**AUTOMATED IRRIGATION CONTROL SYSTEM BASED ON ENVIRONMENTAL SENSING**

***TEAM MEMBERS:***

* ***JOSEPH RIJAN A***
* ***INDHUCHUDAN V S***
* ***MAHESH A***
* ***POORNESH V***

**AIM:**

To design and implement an **automated irrigation control system** that efficiently manages water delivery to crops by sensing environmental factors such as **soil moisture, temperature, and humidity, using the ESP8266 microcontroller** for real-time monitoring, control, and remote management.

**COMPONENTS REQUIRED:**

* ESP8266 microcontroller
* Soil moisture sensor
* DHT22 temperature and humidity sensor
* Water level sensor
* Motor pump
* Relay module (2 channel)
* Power supply
* OLED display
* LED
* Connecting wires

***FUNCTIONS:***

**1. ESP8266 Microcontroller**

* **Purpose:** Acts as the central controller, processing sensor data, controlling irrigation actuators, and providing Wi-Fi connectivity for remote monitoring.
* **Range & Features:** Operates at 3.3V, has multiple GPIO pins, supports Wi-Fi.
* **Working Levels:** Logic level at 3.3V (input voltage max 3.6V); operating temperature -40°C to +125°C.
* **Why Use:** Cost-effective, widely supported, low power with IoT capability.

**2. Soil Moisture Sensor**

* **Purpose:** Measures volumetric water content in soil.
* **Range:** 0% (dry) to 100% (saturated).
* **Operating Voltage:** Typically 3.3V to 5V.
* **Working Levels:** Provides analog voltage corresponding to soil moisture level; threshold typically around 30-40% moisture for irrigation trigger.
* **Why Use:** Directly measures soil condition to avoid over/under irrigation.

**3. DHT22 Temperature and Humidity Sensor**

* **Purpose:** Measures ambient temperature and relative humidity.
* **Range:** Temperature -40°C to +80°C (±0.5°C accuracy), Humidity 0-100% RH (±2-5% accuracy).
* **Operating Voltage:** 3.3V to 5V.
* **Working Levels:** Digital output signal, sensor wakes on request.
* **Why Use:** Environmental data helps adjust irrigation based on climate conditions

**4. Relay Module**

* **Purpose:** Electrically isolates and switches water pumps or solenoid valves.
* **Range:** Controls AC or DC loads; relay coil operates at 5V (can be controlled from ESP8266 with a transistor).
* **Max Load:** Typically 10A/250V AC or 10A/30V DC.
* **Working Levels:** Input control signal from ESP8266 GPIO pin.
* **Why Use:** Necessary for switching high voltage/current devices safely.

**5. Water Level Sensor**

* **Purpose:** Monitors water level in storage tanks.
* **Range:** Usually three-point detection - low, medium, high (can be customized).
* **Operating Voltage:** 3.3V to 5V.
* **Working Levels:** Digital or analog output.
* **Why Use:** Prevents pump dry-run by alerting low water levels.

**6. Power Supply**

* **Purpose:** Provides stable power to ESP8266, sensors, and relays.
* **Specification:** 5V regulated supply, with 3.3V regulator for ESP8266.
* **Why Use:** Ensures system reliability and component safety.

**7. Motor Pump**

* **Purpose:**Pumps water from the storage tank to the field when irrigation is activated.
* **Operating Range:**Typical small DC pumps run on 6V to 12V, with currents around 0.5A to 2A.
* **Why Use:**Essential for delivering water; controlled remotely by the relay module.
* **Max/Min Levels:**Voltage must match pump specs; overvoltage can damage the pump, undervoltage may reduce performance.

**8. OLED Display**

* **Purpose:** Provides real-time system feedback locally, such as soil moisture, temperature, and pump status.
* **Range:** Small displays often 0.96 to 1.3 inches diagonally; operating voltage typically 3.3V.
* **Why Use:** Offers quick, on-site status without needing external devices.
* **Working Levels:** Driven via I2C (uses GPIO pins D1 for SCL, D2 for SDA).

**9. LED Indicators**

* **Purpose:** Provide visual signals for system states (e.g., power on, irrigation active, alert).
* **Range:** Typically 3V forward voltage with 20mA current.
* **Why Use:** User-friendly indication to quickly understand system status.
* **Working Levels:** Use series resistors; exceeding ratings will burn out the LED.

**10. Breadboard**

* **Purpose:** Temporary platform to build and test the circuit without soldering.
* **Range:** Supports standard 0.1 inch pin spacing compatible with ESP8266 and sensors.
* **Why Use:** Enables easy prototyping and debugging.

**11. Connecting Wires**

* **Purpose:** To connect all components, transmit signals, and power between modules.
* **Range:** Typically 22 to 28 AWG stranded or solid wires.
* **Why Use:** Reliable, flexible medium for electrical connections.

