****

**IoT-Based Soil Moisture Monitoring & Irrigation System**

**Project Title:** IoT-Based Soil Moisture Monitoring & Irrigation System

**Submitted By:**

* 09 Dhanashree Kamble
* 12 Nikita Lendghar
* 13 Nishant Mandal
* 32 Vaibhavi Raut

**Department & College Name:** PCCOE | IOT BRANCH

**Guide/Professor Name:** GANESH RAHATE

**Date of Submission:** 04/02/2025

**INDEX**

|  |  |  |
| --- | --- | --- |
| 01 | INTRODUCTION | 04 |
| 02 | COMPONENTS | 06 |
| 03 | WORKING OF SYSTEM | 08 |
| 04 | TROUBLESHOOT | 10 |
| 05 | BENEFITS OF PROJECT | 14 |
| 06 | MOBILE APP FEATURES | 16 |
| 07 | CONCLUSION | 17 |
| 08 | CODE OF SOIL MOISTURE | 19 |
| 09 | REFERENCE | 25 |

**CERTIFICATE**



**Project Report**

**on**

**Project ID:**

**<TITLE>**

Submitted for Fulfillment of the Requirements for the Degree of Bachelor of Vocation in the Department of Internet Of Things, Pimpri Chinchwad College of Engineering, Savitribai Phule University of Pune, Pune.

By

**Name of the students Enrollment NO.**

Dhanashree Kamble 123V1I009

Nikita Lendghar 123V1I012

Nishant Mandal 123V1I013

Vaibhavi Raut 124V2I032

**Ganesh Rahate Dr. U. G. Potdar Prakash Sonatake**

**(Project Guide) (H.O.D) (Director)**

**Department of Bachelor of Vocation – Internet of Things,**

**Pimpri Chinchwad College of Engineering, Pune**

**Savitribai Phule University of Pune, Pune.2024-2025**

**Abstract**

This project addresses the challenges of traditional farming, where manual soil moisture monitoring often results in inefficient water use and suboptimal crop growth. The IoT-based system automates the process of measuring soil moisture, temperature, and humidity, and controls irrigation accordingly. Using a NodeMCU for data processing and wireless communication, along with sensors to gather environmental data, the system optimizes water usage by activating a water pump only when necessary. Remote control via a mobile app offers farmers convenience and precise control over their irrigation schedules. This report details the system’s design, components, working principles, benefits, and potential future improvements for a sustainable agricultural solution.

**1. Introduction**

**1. Overview of the Problem**

**Traditional Farming Challenges:**

* **Manual Monitoring:** Farmers traditionally check soil moisture manually, which is labour-intensive and often inaccurate.
* **Inefficient Water Use:** Without precise measurements, watering may be excessive or insufficient, leading to resource wastage.
* **Impact on Crop Health:** Inconsistent irrigation affects soil nutrient balance and plant health, reducing crop yield.
* **Time Constraints:** Regular manual checks divert time from other critical farming activities.

**1.2. Proposed Solution**

**The proposed system leverages IoT technology to provide a robust solution:**

* **Automate Monitoring:**  
  IoT-enabled soil moisture sensors continuously track the water content in the soil, providing data to a central microcontroller.
* **Real-Time Data:**  
  Integration of a DHT11 sensor ensures that temperature and humidity are also monitored, offering a comprehensive view of the environment.
* **Automated Irrigation:**  
  The system activates a water pump automatically via a relay when moisture levels drop below a predefined threshold.
* **Remote Management:**A mobile app facilitates remote monitoring and control, allowing farmers to adjust irrigation settings and view live data from anywhere.

**1.3. Main Objective**

Improve farming efficiency and crop yield by automating the monitoring and control of soil moisture using IoT technology, thereby ensuring that plants receive the optimum amount of water.

