# Plant Care IoT System

By

Zakiyyah Ahmed

Student number: 190178699

University of London

BSc Computer Science

**Module**: Physical Computing and Internet-of-Things (IoT)

**Module code**: CM3040

**Coursework**: Final Project

**Date of submission:** 7 March 2022

**Word count:** 2488 (Excludes title page, abstract, table of contents, list of figures, figures, references, appendices)

# Abstract

People face many problems in ensuring the survival of their plants. Many people do not have the essential knowledge needed to take care of plants. They may not have the time to care for their plants or may even forget to tend to them.

Therefore, I have developed a plant care system that solves these problems.

The plant care system will use one ESP8266 microcontroller and multiple sensors (soil moisture sensor, DHT11 temperature and humidity sensor, ultrasonic distance sensor) that will capture data such as soil moisture, temperature and humidity.

The system provides functionality for automatic watering of the soil. This is done by using the following components – water pump, relay and a battery pack to power the pump. This functionality will be automatically triggered based on the soil moisture sensor data.

To enable the automatic watering system, a water tank or reservoir, containing a battery powered pump and plastic tubing, will be placed next to the pot plant. An ultrasonic distance sensor will be placed in the water tank to measure the level of water inside the tank. If the water level is below a certain threshold value, the system will notify the user, via email, to refill the tank.

An LCD screen will display the current temperature, humidity, soil moisture and water level status in the reservoir.

Plant sensor data and other relevant information will be displayed on a dashboard. Users can also view this data on another webpage in JSON format.

In the System Implementation section in the report below, I have described in detail every component of the system, explained and

justified my design decisions and listed the changes I have made to the initial project proposal developed for the midterm coursework assignment. I have also supplied a circuit diagram and other relevant diagrams such as architecture diagrams and system flow diagrams. In the System Development Lifecycle section, I have discussed the major stages of development and provided code extracts as evidence. I have also provided test documentation which discusses how the tests were performed and whether the system had passed or failed the tests. The Critical Analysis and Reflection section analyzes my product and my process or approach to the development of the plant care system.

Please see the Appendices section for additional information.

# Table of Contents

# List of Figures

# 1 Introduction and Problem Background

People face many problems when trying to care for their houseplants. They may not have the necessary knowledge or time to maintain their plants, which may lead to plant death.

Therefore, I have developed a plant care system that helps people to maintain the health of their plants.

In my system, multiple sensors will capture the temperature, humidity and soil moisture readings of pot plants. The watering of plants will be fully automated. I have also implemented a system whereby the level of water in a reservoir (that is placed next to the plant) will be continuously monitored. If water is not sufficient, an email notification will be sent to the user to refill the reservoir. In this way, users do not have to physically water their plants or have to consciously remember to fill water in the reservoir.

Users can also view data via a dashboard and LCD screen.

The system code can be updated remotely using the Over-the-Air functionality that I have enabled.

# 2 System Implementation

## 2.1 Component Descriptions

### Webpages

All webpages are served on an asynchronous web server, using the ESPAsyncWebServer library.

### Dashboard

I have created a dashboard that displays the following sensor data – soil moisture, temperature and humidity. Important notifications, such as the status of the water level of the reservoir are also displayed. Users can also set plant soil moisture, temperature and humidity requirements via a user input form. This functionality ensures that the system is compatible with any plant species the user may purchase.

To visit the webpage, navigate to your ESP8266 IP address.

**REST-based HTTP requests in a JSON format**

Sensor data will also be accessible to the user in a JSON format via a webpage.

To visit the webpage, navigate to <ESP8266 IP/json>

**Over-The-Air (OTA) Updates**

To enable OTA updates, I used the AsyncElegantOTA library which allows you to update a new sketch to your board remotely.

To perform OTA updates:

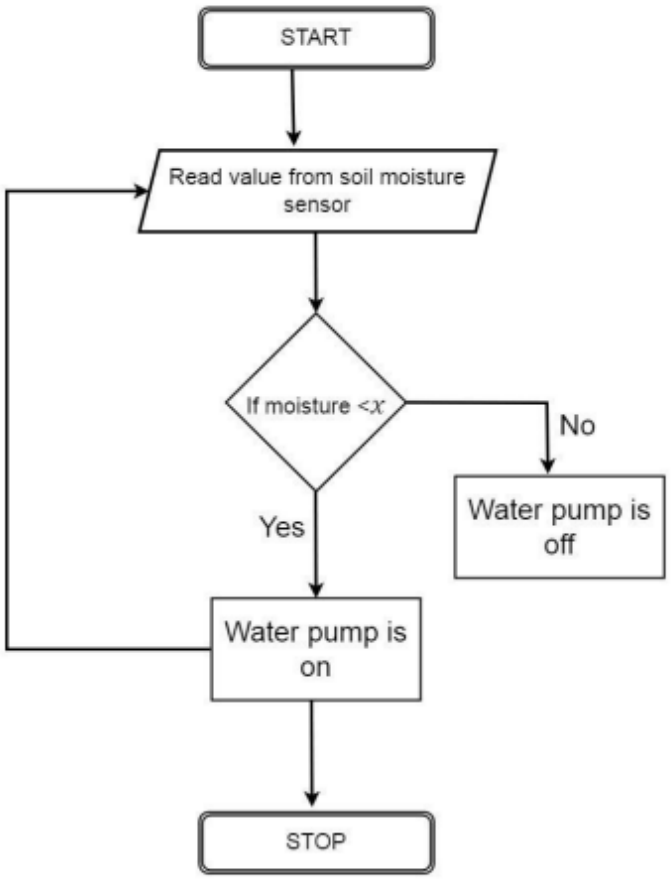
1. Update your sketch
2. Go to Sketch > Export Compiled Binary
3. Go to your ESP IP address followed by /update
4. Click on Choose File and select the *.bin* file you've just generated.

Please refer to Appendix A to view an image of the all the webpages.

**Soil Moisture and Automated Watering System**

The soil moisture sensor will measure the moisture level in the soil. If the level is below the predetermined threshold, the relay connected to a water pump will be programmatically switched on to provide water to the plant from a water reservoir. When the moisture level is above the threshold, the relay connected to the pump will be switched off.

Fig. 2 : System design flow for the watering system




## Temperature and Humidity

The DHT11 sensor will measure the temperature and humidity values of the air surrounding the plant. If the temperature is too low, the user is directed, via the dashboard, to place the pot plant in a cooler area. If the humidity is too low, the user is directed, to place a humidity tray [4] next to the plant.

## Water Level Detection and Email Notification System

The ultrasonic distance sensor measures the distance from the top of the reservoir to the top of the water level. If the distance from the top of the reservoir to the top of the water level is above the specified threshold distance, the LCD screen and dashboard displays a message to the user stating that the reservoir needs to be refilled. The user can also enter, via the dashboard input form, a specific threshold distance. This functionality ensures that the system is compatible with any water container size.

Users will also be notified via email to refill the reservoir.