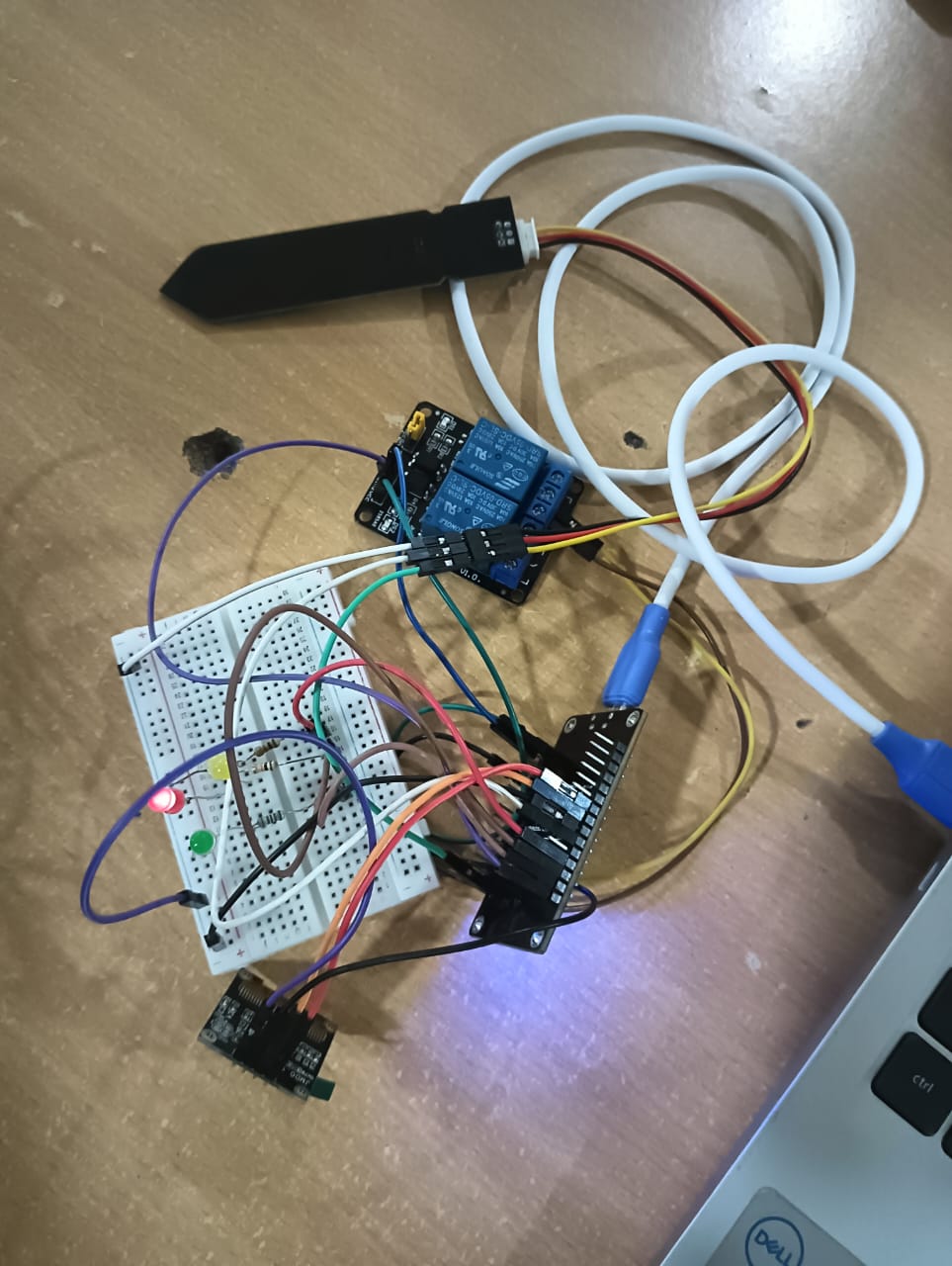
**PIN DIAGRAM:**

ESP8266 Pin Configuration (NodeMCU Typical)

| **Pin Name** | **GPIO Number** | **Function** |
| --- | --- | --- |
| D0 | GPIO16 | Can be used for sensor input |
| D1 | GPIO5 | I2C SCL (optional for sensors) |
| D2 | GPIO4 | I2C SDA (optional for sensors) |
| A0 | ADC0 (Pin) | Analog input (e.g., soil sensor) |
| D5 | GPIO14 | Relay control output |
| D6 | GPIO12 | Relay control output or sensor |
| D7 | GPIO13 | General purpose I/O |
| 3V3 | Power | 3.3V power supply |
| GND | Power Ground | Ground reference |