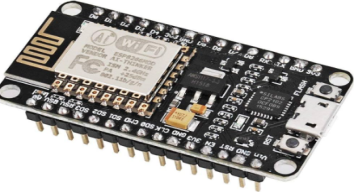
**1.4. Additional Features**

* **Remote Operation:**The mobile app enables farmers to remotely start or stop the water pump, offering flexibility and real-time intervention.
* **Data-Driven Decisions:**  
  Real-time monitoring helps in adjusting irrigation schedules based on environmental conditions, leading to better resource management.
* **Scalability:**  
  The system can be expanded to incorporate additional sensors or control modules for larger or multi-field applications.
* **Optimized Water Usage:** Minimizes water wastage and supports sustainable agricultural practices.

**2. Components**

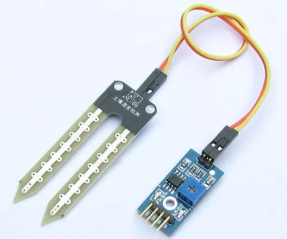
**2.1. Hardware Components**

**1.NodeMCU (ESP8266):**

****

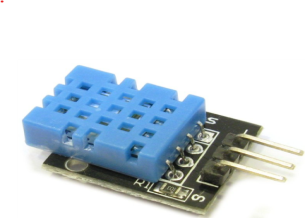
* + Role: Acts as the central controller, processing sensor data and communicating with the mobile app via Wi-Fi.
  + Key Features: Integrated Wi-Fi, sufficient I/O pins, and support for Arduino IDE programming.

**2.Soil Moisture Sensor:**

****

* + Function: Measures the volumetric water content in the soil.
  + Types: Typically, resistive or capacitive sensors; capacitive types tend to be more reliable for long-term use.

**3.DHT11 Temperature and Humidity Sensor:**

****

* + Function: Captures ambient temperature and humidity data.
  + Accuracy: Suitable for basic environmental monitoring; upgrades like DHT11 can be considered for more precision.

4.**Relay Module:**

****

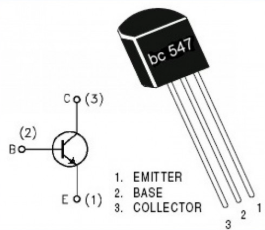
* + Function: Acts as an electronic switch to control the water pump.
  + Operation: Receives a low-voltage signal from the NodeMCU and switches the high-voltage circuit for the pump.

**5.Water Pump:**

****

* + Function: Delivers water to the soil during irrigation cycles.
  + Specifications: Typically, a small DC pump; selection depends on field size and water requirements.

**6.BC547 Transistor:**

****

* + Function: Serves as a switch in the control circuit, ensuring that the relay is activated safely.
  + Application: Works with a current-limiting resistor to drive the relay without damaging the NodeMCU.

**7.220 Ohm Resistor:**



* + Function: Limits the current in the transistor circuit to protect sensitive components.
  + Usage: Placed in series with the base of the BC547 transistor.

.

**3. Working of the System**

**3.1. Data Collection**

* Continuous Monitoring:  
  The soil moisture sensor is embedded in the soil, providing continuous readings of the moisture content.
* Environmental Sensing:  
  The DHT11 sensor captures the ambient temperature and humidity, supplying additional context for irrigation decisions.
* Data Transmission:  
  Both sensors send their analog/digital data to the NodeMCU for processing.

**3.2. Decision-Making Process**

* Data Processing:  
  The NodeMCU reads the sensor values using its ADC (Analog-to-Digital Converter) and digital input pins.
* Threshold Comparison:  
  Predefined threshold values for soil moisture (and possibly temperature/humidity) are set within the NodeMCU’s code.
* Logic Implementation:  
  A simple conditional algorithm checks whether the current moisture level is below the threshold, indicating the need for irrigation.

**3.3. Actuation**

* Relay Activation:  
  If the soil moisture is too low, the NodeMCU sends a signal to the BC547 transistor, which in turn triggers the relay.
* Water Pump Operation:  
  Once the relay is activated, the water pump starts, irrigating the soil until the moisture level is restored to optimal conditions.
* Safety Measures:  
  The use of the BC547 transistor and the 220 Ohm resistor ensures that the NodeMCU is isolated from high currents, protecting both the microcontroller and other components.

