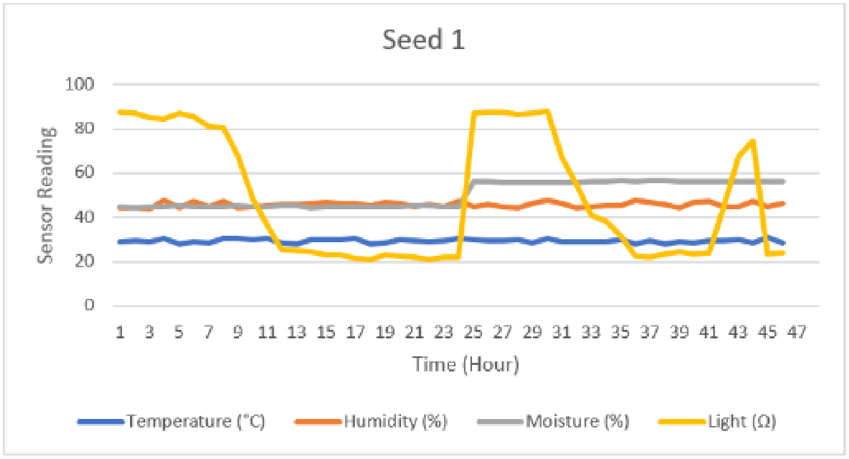
Detect Color Brown Function

Input	Expected Output	Actual Output	Remarks
	A mask with the brown leaf segmented should be shown		Result here is fairly accurate, the shape of the leaf is mostly segmented with some noises in the inner portion of the leaf.
	A mask with the brown leaf segmented should be shown		Result here is accurate, the shape of the leaf and inner portion of the leaf is segmented.
	A mask with the brown leaf segmented should be shown		Result here is accurate, the shape of the leaf including the tear and the inner portion of the leaf are segmented.
	A mask with the wallet outline segmented should be seen.		The wallet was not segmented in this case. The brown color of the wallet in this case was on the far-right end of the Hue spectrum.
	A mask with the belt segmented should be seen.		Similarly, the belt is not segmented as the Hue value for brown in this case falls on the far-right end of the spectrum.
	A mask with the brown side of the textbook segmented should be seen.		The brown side of the textbook here has mixture different shades of brown that is on both end of the Hue spectrum for brown. It is only able to segment the left end of the Hue spectrum for the brown.