**Circuit Connection Diagram (Summary)**

* Soil Moisture Sensor analog output to ESP8266 A0 pin.
* DHT22 data pin connected to GPIO D4 (GPIO2).
* Relay module input connected to GPIO D5 or D6 through transistor driver circuit.
* Relay common connected to pump or valve power line.
* Water level sensor to GPIO D7 (optional).
* ESP8266 powered with 3.3V regulated supply.
* Common GND for all components

 **CIRCUIT CONNECTION:**

**FLOW CHART**

Start

↓

Initialize system (pins, serial)

↓

Read soil moisture sensor value

↓

Is moisture < thresholdLow

├─ Yes → Turn ON pump (relay ON), Blink LED1 (Dry)

└─ No → Turn OFF pump

↓

Is moisture < thresholdMid

├─ Yes → Turn ON pump (relay ON), Blink LED2 (Moderate)

└─ No → Turn OFF pump (relay OFF), Blink LED3 (Wet)

↓

Wait for 1 second

↓

Repeat

**CODE SNIPPET**

// Pin definitions using GPIO numbers

#define RELAY\_PIN 5    // GPIO5 (D1) -> Relay IN1

#define SCL\_PIN   14   // GPIO14 (D5) -> OLED SCL

#define SDA\_PIN   12   // GPIO12 (D6) -> OLED SDA

#define SOIL\_PIN  A0   // Analog pin (A0) -> Soil sensor

// Extra LEDs

#define LED1\_PIN  4    // GPIO4 (D2)

#define LED2\_PIN  0    // GPIO0 (D3)

#define LED3\_PIN  2    // GPIO2 (D4)

void setup() {

  Serial.begin(115200);

  // Relay setup

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, HIGH); // Relay OFF initially (active LOW)

  // I2C pins (just declared, not used without OLED library)

  pinMode(SCL\_PIN, OUTPUT);

  pinMode(SDA\_PIN, OUTPUT);

  // LED pins

  pinMode(LED1\_PIN, OUTPUT);

  pinMode(LED2\_PIN, OUTPUT);

  pinMode(LED3\_PIN, OUTPUT);

  // Start with all LEDs OFF

  digitalWrite(LED1\_PIN, LOW);

  digitalWrite(LED2\_PIN, LOW);

  digitalWrite(LED3\_PIN, LOW);

  Serial.println("System Ready");

}

void loop() {

  int soilValue = analogRead(SOIL\_PIN);

  // Convert to percentage (0=wet, 1023=dry)

  int moisturePercent = map(soilValue, 1023, 0, 0, 100);

  // Print values

  Serial.print("Soil Value: ");

  Serial.print(soilValue);

  Serial.print(" | Moisture: ");

  Serial.print(moisturePercent);

  Serial.println("%");

  // Relay control: pump ON if soil moisture < 40%

  if (moisturePercent < 40) {

    digitalWrite(RELAY\_PIN, LOW);  // Relay ON

    Serial.println("Pump: ON");

  } else {

    digitalWrite(RELAY\_PIN, HIGH); // Relay OFF

    Serial.println("Pump: OFF");

  }

  // LED indicators (example usage)

  if (moisturePercent < 30) {

    digitalWrite(LED1\_PIN, HIGH); // Very dry

    digitalWrite(LED2\_PIN, LOW);

    digitalWrite(LED3\_PIN, LOW);

  } else if (moisturePercent < 60) {

    digitalWrite(LED1\_PIN, LOW);

    digitalWrite(LED2\_PIN, HIGH); // Moderate

    digitalWrite(LED3\_PIN, LOW);

  } else {

    digitalWrite(LED1\_PIN, LOW);

    digitalWrite(LED2\_PIN, LOW);

