ARCS

Electro Mavericks

AUTOMATIC RETRACTABLE CLOTHESLINE SYSTEM

Final Report

PRABODHA K.P.K.A	210490L
VIDMAL H.V.P	210668P
NAYANTHARA J.N.P	210418C
SURENDRA S.A.J.E	210625H

Semester 2 – Group Project (Electro Mavericks)

By

Electro Mavericks

Contents

Abstract

Although traditional clothesline systems have been in use for decades, they have issues and limitations that make it difficult to properly dry clothes, especially with unpredictable rains and busy life schedules. So, we identified a definite need for an automatic clothesline system which solves the above problems. Our objective was to design a product which detects weather conditions and automatically acts accordingly and enables customers to monitor the current situation via an IOT system. Initially, the complete code was implemented using Arduino C++. For testing purposes, breadboard implementation was used. In parallel to the code, schematics and PCB layout were developed. The enclosure design was developed using Solid works. Further, the physical product was developed to get result, correcting all the impairments beforehand using breadboard implementation. Throughout this project, we gained PCB design skills, enclosure design skills, improved logical thinking, programming skills, and improved teamwork skills.

1) Introduction

Conventional clothesline systems face challenges with unpredictable weather and user inconvenience. Users may not be able to bring their clothes inside during rain if they are away from home or unable to monitor the rainy weather, which exacerbates the issue. As a result, clothing may be exposed to the rain for extended periods of time, potentially harming the fabric or shortening its lifespan. Similar to this, customers who have busy schedules or who are away from home could find it difficult to manage their laundry. As a result, clothes may not dry during the day if the user is not home to leave them outside or bring them inside, and it may be challenging to constantly put clothes outside and bring them inside when it rains throughout the day. (Make this to less sentences)

2) Method

2.1) Firmware Development

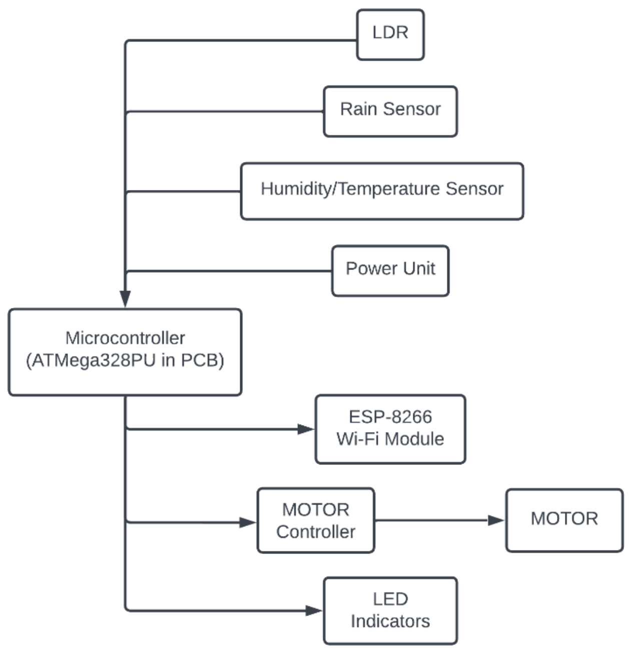


Figure 1. Block-Diagram of the System

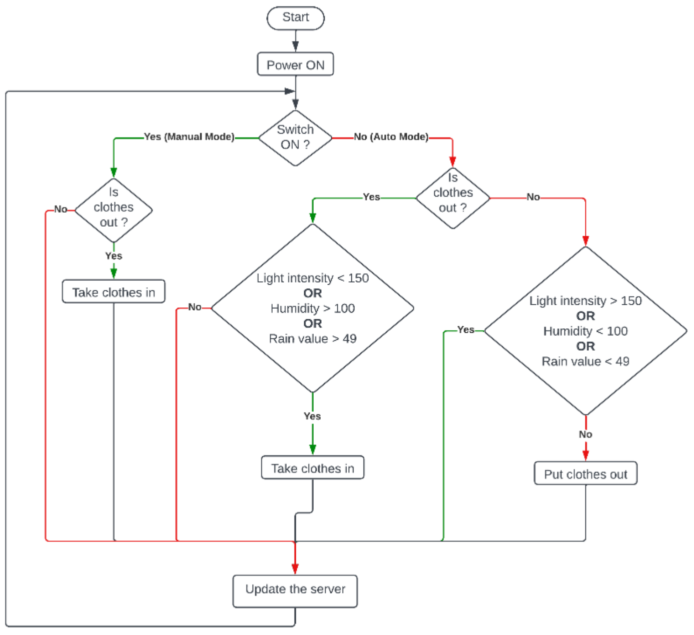


Figure 2. Main Algorithm

2.2) PCB Design

2.2.1. Schematic diagram

The Schematic diagram made for the design of PCB can be found in Appendix 1.

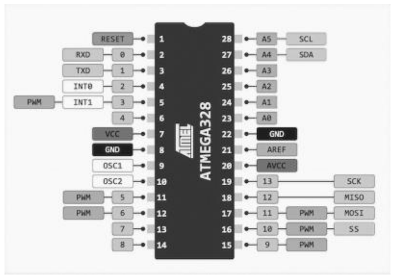
2.2.2. PCB layout

The PCB Layout also made for the design of PCB can be found in appendix 2. The dimensions of the final PCB were 44.5 mm * 39 mm. The top and bottom of the 2D and 3D views can be found in appendix 2 and appendix 3 respectively.

2.3. Components

2.3.1. ATMEGA328P

ATmega328P is a popular low-power CMOS 8-bit microcontroller which achieves high processing speeds of up to 1 MIPS per MHz because of its robust RISC design. It is a flexible option for a variety of embedded systems because it has 32KB of programmable flash memory, 1KB of EEPROM, and 2KB of SRAM. Additionally, it is compatible with a wide range of programming tools and provides a variety of peripherals, including ADC, USART, SPI, and I2C interfaces.



2.3.2. ESP8266 Wi-Fi Module

We have chosen to use the ESP8266 for our clothesline system since it is a practical choice that satisfies our needs for fundamental Wi-Fi connectivity. Our main goal is to produce a low-cost solution, therefore the ESP8266's price is a significant benefit. Furthermore, the ESP8266 has Wi-Fi connectivity, which is all we want for our clothesline system, and sufficient computing power to handle routine duties.