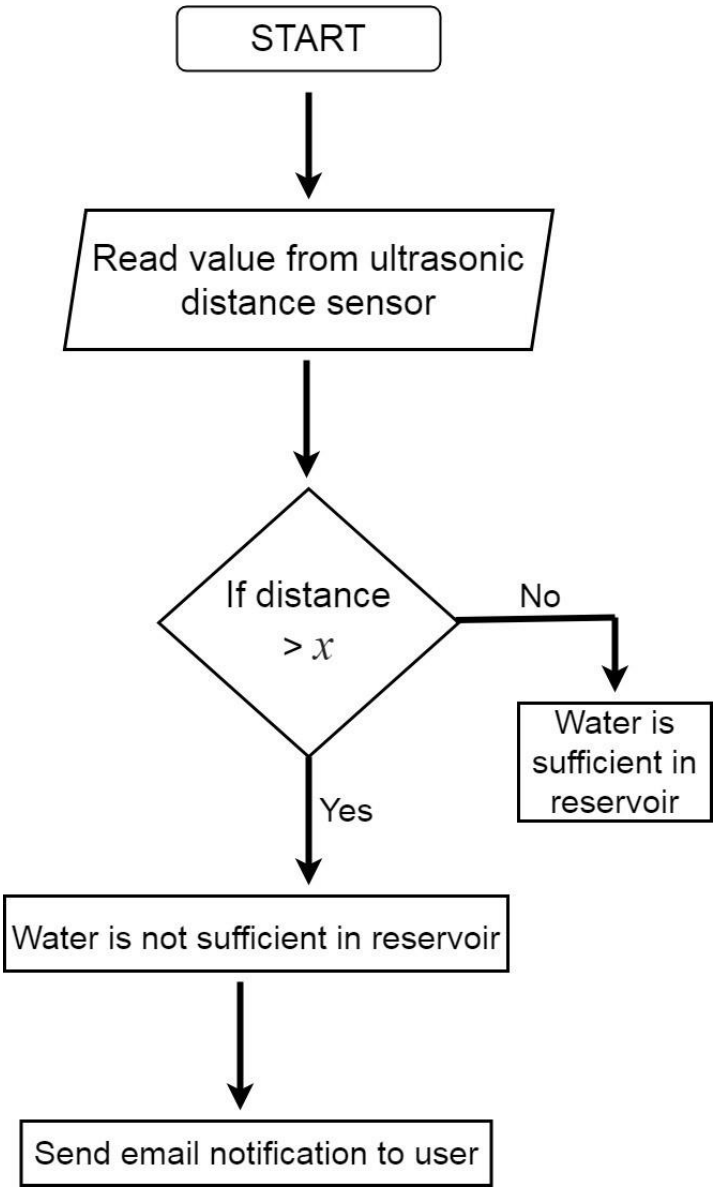


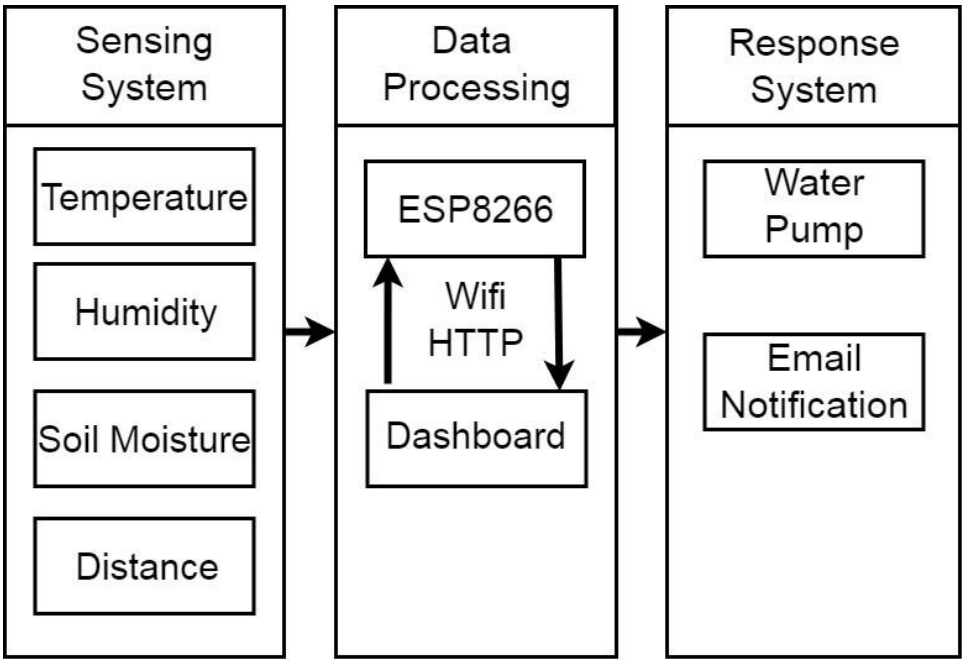

Fig. 3 Water level detection and email notification system

Please refer to Appendix B to view an image of the email sent by the microcontroller to the user.

## LCD Screen

An LCD screen will display the current temperature, humidity and soil moisture readings and reservoir water level status.

Fig. 4 System Architecture Diagram




Please refer to Appendix D for a list of all the components needed to build the system.

## 2.2 Changes made to the initial project proposal

In my initial proposal, I stated that I would use two ESP8266 microcontrollers since I required two analog sensors – one for the soil moisture sensor and the other for a phototransistor.

In the new system design, I have decided to use only one node. The reason for this is that the painlessMesh library used to allow multiple nodes to communicate with each other, was not compatible with two project requirements. Those are Over-The-Air (OTA) updates and the

email notification system. Communication between the nodes had failed after I enabled OTA and email notifications. Therefore, I decided to forego the second microcontroller and the phototransistor.

Thereafter, I moved all the sensors that were originally on the second node to the first node.

The second change I made to the proposal was the removal of the humidity fan powered by the DC Motor. Once the humidity readings are greater than a certain threshold, the fan would turn on to reduce the humidity. The reasons for this change are two-fold. The first reason is that I had difficulty in powering up the DC Motor. The second reason is that I had decided to use only one microcontroller, therefore I decided to remove the DC Motor since the microcontroller already had multiple sensors and not enough digital pins.

## 2.3 Important Design Decisions

Initially, I built a synchronous web server to display the webpages. However, upon further research [1], I decided to build an asynchronous web server since I discovered it has several advantages:

- It can handle more than one connection a time. Since I serve three different web pages, it is an important advantage.
- When sending a response, you are immediately ready to handle other connections
- The speed is excellent
- It is extensible and can handle any content type.

To enable OTA, I initially used the ArduinoOTA library. After conducting extensive research [2][3], I decided to use the AsyncElegantOTA library. The reasons for this are as follows:

- It is easy to integrate with asynchronous web server projects [3].

- The library requires considerably less code than ArduinoOTA.
- It also provides a modern and aesthetically pleasing webserver interface.