**3.4. Remote Control Functionality**

* Mobile App Connectivity:  
  The NodeMCU connects to the internet via Wi-Fi and communicates with a mobile app. This app displays sensor data and offers manual control options.
* User Interaction:  
  Farmers can log in to the app, view real-time updates, and manually override the automatic system if needed.
* Alerts & Notifications:  
  The system can be programmed to send notifications (via SMS, email, or in-app alerts) when soil moisture levels reach critical levels.

**3.5 Block Diagram**

Soil Moisture DHT11 Mobile App

Moisture Data Temp & Humidity Data Wi-Fi Control

Node MCU (ESP8266)

Control Signal

BC547 Transistor

switching

Relay Module

Power to pump

Water Pump

## **4. Troubleshooting**

### **4. No Data from Sensors (Soil Moisture, DHT11)**

**Possible Causes:**

* Loose or incorrect wiring connections.
* Faulty sensor or damaged components.
* Power supply issue.
* Incorrect programming or firmware issues in NodeMCU.

**Solutions:**  
✅ Check wiring connections and ensure correct pin assignments in the code.  
✅ Verify that the sensors are powered correctly.  
✅ Replace faulty sensors if necessary.  
✅ Re-upload the firmware/sketch to the NodeMCU and check the serial monitor for error messages.

### **2. Water Pump Not Turning On/Off Automatically**

**Possible Causes:**

* Faulty relay module or incorrect wiring.
* Low voltage or power supply issue.
* Incorrect logic in the program.
* Faulty moisture sensor reading incorrect values.

**Solutions:**  
✅ Check relay wiring and ensure it is connected properly to NodeMCU.  
✅ Measure the voltage supply to the relay and pump using a multimeter.  
✅ Manually test the relay by triggering it using code.  
✅ Replace the relay or water pump if necessary.

### **3. Inaccurate Soil Moisture Readings**

**Possible Causes:**

* Soil moisture sensor placed incorrectly or too deep/shallow in soil.
* Sensor deterioration due to long-term soil exposure.
* Dry soil or improper calibration.
* Electrical interference affecting the readings.

**Solutions:**  
✅ Place the sensor at an optimal depth (2-3 inches in the soil).  
✅ Clean the sensor and check for corrosion.  
✅ Calibrate the sensor by comparing with manual moisture tests.  
✅ Keep the sensor away from direct sunlight and other electrical components.

### **4. Mobile App Not Connecting to System**

**Possible Causes:**

* Weak or no Wi-Fi signal.
* Incorrect IP address or network configuration.
* NodeMCU not connected to the internet.
* Firewall or security settings blocking communication.

**Solutions:**  
✅ Ensure the Wi-Fi router is working and has a stable internet connection.  
✅ Check if the NodeMCU is connected to the correct Wi-Fi network.  
✅ Restart both the NodeMCU and mobile app, then retry.  
✅ Verify the correct IP address of the system and update it in the app.

### **5. System Not Responding to Remote Commands**

**Possible Causes:**

* Delay in internet connection.
* Server timeout or API request failure.
* NodeMCU crashing due to memory issues.
* Mobile app bug or outdated version.

**Solutions:**  
✅ Check the internet speed and stability.  
✅ Restart the system and clear any excessive cache or logs.  
✅ Optimize the NodeMCU code to prevent memory overload.  
✅ Update the mobile app to the latest version.

### **6. System Keeps Restarting or Crashing**

**Possible Causes:**

* Insufficient power supply to the NodeMCU.
* Overloaded processing due to continuous data logging.
* Faulty or shorted components.

**Solutions:**  
✅ Use a stable 5V power supply with sufficient current rating.  
✅ Reduce data logging frequency in the code.  
✅ Check for any short circuits in the wiring.

### **7. Relay Stuck in ON/OFF Position**

**Possible Causes:**

* Relay contacts burned out due to excessive current.
* Faulty transistor (BC547) or incorrect resistor value.
* Software logic error.

**Solutions:**  
✅ Replace the relay module if the contacts are damaged.  
✅ Check the transistor and ensure the correct resistor is used.  
✅ Review and debug the control logic in the NodeMCU code.

### **8. DHT11 Sensor Not Providing Temperature/Humidity Data**

**Possible Causes:**

* Faulty sensor or improper wiring.
* Sensor too close to heat sources, affecting accuracy.
* Incorrect data format in the code.