2.3.3. DC Stepper Motor

The major reason for using a DC motor in our clothesline system is that DC motors are better suited since they have a larger starting torque and can maintain their torque at lower speeds, making them more suitable for applications where a high level of control and precision is required, such as in our clothesline system. This will guarantee that the clothesline operates effectively and smoothly and enable us to program it to stop at predetermined intervals.

2.3.4. Rain Sensor Module

A raindrop sensor module, also known as a rain sensor or rain detector, is an electronic component used to detect rainfall or the presence of water. Its primary purpose is to provide a signal when raindrops fall on its surface or when there is water on it, enabling various applications in weather monitoring

2.3.5. Temperature Humidity Sensor

The DHT22 temperature humidity sensor is a popular and versatile sensor module used to measure both temperature and relative humidity in the surrounding environment. Its purpose is to provide real-time data on temperature and humidity levels, making it useful for a wide range of applications, including weather monitoring and various IoT (Internet of Things) projects.

2.3.6. Motor Controller

The L298N is a motor controller chip used to control the direction and speed of DC motors or stepper motors. It has two H-bridges for bi-directional control and works with microcontrollers like ATMEGA328P. Commonly used in robotics and projects needing motor control, it handles higher voltages, allows speed control, and requires additional components for proper operation.

2.3.6. Breadboard Power Supply

The MB102 breadboard power supply module provides regulated voltage (typically 3.3V or 5V) to power components on a breadboard during electronics prototyping. It's compact, fits directly onto the breadboard, and requires an external DC power source. It's a convenient tool for simplifying power distribution in projects.

2.3.8. LDR

Using an LDR (Light Dependent Resistor) in an automatic retractable clothesline system can serve the purpose of controlling the system's operation based on ambient light conditions. The LDR is used as a sensor to detect the presence or absence of sunlight or light levels in the surrounding environment. It enables the clothesline system to automatically retract or extend the clothesline based on the user's preferences and prevailing lighting conditions.

2.3.9. Other Components

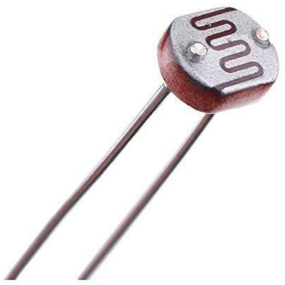


Table 1 listed the other components we have used except the components listed above.

- Crystal oscillator 16MHz 1
- Push button - 2
- Capacitor 10uF - 1

  22pF - 2
- Resistor 1kOhm - 2
- LED - 2

2.4. Enclosure Design

The enclosure is designed with a glossy finish to give a futuristic appearance. (See appendix 4, 5)

Material: Plastic

Enclosure I:

- Dimensions of enclosure - 148*83*40 mm
- Weight (without PCB, sensors, and modules) – 146 g

Enclosure II:

- Dimensions of enclosure – 85*55*26 mm
- Weight (without PCB, sensors, and modules) - 40 g

3) Result

The rain sensor, LDR and the DHT22 sensor and Wi-Fi module worked properly without any failure.
The results were observed using the breadboard prototype (See figure 1). The final results were observed after assembling PCB and other circuit components to the enclosure (See figure 2)
The functions of the final code have been attached to appendix 6.

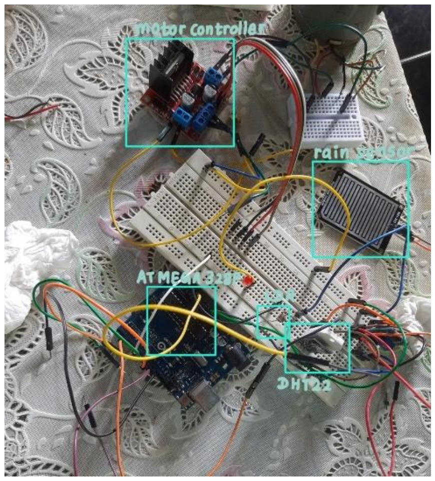


Figure 1

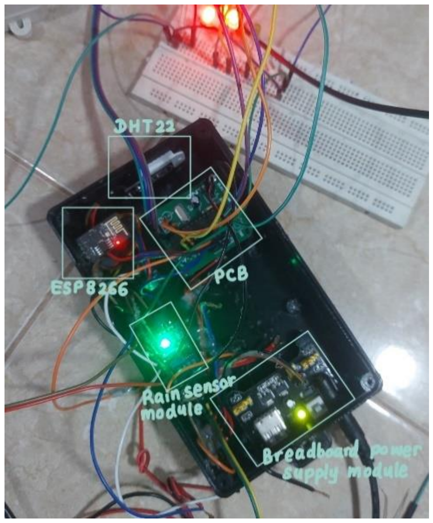
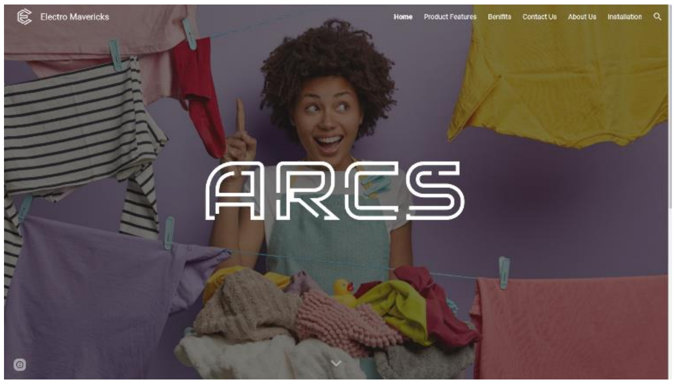
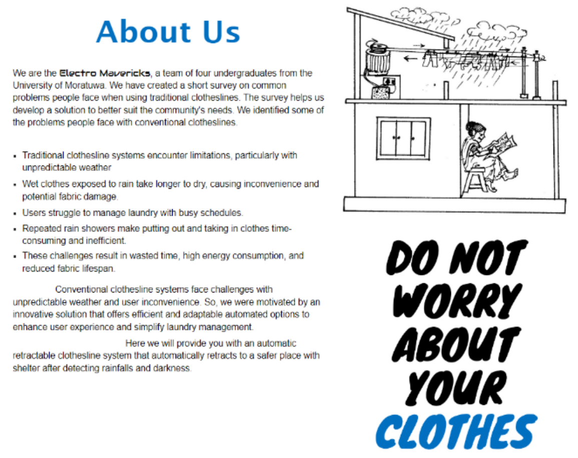


Figure 2

We introduce a mobile app and a website(www.electromavericks.systems) to help the customer to manage this product easily. We can use the mobile app to operate this product manually. We include some graphical charts about the variations that happened up to the current situation in this app.





