|  |
| --- |
| **GrowSensse: Automated Irrigation and Climate Monitoring** |
| **Team: Krish Lakhan, MD Muhtashim Jahin**  Date: 30 Nov 2024 |

## 

## **1. Project Overview**

### **Objective:**

The goal of the **GrowSensse** project is to design and implement an advanced, fully automated greenhouse system by leveraging IoT technologies to ensure optimal plant growth through precise monitoring and control of environmental factors. This system integrates multiple sensors including **soil moisture**, **temperature**, **humidity**, **and ultrasonic sensors** to dynamically manage irrigation, climate control, and access. The system is designed for remote monitoring and management through a web interface integrating the **MQTT** protocol powered by **Node-RED**, allowing users to efficiently monitor critical parameters in real-time. By utilizing automated **relay-controlled pumps**, **exhaust fans**, and **servo** door mechanisms, GrowSensse offers an innovative solution for sustainable and scalable agricultural practices

### **Features:**

* **Automated plant watering system** based on soil moisture data.
* **Real-time monitoring** of temperature and humidity using the DHT22 sensor.
* **Node-RED dashboard** for visualizing sensor data and controlling the watering system.
* **Remote monitoring** and control capabilities, allowing users to manage their plants from anywhere.

## **2. Rubric Alignment**

### **Project Goals & Rubric Alignment:**

This project successfully aligns with the grading rubric by fulfilling key project goals:

* **Project Design & Creativity:** The use of **Node-RED** in this project gives an exciting and creative way of monitoring the grow climate using a web interface
* **Use of MCU and Sensors:** The project integrates the **DHT22** and **Sonic** sensors introduced in class as well as a **Soil moisture** sensor that needed to be researched and configured independently
* **Code Explanation & Ownership:** The code is self-developed, with modifications made to the templates from class and other templates researched online. Operational logic and reading data were developed independently. Mqtt publishing, servo operation, and wi-fi were integrated with minimal modifications to templates.
* **Functionality & Testing:** There were many hours of testing and troubleshooting throughout the development process. Multiple environmental conditions were simulated to test the response and functionality of the sensors and relays.
* **Report Submission & Quality:** This report clearly explains the system's design, functionality, and testing process, meeting all reporting requirements.
* **Presentation & Communication:** The use of visual aids, including dashboard screenshots and system diagrams, helps in clearly communicating the project.
* **Teamwork & Contribution:** Both team members contributed to all areas of the project including brainstorming ideas, designing and building the demonstration, coding, research, reporting, development and debugging.

## **3. Design & Development**

### **3.1 Team Member Contributions**

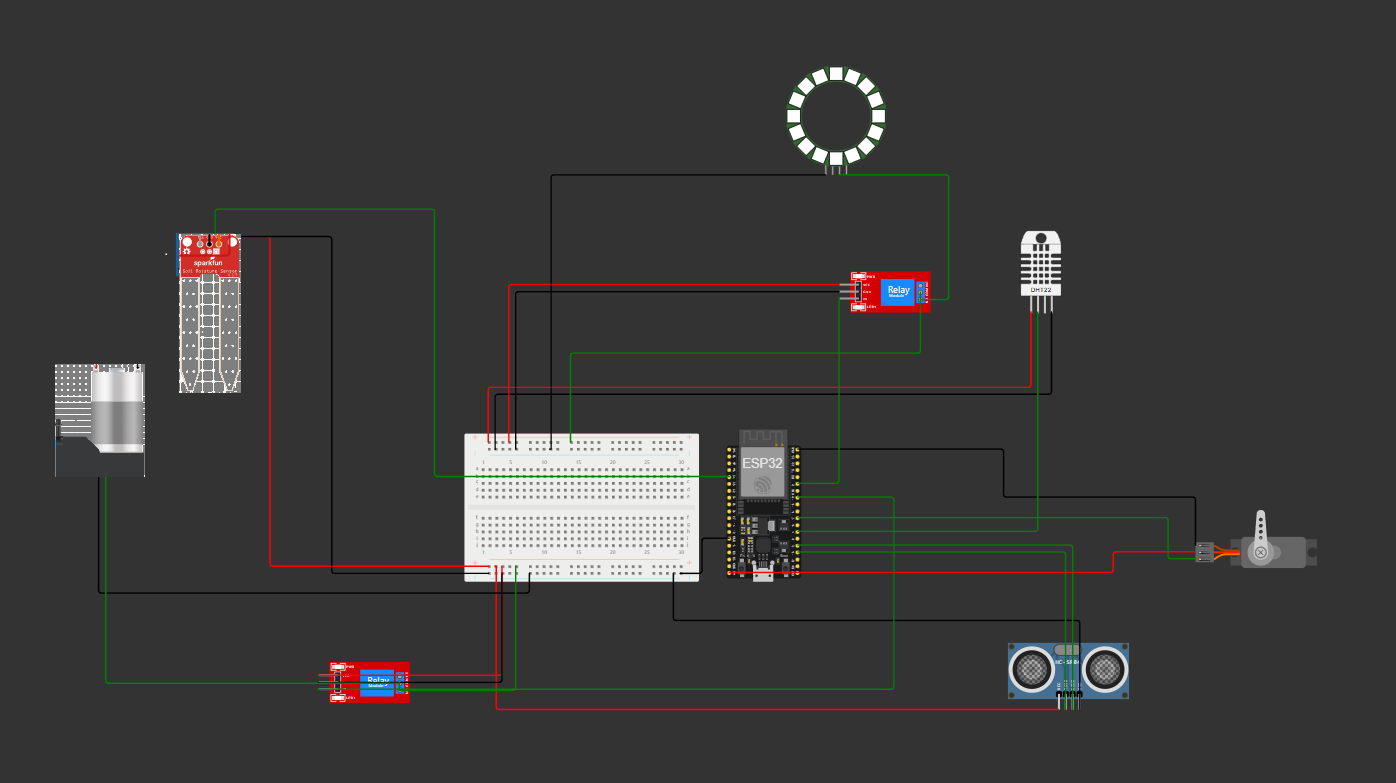
* **[Krish Lakhan]**: Focused on coding the control logic and sensor data handling for the design and development phase.
* **[MD Muhtashim Jahin]**: Focused on integrating MQTT with Node-RED and developing the web interface to display the data for the design and wiring up the circuit.
* **[Team Effort]**: Debugging, troubleshooting, research and the demo were prepared together.

