DMP Project
Plant Monitoring System

Morari Camelia
Sîrca Florentina

January 2024

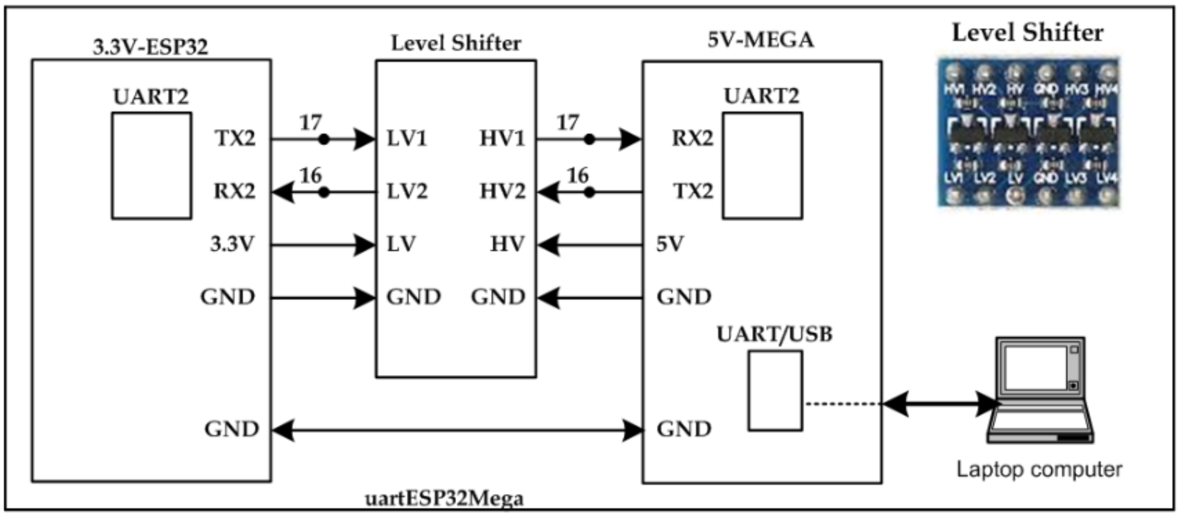
Contents

1 Introduction

The project is a Plant Monitoring System, an application that takes data from multiple sensors connected to a plant, and displaying them into a HTML page, with conclusions that tell you what your plant needs!

2 Analysis

For this project, the theme was a Plant Monitoring System, we used an Arduino Mega 2560 and ESP32, to see the results regarding some parameters about the plant. This system employs a network of sensors, including humidity, temperature, soil moisture, and water level sensors. All the sensors are connected to the Arduino mega board, which takes into consideration the results and sends them with the help of a level shifter through the UART Communication Interface to the ESP32. We used this sceheme to make the required connections:



The humidity and temperature sensors contribute to understanding the plant's microclimate, while the soil moisture sensor ensures precise irrigation management. The water level sensor check the level of water in the plant's bowl. The user is told when it's time to change something in its plant habitat by seeing the changes in the HTML page regarding each sensor component result.

3 Components

3.1 Hardware Components

- Arduino Mega 2560
- ESP32
- DHT11 Sensor
- Water Sensor
- Soil Moisture Sensor
- Breadboard
- Level Shifter
- Wires

3.2 Software Components

- Arduino IDE
- HTML page with IP: 192.168.4.1

4 Implementation

4.1 Arduino Code Implementation

This code is written for an Arduino-like platform and involves interfacing with various sensors to monitor environmental parameters such as humidity, temperature, soil moisture, and water level.

