Plant Communication System with IoT interfacing Using IFTTT & Thingspeak

FINAL REPORT
ECE3003
MICROCONTROLLER AND ITS APPLICATIONS

AJINKYA BAGMAR 20BEC0781
VINYAS SHETTY 20BEC0780
KARTHIK MEHROTRA 20BEC0691

Under the kind guidance of

Prof Karthikeyan B

School of Electronics and Communications EngineeringVellore

Institute of Technology, Vellore

November, 2022

1. ABSTRACT

We are all aware that houseplants are beneficial to our health. In addition to purifying the air, they destroy hazardous contaminants. Studies have also shown that indoor plants enhance focus and productivity (by up to 15 percent!), lower stress levels, and improve mood, making them ideal not just for the home, but also for the workplace.

Keeping your plants alive may be difficult due to their poor communication skills. In addition, during the last several decades, life has sped and gotten faster than ever before. The contemporary world continuously raises the standard and encourages us to take the fast lane. Herein lies the dilemma; due to our hectic lifestyles, we often neglect to care for our plants.

To address this issue, we have chosen to develop a communication tool that always informs us of the plant's state and reminds us to meet its demands.

2. ACCOMPLISHMENTS AND LEARNINGS

This project included the construction of a plant communication system that will assist us in monitoring the plant's requirements. It will inform us if the plant is experiencing any problems so that we can assist it immediately.

Following are the accomplishments at the conclusion of this project:

- Comprehend how the numerous sensors used in our project function.
- Examine the numerous Pros and Cons of the used sensors.
- Know the fundamentals of Arduino and its operation
- We have learned more about coding in the Arduino IDE using the C programming language.
- We learnt about the optimal growing conditions for plants.
- We also made use of Thingspeak and IFTTT which help us to get real time updates about the rating(condition) of our plant. This is beneficial as we can take the appropriate steps required for maintaining the condition of our plant.

3. HARDWARE COMPONENTS

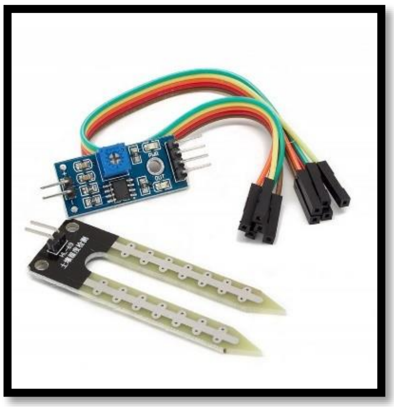
• Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P.

It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 Analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power, it with an AC-to-DC adapter or battery to get started.

• Soil Moisture Sensor

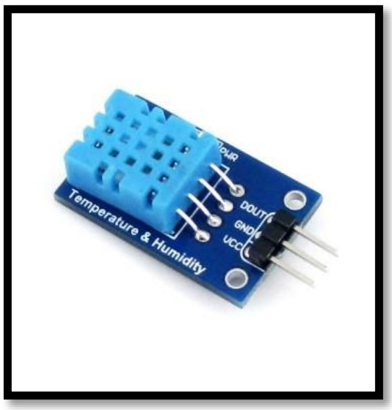


This sensor will be used to measure the amount of water present in the soil. It will be used to check if the plant has been watered properly.

This sensor will provide us with Analog data ranging from 0 -1024, which will then be programmed to identify the suitable amount of water required by the plant.

The Analog input pin for the soil moisture sensor is connected to the Analog pin A1 in the Arduino in our circuit.

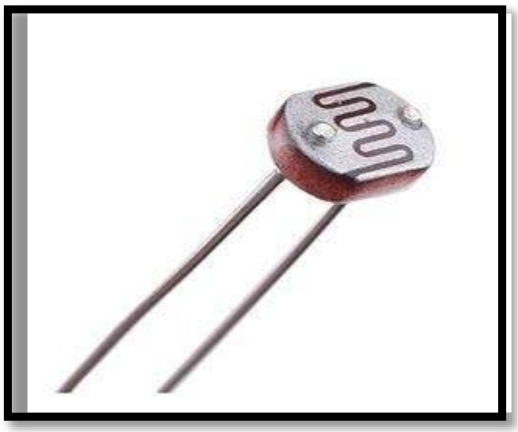
• DHT11 Temperature and Humidity sensor



This is a temperature and a moisture sensor. It will help us monitor if the plant is in suitable temperature and moisture conditions.

We will use an Arduino library for this sensor. The Data pin for this sensor is connected to the Digital pin 7 in our circuit

- **Photoresistor**



This is a light intensity-based resistor.

More the light intensity, more is the resistance.

This will help us in determining if the plant is in suitable light conditions.

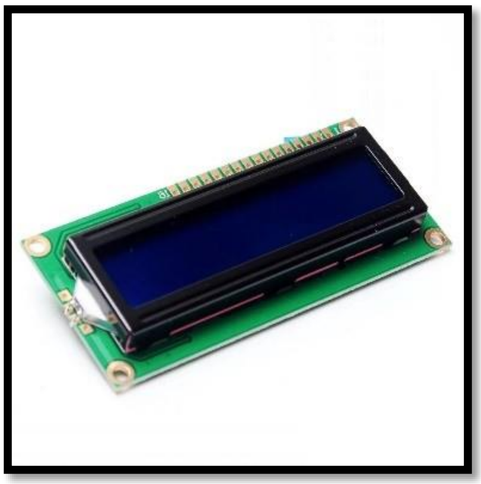
We have used the Analog pin A0 to get the voltage readings across the resistor .