### **3.2 Circuit Design and Setup**

#### **Component List:**

* **DHT22 sensor**
* **Soil moisture Sensor**
* **ESP32 microcontroller**
* **Relay module**
* **Water pump 5V**
* **Mechanical motor 5V**
* **Servo motor 3.3V**
* **Power supply 5V**
* **Jumper wires**
* **Breadboard**
* **Wifi hotspot**
* **Node-RED server**

#### **Schematic Diagram:**



#### **Circuit Description:**

* The **DHT22 sensor** is connected to the microcontroller to measure environmental temperature and humidity.
* The **soil moisture sensor** is also connected to the microcontroller to detect the moisture level in the soil.
* The **sonic sensor** is used to detect movement near the entrance of the greenhouse.
* The **servo** opens to 160° to open the door when movement is detected within 10 cm.
* The **relay module** is used to control both the watering system based on soil moisture levels and a fan for ventilation based on the temperature.
* The **Node-RED** server and **ESP32** are both connected to a wifi LAN.
* The ESP32 sends the processed sensor data to the **Node-RED** server; displayed on the dashboard.

### **3.3 Control Code**

#### **Code Overview:**

The control code for this project is written for the **ESP32** microcontroller using the **Arduino IDE**. It reads sensor data, controls relays, and sends data to a **Node-RED** server via **MQTT** for remote monitoring. The code incorporates the sensors shown in the schematic including **DHT22 sensor** to monitor humidity and temperature and controlling the fan relay, **soil moisture sensor** for controlling the water pump relay, and **ultrasonic sensor** for triggering the servo motor.

**Program Narrative:**

1. **Wi-Fi and MQTT Setup**: The program connects to a Wi-Fi network and an MQTT broker (EMQX), allowing for communication between client and broker.
2. **Sensor Initialization**: The DHT22, soil moisture sensor, and sonic sensors are initialized, with the DHT temperature used to control the HVAC, analog soil moisture reading used to control the watering system and sonic sensor used to control servo action.
3. **Control Logic**: The system reads data from the sensors and triggers actions like activating the HVAC or the watering system via relay and controlling a servo motor based on sensor input.
4. **MQTT Publishing**: Temperature, humidity, soil moisture and relay status is published to specific MQTT topics for display on the Node-RED dashboard.

