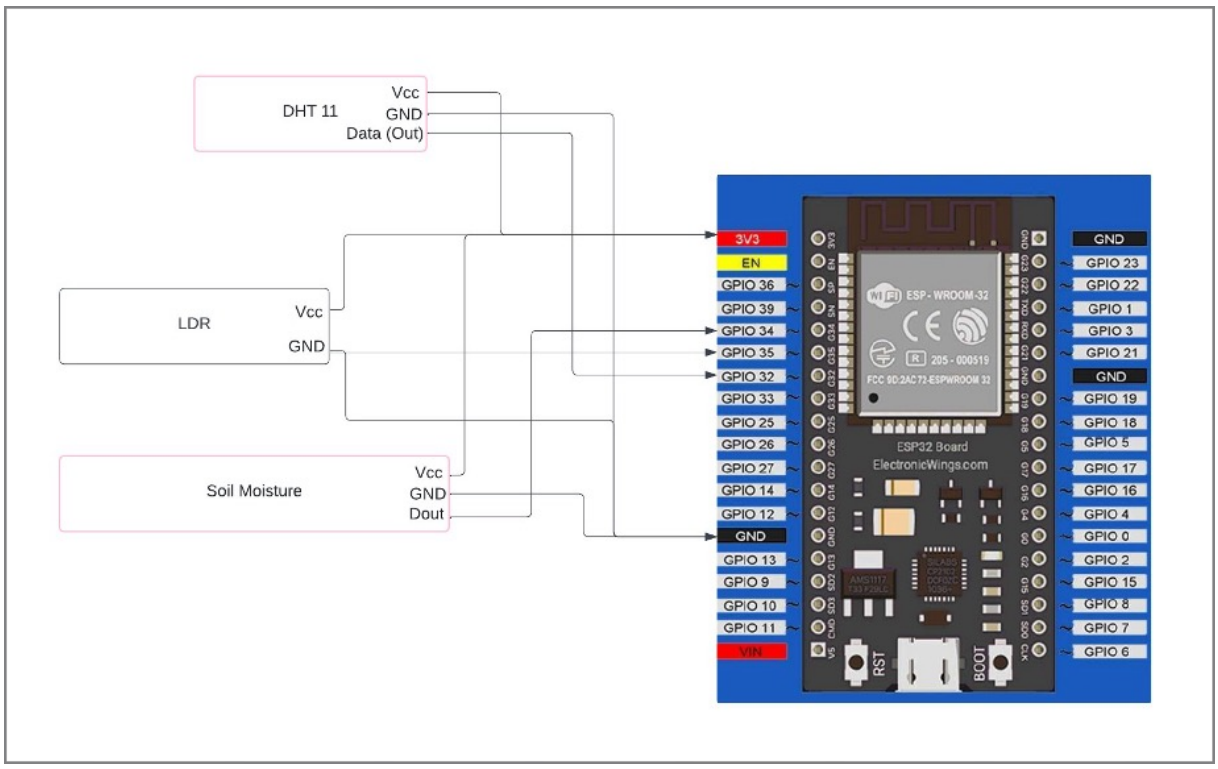
Smart Crop Management System

Aim:- The aim of this experiment is to create an IoT crop management system using an ESP32 microcontroller. This system will monitor various environmental parameters such as temperature, humidity, light intensity, and soil moisture, and send email notifications when any of these parameters exceed predefined thresholds.(Hardware+Software)

Components :- ESP32 microcontroller, DHT11 temperature and humidity sensor, Soil moisture sensor, LDR (Light Dependent Resistor), Breadboard and jumper wires

