Smart Plant Watering System

IoT PROJECT REPORT

Abstract

This project report introduces a smart plant watering system for home use, enabling users to remotely care for their plants even when away. The system combines a NodeMCU microcontroller, soil moisture sensor and a weather API to optimize watering schedules based on real-time environmental conditions. By monitoring both soil moisture level and local weather forecasts, the system intelligently adjusts watering needs to conserve water while ensuring plant health. Users can access the system through a mobile application, allowing for flexible, remote management.

1. Introduction

With the rise of smart home technologies, automation has become an integral part of modern life, offering convenience and efficiency in various daily tasks. One area where this transformation is particularly beneficial is in plant care and gardening. Busy lifestyles, frequent travel, and unpredictable weather conditions make it challenging for plant owners to maintain an ideal watering routine, often leading to under-watering or over-watering, which can harm plant health.

This project aims to address these challenges by developing a smart plant watering system for home environments, allowing users to remotely monitor and control the watering of their plants. By incorporating a NodeMCU microcontroller, a soil moisture sensor, and an integrated weather API, the system ensures that plants receive the necessary amount of water without wastage. The soil moisture sensor continuously monitors the soil's hydration level, while the weather API provides real-time information on expected rainfall and temperature, which the system uses to determine when watering is necessary. This approach helps prevent over-watering on rainy days and avoids the need for manual adjustments during weather changes, contributing to water conservation.Users can control the system through a mobile application, enabling remote watering of plants and enhancing flexibility and convenience.

By automating the plant watering process and integrating real-time weather data, this smart watering system offers an innovative solution for plant care in home settings, aiming to make plant maintenance more accessible and sustainable.

2. Objectives

The objective of this project are as follows -

- **Automate Plant Watering**: Design a system that automatically waters plants based on soil moisture levels, reducing the need for manual watering and ensuring consistent plant care.
- **Enable Remote Control**: Provide users with the ability to control the watering system remotely through a mobile application, allowing for flexibility in plant care regardless of the user's location.
- **Optimize Water Usage**: Integrate a weather API to adjust watering schedules based on real-time weather conditions, minimizing water waste by accounting for expected rainfall and temperature changes.

3. Proposed Methodology

Hardware

The hardware components used in our project are -

1. Soil Moisture Sensor - The soil moisture sensor measures the moisture level in the soil. When the moisture level falls below a set threshold (indicating dry soil), it sends a signal to the NodeMCU to trigger the irrigation system. The sensor helps to ensure plants are watered only when necessary, preventing overwatering.

2. NodeMCU ESP8266 - The NodeMCU is the central microcontroller in this system. It is based on the ESP8266 Wi-Fi module, which allows the system to be controlled remotely. The NodeMCU receives data from the soil moisture sensor, processes it, and controls the relay module to turn the water pump on or off. It is also connected to a mobile application for remote monitoring and control.

3. Relay Module - The relay module is used to control the water pump. When the soil moisture sensor detects dry soil, the NodeMCU sends a signal to the relay, which closes the circuit and activates the water pump. The relay acts as a switch, ensuring that the pump operates only when needed and can be controlled remotely.

4. Water Pump - The water pump is responsible for watering the plants by pumping water from a reservoir to the plant's roots. It is controlled by the relay module, which ensures that water is only supplied when required. The pump is powered by the NodeMCU and is essential for automating the watering process.

Software

The softwares used in our project are -

1. Arduino - The Arduino is programmed to read input from the soil moisture sensor and control the relay to activate the water pump. The code is designed to continuously monitor the soil moisture levels. When the moisture level falls below the predefined threshold, the Arduino triggers the relay to turn on the water pump and start the watering process. Once the moisture level reaches the required threshold, the Arduino stops the pump by deactivating the relay. Additionally, the Arduino code is configured to handle any real-time adjustments based on sensor data, ensuring that the plants are watered effectively and efficiently.

2. Blynk App - The Blynk app is integrated with the system to enable remote monitoring and control of the watering process. The app connects to the Arduino via Wi-Fi (using the ESP8266) and provides a user-friendly interface for monitoring soil moisture levels and controlling the water pump.

Circuit Diagram

The diagram is showing how the Soil Moisture Sensor is connected to the NodeMCU, which in turn controls the Relay to switch the Water Pump. The flow of data and power is controlled through these components, ensuring that the plant is watered automatically based on soil moisture levels.