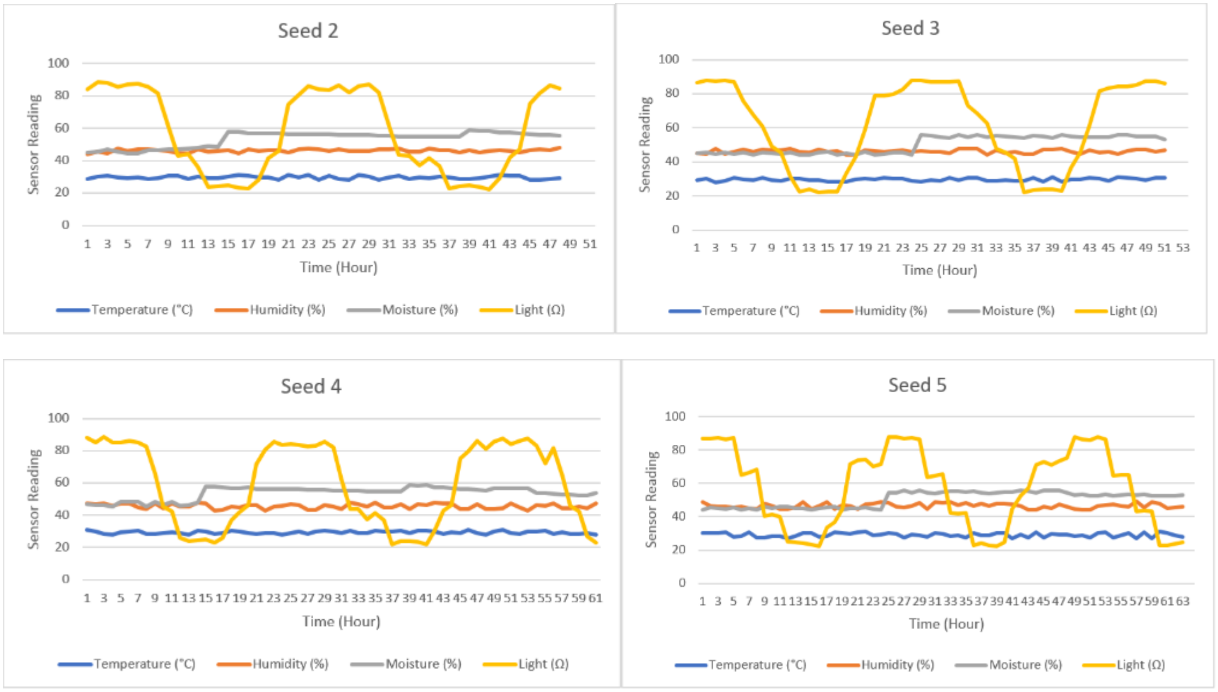
4.3 Germination Rate

The smart pot will make use of data gathered from AM2302, FC-28 and KY-018 to predict how long, in hours, the seed will take to germinate into a seedling. Five experiments were conducted using the same seed type at the same location to test how long it took for each seed to germinate. This is to ensure accuracy of results. Temperature, humidity, moisture, and light data are gathered for data analysis. With the data gathered, median will be used to predict the duration for the seeds to germinate.

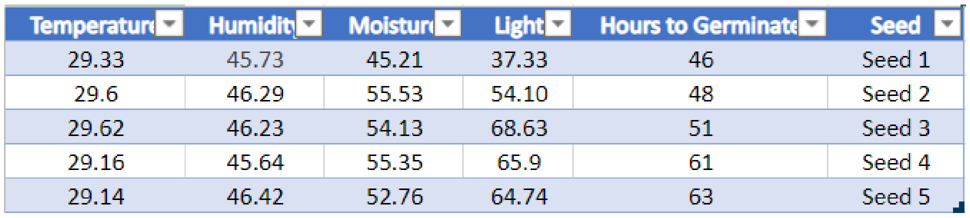
During the first experiment, Seed 1 had lesser amount of light provided during the experiment. This caused the result of the graph to differ greatly when compared to the remaining seeds.



The experiment results for Seed 2, Seed 3, Seed 4 and Seed 5 were similar because the amount of light source provided were the same. Hence, the graphs did not differ much from one another.



After conducting the experiments, it was observed that Seed 1 took 46 hours to germinate, Seed 2 took 48 hours to germinate, Seed 3 took 51 hours to germinate, Seed 4 took 61 hours to germinate and Seed 5 took 63 hours to germinate.



Temperature	Humidity	Moisture	Light	Hours to Germinate	Seed
29.33	45.73	45.21	37.33	46	Seed 1
29.6	46.29	55.53	54.10	48	Seed 2
29.62	46.23	54.13	68.63	51	Seed 3
29.16	45.64	55.35	65.9	61	Seed 4
29.14	46.42	52.76	64.74	63	Seed 5

Figure x. Hours to germinate each seed

However, the team has decided to eliminate Seed 1 as the team believes that the amount of light provided would have affected the results of the germination rate, thus causing inaccuracy of experiment results.

With the gathered information, the team decided to use median to calculate the germination rate of the seeds.

Blackbox Testing

The initial data that were used to plot the multi-linear regression (MLR) was put back into the graph to see if the graph is properly plotted

The input values are Temperature, Humidity, Moisture and Light.