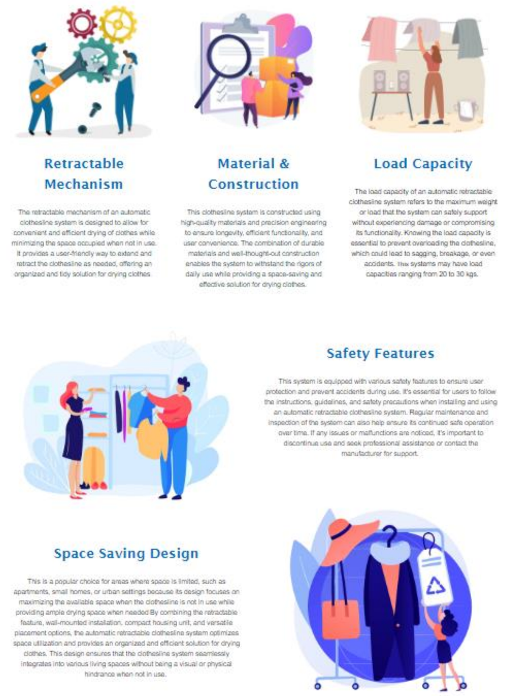
About Us

We are the **Electro Mavericks**, a team of four undergraduates from the University of Moratuwa. We have created a short survey on common problems people face when using traditional clotheslines. The survey helps us develop a solution to better suit the community's needs. We identified some of the problems people face with conventional clotheslines.

- Traditional clothesline systems encounter limitations, particularly with unpredictable weather
- Wet clothes exposed to rain take longer to dry, causing inconvenience and potential fabric damage.
- Users struggle to manage laundry with busy schedules.
- Repeated rain showers make putting out and taking in clothes time-consuming and inefficient.
- These challenges result in wasted time, high energy consumption, and reduced fabric lifespan.

Conventional clothesline systems face challenges with unpredictable weather and user inconvenience. So, we were motivated by an innovative solution that offers efficient and adaptable automated options to enhance user experience and simplify laundry management.

Here we will provide you with an automatic retractable clothesline system that automatically retracts to a safer place with shelter after detecting rainfalls and darkness.



Retractable Mechanism

The retractable mechanism of an automatic clothesline system is designed to allow for convenient and efficient drying of clothes while minimizing the space occupied when not in use. It provides a user-friendly way to extend and retract the clothesline as needed, offering an organized and tidy solution for drying clothes.

Material & Construction

This clothesline system is constructed using high-quality materials and precision engineering to ensure longevity, efficient functionality, and user convenience. The combination of durable materials and well-thought-out construction enables the system to withstand the rigors of daily use while providing a space-saving and effective solution for drying clothes.

Load Capacity

The load capacity of an automatic retractable clothesline system refers to the maximum weight or load that the system can safely support without experiencing damage or compromising its functionality. Knowing the load capacity is essential to prevent overloading the clothesline, which could lead to sagging, breakage, or even accidents. This systems may have load capacities ranging from 20 to 30 kgs.

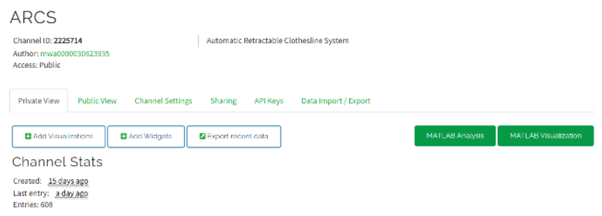
Safety Features

This system is equipped with various safety features to ensure user protection and prevent accidents during use. It's essential for users to follow the instructions, guidelines, and safety precautions when installing and using an automatic retractable clothesline system. Regular maintenance and inspection of the system can also help ensure its continued safe operation over time. If any issues or malfunctions are noticed, it's important to discontinue use and seek professional assistance or contact the manufacturer for support.

Space Saving Design

This is a popular choice for areas where space is limited, such as apartments, small homes, or urban settings because its design focuses on maximizing the available space when the clothesline is not in use while providing ample drying space when needed By combining the retractable feature, wall-mounted installation, compact housing unit, and versatile placement options, the automatic retractable clothesline system optimizes space utilization and provides an organized and efficient solution for drying clothes. This design ensures that the clothesline system seamlessly integrates into various living spaces without being a visual or physical hindrance when not in use.

Mobile App



ARCS

Channel ID: **2225714** | Automatic Retractable Clothesline System

Author: mwa0000030623935

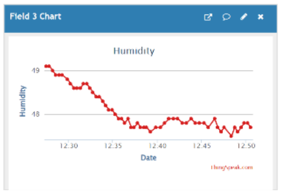
Access: Public

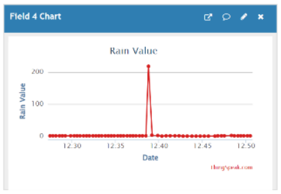
Channel Stats

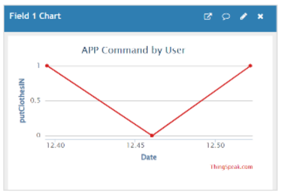
Created: 15 days ago

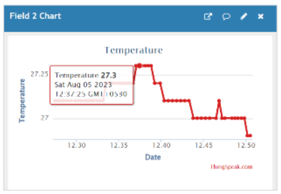
Last entry: a day ago

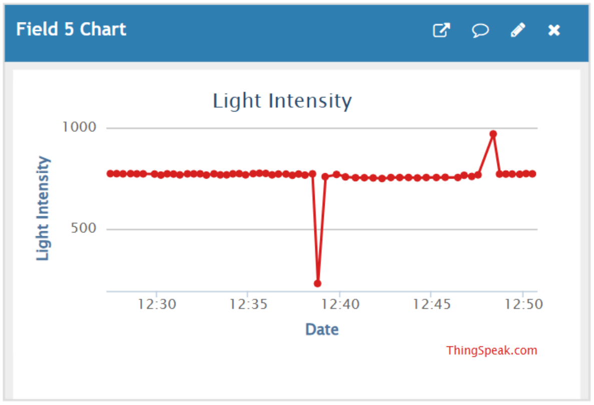
Entries: 608











4) Budget

Components	Quantity	Price (Rs.)
Rain Sensor (YL-83-FC-37)	1	300.00
Humidity/Temperature Sensor	1	1100.00
Atmel ATMEGA 328P-PU	1	1500.00
ESP8266	1	880.00
LDR	1	20.00
Crystal Oscillator	1	35.00
LED	2	10.00
10uF Capacitors	1	5.00
22pF capacitors	2	10.00
Resistors	2	5.00
Motor controller	1	500.00
Power Supply Module	1	350.00
Level Logic Converter	1	120.00
PCB Design	1	450.00
Enclosure	2	2080.00
DC Step Motor	1	2500.00
Total Cost		**9865.00**

