Mini Project

Smart Plant Watering System

Member

Student ID 65107690 Ms.Waraporn Prawat

Student ID 65132185 Ms.Kewalin Savatadul

Present

Dr.Eshrat E Alahi

This report is part of COE64-213 Microprocessors and Embedded Systems

Engineering and Technology Program

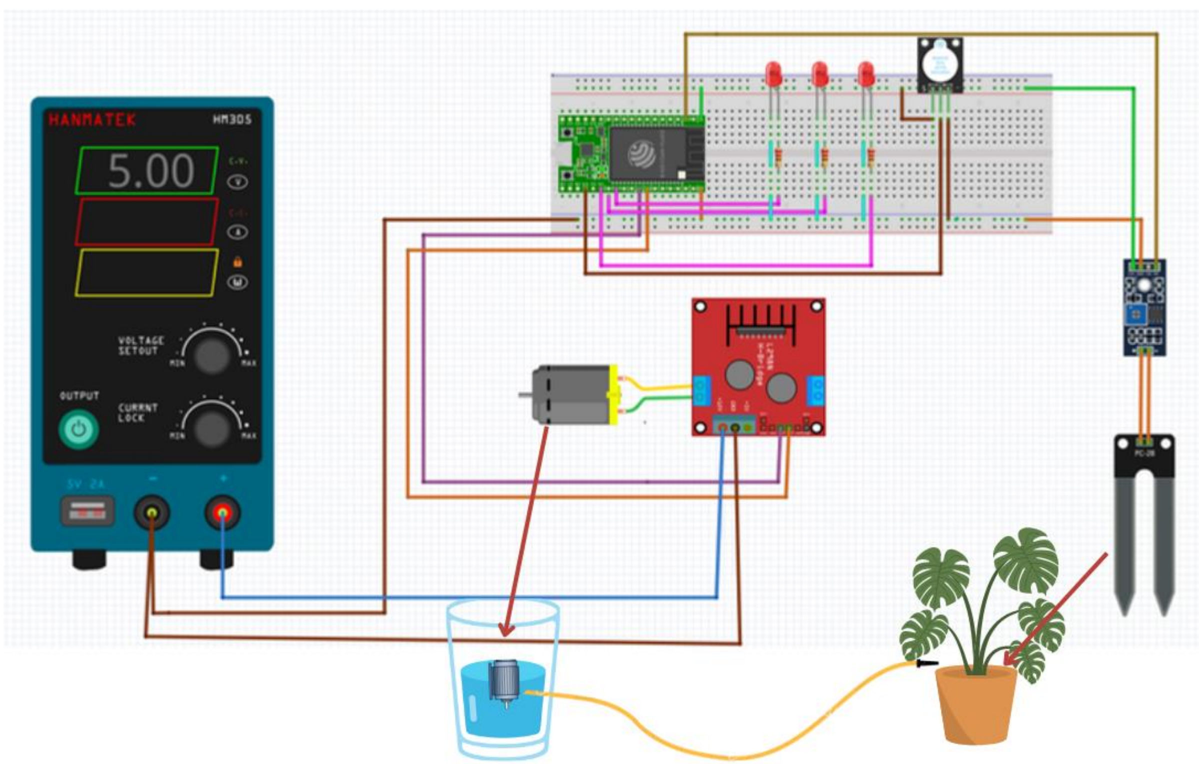
Computer Engineering and Artificial Intelligence

Objective

Design and implement a smart plant watering system that automates watering plants based on their moisture levels. The system will use an ESP32 microcontroller, soil moisture sensors, and a water pump. The system will integrate functions such as Soil Moisture Sensing, Motor Control, Remote Monitoring, and Alerts and Notifications.

The smart plant watering system will provide an innovative and efficient solution for maintaining healthy plants while reducing the time and effort required for manual watering. Additionally, it will contribute to sustainable practices by conserving water resources and promoting optimal plant care.

Including Circuit Diagrams



Working Principle

Components

1. ESP32 Development Board : Acts as the central controller for the plant watering system.

2. Soil Moisture Sensors : Measure the moisture level in the soil for different plants.

3. Water Pump : Controls the water flow to irrigate the plants.

4. Relay Module : Enables the ESP32 to control the water pump.

5. Tubing and Nozzle : Distributes water to the plants.

6. Buzzer/LEDs : Provides feedback on watering events.

Working Principle

- If the moisture percentage is less than 40, it indicates low moisture, so the third LED (LED 3) is turned on to indicate watering is needed. Additionally, a speaker connected to SPEAKER_PIN will turn on notifications. Water pump 1 (WATER PUMP 1) is turned on to supply water.
- If the moisture percentage is between 40 and 70, it indicates moderate moisture, so the second LED (LED 2) is turned on to indicate the system is monitoring the moisture level. The speaker and Both water pumps are turned off.
- If the moisture percentage is between 70 and 100, it indicates high moisture, so the first LED (LED 1) is turned on to indicate excess moisture. The speaker and Both water pumps are turned off.
- If the moisture percentage is out of range (above 100 or below 0), all LEDs, the speaker, and both water pumps are turned off.
- The loop then waits for 500 milliseconds before starting the next iteration.