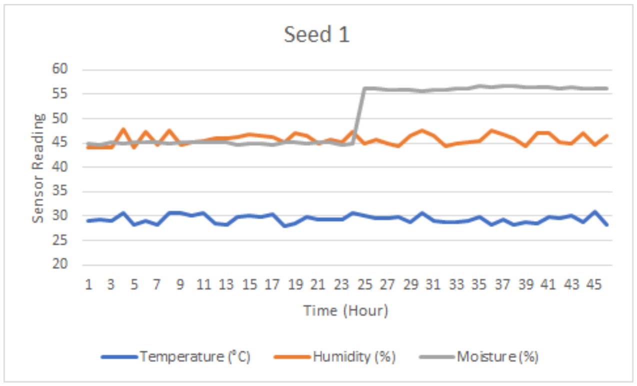
These testing are conducted to ensure validity of the predicted algorithm before implementing and use in our system.

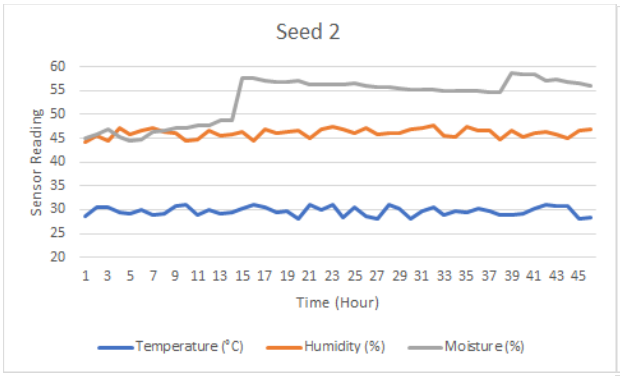
Input				Expected Output	Actual Output	Remarks
Temperature	Humidity	Moisture	Light			
29.33	45.73	45.21	37.33	46	46	Result is as expected
29.6	46.29	55.53	54.10	48	48	Result is as expected
29.62	46.23	46.29	68.63	51	51	Result is as expected
29.16	45.64	55.35	65.9	61	61	Result is as expected
29.14	46.42	52.76	64.74	63	63	Result is as expected

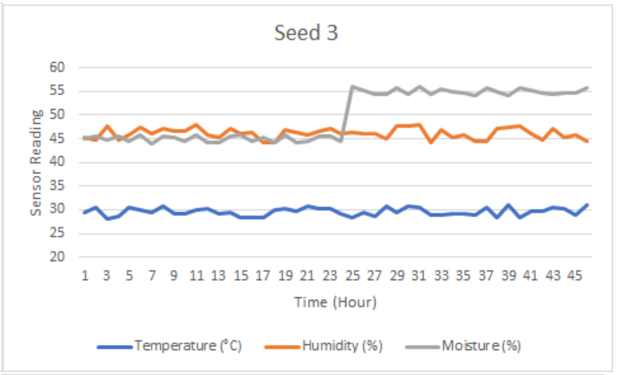
4.4 Evaporation Rate

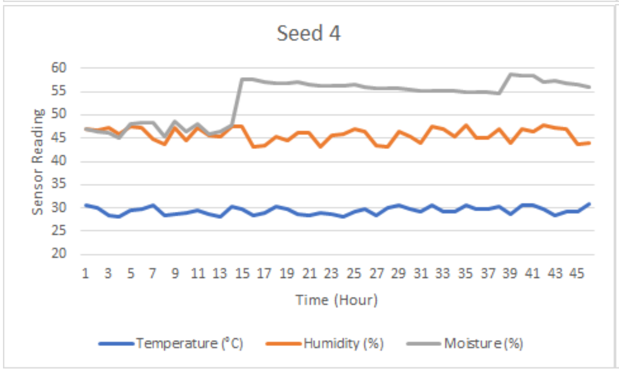
The smart pot will make use of data gathered from AM2302 and FC-28 to predict the evaporation rate of the soil. Five experiments were conducted using the same seed type with individual pots at the same location to determine the evaporation rate of the soil. This is to ensure accuracy of results. Results were recorded hourly for a period of 46 hours and plotted into a graphical format for easier visualization.