- Includes and Definition
  - #include ”DHT.h” and #include Adafruit_Sensor.h: These lines include libraries for the DHT (Digital Humidity and Temperature) sensor.
  - #include Wire.h and #include BH1750.h: These lines include libraries for using the I2C communication protocol (Wire.h) and the BH1750 light sensor (BH1750.h).
  - #define: Various constants are defined here, such as AOUT_PIN for soil moisture sensor analog output, THRESHOLD, DHTPIN and DHTTYPE for the DHT sensor, and pins for power and signal of the water level sensor.
- Global Object Declarations
  - DHT dht(DHTPIN, DHTTYPE);: Creates a DHT object for the humidity and temperature sensor.
  - BH1750 lightMeter;: Creates an object for the BH1750 light sensor.
- Setup Function
  - setup(): This function initializes serial communication, sets pin modes, and starts the sensors. Begins serial communication at 9600 baud rates for debugging and data display. Initializes the DHT sensor. Sets the POWER_PIN as an output and initially turns it off.
- Loop Function
  loop(): This function is the main loop that repeatedly executes as long as the device is powered. Reads humidity and temperature from the DHT sensor and prints them to the serial monitor. Reads soil moisture level from an analog pin and prints it. Temporarily powers on the water level sensor, reads its value, then powers it off, and prints the value. After a delay, it transmits the sensor readings over Serial2 for communication with the ESP32 board.
- Sensor Readings and Data Transmission
  The code reads various environmental parameters and outputs them via serial communication. Additionally, it sends the data over a secondary serial interface, Serial2, for communication with the ESP32 board.

4.2 ESP32 Code Implementation

This code is for a plant monitoring system that uses an Arduino or a similar microcontroller with WiFi capability. It receives sensor data from another device (presumably over Serial2), analyzes the data to determine the health of the plant, and then serves this information on a simple web page accessible via a WiFi network. Here’s a breakdown of the various components of the code:

- Libraries and Global Variables
  - WiFi.h, WiFiAP.h, WebServer.h: Libraries for WiFi functionalities and hosting a web server.
  - SSID: The name of the WiFi network.
  - WebServer server(80): Initializes a web server that listens on port 80.
  - String sensorData, humidity, temperature, soilMoisture, waterLevel: Strings to store sensor data.
  - RX2, TX2: Pin definitions for Serial2 communication.

  - humidityOK, temperatureOK, soilMoistureOK, waterLevelOK: Variables to store the status of each measurement (OK or not OK).

- Setup Function
  - Initializes Serial communication for debugging and Serial2 for receiving data.
  - Sets up a WiFi Access Point (AP) with the specified SSID.
  - Prints the IP address for the HTTP server.
  - Sets up a web server to handle HTTP GET requests on the root path (/), responding with HTML generated by generateHTML().

- Loop Function
  Continuously checks for incoming data on Serial2. Parses the data into humidity, temperature, soil moisture, and water level readings. Calls plantState() to analyze the readings. Handles incoming HTTP requests by responding with the current sensor data and plant state.