- **LCD display**

An electronic device that is used to display data and the message is known as LCD 16×2. As the name suggests, it includes 16 Columns & 2 Rows so it can display 32 characters (16 x 2=32) in total & every character will be made with 5×8 (40) Pixel Dots. So, the total pixels within this LCD can be calculated as 32 x 40 otherwise 1280 pixels.

16 x 2 displays mostly depend on multi-segment LEDs. There are different types of displays available in the market with different combinations such as 8×2, 8×1, 16×1, and 10×2, however, the LCD 16×2 is broadly used in devices, DIY circuits, electronic projects due to less cost, programmable friendly & simple to access.

In our Project this display is mainly used to display necessary information to the user along with the score that the user has received for taking care of the plant. This display is connected to the Arduino with the help of an I2C bus, which helps in reducing the number of cables.



16x2 LCD Display

I2C bus

8×8 LED dot matrix

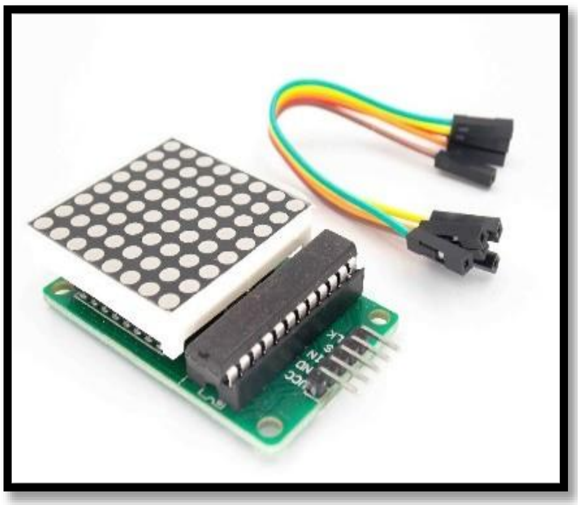
An 8 x 8 LED matrix display is used in this project to display the information. LED matrices are available in different styles like single colour, dual colour, multi-colour or RGB LED matrix.

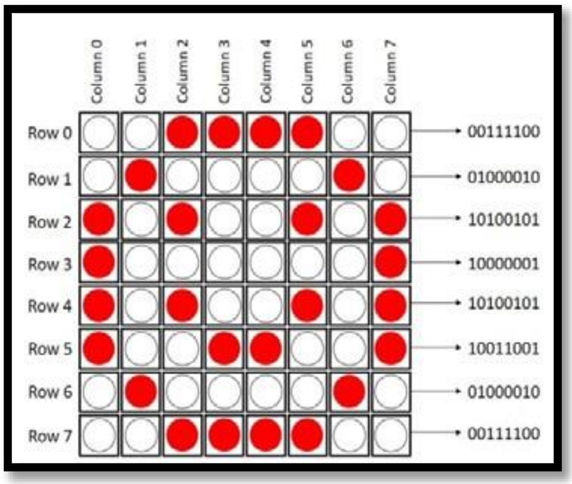
They are also available in different dimensions like 5 x 7, 8 x 8, 16 x 16, 32 x 32 etc. Based on the arrangement of the LEDs in the matrix, an LED matrix can be either common row anode or common row cathode. In case of common row anode type LED matrix, the current sources (high or positive voltage) are given to the rows A-D and the current sinks (low or negative voltage or ground) are given to the columns 1-4.

In case of common row cathode type LED matrix, the current sources (high or positive voltage) are given to the columns 1-4 and the current sinks (low or negative voltage or ground) are given to the rows A-D.

The LED matrix used in this project is a common row cathode type LED matrix. While developing the project, the type of LED matrix must be known and the program must be written accordingly.

This 8x8 matrix of LED will be used to project a face for the plant. If the plant is in healthy conditions, it will display a smile, if not it would display a sad face



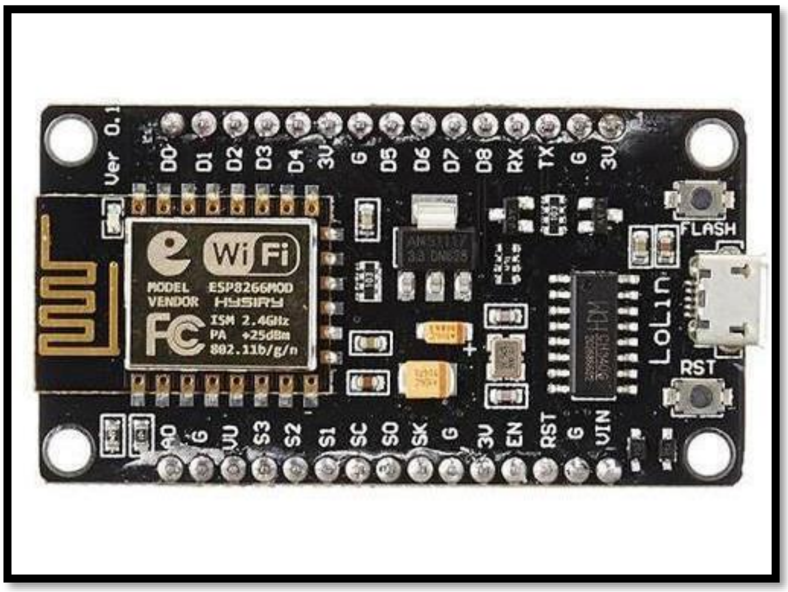


- **SERVO MOTOR**

- **NODE MCU/ESP32**

NodeMCU [ESP 32} is a microcontroller board and is a low cost open source IoT platform. NodeMCU acts as firmware and a microcontroller unit which is further used as IoT development kit.