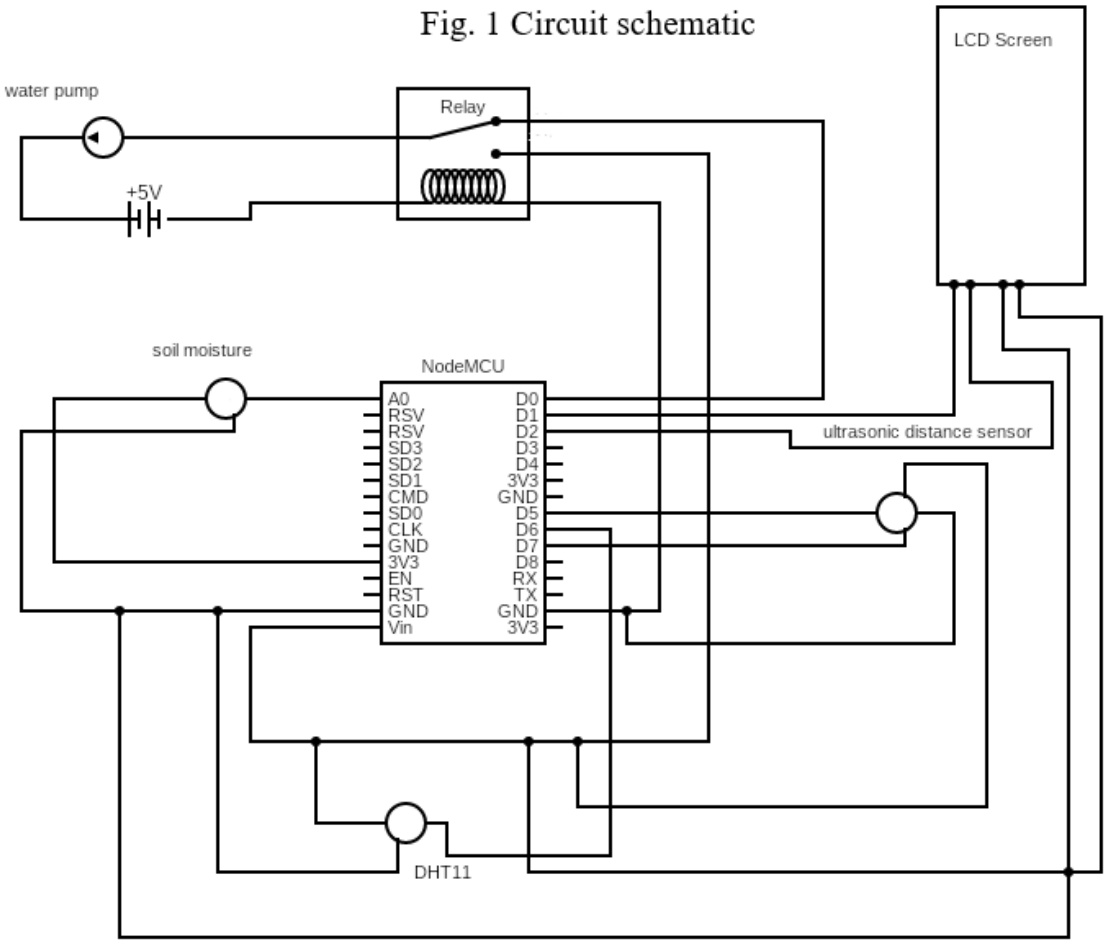
I have decided to use an LCD screen to make the system more user-friendly and to enhance the user experience. The LCD screen ensures that the user is aware of their plant status without having to visit the dashboard webpage.

I have also decided to automate the watering of plants since it is an important aspect of the system. Plants will get watered even if their owners are away or forget to tend to them.

I have integrated an email notification system to improve user experience and usability. An email will be sent to the user to fill water into the reservoir if the water level is too low. In this way, the user does not have to remember to fill water, instead he/she will be notified to do so.

## 2.4 Circuit Diagram and Pin Connections

Fig. 1 Circuit schematic




Refer to Appendix C For pin connections and Appendix F for an image of the circuit I have built.

# 3 System Development Lifecycle

## Stage 1: Soil Moisture Sensor

### Design and Implementation

The first task I performed was to get the soil moisture sensor working. The soil moisture sensor is an analog device, therefore, I had to connect

the wire to pin A0 on the ESP8266 node. I then connected the remaining wires to the node. Thereafter, I wrote the code to read values from the sensor.

```
void read_current_soil_moisture_value() {

  soil_moisture_value= analogRead(soil_sensor_pin);
  // map soil moisture value to a percentage - for usability
  soil_moisture_value = map(soil_moisture_value,550,0,0,100);

  // display on serial monitor - for testing
  Serial.print("Moisture : ");
  Serial.print(soil_moisture_value);
  Serial.println("%");

}
```

### Testing and Validation

I placed the sensor in a bowl of water. The readings showed that once the sensor was placed in the water, the moisture value increased. After taking the sensor out of the water, the readings decreased. Thus, the sensor readings were validated.

## Stage 2: Automated Watering System

### Design and Implementation

To enable the pump to automatically turn on, I connected a relay to the pump and the circuit. I then wrote the code to turn on and off the pump under certain conditions.

```
// if the soil moisture reading is less than the threshold
if (soil_moisture_value < soil_moisture_threshold) {
  // see water_pump_control tab
  turn_on_water_pump();
}
else {
  turn_off_water_pump();
}
```

I also connected a battery pack to the pump.

### Testing and Validation

To test that the pump automatically turns on only when the moisture reading is below a threshold, I placed the water pump into a bowl of water and ensured that the sensor remained dry. The result was that the pump turned on.

To test that the pump automatically is off under all other conditions, I placed the water pump and moisture sensor into a bowl of water. Since the sensor reading was above the specified threshold, the pump was off and did not switch on.

Thus, the water automation system was validated and verified.

## Stage 3: DHT11 Temperature and Humidity Sensor

### Design and Implementation

I connected the DHT11 sensor to the circuit with the power connected to the 3V3 pin of the microcontroller. I then installed the necessary library (DHT.h) and wrote the code to enable the sensor to capture the readings.

```
void read_temperature_humidity(){

   // Read temperature as Celsius (the default)
    float newTemperature = dht.readTemperature();

    // if temperature read failed, don't change temp value
    if (isnan(newTemperature)) {
      Serial.println("Failed to read emperature!");
    }
    else {
      temperature = newTemperature;
      Serial.print("Temperature: ");
      Serial.println(temperature);
    }
    // Read Humidity
    float newHumidity = dht.readHumidity();
    // if humidity read failed, don't change humidity value
     if (isnan(newHumidity)) {
      Serial.println("Failed to read humidity!");
    }
    else {
      humidity = newHumidity;
      Serial.print("Humidity: ");
      Serial.println(humidity);
    }

}
```

### Testing and Validation

Initially, the serial monitor printed a value of 0 for both humidity and temperature readings. This was clearly incorrect. After doing research [5], I discovered that, in some cases, powering the DHT with 5V solves the problem. Therefore, I connected the power to the Vin pin of the ESP8266. The readings were now giving me the expected non-zero values.