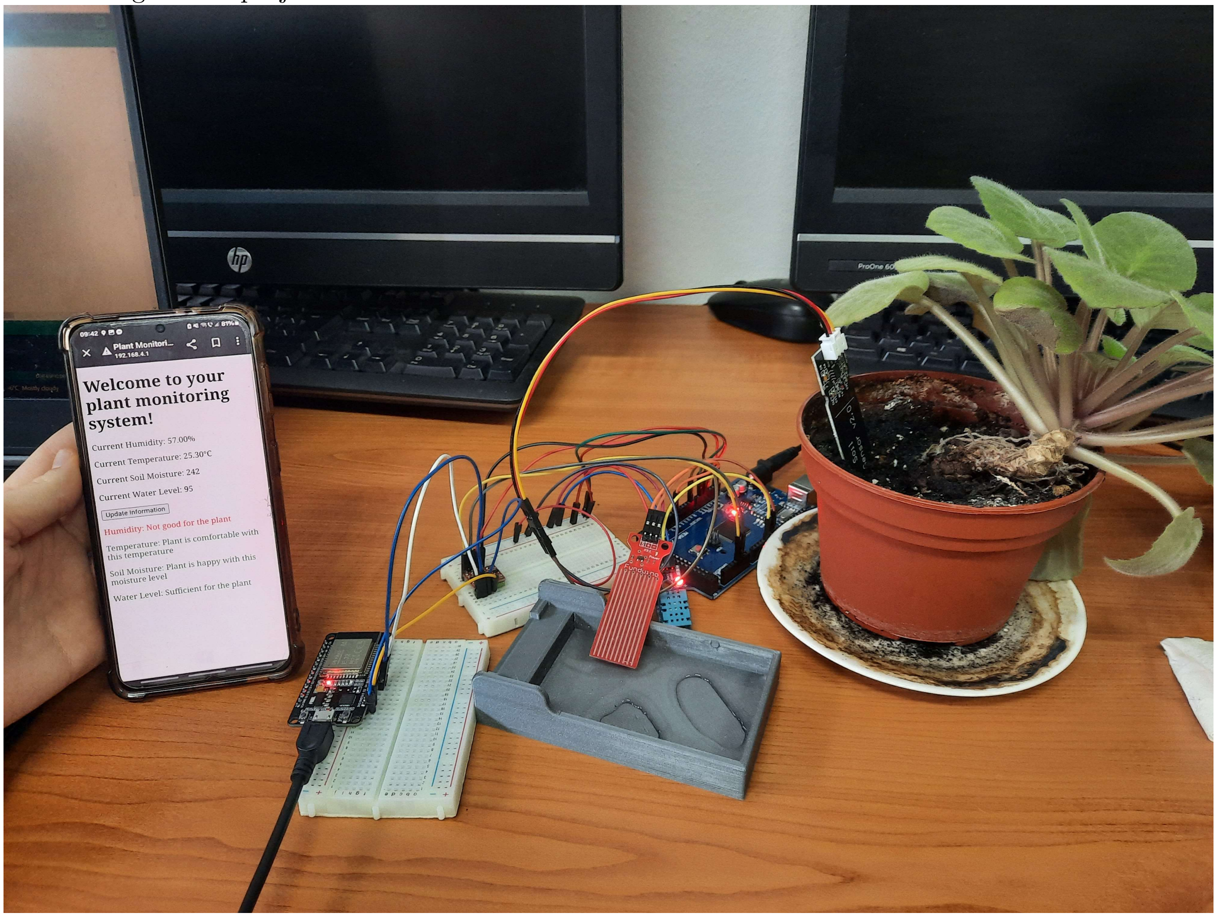
- plantState Function
  Converts sensor data strings to appropriate data types for comparison. Checks each sensor reading against predetermined healthy ranges and updates the corresponding status variables.

- generateHTML Function
  Generates an HTML string that displays the plant's current status. Updates the status with green text if conditions are good and red text if conditions are poor. Provides a button to refresh the page and get updated information.

- Key Features
  The system creates a WiFi access point and serves a simple web page to display sensor data and plant health status. It receives sensor data from another device via serial communication.

5 Results

Final wiring for the project:



Welcome to your plant monitoring system!