Circuit Diagram :-



Source Code :-

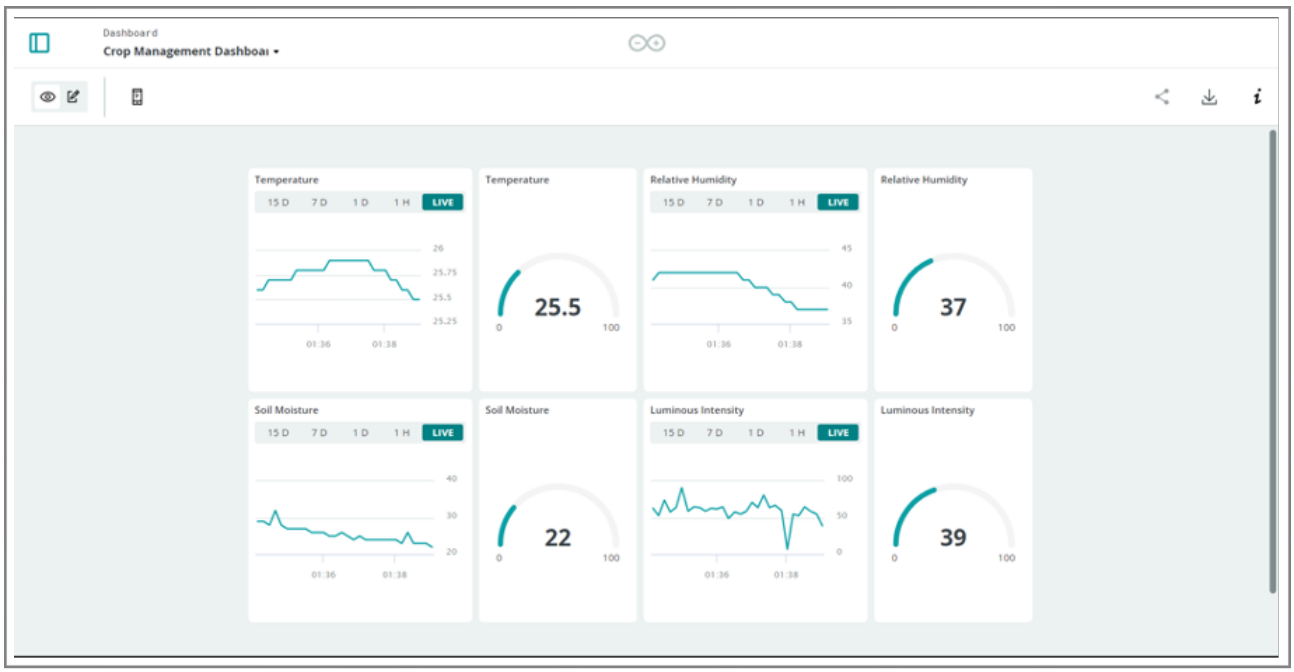
```
#include <WiFi.h>
#include <ESP_Mail_Client.h>
#include <Arduino.h>
#include "DHT.h"

// ----------- Pin Definitions -----------
#define DHTPIN 32            // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11        // DHT 11
#define SOIL_MOISTURE_PIN 34  // Soil moisture sensor pin
#define LDR_PIN 35           // LDR sensor pin

// ----------- WiFi Credentials -----------
#define WIFI_SSID "YOUR_WIFI_NAME"
```

```
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

// ----------- Email Settings -----------
#define SMTP_server "smtp.gmail.com"
#define SMTP_Port 587
#define sender_email "your_email@gmail.com"
#define sender_password "your_app_password"
#define Recipient_email "recipient_email@gmail.com"
#define Recipient_name "Recipient"

// ----------- Threshold Values -----------
const float TEMPERATURE_THRESHOLD = 20.0;   // °C
const float HUMIDITY_THRESHOLD = 70.0;      // %
const int LIGHT_THRESHOLD = 10;             // analog value
const int SOIL_MOISTURE_THRESHOLD = 30;     // %

SMTPSession smtp;
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.print("Connecting to WiFi...");

  // Connect to WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(200);
  }
  Serial.println("\nWiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Initialize SMTP session
  smtp.debug(1);

  // Configure email session
  ESP_Mail_Session session;
  session.server.host_name = SMTP_server;
  session.server.port = SMTP_Port;
  session.login.email = sender_email;
  session.login.password = sender_password;
  session.login.user_domain = "";

  if (!smtp.connect(&session)) {
    Serial.println("Failed to connect to SMTP server");
```

```
    return;
  }

  pinMode(SOIL_MOISTURE_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);
  dht.begin();

  // Read initial sensor data
  sendSensorData("Sensor Threshold Exceeded on Boot");
}

void loop() {
  sendSensorData("Sensor Threshold Exceeded");
  delay(60000); // Delay for 1 minute before next reading
}

void sendSensorData(String subject) {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  int lightIntensity = analogRead(LDR_PIN);
  int soilMoisture = (100 - (analogRead(SOIL_MOISTURE_PIN) / 4095.00) * 100);

  if (temperature > TEMPERATURE_THRESHOLD ||
      humidity > HUMIDITY_THRESHOLD ||
      lightIntensity > LIGHT_THRESHOLD ||
      soilMoisture < SOIL_MOISTURE_THRESHOLD) {

    SMTP_Message message;
    message.sender.name = "ESP32";
    message.sender.email = sender_email;
    message.subject = subject;

    String content = "Sensor readings exceeded threshold:\n";
    content += "Temperature: " + String(temperature) + "°C\n";
    content += "Humidity: " + String(humidity) + "%\n";
    content += "Light Intensity: " + String(lightIntensity) + "\n";
    content += "Soil Moisture: " + String(soilMoisture) + "\n";
    message.text.content = content.c_str();

    message.addRecipient(Recipient_name, Recipient_email);

    if (!MailClient.sendMail(&smtp, &message)) {
      Serial.println("Error sending Email, " + smtp.errorReason());
    } else {
      Serial.println("Email sent successfully!");
    }
  }
}
```