Code Connection

Various components connected to the ESP32 as follows :

- AOUT_PIN: Pin connected to the soil moisture sensor's analog output.
- LED_PIN_1, LED_PIN_2, LED_PIN_3: Pins connected to LEDs for indicating system status.
- SPEAKER_PIN: Pin connected to a speaker (or buzzer) for sound feedback.
- WATER_PUMP_PIN_1, WATER_PUMP_PIN_2: Pins connected to water pumps.

In the **void setup()** section :

- Serial communication is initialized at 9600 bps.
- The pinMode of all used pins is set to OUTPUT.

In the **void loop()** section :

- Soil moisture level is read from the sensor.
- The analog value is mapped to a percentage representing soil moisture.
- Conditional statements determine the soil moisture level and control the LEDs, speaker, and water pumps accordingly.
- The system delays for 500 milliseconds before looping again.

Coding

```
#define AOUT_PIN 36 // ESP32 pin GPIO36 (ADC0) that connects to AOUT
pin of the moisture sensor

#define LED_PIN_1 16 // GPIO17 for the first LED

#define LED_PIN_2 4 // GPIO18 for the second LED

#define LED_PIN_3 0 // GPIO19 for the third LED

#define SPEAKER_PIN 15 // GPIO32 for the speaker

#define WATER_PUMP_PIN_1 18 // GPIO33 for water pump 1

#define WATER_PUMP_PIN_2 19 // GPIO34 for water pump 2
```

```
void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN_1, OUTPUT);
  pinMode(LED_PIN_2, OUTPUT);
  pinMode(LED_PIN_3, OUTPUT);
  pinMode(SPEAKER_PIN, OUTPUT);
  pinMode(WATER_PUMP_PIN_1, OUTPUT);
  pinMode(WATER_PUMP_PIN_2, OUTPUT);
}

void loop() {
  int value = analogRead(AOUT_PIN); // Read analog value from the
sensor

  // Convert moisture value to a range of 0-100, where 4095 is mapped
to 0% and 0 is mapped to 100%
  float moisturePercentage = map(value, 0, 4095, 100, 0);
  Serial.print("Analog value: ");
  Serial.print(value);
  Serial.print(" (Moisture value: )");
  Serial.print(moisturePercentage);
  Serial.println("%");

  if (moisturePercentage < 40) {
    digitalWrite(LED_PIN_1, LOW);
    digitalWrite(LED_PIN_2, LOW);
    digitalWrite(LED_PIN_3, HIGH);
    digitalWrite(SPEAKER_PIN, LOW);
    delay(100);
    digitalWrite(SPEAKER_PIN, HIGH);
    delay(100);
```

```
    digitalWrite(WATER_PUMP_PIN_1, HIGH);
    digitalWrite(WATER_PUMP_PIN_2, LOW);
  } else if (moisturePercentage >= 40 && moisturePercentage < 70) {
    digitalWrite(LED_PIN_1, LOW);
    digitalWrite(LED_PIN_2, HIGH);
    digitalWrite(LED_PIN_3, LOW);
    digitalWrite(SPEAKER_PIN, HIGH);
    digitalWrite(WATER_PUMP_PIN_1, LOW);
    digitalWrite(WATER_PUMP_PIN_2, LOW);
  } else if (moisturePercentage >= 70 && moisturePercentage <= 100) {
    digitalWrite(LED_PIN_1, HIGH);
    digitalWrite(LED_PIN_2, LOW);
    digitalWrite(LED_PIN_3, LOW);
    digitalWrite(SPEAKER_PIN, HIGH);
    digitalWrite(WATER_PUMP_PIN_1, LOW);
    digitalWrite(WATER_PUMP_PIN_2, LOW);
  } else {
    digitalWrite(LED_PIN_1, LOW);
    digitalWrite(LED_PIN_2, LOW);
    digitalWrite(LED_PIN_3, LOW);
    digitalWrite(SPEAKER_PIN, LOW);
    digitalWrite(WATER_PUMP_PIN_1, LOW);
    digitalWrite(WATER_PUMP_PIN_2, LOW);
  }

  delay(500);
}
```