Current Humidity: 57.00%

Current Temperature: 25.30°C

Current Soil Moisture: 242

Current Water Level: 95

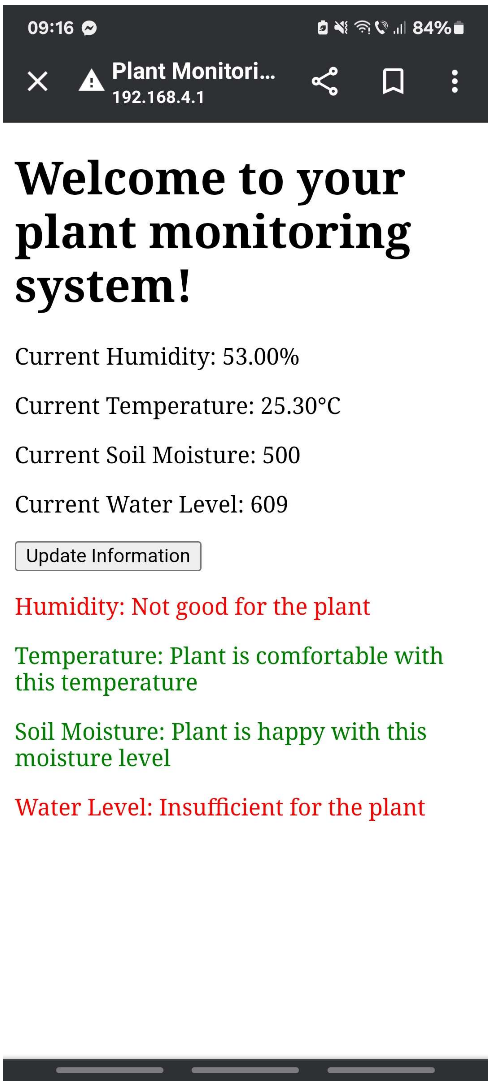
Humidity: Not good for the plant

Temperature: Plant is comfortable with this temperature

Soil Moisture: Plant is happy with this moisture level

Water Level: Sufficient for the plant

Final HTML page:


09:16 84%

Plant Monitori...
192.168.4.1

Welcome to your plant monitoring system!

Current Humidity: 53.00%

Current Temperature: 25.30°C

Current Soil Moisture: 500

Current Water Level: 609

Humidity: Not good for the plant

Temperature: Plant is comfortable with this temperature

Soil Moisture: Plant is happy with this moisture level

Water Level: Insufficient for the plant

6 Code

6.1 Arduino Code

```
#include "DHT.h"
#include <Adafruit_Sensor.h>

#include <Wire.h>
#include <BH1750.h>

#define AOUT_PIN A0
#define THRESHOLD 2000

#define DHTPIN  4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

#define POWER_PIN  7
#define SIGNAL_PIN A5
int valueWater = 0;
```

```
BH1750 lightMeter;

void setup() {
  Serial.begin(9600);
  Serial2.begin(9600);

  pinMode(POWER_PIN, OUTPUT);
  digitalWrite(POWER_PIN, LOW);
  Wire.begin(); // For BH1750
  dht.begin();
}

void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println("%");

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println("°C");

  int soilMoisture = analogRead(AOUT_PIN);
  Serial.print("Soil Moisture: ");
  Serial.println(soilMoisture);

  digitalWrite(POWER_PIN, HIGH);
  delay(10);
  valueWater = analogRead(SIGNAL_PIN);
  digitalWrite(POWER_PIN, LOW);
  Serial.print("Water Level: ");
  Serial.println(valueWater);

  delay(2000);

  Serial2.println("Transmit: ");

  Serial2.println(humidity);
  Serial2.println(temperature);
  Serial2.println(soilMoisture);
  Serial2.println(valueWater);
}
```