5) Discussion

We had to face a lot of challenges in designing and implementing this Automatic Retractable Clothesline System. ATMega328p also were not in the market then we found it from Arduino board. There was not any way to check if our circuit worked properly or not. So, we entered two LEDs into our product. We got some help from a mechanist to build our prototype. We must bring some cables from one side of the prototype clothesline to the other side via wiring cables.

6) Acknowledgment

We would like to thank each person who has helped us, even in a very small manner, to achieve good results in this project. We would like to especially thank Dr. Ajith Pasqual. He motivated us to learn the extra subjects required for this project ourselves. He also helped us in clearing up our doubts and ambiguities regarding the project.

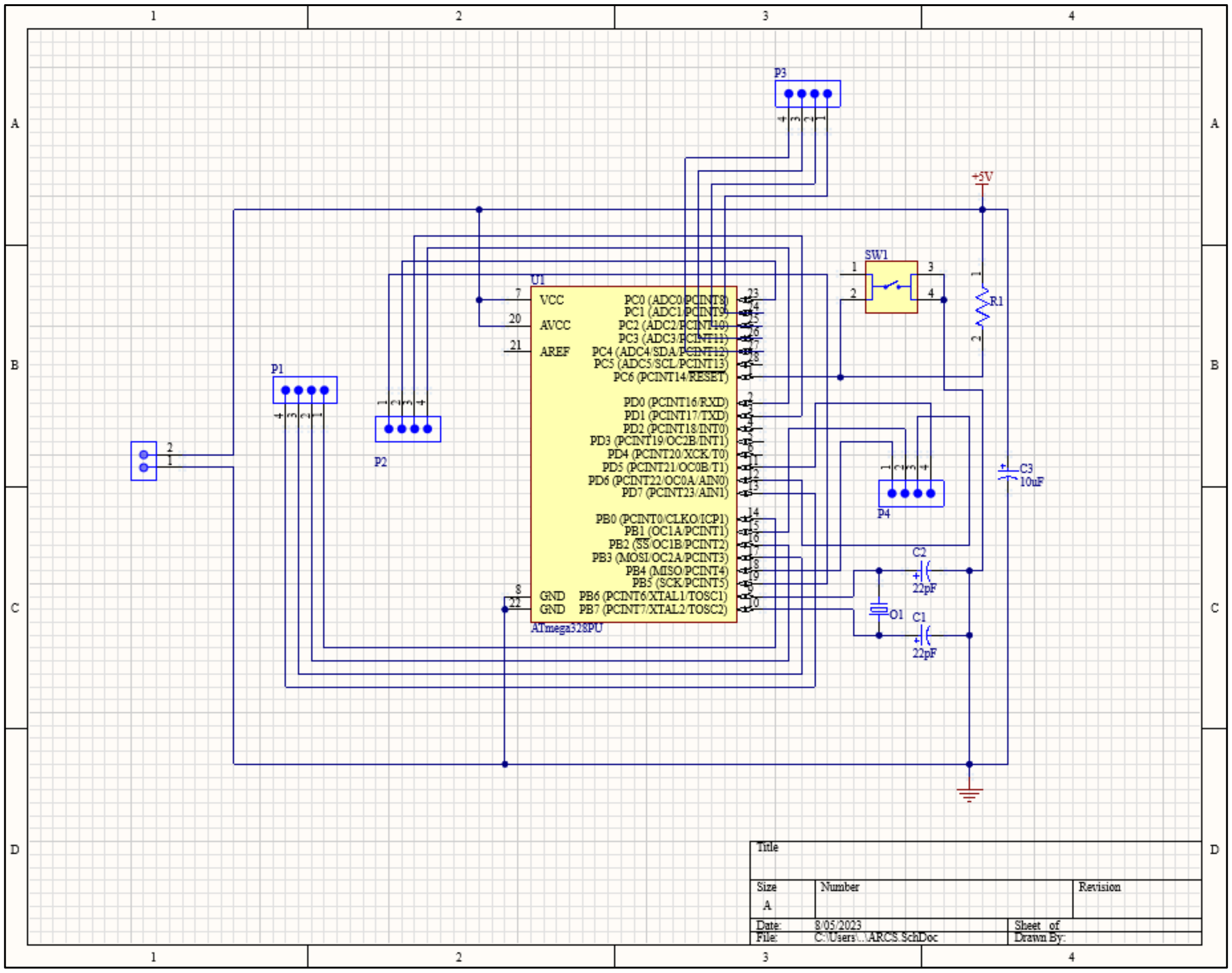
7) Reference

https://www.electronicsdatasheets.com/

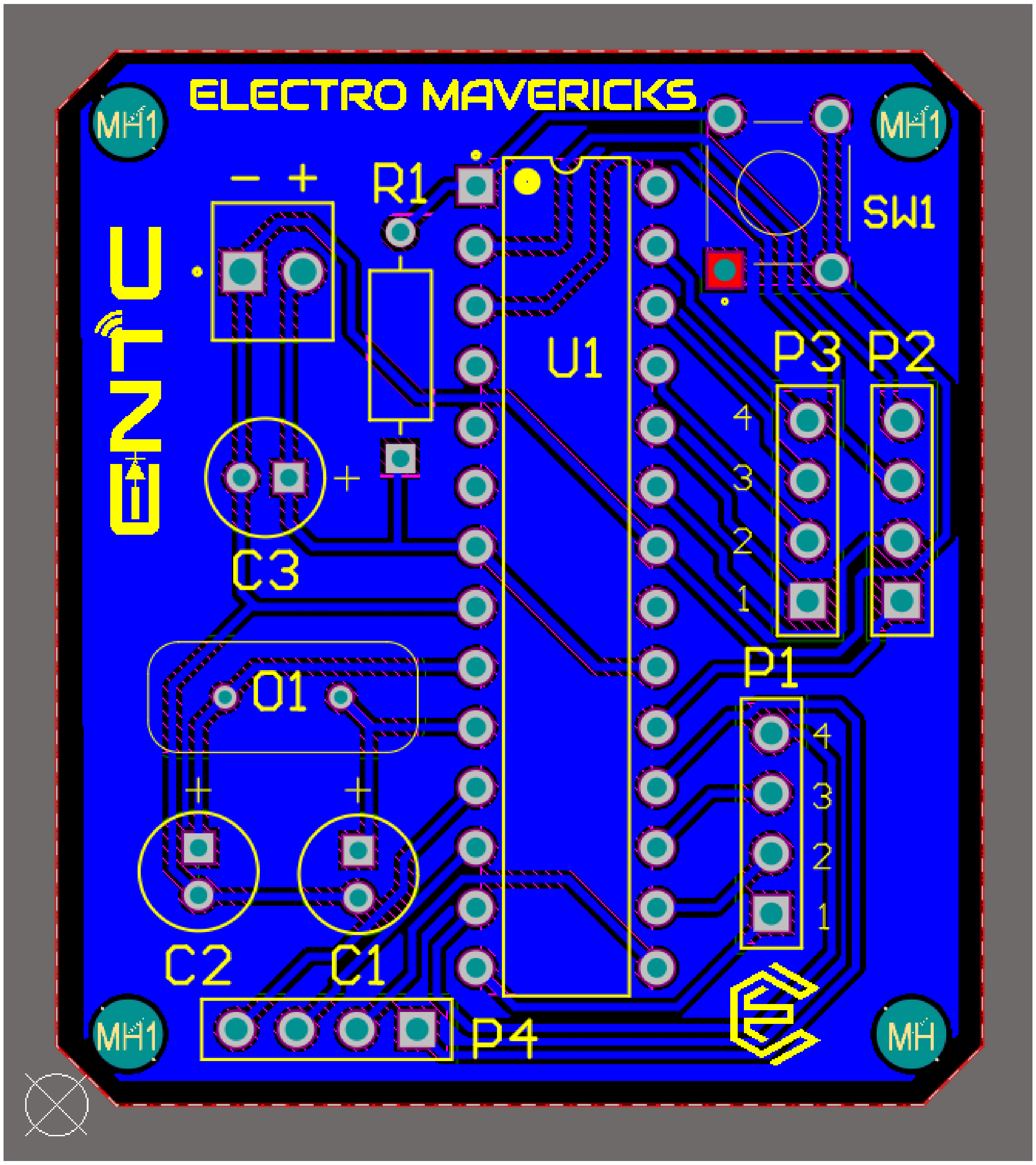
https://www.alldatasheet.com/

8) Appendices

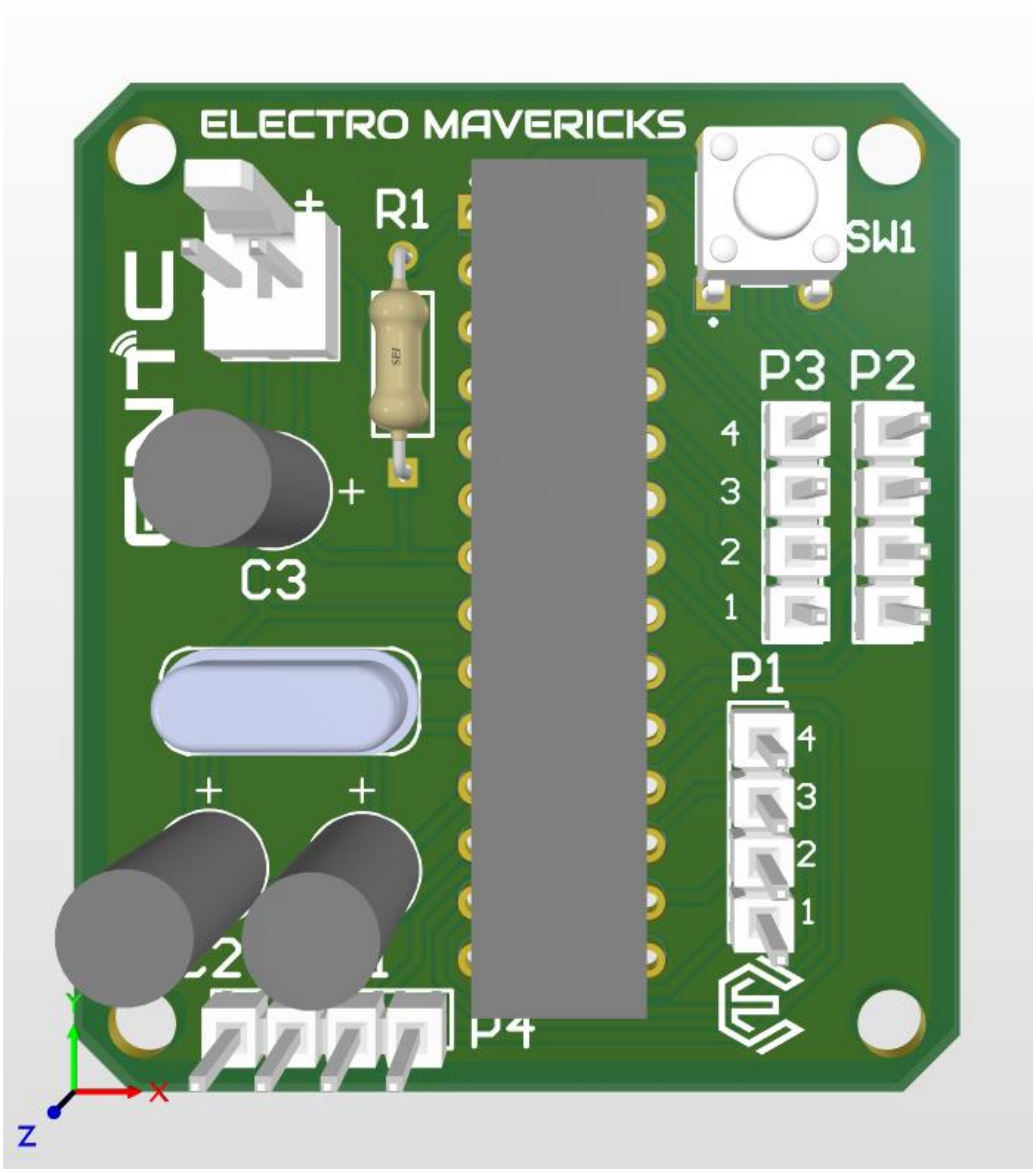