I first placed the sensor in a sunny and humid area and then in a shaded area and noted both sensor readings. As expected, the readings were higher in the sunny area than in the shaded area.

## Stage 4: Water Level Detection and Email Notification

### Design and Implementation

I connected an ultrasonic distance sensor to the node. Thereafter, I wrote the code to enable the sensor to measure the distance to the nearest object in centimeters.

I have also enabled the microcontroller to send emails to the user. I have used the EmailSender library to create this functionality.

If the distance from the top of the reservoir to the top of the water level is above the specified threshold distance, an email notification will be sent to the user.

```
// get distance from ultrasonic sensor - see get_distance tab
get_distance();

// if the distance reading is less than the threshold
if (distance < distance_threshold) {
  reservoir_level_status = "Water reservoir is sufficient";
  Serial.println(reservoir_level_status);
}
else {
  reservoir_level_status = "Water in reservoir is not sufficient. Please fill water into the reservoir";
  Serial.println(reservoir_level_status);
  //send email to user - see send_email tab
  send_email();
}
```

### Testing and Validation

After specifying a threshold distance, I placed the distance sensor on the top of the reservoir. Since there was no water in the reservoir, the distance sensor measured a reading greater than the threshold distance and an email was sent to my Gmail address. This system worked as expected.

## Stage 5: Using the ESPAsyncWebserver and OTA

### Design and Implementation

I have used the ESPAsyncWebServer library to build the server that serves the webpages.

I created the dashboard that displays data to the user and have also added input forms so that the user can set their own unique plant requirements.

I then created a webpage that serves sensor data to users in a JSON format. For this, I have used the ArduinoJson library.

To enable OTA updates, I installed the AsyncElegantOTA library. It was incredibly easy to enable OTA updates. This functionality only required 2 to 3 lines of code.

### Testing and Validation

The dashboard was correctly displayed once I navigated to the ESP IP address in a web browser.

The webpage displaying data in JSON format was displayed once I navigated to <ESP_IP>/json.

To validate the OTA functionality, I modified the dashboard html in my sketch and then navigated to <ESP_IP>/update. I then created a .bin file of my sketch and uploaded it to the AsyncElegant interface. After it successfully uploaded, I navigated to the dashboard and saw that the changes were displayed correctly.

## Stage 6: LCD Screen

### Design and Implementation

Initially, I used an LCD screen without a 12C module. It had multiple wires and therefore, required many digital pins. Since, I decided to use only one microcontroller, using this type of LCD screen was not a viable option. I then switched to using the 12C module LCD screen. It required fewer digital pins and the wiring was simple. Thereafter, I wrote the necessary code to power and display data on the LCD screen. Since I had multiple sensors readings to display, there was not enough space on the screen. I then created a function that allows the LCD to display scrolling text. Now all the data could be displayed.

```
void display_data_lcd(){
  String sensorReadings = "Moisture: " + String(soil_moisture_value) + "% ";

  sensorReadings += "Temp: " + String(temperature) + " ";
  sensorReadings += "Hum: " + String(humidity) + "% ";

  scroll_text(0, sensorReadings, 250, lcdColumns);
  scroll_text(1, reservoir_level_status, 250, lcdColumns);
}

// enables scrolling on lcd screen
void scroll_text(int row, String message, int delayTime, int lcdColumns) {

  for (int i=0; i < lcdColumns; i++) {
    message = " " + message;
  }

  message = message + " ";
  for (int pos = 0; pos < message.length(); pos++) {
    lcd.setCursor(0, row);
    lcd.print(message.substring(pos, pos + lcdColumns));
    delay(delayTime);
  }
}
```

### Testing and Validation

At this point all the sensors were connected so I looked at the LCD screen and saw that the data was displayed correctly.

Refer to Appendix E for a list of all libraries needed.

Refer to Appendix G to view images of the physical system set up.

# 4 Testing

Below is a summary of the testing conducted to validate the plant care system.

Fig. 5 Testing

| ID | Test Case | Actual result | Pass/Fail |
|---|---|---|---|
| 1 | Check that the soil moisture senor captures the correct readings. | The soil moisture sensor captures expected results when it was placed in soil that was recently watered and when it was dry | Pass |
| 2 | Check that the DHT11 sensor captures the correct readings. | Temperature and humidity readings captured expected results when it was placed in the sun and a then in a cooler area | Pass |
| 3 | Check that the ultrasonic distance sensor captures the correct distance in centimeters. | When placing the sensor on the top of an empty reservoir at exactly 8cm from the floor, the sensor captures the distance as 8.3 cm. The 0.3 is negligible. | Pass |
| 4 | Check that the system can automatically water the plant. | When the soil moisture reading is below a specified threshold, the water pump is turned on. | Pass |
| 5 | Check that the user receives an email from the system when the water level in the reservoir is low and needs to be refilled. | When the ultrasonic distance sensor measures the distance to be greater than the specified threshold distance, an email notification is sent to the user. | Pass |
| 6 | Check that the LCD screen displays correct real-time sensor data. | The LCD displays the water level status of the reservoir, temperature, humidity and soil moisture readings of the plant. | Pass |

| | | | |
|---|---|---|---|
| 7 | Check that the dashboard displays correct real-time sensor data and users can successfully input values into the forms. | After navigating to the ESP IP address, the dashboard displays real time sensor data. I can also successfully input values into the forms. The watering system and water level detection system respond to user inputted threshold values. | Pass |
| 8 | Check that the user can access a webpage with sensor data displayed in a JSON format. | After having navigated to <ESP_IP>/json on a web browser, a JSON object with sensor data is displayed. | Pass |
| 9 | Check that OTA updates have been enabled | I modified the dashboard html in my sketch and then navigated to <ESP_IP>/update. I then created a .bin file of my sketch and uploaded it to the AsyncElegant interface. After it successfully uploaded, I navigated to the dashboard and saw that the changes were displayed correctly. | Pass |

# 5 Critical Analysis and Reflection

## 5.1 Product Analysis

The end-product is usable in a real-life context. The system ensures that people are aware of their plant status at any given time.

The automated watering system is a core feature of the product. It ensures that plants are always watered in correct amounts. This functionality is augmented by the water level detection and email

notification system. Together, these two core features make the plant care system less dependent on the plant owner.

The email notification system is an excellent feature, however, since not all people access their emails regularly, a better feature would be an SMS notification. This would ensure that people are getting the message to fill the reservoir with water, in a timely manner.

Another improvement to the system would be to add a phototransistor (to a second microcontroller) to detect the total amount of light received by the plant. This feature was part of my original design, but I had to remove it since the painlessMesh library that allows two nodes to communicate was incompatible with two other libraries. In the future, I would like to try and find a solution to this issue and add a second node to the system.

## 5.2 Process Analysis

The general process I followed was dividing the development into different stages and thereafter testing the code written during that stage of development. I thereafter combined the code into one file and conducted integrated testing.

I also divided the code into separate functional modules or files. This made it easier to debug and test code.