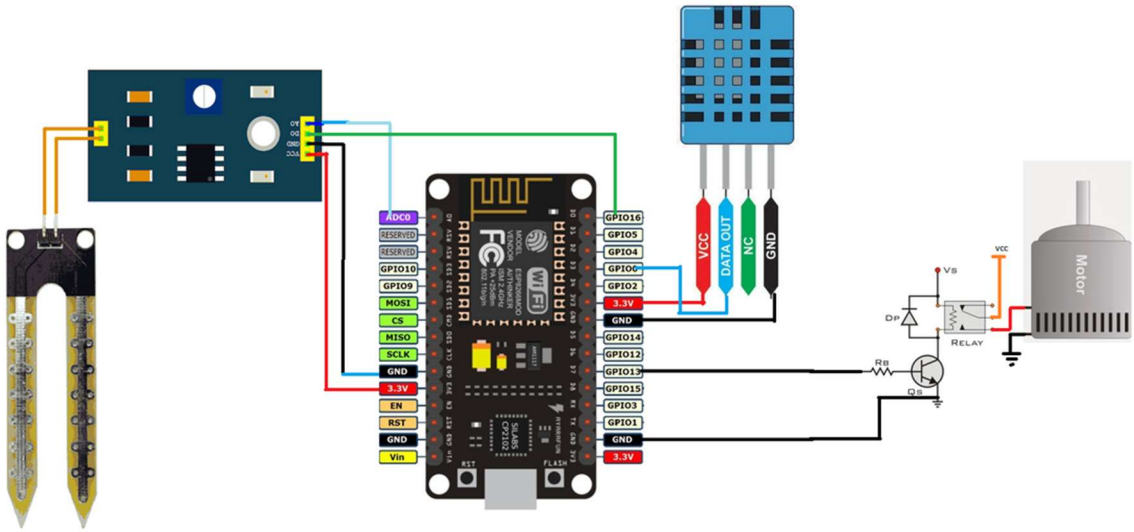

Fig. 1. Circuit Diagram for Smart Plant Watering System

Block Diagram

This block diagram represents a system that collects data from soil moisture sensor, processes this data and weather API data on NodeMCU and uses the Blynk platform to provide remote monitoring and control of a water pump via a relay.

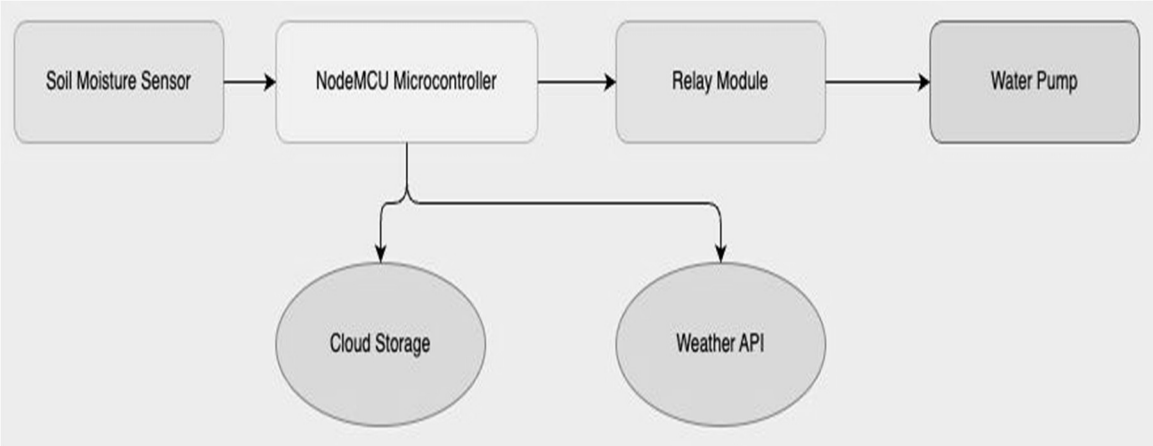

Fig. 2. Block Diagram

4. Results

The smart plant watering system successfully met its design objectives, demonstrating reliable functionality and water conservation:

1. **System Functionality**: The NodeMCU and soil moisture sensor accurately detected moisture levels, triggering watering when needed. The weather API adjusted watering schedules based on real-time weather data, preventing overwatering on rainy days.
2. **Remote Control and User Experience**: The mobile app allowed users to remotely control watering, monitor soil moisture, and access weather forecasts, providing a seamless and flexible user experience.

Prototype Design

This is the actual implementation of our smart plant watering system.