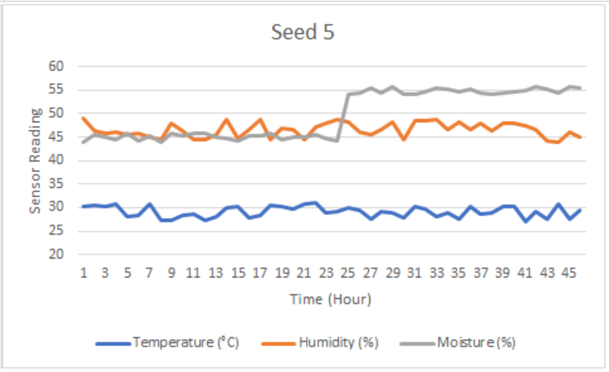
For this experiment, the team watered the seeds at an interval of 24 hours with 50ml of water each time. The spike in moisture level shows that the plant was being watered. It is difficult to show the evaporation rate of the soil moisture as the evaporation rate is minimal every hour.



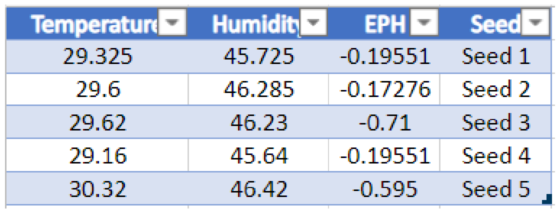








After conducting the experiment, despite having the same experimental conditions, it was observed that the evaporation per hour (EPH) for Seed 3 was the highest of -0.71, followed by Seed 5 of -0.595, then Seed 1 and 4 of -0.19551 for both, and lastly Seed 2 of -0.17276.



Temperature	Humidity	EPH	Seed
29.325	45.725	-0.19551	Seed 1
29.6	46.285	-0.17276	Seed 2
29.62	46.23	-0.71	Seed 3
29.16	45.64	-0.19551	Seed 4
30.32	46.42	-0.595	Seed 5

Blackbox Testing

The initial data that were used to plot the multi-linear regression (MLR) was put back into the graph to see if the graph is properly plotted

The input values are Temperature and Humidity.

These testing is conducted to ensure validity of the predicted algorithm before implementing and use in our system

Input		Expected Output	Actual Output	Remarks
Temperature	Humidity			
29.325	45.725	-0.19551	-0.19551	Result is as expected
29.6	46.285	-0.17276	-0.17276	Result is as expected
29.62	46.23	-0.71	-0.71	Result is as expected
29.16	45.64	-0.19551	-0.19551	Result is as expected
30.32	46.42	-0.595	-0.595	Result is as expected

4.5 Overall Limitation

There are a couple of limitations for this project.

The first limitation is that only 1 type of seed will be used for the sample experiments. This is to standardize the growing factors for the experiments. Basil seeds will be used as they only require 3-5 days to germinate, which is considered one of the few fast-growing plants considering Singapore's climate.

Next, time constraint is another limitation which will lead to lower accuracy of predictive data as there is limited time to research on information about the basil seeds, program the system, test the system and conduct experiments on the plants.

Following, temperature results will not vary too much as the project is tested in Singapore. This is because Singapore is located near the equator. Hence, Singapore does not experience the four seasons – spring, summer, autumn, winter.

Subsequently, it is difficult to ensure that the entire pot of soil is moist. After watering the plant, there may be parts which the water did not reach or there may be moisture in the soil, but it was not detected or reached by the soil moisture sensor. Not forgetting, the sensor will also take in surrounding noises. Therefore, results of the experiment may be affected.

Additionally, the current color detection can only detect colors and will not differentiate objects. Therefore, if an object is placed under the camera, it will simply detect the colors and display the results accordingly to what it was being set.

Furthermore, as our FC-28 Moisture Sensor and KY-018 Photoresistor are using the same Analogue Digital converter, there is a conflict of data being collected since they are both sharing the same address but different registers. The register that is being used cannot be defined in Python code. Therefore, we have hardcoded the moisture level with *random.uniform(58, 59)* to keep the values within the range of 58-59%. The rest of the code including MQTT are working fine in Python. This issue can be solved by using C code, however in C we are unable to find a working MQTT library and thus have not been successful in pushing the data onto ThingsBoard. Converting the C code into a library to be called in Python was also not possible due to the other dependencies used to C to define the registers and address for the ADC.