While dividing the development and testing into different stages proved to be efficient, I later came across compatibility issues when combining all the code. I then had to rewrite some of the code.

To improve this process, I should first determine all the libraries required for my project and try to find out if there exist any compatibility issues. Solutions would be to find different libraries to solve these issues. If there are no other libraries, I should then alter my system design. This should be done before any code has been written.

# 6 Conclusion

This course was the first time that I had been exposed to the world of IoT. I learned a lot during the development of this project and the skills I acquired will serve me well in future IoT project endeavors.

# 7 References


**[1]** GitHub (2021) *ESPAsyncWebServer* https://github.com/me-no-dev/ESPAsyncWebServer [Accessed: 21 February 2022]

**[2]** Random Nerd Tutorials (2020) *ESP8266 NodeMCU OTA (Over-the-Air) Updates – AsyncElegantOTA using Arduino IDE* *https://randomnerdtutorials.com/esp8266-nodemcu-ota-over-the-air-arduino/* [Accessed: 22 February 2021]

**[3]** Microcontrollers Lab (2021) ESP8266 OTA Over The Air Programming using AsyncElegantOTA Library and Arduino IDE. https://microcontrollerslab.com/esp8266-ota-over-the-air-programming-arduino-ide/#Key_Advantages_of_using_AsyncElegantOTA_Library

[Accessed: 22 February 2022]

**[4]** Fifth Season Gardening (2018) *Humidity and Houseplants* https://fifthseasongardening.com/humidity-and-houseplants#:~:text=Humidity%20Trays%20%E2%80%93%20They%20are%20easy,just%20below%20the%20medium%20level.[Accessed: 15 February 2022]

**[5]** Random Nerd Tutorials (2021) *[SOLVED] DHT11/DHT22 – Failed to read from DHT sensor*. https://randomnerdtutorials.com/solved-dht11-dht22-failed-to-read-from-dht-sensor/ [Accessed: 24 February 2022]

## Code References

**[6]** Random Nerd Tutorials (2021) *ESP8266 NodeMCU with HC-SR04 Ultrasonic Sensor with Arduino IDE* *https://randomnerdtutorials.com/esp8266-nodemcu-hc-sr04-ultrasonic-arduino/* [Accessed: 26 February 2022]

**[7]** Random Nerd Tutorials (2021) *ESP8266 DHT11/DHT22 Temperature and Humidity Web Server with Arduino IDE* https://randomnerdtutorials.com/esp8266-dht11dht22-temperature-and-humidity-web-server-with-arduino-ide/ [Accessed: 28 February 2022]

**[8]** Random Nerd Tutorials (2021) *How to Use I2C LCD with ESP32 on Arduino IDE (ESP8266 compatible)* https://randomnerdtutorials.com/esp32-esp8266-i2c-lcd-arduino-ide/ [Accessed: 1 March 2022]

## Academic papers used for ideation development

**[9]** Elangovan, Ramkumar & Santhanakrishnan, Nagarani & Rozario, Roger & Banu, Arjuman. (2018). Tomen: A Plant monitoring and smart gardening system using IoT. International Journal of Pure and Applied Mathematics. Volume 119.

**[10]** Kawakami, Ayumi & Tsukada, Koji & Kambara, Keisuke & Siio, Itiro. (2011). PotPet: pet-like flowerpot robot. 263-264. 10.1145/1935701.1935755.

**[11]** Velner, Koppen, Dolstra, de Vink. (2020). SmartPlant: an automatic plant care-taking service.

**[12]** Kush, Gabrani. (2020) IoT based Computing to Monitor Indoor Plants by using Smart Pot. Proceedings of the International Conference on Innovative Computing & Communications (ICICC).

**[13]** Yeong-Keun Lee, Koo-Rack Park, Dong-Hyun-Kim. (2019). Implementation of Smart Pot System using USB Plug-in Sensor.

International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-2S6

**[14]** Yawichai, Poomipiew, Boonyeun and Chucherd. (2020). Smart Mini Plant Pot. NU. International Journal of Science 2020; 17(1): 120-134.

# 8 Appendices

## Appendix A
### Dashboard

# Plant Care Dashboard

## Current Sensor Readings

**Current Soil Moisture(%):**

-38

**Current Temperature(degree Celsius):**

28.50

**Current Humidity(%):**

55.00

---

## Notifications

**Water Pump:**

ON

**Water Reservoir:**

Water reservoir is sufficient

**Temperature:**

The air temperature surrounding the plant is too low. Please move the plant to a warmer area