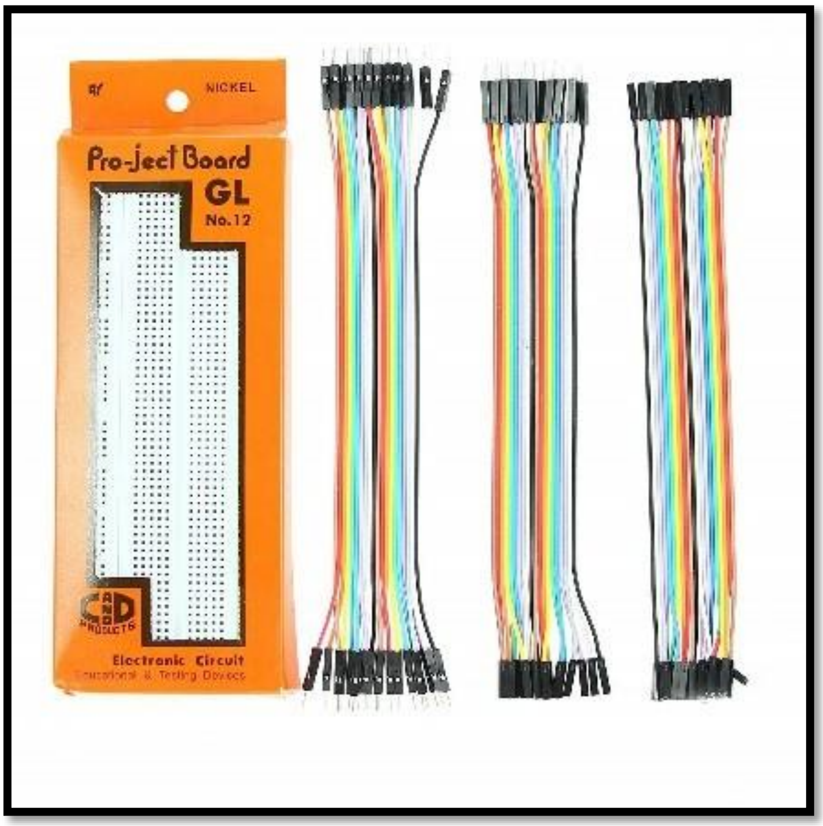
The RX2 and TX2 pins of the NodeMCU are connected with RX and TX pins of Arduino UNO which helps in transmitting the required data over WiFi to the ThingSpeak platform.


Ver 0.1
D0 D1 D2 D3 D4 3V G D5 D6 D7 D8 RX TX G 3V
WiFi
MODEL ESP8266MOD
VENDOR HYSIRY
ISM 2.4GHz
PA +25dBm
802.11b/g/n
FLASH
LoLin
RST
A0 G VU S3 S2 S1 SC SO SK G 3V EN RST G VIN

• Breadboard along with Jumper wires

It is an electric wire or group of them in a cable with a connector or pin at each end. A jump wire also known as jumper, jumper wire, jumper cable, DuPont wire or cable is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.



4. SOFTWARE USED

- **THINGSPEAK**

ThingSpeak is an open-source software written in Ruby which allows users to communicate with internet enabled devices. It facilitates data access, retrieval and logging of data by providing an API to both the devices and social network websites.ThingSpeak includes a Web Service that lets us to collect and store sensor data in the cloud(pall) and develop Internet of Things operations.

It works with Arduino, Rasberry Pi , Matlab, should work on all kinds of programming languages since it uses REST API and HTTP. In this study we get data (score) from our hardware model and analyze this data we use ThingSpeak inside which we create channel name and channel field (i.e Plant Feelings Communicator and data score in our case).

Inside channel we generate an API key for our project which important for our later steps. Then we add suitable IOT package for ESP 32 and we connect ESP 32 with our own wifi by filling wifi SSID and KEY with API key in the code blocks inside Madecode microbit.org.

The data is viewed that was sent from our channel to the IOT platform which updates every time we make changes in parameters in our hardware model.

- **IFFT**

IFTTT stands for "If This Then That." It's a free web service that helps users automate web-based tasks and improve productivity. IFTTT connects various developers devices, services and apps to create "applets" that perform automations.