#### **Key Functionalities:**

1. **Wi-Fi & MQTT Connection**:
   * The **setup\_wifi()** function connects the ESP32 to a local Wi-Fi network.
   * The **reconnect()** function ensures that the device reconnects to the MQTT broker if the connection is lost.
2. **Sensor Data Reading**:
   * The **DHT22 sensor** measures temperature and humidity every loop iteration, which controls the HVAC. The **soil moisture sensor** reads the moisture level from the soil, which controls the watering system.
   * **Ultrasonic Sensor**: The distance is calculated by sending a pulse from the **trigPin** and receiving it on the **echoPin**. If the distance is below the set threshold (10 cm), it triggers the servo to open the door.
3. **Control Logic**:
   * **Soil Moisture Control**: If the soil moisture value is above a threshold (1600), the relay is activated to turn on the watering system. The threshold value was determined using experimentation.
   * **Temperature Control**: If the temperature exceeds 25°C, the second relay connected to the HVAC system is activated.
   * **Servo Control**: If the ultrasonic sensor detects an object within 10 cm, it triggers the servo motor to open (to 160 degrees) for a period of 10 seconds, then closes it back to 0 degrees. This effectively acts as an automated door for the greenhouse.
4. **MQTT Publishing**:
   * The system regularly publishes the temperature, humidity, soil moisture, and relay status to specific topics ("dht/temperature", "soil/moisture", "relay/soil", “relay/temperature”) for monitoring in Node-RED.

#### **Challenges & Solutions:**

* **Challenge 1: Sensor Calibration**
  + The soil moisture sensor was sensitive to environmental conditions, requiring fine-tuning for accurate readings.
  + **Solution**: We added calibration logic to filter noise in the sensor data and adjusted the code to define clear moisture thresholds for better decision-making regarding the watering system.
* **Challenge 2: Sensor Interference**
  + The use of significant current to control the watering system, HVAC and servo on a single board caused interference with sensor readings.
  + **Solution**: We added a **resistor** to filter the noise from the analog output of the sensor to the input of the microcontroller.
* **Challenge 3: MQTT Connectivity Issues**
  + There were intermittent issues with maintaining a stable MQTT connection with the use of a mobile hotspot.
  + **Solution**: The **reconnect()** function ensures that the system automatically attempts to reconnect if the connection to the MQTT broker is lost.
* **Challenge 4: Servo Control**
  + The servo motor initially faced issues with smooth operation due to long delay times and response delays in the code. The circuit design also created voltage drops that simulated a PWM signal causing irregular operation.
  + **Solution**: Shortened the delays and ensured the servo commands were timed appropriately to improve responsiveness and prevent delays in the system's operation. The power for the servo was separated from heavier loads to prevent voltage drops.
* **Challenge 5: Over Watering**
  + There are built-in delays to the code to smooth functionality and stop excessive serial printing. Initially these delays caused the pump to run excessively and over water the plants.
  + **Solution**: The control functions were adjusted to cycle the pump for 3 seconds for each sensor reading , rather than running the pump until the next iteration of the loop.

### **3.4 Dashboard or Interface Design**

#### **Dashboard Layout:**

* **GrowSensse Climate Monitor**: Displays real-time data from the DHT22 sensor.
* **Exhaust Fan:** Displays the current status of the HVAC system ON/OFF.
* **Soil Moisture Level**: Raw analog output from soil moisture sensor representing the current moisture level.
* **Water Pump Status Monitor**: Displays the current status of the watering system On/OFF.

#### **User Interface Display:**

## **4. Testing and Debugging**

### **Testing Process:**

* **Simulated Test Conditions:**
  + The sensor testing was done under different environmental conditions (e.g., dry soil, high temperature) to verify the system's responsiveness. The soil sensor was placed deeper and removed from damp soil to test the output and calibrate an acceptable moisture range for the soil.
  + The DHT22 was tested by heating the sensor and cooling the sensor with body heat and blowing on the sensor to test humidity.
  + The relay devices were tested by simulating the on condition.
* **Real-World Test:** The system has only been tested using simulated environments and not yet tested in the field.

### **Results:**

* The watering system responded correctly to soil moisture levels.
* The HVAC system turns on expectedly to temperatures above 25 degrees.
* The sonic sensor properly opens the door after detecting objects within 10 cm.
* The Node-RED dashboard displays accurate temperature, humidity, and moisture data.

## **5. Final Outcomes**

### **Project Results:**

The automated plant watering system successfully updates the UI for monitoring and waters plants based on real-time data from the soil moisture and environmental sensors. The Node-RED dashboard provides an intuitive interface for remote monitoring.

### **Screenshots/Photos:**

## **6. Conclusion and Reflections**

### **Challenges Faced:**

* Integrating different sensors and actuators to create a functional system was challenging.
* The calibration of the soil moisture sensor took more time than expected due to environmental variability and noise in the signal generated from the sensor.
* There were problems with some GPIO pins with our ESP32 module that did not function properly and required reassignment.
* There were no data sheets for many of the components used in the project and required trial and error to figure out.