**Humidity:**

The air humidity is too low. Place a humidity tray next to the plant

---

## Requirements Set By User

**Soil moisture level set by user(%):**

10

**Temperature set by user(degree Celsius):**

40

**Humidity set by user(%):**

**Humidity set by user(%):**

60

**Distance set by user between the top of the reservoir and the water level (cm):**

9

---

## Enter Plant Requirements

**Set the minimum soil moisture level required by your plant**

Soil Moisture Level:
Submit

**Set the minimum temperature required by your plant**

Temperature:
Submit

**Set the minimum humidity required by your plant**

Humidity:
Submit

**Set the maximum distance(cm) between the top of the reservoir and the water level**

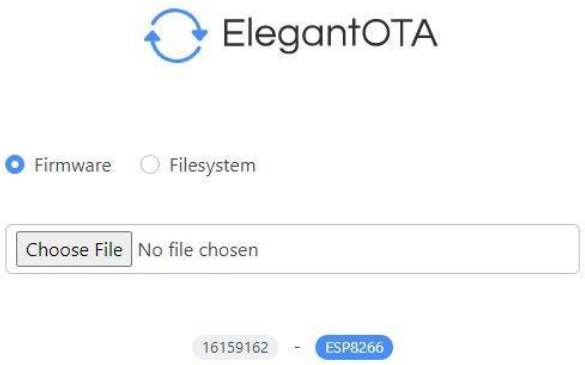
Distance:
Submit

# JSON Web page



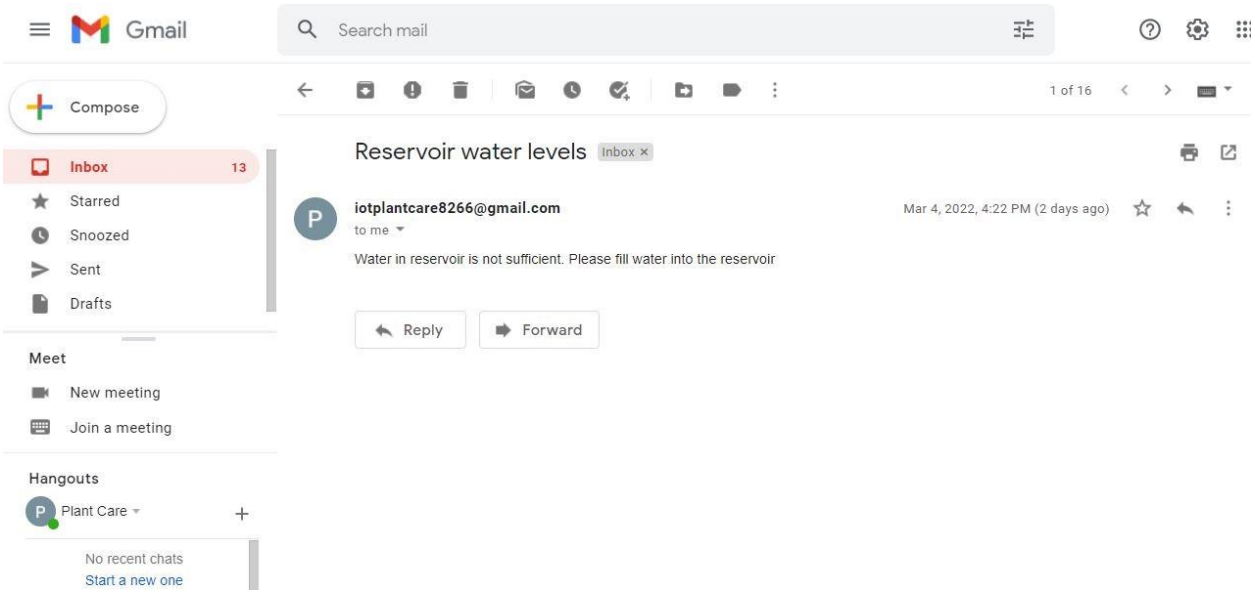
```
{
  "Content-Type": "application/json",
  "Status": 200,
  "Current Sensor Data": {
    "Current Soil Moisture Value": -40,
    "Current Temperature Value": 0,
    "Current Humidity Value": 0
  },
  "Notifications": {
    "Water Pump Status": "ON",
    "Reservoir Level Status": "Water reservoir is sufficient",
    "Temperature Status": "The air temperature surrounding the plant is too low. Please move the plant to a warmer area",
    "Humidity Status": "The air humidity is too low. Place a humidity tray next to the plant "
  },
  "Plant Requirements  Set by User": {
    "Minimum Soil Moisture Level": 10,
    "Minimum Temperature": 40,
    "Minimum Humidity": 60,
    "Maximum Distance": 9
  }
}
```


# OTA Web page



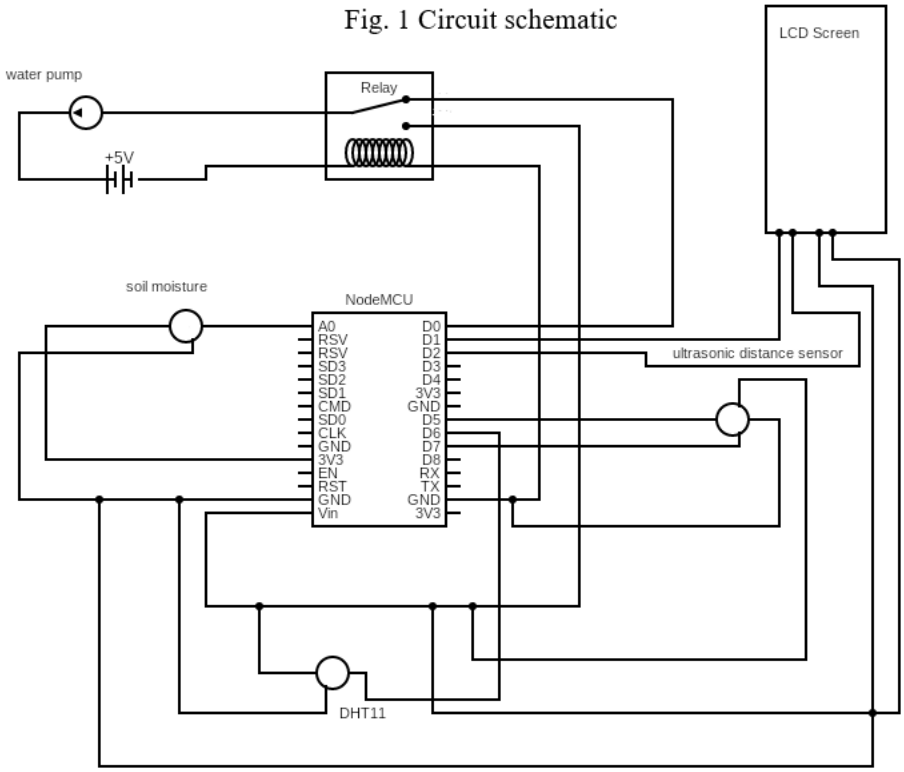

# Appendix B

## Image of the email sent to the user by the microcontroller



# Appendix C

Fig. 1 Circuit schematic




# Pin Connections

**Soil Moisture Sensor** => [A0, GND, 3V3]

**Ultrasonic Distance Sensor**

Trig => D7

Echo => D5

5V pin => Vin pin of microcontroller

GND pin of device => GND pin of microcontroller

**DHT11** => [D6, GND, Vin]

**LCD**

SCL => D1

SDA => D2

GND => GND

VCC => Vin pin of microcontroller

### Relay

The relay is connected to pin **D0** of the microcontroller. The pump is connected to the relay and a battery pack.

# Appendix D

## Component List

1. Capacitive Soil Moisture sensor
2. DHT11 Temperature and Humidity sensor
3. HC-SR04 Ultrasonic Sensor
4. Water pump
5. 12C 16x2 LCD screen
6. ESP8266 microcontroller

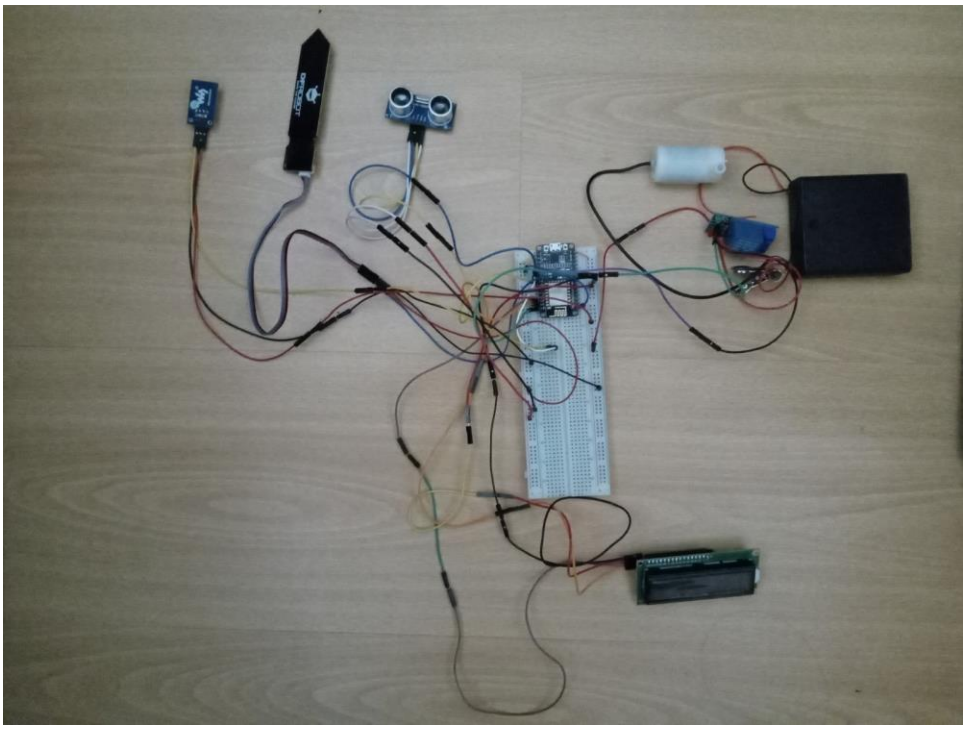
### Other Components

- Relay
- Breadboard
- Micro USB cable
- Jumper wires
- Battery pack
- Power bank
- Resistors
- Water container
- Plastic tubing
- Humidity tray
- Pot plant