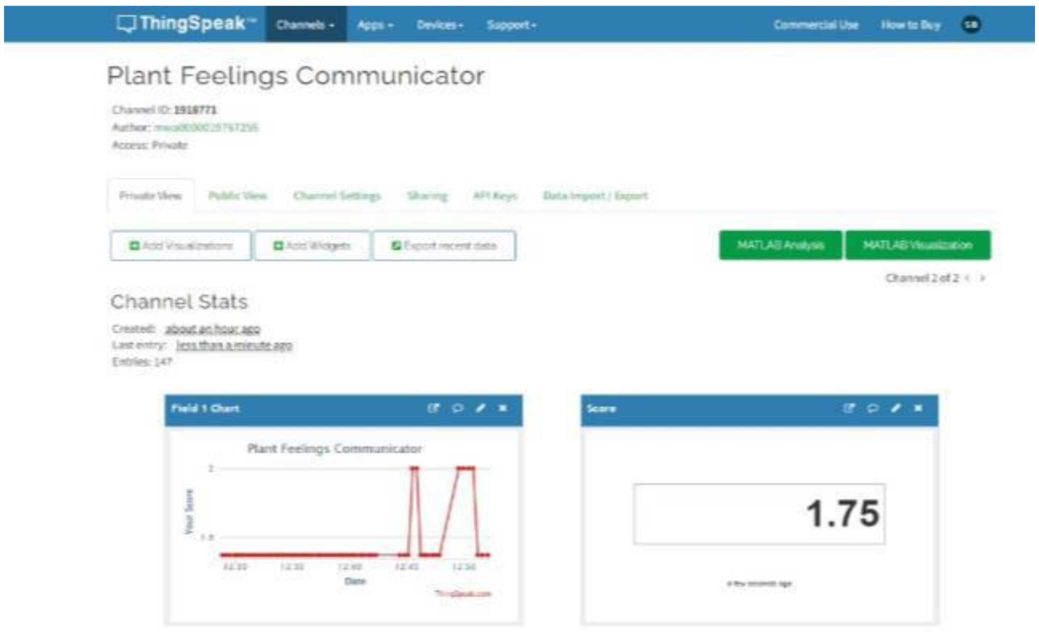
This will be an important part of our project as we will get notified about the score(condition of our plant) and we can take the appropriate steps to to take care of our plant if the rating is less.

Now in this study, IFTTT is used to receive an email whenever channel field score arrives at threshold value where it activates the trigger to perform the designated action.

(i) this operation: From WebHook settings an applet is created where a trigger is choosen by setting down the trigger fields.

Thus when trigger fires everytime our service receives a web request to notify it for an event.

(ii) that operation: It sends us an HTML based email where parameters of trigger is written