**Solutions:**  
✅ Ensure proper wiring and correct data pin connection.  
✅ Move the sensor to a shaded area away from heat sources.  
✅ Verify and correct data formatting in the NodeMCU script.

**5. Benefits of the Project**

**5.1. Efficient Watering**

* **Precision Irrigation:**Automated control ensures that water is used only when necessary, reducing wastage.
* **Optimized Schedules:**Real-time data allows for dynamic irrigation schedules that adjust based on current conditions.

**5.2. Improved Crop Health**

* **Consistent Moisture Levels:**Maintaining optimal soil moisture prevents crop stress and promotes healthier growth.
* **Reduced Waterlogging:**Controlled irrigation minimizes the risks of overwatering and the associated problems like root rot.

**5.3. Remote Management**

* **Ease of Use:**Farmers can monitor and control their irrigation systems from anywhere using the mobile app.
* **Multi-Field Management:**The system can be scaled to monitor and control multiple fields simultaneously**.**

**5.4. Cost Savings**

* **Labor Reduction:**Automating irrigation reduces the need for manual labour, saving time and operational costs.
* **Resource Optimization:**Efficient water use leads to lower utility bills and a reduced environmental footprint.

**6.Mobile App Features**

**6.1. Real-Time Monitoring**

* **Dashboard:**A user-friendly interface displaying current soil moisture, temperature, and humidity levels.
* **Historical Data:**Graphs and charts to track changes over time, enabling trend analysis.

**6.2. Remote Control**

* **Manual Override:**Allows users to activate or deactivate the water pump remotely, providing flexibility during unexpected conditions.
* **Automated Alerts:**Push notifications or SMS alerts to inform users when soil moisture falls below or exceeds set thresholds.

**6.3. User-Friendly Interface**

* **Intuitive Design:**Simple menus and controls designed for farmers with limited technical background.
* **Customization:**Options to set threshold values, notification preferences, and scheduling parameters.

**7.Conclusion**

**7.1. Summary**

The IoT-based soil moisture monitoring and irrigation system offers a comprehensive solution to the limitations of traditional farming practices. By automating sensor-based monitoring and integrating remote control capabilities via a mobile app, the system ensures that crops receive precise irrigation, thereby enhancing crop health and optimizing water usage. The system’s scalability and ease of use make it an attractive option for both small-scale and commercial farming operations.

**7.2. Future Scope**

* **Enhanced Sensor Integration:**  
  Future versions may include additional sensors to monitor parameters such as soil pH and nutrient levels, offering a more comprehensive analysis of soil health.
* **Automated Fertilization and Pest Control:**  
  Expanding the system to include automated controls for fertilization and pest management could further optimize agricultural practices.
* **Energy-Efficient Operations:**Integration of renewable energy sources (e.g., solar panels) could improve the sustainability and operational efficiency of the system.
* **Sensor Expansion:**  
  Incorporate additional sensors (e.g., soil pH, nutrient content) to provide a fuller picture of soil health.
* **Advanced Analytics:**  
  Use machine learning algorithms to predict irrigation needs based on weather forecasts and historical data.
* **Automated Fertilization:**  
  Integrate systems to automate fertilization alongside irrigation, ensuring balanced nutrient distribution.
* **Energy Efficiency:**Explore renewable energy options, such as solar panels, to power the system and further reduce operational costs.
* **Cloud Integration:**  
  Enhance data storage and analysis by integrating cloud-based platforms, enabling remote data logging and trend forecasting.

**09. Code of soil moisture**

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <DHT.h>

#define DHTPIN D6 // Pin connected to the DHT11 data pin

#define DHTTYPE DHT11 // Define sensor type (DHT11)

DHT dht(DHTPIN, DHTTYPE);

// Soil moisture sensor connected to A0

#define SOIL\_MOISTURE\_PIN A0

// Define the D4 pin (Relay/Motor Control Pin)