Appendix 1 – PCB schematic



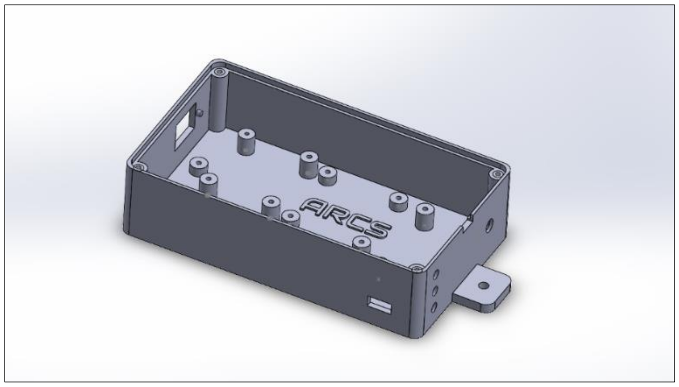
Appendix 2 – PCB Layout



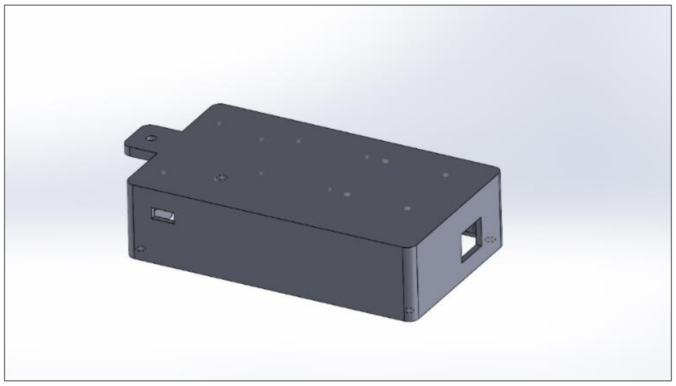
Appendix 3 – PCB 3D view



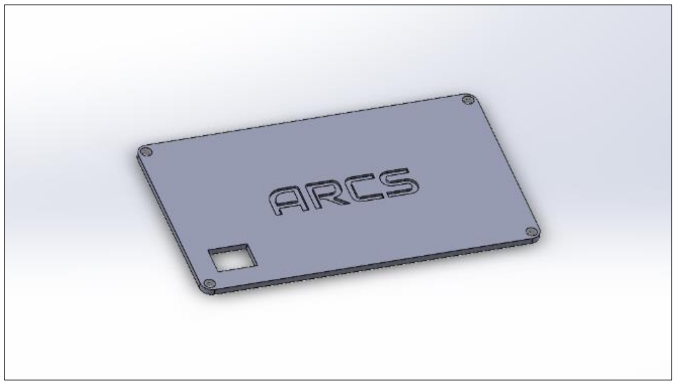
Appendix 4 – Separated parts of Enclosure 1



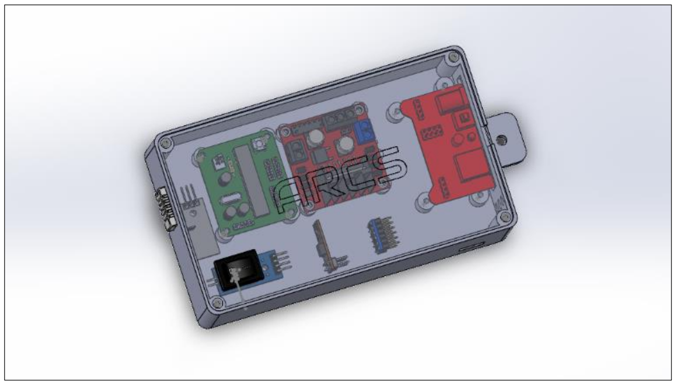
Box-Bottom View



Box-Bottom View

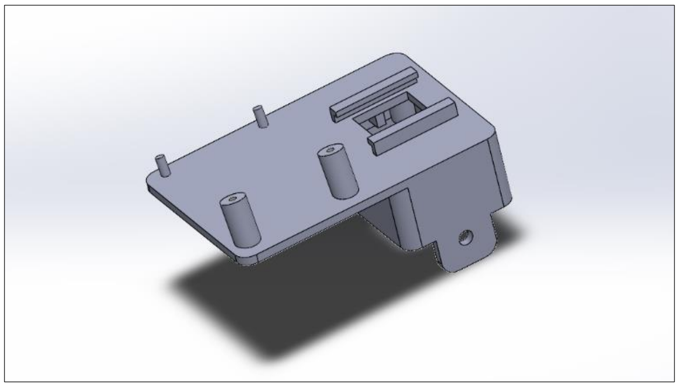


Lid

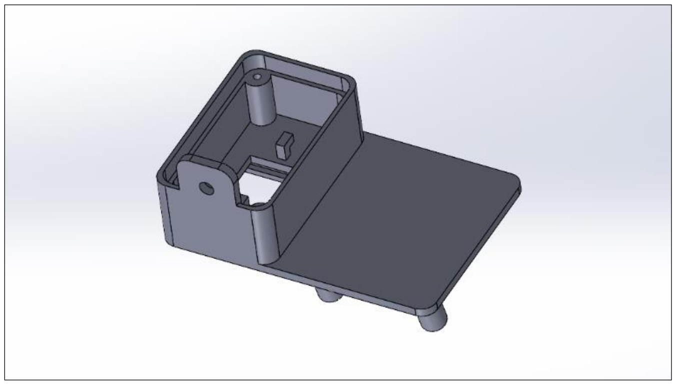


Assembly View

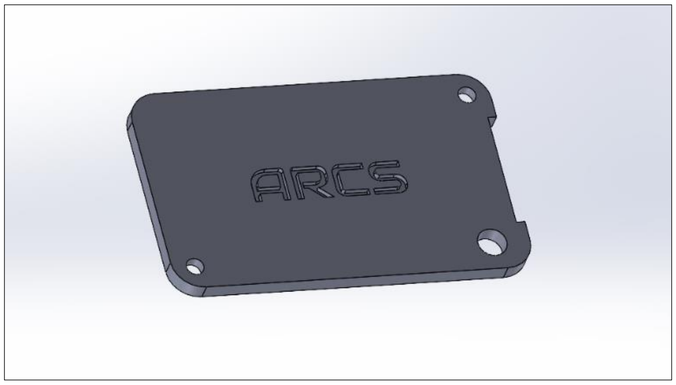
Appendix 5 – Separated parts of Enclosure 2