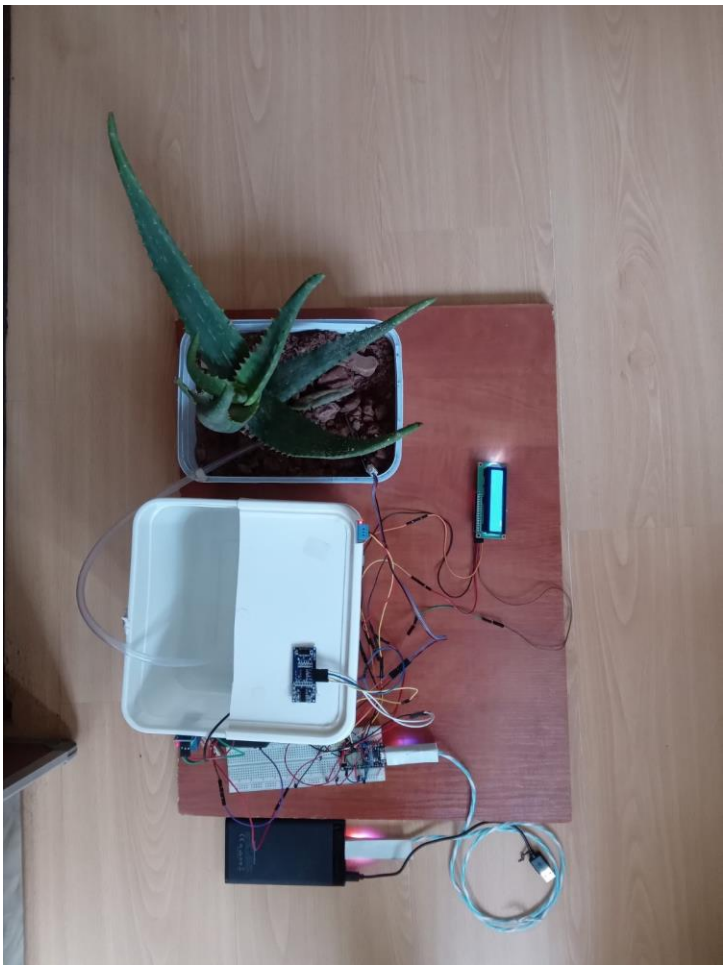
# Appendix E
## Dependency Libraries

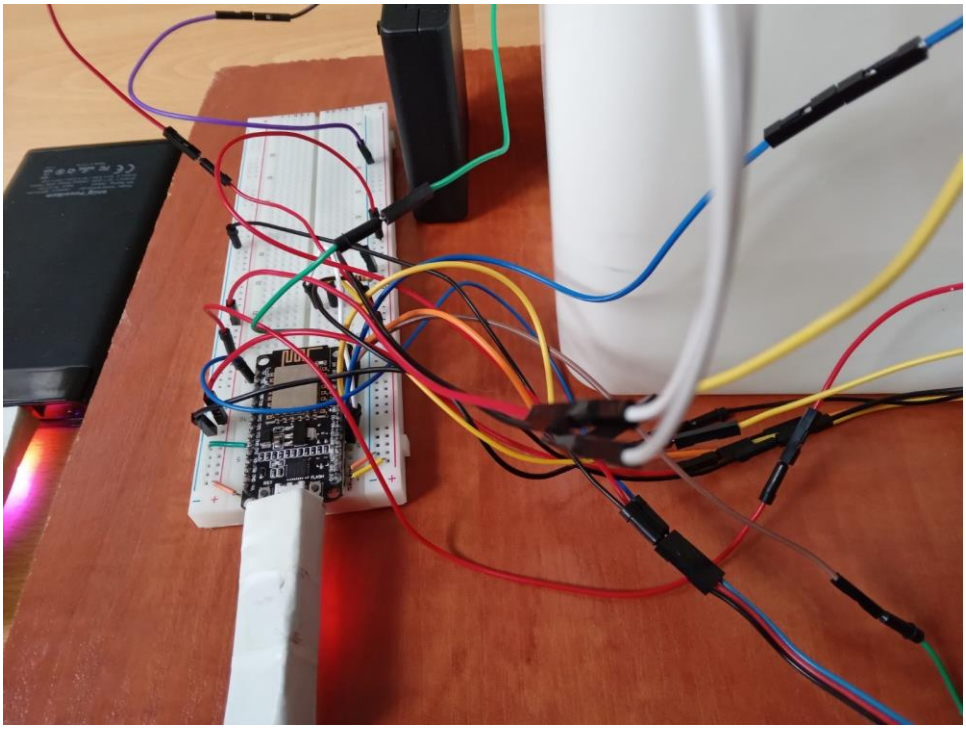
- Arduino.h
- ESP8266WiFi.h
- Hash.h
- ESPAsyncTCP.h
- ESPAsyncWebServer.h
- LiquidCrystal_12C.h
- AsyncElegantOTA.h
- EmailSender.h
- ArduinoJson.h
- Adafruit_Sensor.h
- DHT.h