    digitalWrite(LED3\_PIN, HIGH); // Wet

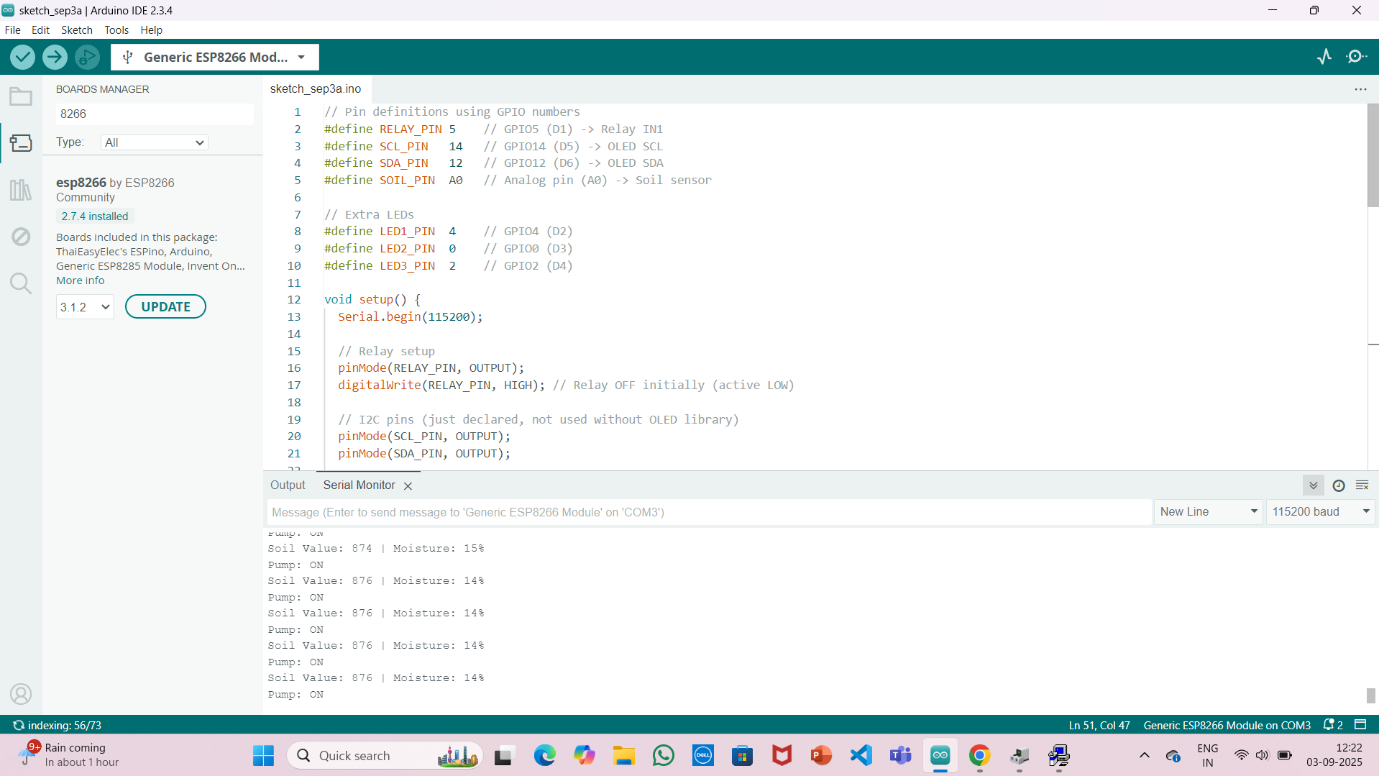
  }

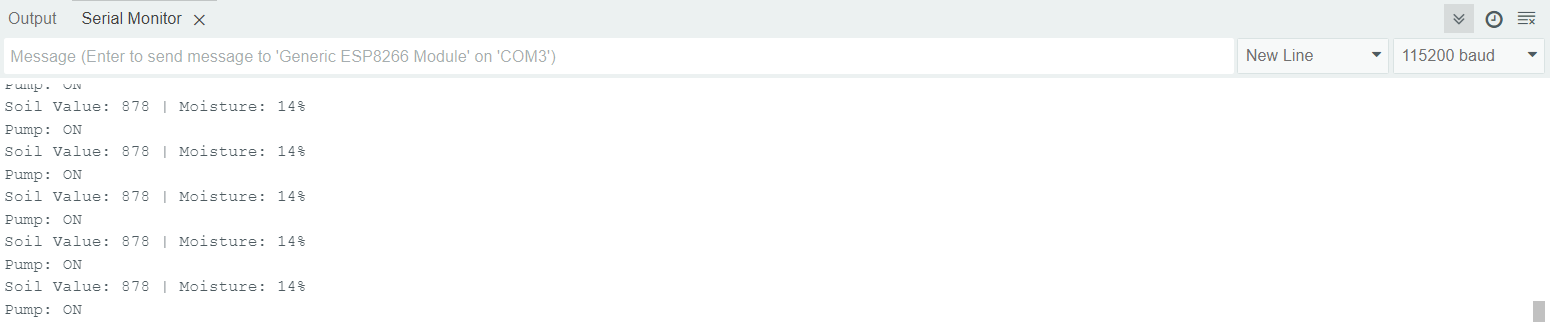
  delay(2000);

}

**Execution and Testing**

* Power the ESP8266 and connect sensors as per circuit.
* Upload the code to ESP8266 using Arduino IDE.
* Observe serial monitor for real-time sensor readings and relay activity.
* Place soil moisture sensor in dry and wet soil to test irrigation trigger.
* Verify relay correctly switches the water pump/valve.
* Optionally integrate with cloud platforms for remote monitoring.
* Adjust soil moisture threshold values to optimize watering schedule as per crop needs.

****



**RESULT:**

The automated irrigation system demonstrated reliable and efficient operation by continuously sensing environmental factors and activating irrigation based on data-driven decisions. It helps conserve water, reduces manual intervention, and enhances crop productivity. The results affirm that this system can be deployed in agricultural settings to improve irrigation practices sustainably.