Plant Feelings Communicator

Channel ID: 1918771
Author: mwa0000025767256
Access: Private

Channel Stats

Created: about an hour ago
Last entry: less than a minute ago
Entries: 147

1.75

- **<u>ARDUINO CODE</u>**

```
#include <Wire.h>                                    // Including Wire library for the I2C Connected
to the LCD Display

#include <LiquidCrystal_I2C.h>                          // Including Liquid crystal library for I2C

#include "DHT.h"                                     // Including DHT library for DHT11 Temperature
and Humidity Sensor

#include <LedControl.h>

#include <Servo.h>


// Including LED library for controlling 8X8 dot matrix using MAX7219

#define DHTPIN 7 // Defining the sensor pin of DHT11 sensor connected to Arduino

#define DHTTYPE DHT11 // Defining the type of DHT sensor, since this library is built also for DHT22


const int en = 2,rw = 1, rs = 0,d4 = 4,d5 = 5,d6 = 6,d7 = 7, bl = 3; // LCD pinouts for the I2C
const int i2c_addr = 0x27; // I2C Address, found using another programme
const int msensor = A1; // Moisture Sensor data pin Connected to analog pin A1 in Arduino
const int Vin = A0; // Pin to receive analog readings from the photoresistor
const int DIN = 10; // Data pin of the MAX7219 8X8 dot matrix controller
const int CS = 9; // Another Data pin of the MAX7219 8X8 dot matrix controller
const int CLK = 8; // Clock Data pin of the MAX7219 8X8 dot matrix controller
const int buzz = 12;
int pos = 0;

LedControl lc = LedControl(DIN, CLK, CS, 0); // Defininng pins to control the 8X8 dot matrix
LiquidCrystal_I2C lcd(0x27, 16, 2); // Defining LCD connections to I2C bus
DHT dht(DHTPIN, DHTTYPE); // Defining and Setting up DHT sensor
Servo myservo;
int msvalue = 0; // Initial Value of Moisture Sensor
int light = 0; // Initial value of voltage across photoresistor
float tempC = 0.0; // Assigning variable to calculate the Temperature score
float humC = 0.0; // Assigning variable to calculate the Humidity score
float msenC = 0.0; // Assigning variable to calculate the Moisture score
float lightC = 0.0; // Assigning variable to calculate the Sunlight score
float hum; // Stores humidity value in percent
float temp; // Stores temperature value in Celcius
float score = 0.0;

byte smile[8] = {0x3C,0x42,0xA5,0x81,0xA5,0x99,0x42,0x3C}; // Byte array to store the "Smile Expression"
byte neutral[8] = {0x3C,0x42,0xA5,0x81,0xBD,0x81,0x42,0x3C}; // Byte array to store the "Neutral Expression"
byte sad[8] = {0x3C,0x42,0xA5,0x81,0x99,0xA5,0x42,0x3C}; //Byte array to store the "Sad Expression"

void setup() {
  Serial.begin(9600); //Setting up serial monitor
  pinMode(msensor, INPUT); //giving analog pin address (A1) to Arduino for Moisture sensor
  pinMode(Vin, INPUT); //giving analog pin address (A2) to Arduino for photoresistor
  dht.begin(); //Initialize DHT-22
  myservo.attach(13);
  lcd.begin(); // Set display type as 16 char, 2 rows
  lcd.backlight();
  lc.shutdown(0, false); //Initializing the MAX7219 8X8 dot matrix controller
  lc.setIntensity(0, 2); //Toggle to adjust LED brightness, maximum is 15
  lc.clearDisplay(0); //Clearing any previous display from the dot matrix
  lcd.setCursor(0, 0);
  delay(2000);
  lcd.print(" Welcome"); //Intro lines
  lcd.setCursor(0, 1);
  lcd.print(" to"); //Intro lines
  delay(2000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" Plant"); //Intro lines
  lcd.setCursor(0, 1);
  lcd.print(" Communicator"); //Intro lines
  delay(2000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" Micro 3003"); //Intro lines
  lcd.setCursor(0, 1);
  lcd.print("J COMP "); //Intro lines
  delay(2000);
  lcd.clear();
  //Serial.println("Connection to NODE MCU initiated");
}
void loop() {
  hum = dht.readHumidity(); //Get Humidity value
  temp = dht.readTemperature(); //Get Temperature value
```

```
//Serial.println(hum);
//Serial.println(temp);
lcd.clear(); //Clear the display
//Temperature Parameters
if (temp >= 18 && temp <= 26) {
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temp);