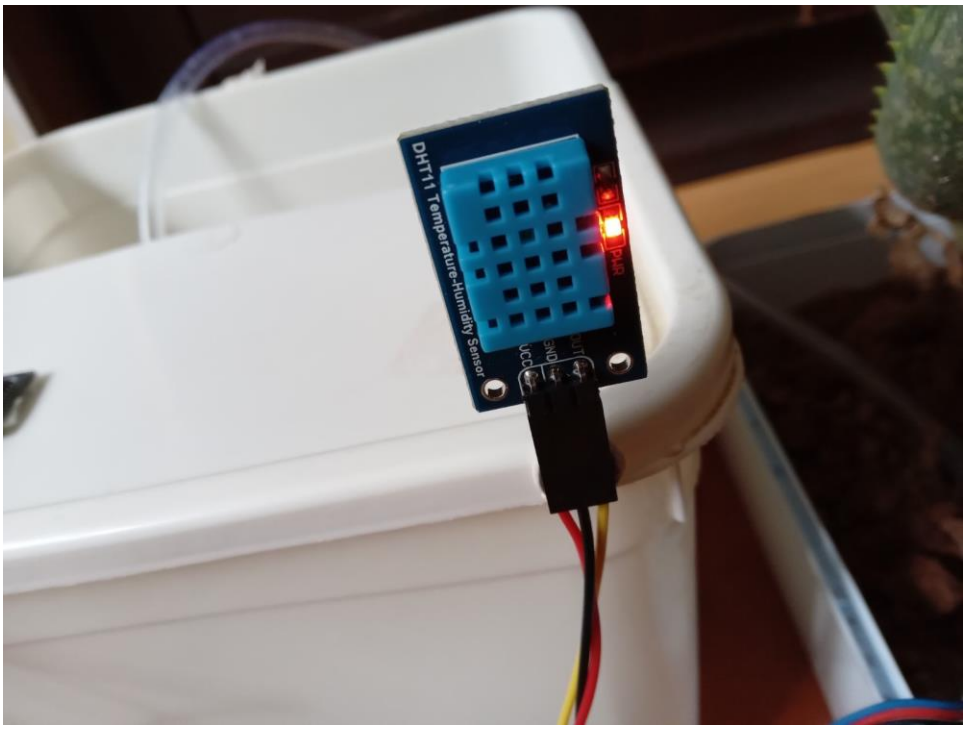
# Appendix F
## Image of Physical Circuit

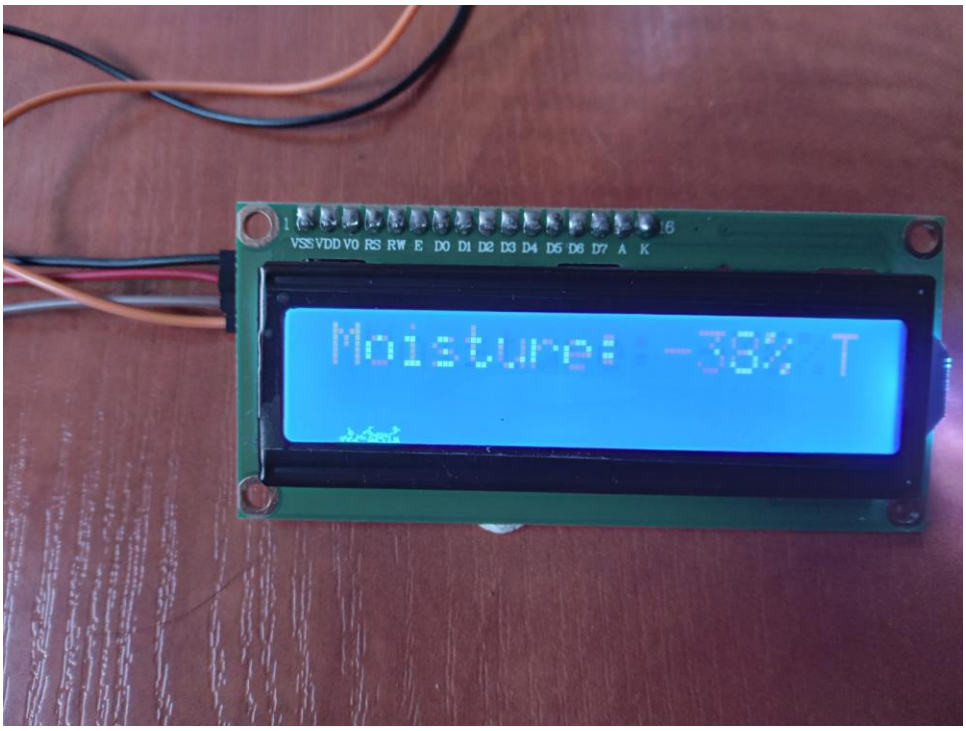


# Appendix G
## Images of Physical System Set Up

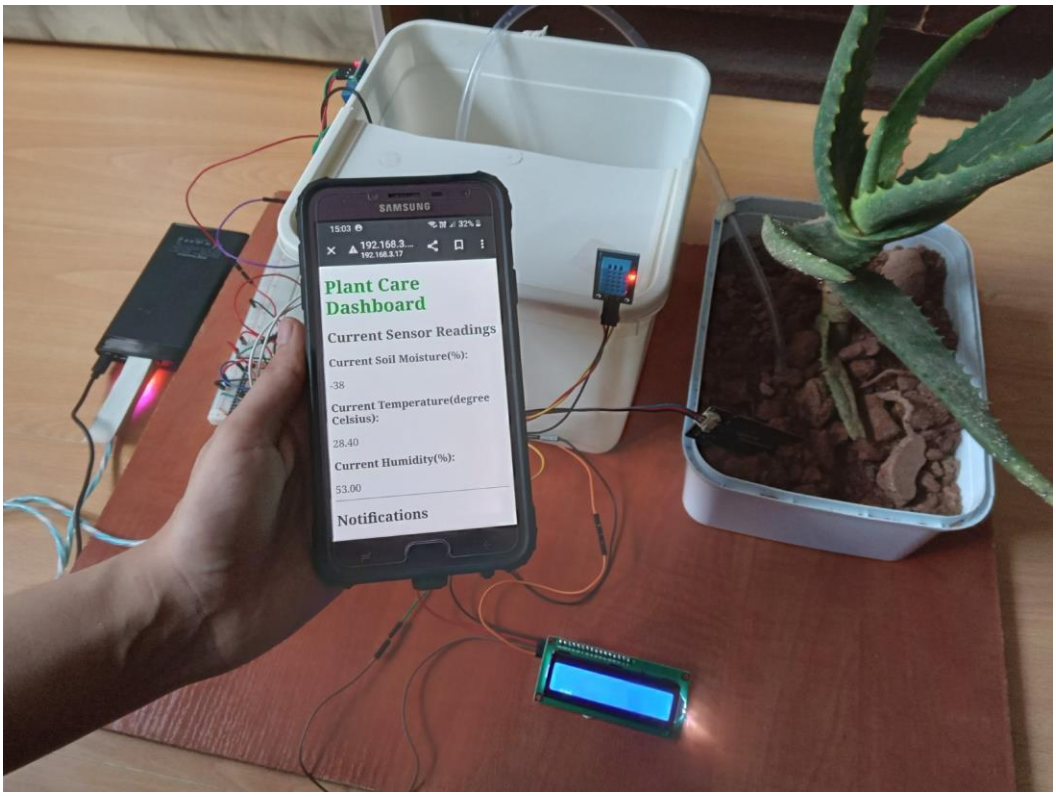





DHT11 Temperature-Humidity Sensor
PWR
GND
VCC


VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
Moisture: -38% T


Capacitive Soil
Moisture Sensor V1.0


SAMSUNG
15:03
192.168.3....
192.168.3.17
Plant Care Dashboard
Current Sensor Readings
Current Soil Moisture(%):
-38
Current Temperature(degree Celsius):
28.40
Current Humidity(%):
53.00
Notifications