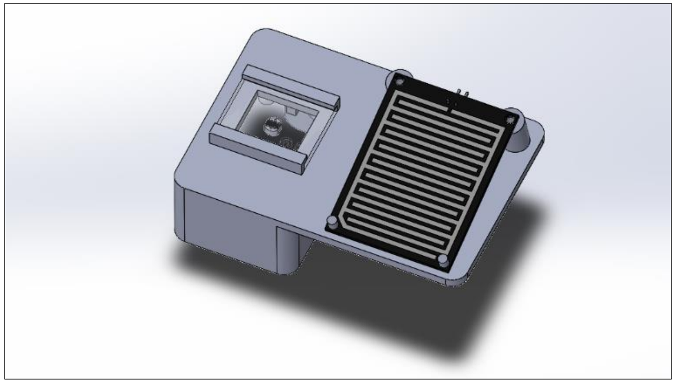
Box-Bottom View



Box-Bottom View



Lid



Assembly View

APPENDIX 6 – CODE FOR ATMEGA328P

```
// Include Libraries                         #include <Arduino.h> // Include the
Arduino core library
#include <DHT.h> // Include the DHT library for the humidity and temperature sensor
#include <Stepper.h> // Include the Stepper library
#include <EEPROM.h>

#include <SoftwareSerial.h>
SoftwareSerial mySerial(0, 1);  // RX, TX

// Pin Definitions
#define DHT_PIN_DATA  7 // Define the pin for DHT data

#define inside_ledPin A2// Define the LED pin
#define auto_ledPin A3
#define outside_ledPin A4
#define sensorPin A0 // Define the pin for the rain sensor
#define ldr_pin A1

//Step Motor
int motorSpeed = 10; // Set the motor speed
Stepper myStepper(2048, 12, 9, 6, 5); // Initialize the Stepper object with the
specified parameters


bool isClothsOutside; // Flag to track if the motor has rotated
//bool isAuto;



//DHT Sensor
DHT dht(DHT_PIN_DATA); // Create a DHT object


//////////////Thingspeak Read and Write //////////////////////////////////

// Pin Definitions
#define WIFI_PIN_TX 8
#define WIFI_PIN_RX 13

#define LED_PIN_off 10
#define LED_PIN_on 11

SoftwareSerial espSerial(WIFI_PIN_RX, WIFI_PIN_TX);
```

```
const char* SSID = "admin";// Enter your Wi-Fi name
const char* PASSWORD = "admin"; // Enter your Wi-Fi password

String HOST = "api.thingspeak.com";
String PORT = "80";
String API = "ZQRO45V38G5PA209"; // Write API KEY

String motorStatus = "/channels/2225714/fields/1/last.txt";
//String isClothOutsideField = "/channels/2225714/fields/6/last.txt";
//String isAutoField = "/channels/2225714/fields/7/last.txt";


// Sample Values for Sensor Data
float Temperature;
float Humidity;
float rainValue;
float lightIntensity;

int countTrueCommand;
int countTimeCommand;
boolean found = false;

void setup() {
  Serial.begin(9600); // Initialize the serial communication
  mySerial.begin(9600); // SoftwareSerial communication

  myStepper.setSpeed(motorSpeed); // Set the motor speed

  while (!Serial); // Wait for the serial port to connect (needed for native USB)
  Serial.println("start"); // Print "start" message
  mySerial.println("Start");


  pinMode(inside_ledPin, OUTPUT); // Set the LED pin as an output
  pinMode(auto_ledPin, OUTPUT);
  pinMode(outside_ledPin, OUTPUT);

  pinMode(DHT_PIN_DATA, INPUT);


  digitalWrite(inside_ledPin, LOW); // Turn off the LED
  digitalWrite(auto_ledPin, LOW);
  digitalWrite(outside_ledPin, LOW);

  dht.begin(); // Initialize the DHT sensor
  ///////////////// Thingspeak //////////////////////////
  // Initialize ESP8266 serial communication
  espSerial.begin(9600);

  connectToWiFi();

  //Define testing LED pin output
  pinMode(LED_PIN_off, OUTPUT);
  pinMode(LED_PIN_on, OUTPUT);

  //isClothsOutside = readThingSpeak(isClothOutsideField);

  //delay(30);

  //isAuto = readThingSpeak(isAutoField);

  //delay(30);
  isClothsOutside = EEPROM.read(0);
  //isAuto = EEPROM.read(1);

}

void loop() {

  //Serial.print("isClothsOutside = ");
  //Serial.println(isClothsOutside);
  //mySerial.print("isClothsOutside = ");
  //mySerial.println(isClothsOutside);
  //Serial.print("isAuto = ");
  //Serial.println(isAuto);


  // LDR
  float lightIntensity = analogRead(ldr_pin); // Read the ldr_value intensity from
analog pin A1

  delay(30);

  //DHT
  float Humidity = dht.readHumidity(); // Read the humidity value from the DHT
sensor
  float Temperature = dht.readTempC(); // Read the temperature value from the DHT
sensor in Celsius
```

```
  delay(30);

  //Rain Sensor
  float rainValue = readRainSensor(); // Read the rain sensor value

  delay(30);

  printToSerial(float(lightIntensity), float(Humidity), float(Temperature),
float(rainValue));

  ////////////////////////////////////// From Here Onwards thingspeak ////////////

  float checkFieldStatus = readThingSpeak(motorStatus);
  Serial.println(checkFieldStatus);
  mySerial.println(checkFieldStatus);


  if (checkFieldStatus) {
    manualModeon();
  }


  else  {
    autoMode(lightIntensity, Humidity, rainValue);
  }
  EEPROM.write(0, isClothsOutside);
  //EEPROM.write(1, isAuto);

  writeThingSpeak(Temperature, Humidity, rainValue, lightIntensity);

}

void manualModeon() {
  Serial.println(F("Manual mode on"));
  mySerial.println("Manual mode on");
  digitalWrite(LED_PIN_on, HIGH);
  digitalWrite(LED_PIN_off, LOW);
  digitalWrite(auto_ledPin, LOW);
  turnOnInsideClothes();

}


void autoMode(int lightIntensity, int Humidity, int rainValue) {
  digitalWrite(LED_PIN_on, LOW);
  digitalWrite(LED_PIN_off, HIGH);
  Serial.println(F("Auto mode on"));
  mySerial.println("Auto mode on");
  digitalWrite(auto_ledPin, HIGH);

  if (lightIntensity < 20 || Humidity > 100 || rainValue > 150)  {
    turnOnInsideClothes();
  } else  {
    turnOnOutsideClothes();
  }
}

void turnOnInsideClothes() {

  if (isClothsOutside) {
    myStepper.step(1024);