  lcd.print(" C");
  lcd.setCursor(0, 1);
  lcd.print(" PERFECT");
  tempC = 3.0;
  delay(2000);
} else if (temp >= 6 || temp <= 34) {
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temp);
  lcd.print(" C");
  lcd.setCursor(0, 1);
  lcd.print(" FINE");
  tempC = 2.0;
  delay(2000);
} else {
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temp);
  lcd.print(" C");
  lcd.setCursor(0, 1);
  lcd.print(" BAD");
  tempC = 1.0;
  delay(2000);
}
lcd.clear();
//Humidity Parameters
if (hum >= 50 && hum <= 60) {
  lcd.setCursor(0, 0);
  lcd.print("Humidity: ");
  lcd.print(hum);
  lcd.print("%");
  lcd.setCursor(0, 1);
  lcd.print(" PERFECT");
  humC = 3.0;
  delay(2000);
} else if (hum >= 40 || hum <= 80) {
  lcd.setCursor(0, 0);
  lcd.print("Humidity: ");
  lcd.print(hum);
  lcd.print("%");
  lcd.setCursor(0, 1);
  lcd.print(" FINE");
  humC = 2.0;
  delay(2000);
} else {
  lcd.setCursor(0, 0);
  lcd.print("Humidity: ");
  lcd.print(hum);
  lcd.print("%");
  lcd.setCursor(0, 1);
  lcd.print(" BAD");
  humC = 1.0;
  delay(2000);
}
```

```
  lcd.clear();

  //Soil Moisture Parameters

  msvalue = analogRead(msensor); //Reading Value from Moisture Sensor //Get Soil Moisture values from
analog pin

  //Serial.println(msvalue);

  if (msvalue >= 200 && msvalue <= 600) {

    lcd.setCursor(0, 0);

    lcd.print(" Soil Moisture");

    lcd.setCursor(0, 1);

    lcd.print(" PERFECT");

    msenC = 3.0;

    delay(2000);

  } else if (msvalue > 600) {

    lcd.setCursor(0, 0);

    lcd.print(" Soil Moisture");

    lcd.setCursor(0, 1);

    lcd.print(" LOW");

    msenC = 1.0;

    delay(2000);

  } else {

    lcd.setCursor(0, 0);

    lcd.print(" Soil Moisture");

    lcd.setCursor(0, 1);

    lcd.print("Excess Water");

    msenC = 2.0;

    delay(2000);

  }

  lcd.clear();



  //Sunlight Parameters
```

```
light = analogRead(Vin); //Reading Voltage across the photoresistor to calculate the light intensity
//Serial.print("The PhotoResistor value is:");
//Serial.println(light);
//Serial.println(Vin);
if (light >= 750) {
  lcd.setCursor(0, 0);
  lcd.print(" Sunlight");
  lcd.setCursor(0, 1);
  lcd.print(" PERFECT");
  lightC = 3.0;
  delay(2000);
} else if (light >= 300) {
  lcd.setCursor(0, 0);
  lcd.print(" Sunlight");
  lcd.setCursor(0, 1);
  lcd.print("Could be Better");
  lightC = 2.0;
  delay(2000);
} else {
  lcd.setCursor(0, 0);
  lcd.print(" Sunlight");
  lcd.setCursor(0, 1);
  lcd.print("Needs Sunlight!!");
  lightC = 1.0;
  delay(2000);
}
lcd.clear();


//All parameters have been checked and we now calculate the score
score = ((tempC + humC + msenC + lightC) / 4);
```

```
if (score >= 2.5) {
  lcd.setCursor(0, 0);
  lcd.print("Score: ");
  lcd.print(score);
  lcd.print("/3.00");
  lcd.setCursor(0, 1);
  lcd.print(" Plant is HAPPY "); //Print how the plant feels on bottom line
  printByte(smile); //Display Plant's expression on the DOT Matrix
  delay(3000);
} else if (score >= 2) {
  lcd.setCursor(0, 0); //Print Caretaking score on top line
  lcd.print("Score: ");
  lcd.print(score);
  lcd.print("/3.00");
  lcd.setCursor(0, 1); //Print how the plant feels on bottom line
  lcd.print(" Plant is Fine"); //Display Plant's expression on the DOT Matrix
  myservo.write(0);
  printByte(neutral);
  delay(3000);
} else {
  lcd.setCursor(0, 0); //Print Caretaking score on top line
  lcd.print("Score: ");
  lcd.print(score);
  lcd.print("/3.00");
  lcd.setCursor(0, 1); //Print how the plant feels on bottom line
  lcd.print(" Plant is SAD"); //Display Plant's expression on the DOT Matrix
  printByte(sad);
  tone(buzz,261);
  myservo.write(90);              // tell servo to go to position in variable 'pos'
  delay(15);
  delay(500);
```

```
    tone(buzz,440);
    delay(500);
    noTone(buzz);
    delay(2000);
  }
  lcd.clear();
  Serial.println(score);
}
void printByte(byte character[]) //Function, called to print Expressions on the DOT Matrix Display
{
  int i = 0;
  for (i = 0; i < 8; i++) {
    lc.setRow(0, i, character[i]);
  }
```