#define MOTOR\_CONTROL\_PIN D4

// Wi-Fi and MQTT Broker Info

const char\* ssid = "vivo";

const char\* password = "123456789";

const char\* mqtt\_server = "dev.coppercloud.in"; // Broker address

const char\* mqtt\_topic = "dhanshreekamble094@gmail.com/soil/project"; // Topic to publish data

const char\* control\_topic = "dhanshreekamble094@gmail.com/soil/motor/project"; // Topic to control motor

WiFiClient espClient;

PubSubClient client(espClient);

void setup() {

// Start Serial communication for debugging

Serial.begin(115200);

delay(10);

// Initialize the motor control pin as output

pinMode(MOTOR\_CONTROL\_PIN, OUTPUT);

digitalWrite(MOTOR\_CONTROL\_PIN, LOW); // Make sure the motor is off initially

// Connect to Wi-Fi

WiFi.begin(ssid, password);

Serial.print("Connecting to WiFi");

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

delay(500);

Serial.print(".");

}

Serial.println("Connected to WiFi");

// Initialize DHT11 sensor

dht.begin();

// Set up MQTT client

client.setServer(mqtt\_server, 1883);

client.setCallback(mqttCallback);

// Subscribe to the control topic

client.subscribe(control\_topic);

}

void mqttCallback(char\* topic, byte\* payload, unsigned int length) {

// Handle incoming MQTT messages

String message = "";

for (unsigned int i = 0; i < length; i++) {

message += (char)payload[i];

}

// Check if the received message is "on" or "off" to control the motor

if (String(topic) == control\_topic) {

if (message == "onMoter") {

Serial.println("Turning motor ON");

digitalWrite(MOTOR\_CONTROL\_PIN, HIGH); // Turn motor ON

} else if (message == "offMoter") {

Serial.println("Turning motor OFF");

digitalWrite(MOTOR\_CONTROL\_PIN, LOW); // Turn motor OFF

}

}

}

void reconnect() {

// Loop until connected to MQTT broker

while (!client.connected()) {

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

if (client.connect("ESP8266Client")) {

Serial.println("connected");

// Once connected, subscribe to the control topic

client.subscribe(control\_topic);

} else {

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

Serial.print(client.state());

Serial.println(" trying again in 5 seconds");

delay(5000);

}

}

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

// Read soil moisture

int soilMoistureValue = analogRead(SOIL\_MOISTURE\_PIN);

// Map the soil moisture value from 0-1023 to 0-100 (percentage)

int soilMoisturePercentage = map(soilMoistureValue, 0, 1023, 0, 100);

// Read temperature and humidity from DHT11

float temperature = dht.readTemperature(); // Temperature in Celsius

float humidity = dht.readHumidity(); // Humidity percentage

// Check if readings failed and try again

if (isnan(temperature) || isnan(humidity)) {

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

return;

}

// Create a JSON string with sensor data

String payload = "{\"soil\_moisture\":" + String(soilMoisturePercentage) +

",\"temperature\":" + String(temperature) +

",\"humidity\":" + String(humidity) + "}";

// Send data to MQTT broker

if (client.publish(mqtt\_topic, payload.c\_str())) {

Serial.println("Data sent to MQTT broker");

} else {

Serial.println("Failed to send data");

}

// Wait for 1 second before sending data again

delay(1000); // Increased delay to give time for reading to be updated

}

**REFERENCE**

* 1. Arduino Official Documentation

https://www.arduino.cc/en/Guide/HomePage

2. ESP8266 NodeMCU Documentation

https://www.espressif.com/en/products/socs/esp8266

3. DHT11 Sensor Datasheet

https://components101.com/sensors/dht11-temperature-sensor

4. Soil Moisture Sensor Guide

https://lastminuteengineers.com/soil-moisture-sensor-arduino-tutorial/

5. Relay Module Working Principle

https://www.electronics-tutorials.ws/blog/relay.html

6. Transistor BC547 Datasheet

https://www.componentsinfo.com/bc547-transistor-pinout-equivalent/

7. IoT in Agriculture: Smart Irrigation System

Research Article: "IoT Based Smart Irrigation System" – International Journal of Engineering Research & Technology (IJERT)

https://www.ijert.org/iot-based-smart-irrigation-system

8. Mobile App Development for IoT Projects

https://www.thingspeak.com/ or https://www.blynk.io/