### **Lessons Learned:**

* Learned about different raw sensor data types and how to make them human readable.
* Gained experience in **Node-RED** integration with embedded systems.
* Improved troubleshooting and debugging skills, especially with sensor calibration and communication issues.

### **Future Enhancements:**

* Adding additional sensors, such as a light sensor and ph sensor, to further improve plant care automation.
* Incorporating data logging with MQTT to store historical climate data and relay status for other potential analysis.
* Adding remote control on top of monitoring to control relay actions remotely.
* Adding an alert system to inform of critical changes.

## **7. Appendix**

### **Complete Code:**

**#include <WiFi.h>**

**#include <PubSubClient.h>**

**#include <DHT.h>**

**#include <ESP32Servo.h> // Include the ESP32Servo library**

**// WiFi credentials**

**const char\* ssid = "Muhtashim"; // Replace with your WiFi name**

**const char\* password = "butcher568"; // Replace with your WiFi password**

**// MQTT Broker settings**

**const char\* mqtt\_server = "broker.emqx.io"; // Your provided EMQX broker address**

**const int mqtt\_port = 1883; // MQTT port**

**const char\* mqtt\_user = ""; // Username (if needed)**

**const char\* mqtt\_password = ""; // Password (if needed)**

**// DHT setup**

**#define DHTPIN 4 // ESP32 GPIO pin connected to the DHT sensor**

**#define DHTTYPE DHT22 // DHT11 or DHT22 sensor type**

**DHT dht(DHTPIN, DHTTYPE);**

**// Define pin numbers**

**const int trigPin = 15; // Trigger pin (connected to the Ultrasonic sensor's Trigger)**

**const int echoPin = 2; // Echo pin (connected to the Ultrasonic sensor's Echo)**

**// Define moisture sensor pin numbers**

**const int soilSensorPin = 34; // Define Soil moisture pin input**

**// Define servo control pin (GPIO pin with PWM support)**

**const int servoPin = 17; // Pin for the servo motor**

**Servo myServo; // Create a Servo object**

**// Define relay control pins (GPIO pins)**

**const int soilrelayPin = 19; // Pin for the water pump relay**

**const int tempRelayPin = 21; // Pin for the fan relay**

**// Define threshold for distance measurement**

**float thresholdDistance = 10.0; // Set threshold distance (in cm)**

**// MQTT Client**

**WiFiClient espClient;**

**PubSubClient client(espClient);**

**// Function to connect to WiFi**

**void setup\_wifi() {**

**delay(10);**

**Serial.println();**

**Serial.print("Connecting to ");**

**Serial.println(ssid);**

**WiFi.begin(ssid, password);**

**while (WiFi.status() != WL\_CONNECTED) {**

**delay(500);**

**Serial.print(".");**

**}**

**Serial.println("");**

**Serial.println("WiFi connected");**

**}**

**// Function to reconnect to MQTT Broker**

**void reconnect() {**

**while (!client.connected()) {**

**Serial.print("Attempting MQTT connection...");**

**if (client.connect("ESP32Client", mqtt\_user, mqtt\_password)) {**

**Serial.println("connected");**

**} else {**

**Serial.print("failed, rc=");**

**Serial.print(client.state());**

**Serial.println(" try again in 5 seconds");**

**delay(5000);**

**}**

**}**

**}**

**void setup() {**

**// Start serial communication at 115200 baud rate**

**Serial.begin(115200);**

**setup\_wifi();**

**client.setServer(mqtt\_server, mqtt\_port);**

**dht.begin();**

**// Set up the analog input pin for soil moisture**

**pinMode(soilSensorPin, INPUT);**

**// Set the Trigger and Echo pins**

**pinMode(trigPin, OUTPUT);**

**pinMode(echoPin, INPUT);**

**// Set the relay pins as output**

**pinMode(soilrelayPin, OUTPUT);**

**pinMode(tempRelayPin, OUTPUT);**

**// Initialize both relays as OFF**

**digitalWrite(soilrelayPin, HIGH); // Ensure relay is off at the start**

**digitalWrite(tempRelayPin, HIGH); // Ensure temperature relay is off at the start**

**// Attach the servo to the servo pin**

**myServo.attach(servoPin);**

**// Ensure the servo starts at 0 degrees**

**myServo.write(0);**

**}**

**void loop() {**

**if (!client.connected()) {**

**reconnect();**

**}**

**client.loop();**

**// Read temperature and humidity data**

**float h = dht.readHumidity();**

**float t = dht.readTemperature();**

**// Check if the data was successfully read**

**if (isnan(h) || isnan(t)) {**

**Serial.println("Failed