Lastly, only 1 set of sensors is available to test. This is because only 1 set of sensors is supplied for the project. Therefore, the team can only conduct the tests and experiments one at a time over the project duration.

5. Conclusion

The proposed solution of a smart pot will assist plant owners in keeping track of their plants' expected germination rate and behavior conveniently using ThingsBoard.

Firstly, plant owners will be more informed on when their plants will start to grow and whether the environment is optimal for its growth. The smart pot will be able to keep track of the temperature, humidity and light of the environment to make predictions on the germination rate of the plants (e.g., hours to take to germinate, etc.).

Next, plant owners do not need to be physically present to observe their plants' status. The smart pot can assist plant owners decide whether their plants are suffering by observing the color of the leaves.

A potential upgrade is to make sure plant owners can focus on their priorities and do not need to follow up with their plants' watering schedule as the smart pot will assist them with the schedules. For that to happen, the smart pot must be able to keep track of the soil moisture level to help plant owners decide whether their plants are dehydrated and requires any watering of plants. The watering of plants can also be scheduled to ensure that the plants stay hydrated.

Additionally, to add onto the schedule of watering the plant, the team has an ambitious idea where integrate automation with relay and water pump. This allows user to "switch" on the auto-watering function if the user is away for vacation etc. Where the system will monitor the moisture level of the soil and water once it hits the threshold.

Lastly, another potential upgrade is to have an object recognition with color recognition such that the system can automatically recognize the color of the leaves whenever leaves are being detected. This is because, the current system is only recognizing colors – brown or green and not object. Hence, if there were other objects that are brown or green colored, it would be detected into the system.

Therefore, the current functions of the smart pot will reduce the time and attention required by the plant owners to achieve a healthy growing plant as they will be able to conveniently access their plants' information on their devices to check on their growth without being physically there.

6. Individual Contribution

Student	Work Package
David Chua	• ThingsBoard • Training data • Integration of sensors • Integration of codes • Multiple Linear Regression • Process Flow Chart • Python calling C Code • Pioneer600's LCD (OLED & Joystick) • Timer Interrupt • Video • Report
Ng Jia Cheng	• Pioneer600's LCD (OLED & Joystick) • Training data • Report • Process Flow Chart • Relay [Not implementing] • Multiple Linear Regression • Python calling C Code • Video
Tan Wei Jian	• Raspberry Pi Camera • Color Recognition • Report • Process Flow Chart • Training data • Video
Shirley Chia	• Sensors • Timer Interrupt • Training data • Process Flow Chart • Report • Video
Daren Puan	• Sensors • Video • Python calling C Code • Initial Setup (Arduino) • Process Flow Chart • Convert sensor code to C [Not implementing] • Report

7. Project Contribution

S/N	Contribution Description
1	Able to predict the number of hours to germination based on temperature, humidity, light and moisture (using own training data) [Using multiple linear regression]
2	Able to predict the number of hours the soil moisture will last based on temperature and humidity (using own training data) [Using multiple linear regression]
3	Able to differentiate colors of leaf to determine the healthiness of the plant through raspberry pi camera (using available libraries) [Using decision tree]
4	Finding the suitable frequency for the sensor(s) [To prevent overload and corrosion]
5	Able to see the updated sensors' data in ThingsBoard

7. References

https://www.waveshare.com/wiki/Pioneer600

https://github.com/tayfunulu/Pioneer600 - Pioneer600 - Navigation for joystick and Oled display

http://colorizer.org/ - Online tool for assisting with finding the right HSV value for color recognition

https://maker.pro/raspberry-pi/tutorial/how-to-create-object-detection-with-opencv - Color detection Algorithm

https://www.w3schools.com/python/python_ml_multiple_regression.asp - Machine Learning: Multiple Regression