6.2 ESP32 Code

```
#include <WiFi.h>
#include <WiFiAP.h>
#include <WebServer.h>

const char *SSID = "camiTina";
WebServer server(80);

String sensorData = "";
String humidity = "N/A", temperature = "N/A", soilMoisture = "N/A", waterLevel = "N/A";

#define RX2 16
#define TX2 17
```

```
int humidityOK = 0;
int temperatureOK = 0;
int soilMoistureOK = 0;
int waterLevelOK = 0;

void setup() {
  Serial.begin(9600);
  Serial2.begin(9600, SERIAL_8N1, RX2, TX2);

  if (!WiFi.softAP(SSID)) {
    Serial.println("Unable to start SoftAP mode");
    while (1);
  }

  IPAddress IP = WiFi.softAPIP();
  Serial.print("HTTP server started --> IP addr: ");
  Serial.println(IP);

  server.on("/", HTTP_GET, []() {
    server.send(200, "text/html", generateHTML());
  });

  server.begin();
}

void loop() {
  static int readState = 0;

  if (Serial2.available()) {
    sensorData = Serial2.readStringUntil('\n');
    sensorData.trim();

    if (sensorData == "Transmit:") {
      readState = 1;
    } else {
      switch (readState) {
        case 1:
          humidity = sensorData;
          readState++;
          break;
        case 2:
          temperature = sensorData;
          readState++;
          break;
        case 3:
          soilMoisture = sensorData;
          readState++;
          break;
        case 4:
          waterLevel = sensorData;
          readState = 0;
          plantState();
          break;
      }
    }
  }
  server.handleClient();
```

```
}

void plantState()
{
  float floatHumidity = humidity.toFloat();
  float floatTemperature = temperature.toFloat();
  int intSoil = soilMoisture.toInt();
  int intWater = waterLevel.toInt();
  Serial.println(floatHumidity);
  Serial.println(floatTemperature);
  Serial.println(intSoil);
  Serial.println(intWater);

  if (floatHumidity  >= 60.00 && floatHumidity  <= 70.00)
  {
    humidityOK = 1;
  }
  else
  {
    humidityOK = 0;
  }

  if (floatTemperature >= 20.00 && floatTemperature <= 30.00)
  {
    temperatureOK = 1;
  }
  else
  {
    temperatureOK = 0;
  }

  if (intSoil >= 200 && intSoil <= 300)
  {
    soilMoistureOK = 1;
  }
  else
  {
    soilMoistureOK = 0;
  }

  if (intWater >= 0 && intWater <= 500)
  {
    waterLevelOK = 1;
  }
  else
  {
    waterLevelOK = 0;
  }

}

String generateHTML() {
  String html = "<!DOCTYPE html><html lang=en><head><meta
  charset=UTF-8><meta name=viewport content=width=device-width, initial-scale=1.0><title>Plant
  Monitoring System</title></head><body>";
  html += "<h1>Welcome to your plant monitoring system!</h1>";
  html += "<p>Current Humidity: " + humidity + "%</p>";
  html += "<p>Current Temperature: " + temperature + "°C</p>";
  html += "<p>Current Soil Moisture: " + soilMoisture + "</p>";
  html += "<p>Current Water Level: " + waterLevel + "</p>";

  html += "<button onclick=\"location.reload();\">Update Information</button>";

  // Humidity
  html += "<p>";
  if (humidityOK == 1) {
    html += "<span style='color: green;'>Humidity: Plant is happy with this humidity</span>";
  } else {
    html += "<span style='color: red;'>Humidity: Not good for the plant</span>";
  }
  html += "</p>";

  // Temperature
  html += "<p>";
  if (temperatureOK == 1) {
    html += "<span style='color: green;'>Temperature: Plant is comfortable with this temperature
    </span>";
  } else {
    html += "<span style='color: red;'>Temperature: Not ideal for the plant</span>";
  }
  html += "</p>";

  // Soil Moisture
  html += "<p>";
  if (soilMoistureOK == 1) {
    html += "<span style='color: green;'>Soil Moisture: Plant is happy with this moisture level
    </span>";
  } else {
    int intSoil = soilMoisture.toInt();
    if (intSoil > 300)
    {
      html += "<span style='color: red;'>Soil Moisture: The soil is too dry! The plant needs some
      water. </span>";
    }
    else if (intSoil < 200)
    {
      html += "<span style='color: red;'>Soil Moisture: The soil is too moist, you put too much
      water. </span>";
    }
  }
  html += "</p>";

  // Water Level
  html += "<p>";
  if (waterLevelOK == 1) {
    html += "<span style='color: green;'>Water Level: Sufficient for the plant</span>";
  } else {
    html += "<span style='color: red;'>Water Level: Too much water in the plant bowl</span>";
  }
  html += "</p>";

  html += "</body></html>";
  return html;
}
```