to read from DHT sensor!");**

**} else {**

**// Print temperature and humidity data to the serial monitor**

**Serial.print("Temperature: ");**

**Serial.print(t);**

**Serial.print("°C, Humidity: ");**

**Serial.print(h);**

**Serial.println("%");**

**// Create character buffers to publish the data**

**char temperatureString[8];**

**char humidityString[8];**

**dtostrf(t, 1, 2, temperatureString);**

**dtostrf(h, 1, 2, humidityString);**

**// Publish temperature and humidity data to MQTT topics**

**client.publish("dht/temperature", temperatureString);**

**client.publish("dht/humidity", humidityString);**

**}**

**// Read the analog value from the soil sensor**

**int soilMoistureValue = analogRead(soilSensorPin);**

**// Check soil moisture level and control the relay**

**if (soilMoistureValue > 1600) {**

**Serial.println("Soil is dry - Turning ON Water Pump!");**

**Serial.print("Soil Moisture Value: ");**

**Serial.println(soilMoistureValue);**

**digitalWrite(soilrelayPin, LOW); // Turn on relay**

**client.publish("relay/soil", "ON"); // Publish relay status**

**delay(3000);**

**digitalWrite(soilrelayPin, HIGH); // Turn off relay**

**} else {**

**Serial.println("Soil moisture is in a good range.");**

**client.publish("relay/soil", "OFF"); // Publish relay status**

**}**

**// Publish soil moisture data to MQTT**

**char moistureString[8];**

**itoa(soilMoistureValue, moistureString, 10); // Convert integer to string**

**client.publish("soil/moisture", moistureString);**

**// Temperature relay control based on temperature value**

**if (t > 25.0) { // If the temperature is greater than 25°C**

**Serial.println("It is too hot! Turning ON temp relay.");**

**digitalWrite(tempRelayPin, LOW); // Turn on the second relay**

**client.publish("relay/temperature", "ON"); // Publish relay status**

**} else {**

**digitalWrite(tempRelayPin, HIGH); // Turn off the second relay**

**client.publish("relay/temperature", "OFF"); // Publish relay status**

**}**

**// Send a pulse to trigger the ultrasonic sensor**

**digitalWrite(trigPin, LOW);**

**delayMicroseconds(2); // Short delay to ensure the sensor resets**

**digitalWrite(trigPin, HIGH);**

**delayMicroseconds(10); // 10 microsecond pulse**

**digitalWrite(trigPin, LOW);**

**// Measure the time it takes for the echo to return**

**long duration = pulseIn(echoPin, HIGH);**

**// Calculate the distance (in cm)**

**float distance = duration \* 0.0344 / 2; // speed of sound is 0.0344 cm/μs, divide by 2 for round trip**

**// Print the result to the Serial Monitor**

**Serial.print("Distance: ");**

**Serial.print(distance);**

**Serial.println(" cm");**

**// Check if the distance is less than the threshold (10 cm)**

**if (distance < thresholdDistance) {**

**// If distance is less than 10 cm, open the servo to 160 degrees**

**Serial.println("Opening servo to 160 degrees!");**

**myServo.write(160); // Move the servo to 160 degrees**

**delay(10000); // Wait for 10 seconds**

**myServo.write(0); // Move the servo back to 0 degrees**

**Serial.println("Closing servo to 0 degrees!");**

**}**

**// Delay for a short period before the next loop**

**delay(2000); // 2 seconds delay to prevent excessive serial printing**

**}**

### **References:**

* **Datasheets:**
  + DHT22: <https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0137_Web.pdf>
  + Soil Moisture Sensor: <https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0193_Web.pdf>
  + Relay Module: <https://www.elegoo.com/blogs/arduino-projects/elegoo-dc-5v-relay-module-tutorial>
  + Water Pump: <http://brainzstore.com/admin/productimages/5V%20Mini%20Water%20Pump/water%20pump%20mini%205v.pdf>
  + Mechanical Motor: <https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf>
  + Servo Motor: <https://cdn.sparkfun.com/datasheets/Robotics/Small%20Servo%20-%20ROB-09065.pdf>
  + ESP32: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>
* Node-RED was configured mimicking lecture 9.
* Tutorials and Coding resources
  + Servo Motor:   
    <https://esp32io.com/tutorials/esp32-servo-motor>
  + MQTT: <https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/>
  + Read Analog Data: <https://docs.arduino.cc/tutorials/uno-rev3/AnalogReadSerial/>
  + DHT code was adapted from the lecture