- **<u>ESP 32 CODE</u>**

```
#include "WiFi.h"
#include <ThingSpeak.h>;
#define RXp2 16
#define TXp2 17
WiFiClient client;

char* ssid = "Micro"; //Enter SSID
char* password = "1234567*()"; //Enter Password

unsigned long myChannelNumber = 1927063; //Your Channel Number (Without Brackets)
char * myWriteAPIKey = "M221Z4AHV6N1Z3WJ"; //Your Write API Key

String a;
float dat = 0.0;
```

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
     delay(500);
     Serial.print("*");
  }
  ThingSpeak.begin(client);
  Serial.println("");
  Serial.println("WiFi connection Successful");
  Serial.print("The IP Address of ESP32 Module is: ");
  Serial.println(WiFi.localIP());// Print the IP address
}
void loop() {


    if(Serial2.available() > 0) {
    a = Serial2.readString();
    Serial.println("Value Fetched");
    dat = a.toFloat();
    Serial.println(dat);
    }
    ThingSpeak.writeField(myChannelNumber, 1,dat, myWriteAPIKey); //Update in ThingSpeak
    delay(4000);
}
```

5. WORKING

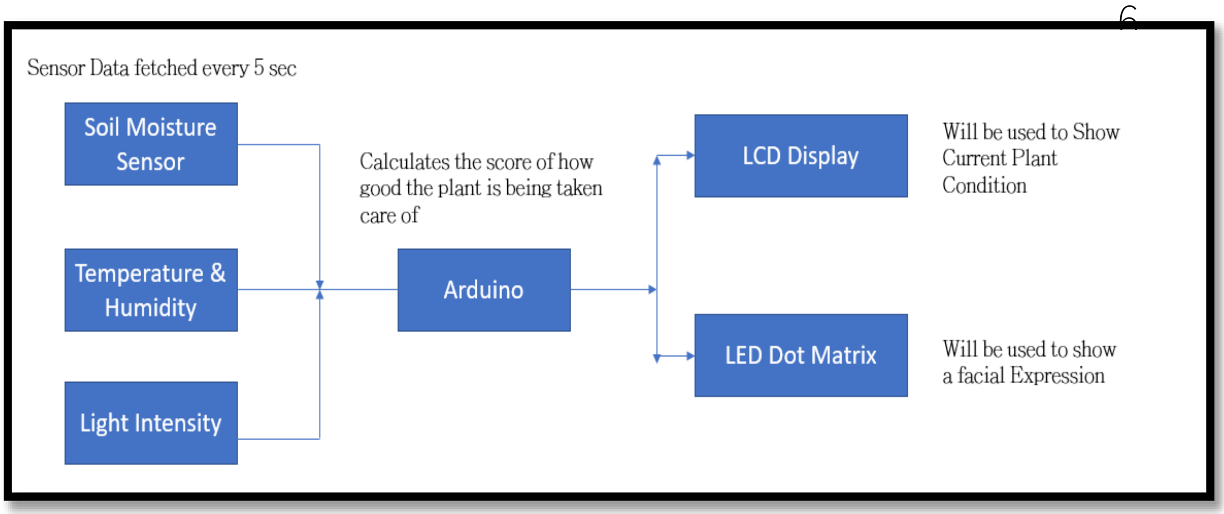
The brains of our project will be an Arduino – Uno. It is programmed in C/C++ language with the help of the open-source IDE "Arduino Software."
All our Sensors, displays will be connected to the Arduino using the jumper cables and the breadboard. Our Code will be plant specific i.e., all the conditions (temperature, light intensity) will be based on a particular plant.
This code will be compiled and uploaded to the Arduino with the Arduino IDE.

At first, all the data from the sensors will be collected, and displayed to the user one by one with a delay. Each of the parameter will have a score from 1-3. Based on the conditions of the plant, the score will be higher/ lower.

Finally the average score of all the parameters will be taken, and the user will be shown the score. This score will help the user know exactly how well they have taken care of the plant.
Now based on this score, the 8x8 LED Dot matrix will display a face. If the score is high, it will show a smiling face, if the score is low, it will show a sad face. This gives the plant a "Face" to communicate.

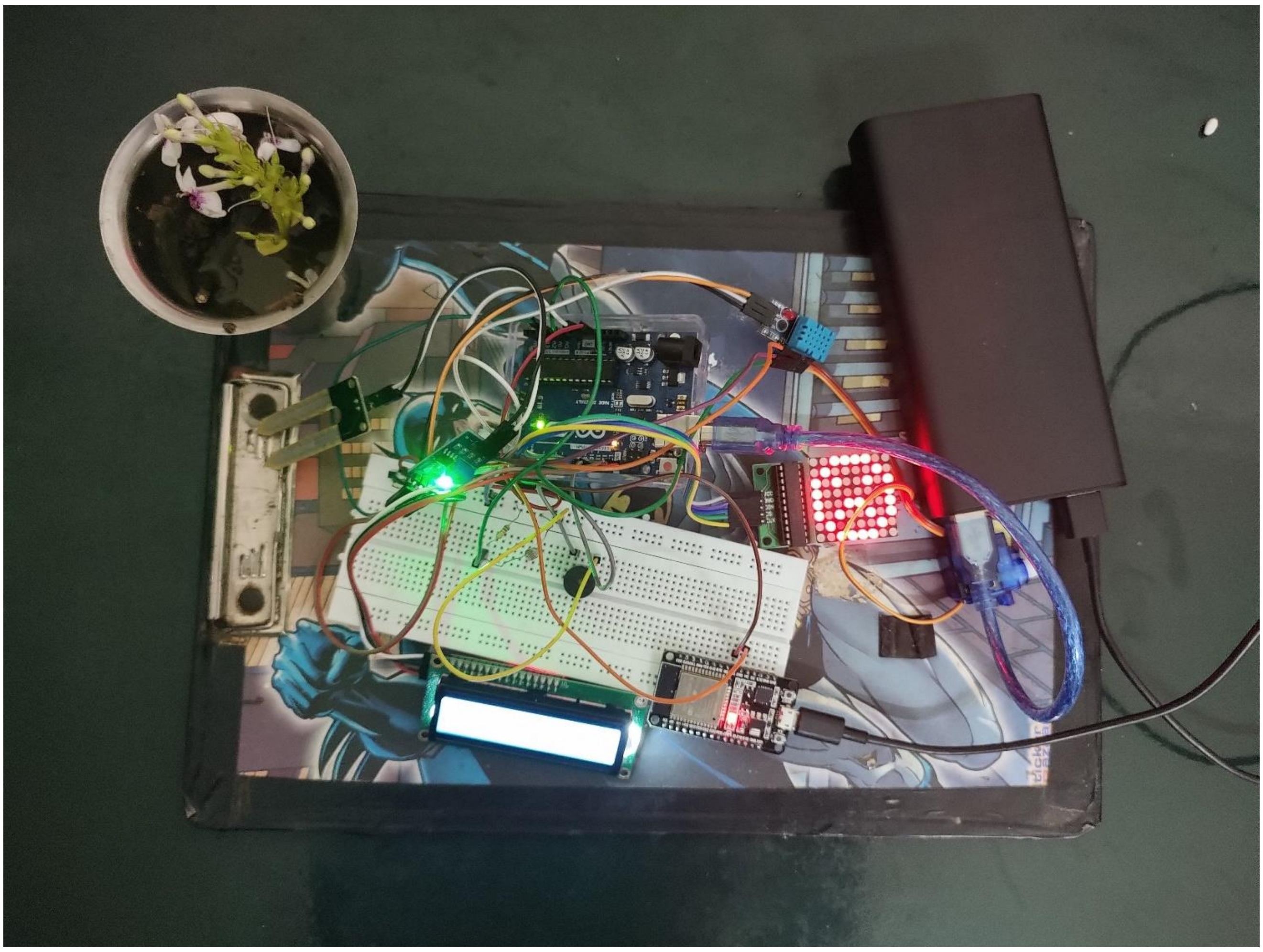
Statistical score of the plant condition is sent to the Thingspeak software and triggered through IFTTT if the condition of the plant is not good.

Block Diagram



Sensor Data fetched every 5 sec
Soil Moisture Sensor
Temperature & Humidity
Light Intensity
Calculates the score of how good the plant is being taken care of
Arduino
LCD Display
LED Dot Matrix
Will be used to Show Current Plant Condition
Will be used to show a facial Expression

6. HARDWARE ASSEMBLED



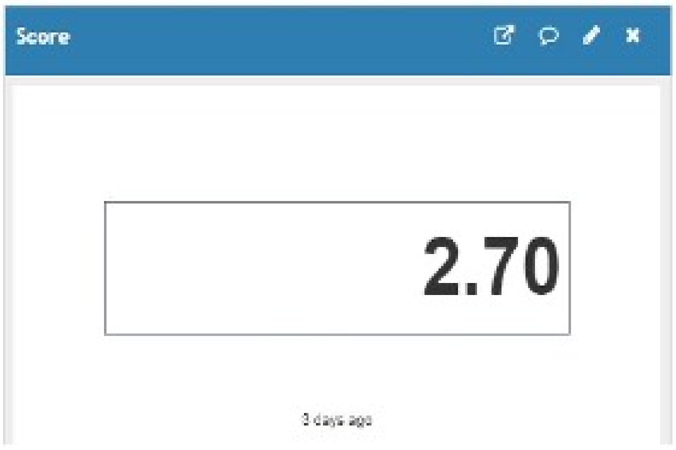
7. Results

Plant Communication system is a simple but effective communication tool which will help us to understand, the basic needs of the plant. This system will indicate and remind us if the plant is facing any scarcity of the basic needs.