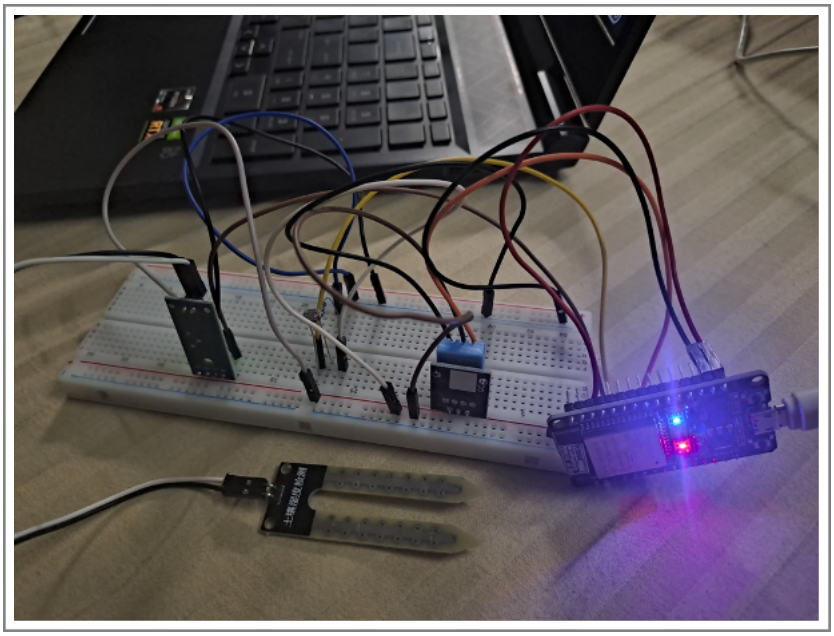
Results :-

The system successfully monitored the environmental parameters including temperature, humidity, light intensity, and soil moisture. Email notifications were sent when any of these parameters exceeded the predefined thresholds.



Snippet of Email :-

Hardware Implementation:-



Conclusion :-

The IoT crop management system demonstrated the ability to effectively monitor environmental conditions and provide timely notifications when thresholds were exceeded. This project successfully demonstrated the monitoring of environmental parameters crucial for crop management. The ESP32 microcontroller effectively interfaced with sensors to collect data of temperature, humidity, light intensity, and soil moisture. Email notifications were successfully sent via SMTP when sensor readings deviated from predefined optimal values, contributing to early detection of environmental stressors affecting crops. Further refinements could be made to enhance the system's robustness and accuracy in real-world agricultural settings.