  }
  digitalWrite(inside_ledPin, HIGH);
  Serial.println(F("Now inside clothes "));
  mySerial.println("Now inside clothes ");
  digitalWrite(outside_ledPin, LOW);
  isClothsOutside = false;
  delay(5000);
}

void turnOnOutsideClothes() {

  if (!isClothsOutside) {
    myStepper.step(-1024);
  }
  digitalWrite(inside_ledPin, LOW);
  Serial.println(F("Now outside clothes "));
  mySerial.println("Now outside clothes ");
  isClothsOutside = true;
  digitalWrite(outside_ledPin, HIGH);
  delay(1000);

}

// This function returns the analog data of Rain Sensor
int readRainSensor() {
  int sensorValue = analogRead(sensorPin);  // Read the analog value from the rain
sensor
```

```
  int outputValue = map(sensorValue, 0, 1023, 255, 0); // Map the 10-bit data to 8-
bit data
  //analogWrite(ledPin, outputValue); // Generate PWM signal
  return outputValue; // Return the analog rain value
}


void printToSerial(float(lightIntensity_), float(Humidity_), float(Temperature_),
float(rainValue_)) {

  Serial.print(F("LDR Value ")); // Print the label
  Serial.println(lightIntensity_); // Print the ldr_value intensity value

  mySerial.print("LDR Value ");
  mySerial.println(lightIntensity_);

  Serial.print(F("Humidity: ")); // Print the label
  Serial.print(Humidity_); // Print the humidity value
  Serial.println(F(" [%]\t")); // Print the unit for humidity

  Serial.print("Temperature: "); // Print the label
  Serial.print(Temperature_); // Print the temperature value
  Serial.println(" C"); // Print the unit for temperature


  Serial.print(F("RainValue ")); // Print the label
  Serial.println(rainValue_); // Print the rain sensor value

  mySerial.print("RainValue ");
  mySerial.println(rainValue_);


}



//////////////////////////// codes for read and write //////


void connectToWiFi() {
  // Reset ESP8266 module
  sendCommand("AT+RST", 5000, "ready");

  // Set ESP8266 to client mode
  sendCommand("AT+CWMODE=1", 2000, "OK");
  // Connect to Wi-Fi network
  Serial.println("Connecting to Wi-Fi...");
  mySerial.println("Connecting to Wi-Fi...");
  sendCommand("AT+CWJAP=\"" + String(SSID) + "\",\"" + String(PASSWORD) + "\"",
10000, "OK");
}


bool isNumber(const String &str) {
  for (unsigned int i = 0; i < str.length(); i++) {
    if (!isdigit(str.charAt(i)) && str.charAt(i) != '.' && str.charAt(i) != '-') {
      return false;
    }
  }
  return true;
}


float readThingSpeak(String FIELD_PATH_) {
  // Make an HTTP GET request to Thingspeak for "FIELD_PATH_"
  String getRequest = "GET " + FIELD_PATH_;
  sendCommand("AT+CIPMUX=1", 5000, "OK");
  sendCommand("AT+CIPSTART=0,\"TCP\",\"" + HOST + "\"," + PORT, 10000, "OK");
  sendCommand("AT+CIPSEND=0," + String(getRequest.length() + 2), 5000, ">");
  espSerial.println(getRequest);
  delay(1000);

  // Read and parse the value from the response
  String response;
  while (espSerial.available())
  {
    char c = espSerial.read();
    response += c;
  }
  Serial.println(response);
  mySerial.println(response);

  // Find the start and end indices of the numeric value
  int valueStartIndex = response.indexOf(":") + 1;
  int valueEndIndex = response.indexOf("0,CLOSED");
  String value_ = response.substring(valueStartIndex, valueEndIndex);

  // Remove any additional characters (e.g., newlines) from the value
  value_.trim();
  //String value;
  //if (isNumber(value_)) {
  //  float value = value_.toFloat(); // Convert the string to a float
  //} else {
  // String value = value_;
  //Serial.println("Error: Invalid input. The string is not a valid number.");

  //}
  // Serial.print("Field 2 Value: ");
  // Serial.println(value);
  float value = value_.toFloat();
  sendCommand("AT+CIPCLOSE=0", 5000, "OK");
  return value;
}


float writeThingSpeak(float Temperature_, float Humidity_, float rainValue_, float
lightIntensity_) {

  String getData = "GET /update?api_key=" + API + "&field2=" + Temperature_ +
                   "&field3=" + Humidity_ + "&field4=" + rainValue_ + "&field5=" +
lightIntensity_;

  sendCommand("AT+CIPMUX=1", 5000, "OK");
  sendCommand("AT+CIPSTART=0,\"TCP\",\"" + HOST + "\"," + PORT, 10000, "OK");
  sendCommand("AT+CIPSEND=0," + String(getData.length() + 4), 5000, ">");

  espSerial.println(getData);
  delay(1500);
  countTrueCommand++;
  sendCommand("AT+CIPCLOSE=0", 5000, "OK");
}


void sendCommand(String command, unsigned int timeout, const char* expected)
{
  espSerial.println(command);
  unsigned long startTime = millis();

  String response;

  while (millis() - startTime < timeout)
  {
    while (espSerial.available())
    {
      char c = espSerial.read();
      response += c;
    }
    if (response.indexOf(expected) != -1)
    {
      found = true;
      break;
    }
  }
  if (found)
  {
    Serial.println("OK");
    mySerial.println("OK");
    countTrueCommand++;
  }
  else
  {
    Serial.println("Fail");
    mySerial.println("Fail");
    countTrueCommand = 0;
  }
  found = false;
}
```