In this busy world all the people are busy in their works this communication system will help them to take care of the plants. This plant communication system can be used for household plants, offices and also in any other places, where growing up plants are necessary.

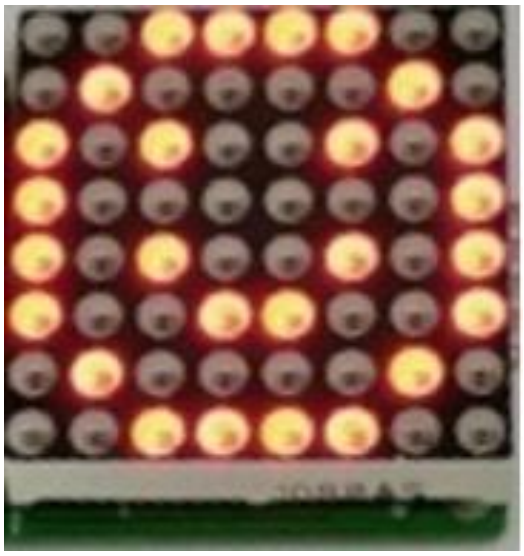
The cost of implementation of this plant communication system is also low and can be further reduced with the aid of process specific printed circuit boards.

If the plant's condition is in ideal state where the humidity temperature soil moisture and sunlight parameter are met we get the following score and a smiley face on our lcd display.

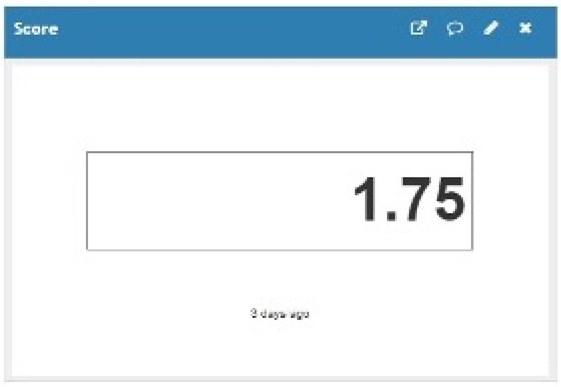

Score

2.70

3 days ago

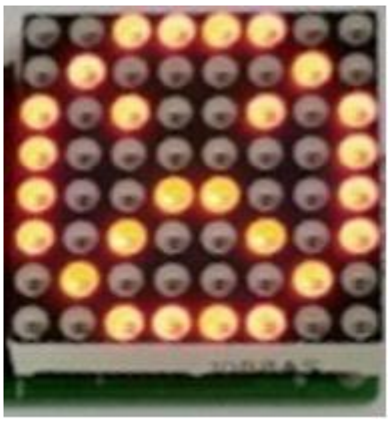


In this case a sad a sad face is displayed on the 8x8 matrix indicating the plants condition is poor. The rating is 1.75 and as a result we get a notification in our ThingHTTP and a service is triggered which helps us get notified through and email via IFFT.


Score

1.75

3 days ago



8.Acknowledgement

We would like to thank our university, VIT Vellore for providing us with the opportunity to do this project. The constant guidance and support from our faculty, Prof. Karthikeyan B enabled us to understand the concepts better and apply them in our project.

We would like to show our greatest of gratitude towards our faculty for allowing us to do this project and being there for us in every step. His enormous faith helped us to focus on our project and complete it on time.

9.References

[1] SHAIKH SHEROZ MOHD HASAN, ‘AUTO IRRIGATION USING ARDUINIO’2016.

[2] Automatic Plant Watering and Monitoring System using NodeMCU

[3] M. Giri and D.N. Wavhal, “Automated Intelligent Wireless Drip Irrigation Using Linear Programming,” International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) vol. 2, no. 1, pp. 1–5, 2013.

[4]] D. Bansal and S. R. N. Reddy, “WSN Based Closed Loop Automatic Irrigation System,” International Journal of Engineering Science and Innovative Technology (IJESIT), vol. 2, no. 3, pp. 229–237, 2013.

[5] Devika, C. M., K. Bose, and S. Vijayalekshmy. "Automatic plant irrigation system using Arduino." IEEE International Conference on Circuits and Systems (ICCS), Thiruvananthapuram, India 20-21 Dec,2017.

[6] M. Ramu and C. H. Rajendra, “Cost effective atomization of Indian agricultural system using 8051

microcontroller,” International Journal of Advanced Research in Computer and Communication Engineering,

vol. 2, no. 7, pp. 2563-2566 , 2013.Devika et al., International Journal of Advanced Research in Computer

Science and Software Engineering 4(10), October -2014, pp. 449-456

[7] S. G. Zareen, K. S. Zarrin, A. R. Ali and S. D. Pingle “Intelligent Automatic Plant Irrigation System,”,

International Journal of Scientific Research and Education, vol. 4, no. 11, pp. 6071–