Pictures and Videos

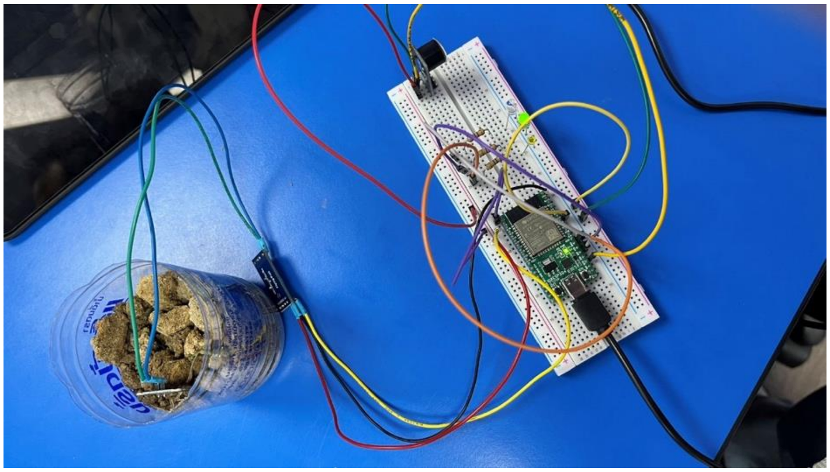


Figure **1 It's a circuit setup without including the motor and water pump yet.**

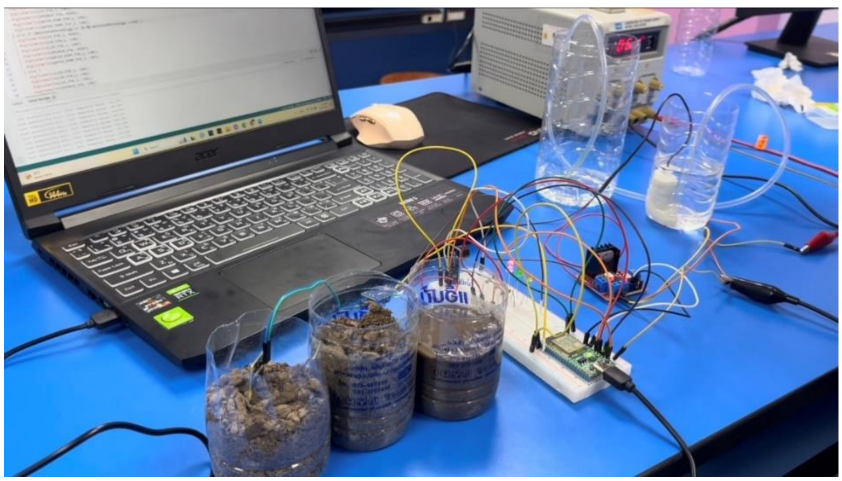


*Figure **2 The entire circuit setup***

QR code to watch the experiment video.

https://drive.google.com/drive/folders/1-3fkL7AmaSav-eQ7lzmPFpLap1PAFge-?usp=drive_link

Results And Discussions

Results

The experiment and testing of the smart plant watering system as designed yielded efficient results in controlling plant watering based on soil moisture levels, as follows

1. The system accurately detects and measures soil moisture levels using the designated sensor.
2. Control of the water pump operation to irrigate plants according to system specifications.
3. The system efficiently sends notifications via mobile devices or online platforms.
4. Control of system status through both visual indicators and sound alerts provides users with real-time feedback.
5. System responsiveness to soil moisture levels may be affected by delays in response when there are issues with the wiring or connections.

Discussion

1. Inaccurate Sensor Readings : Occasionally, this type of sensor may provide inaccurate readings when compared to actual soil moisture levels, potentially leading to ineffective watering control.
2. Fluctuating Values : Wiring issues or problematic connections can cause fluctuations in the values related to the sensor or water pump control, affecting the system's ability to regulate watering effectively.
3. Troubleshooting Wiring Issues : Addressing issues related to faulty or problematic wiring may require significant time and effort, potentially posing a challenge in the maintenance and upkeep of the system.
4. Lack of IoT Integration : While the current system lacks IoT integration for remote control, incorporating IoT features in the future would significantly enhance remote accessibility and control capabilities.

Addressing and resolving issues related to wiring connections and sensor problems will be crucial for ensuring the long-term effectiveness and stability of the system. Testing and continuous improvement are essential steps in the development of this project to ensure usability and efficiency for users.

Summary Of The Project

The project aimed to design and implement a smart plant watering system utilizing an ESP32 microcontroller, soil moisture sensors, and a water pump. The system integrates functions such as soil moisture sensing, motor control, remote monitoring, and alerts/notification. By automating watering based on plant moisture levels, it offers an innovative solution to maintaining healthy plants while reducing manual effort. Despite the lack of IoT integration in the current version, future development could enhance remote accessibility and control, contributing to sustainability by conserving water resources and promoting optimal plant care.