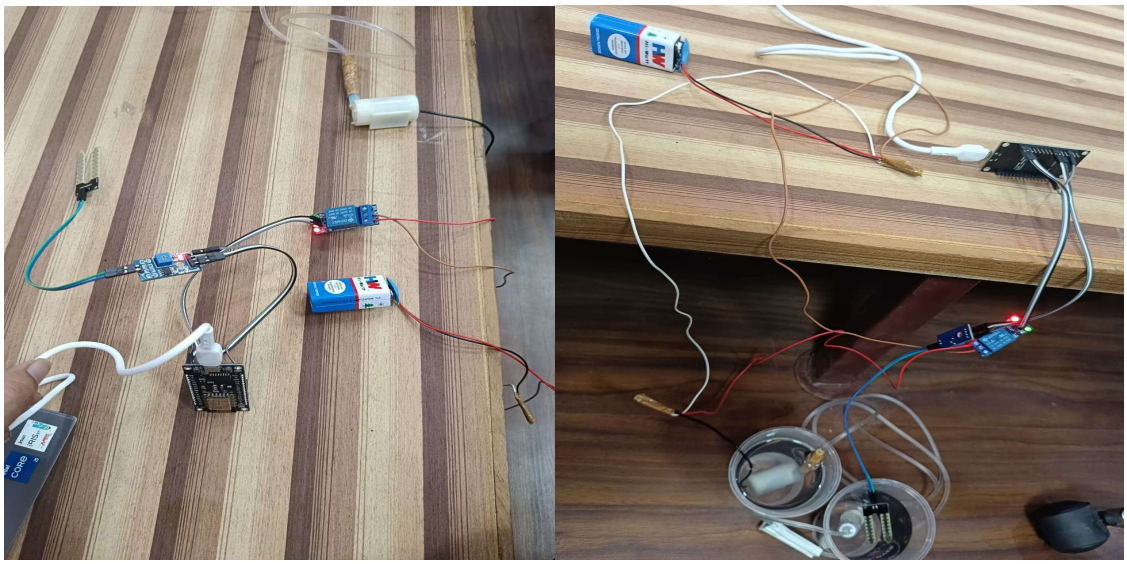

Fig. 3. Prototype Design

Blynk Interface showing results

This is the app that allows users to manually water the plants by switching on the motor remotely or put it in automation mode.

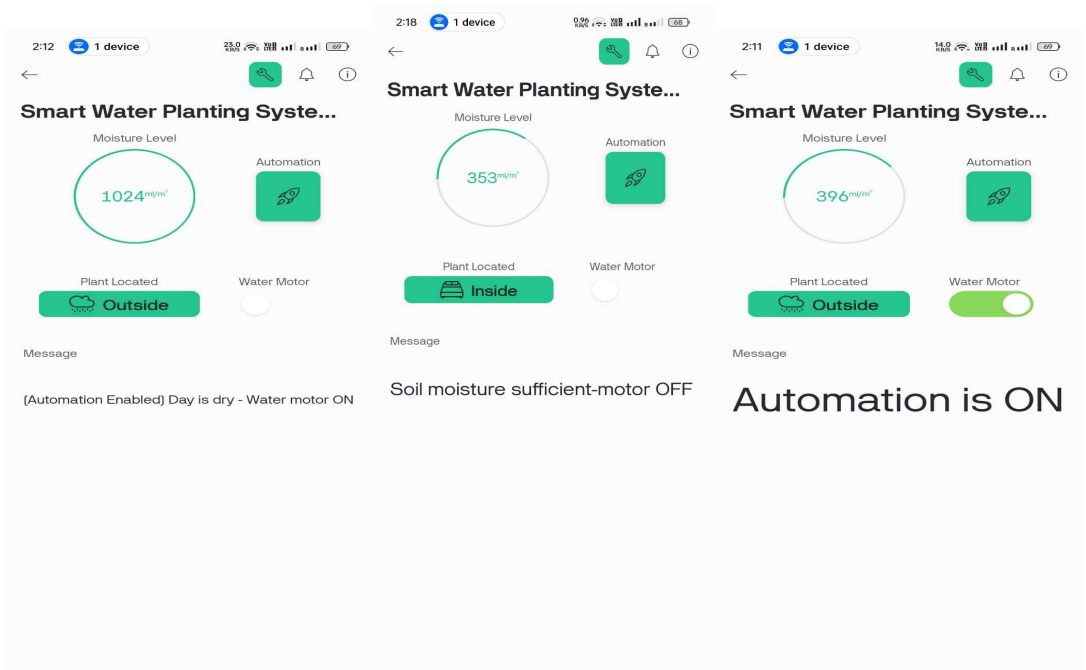

Fig. 4. Showing automation mode in Blynk App

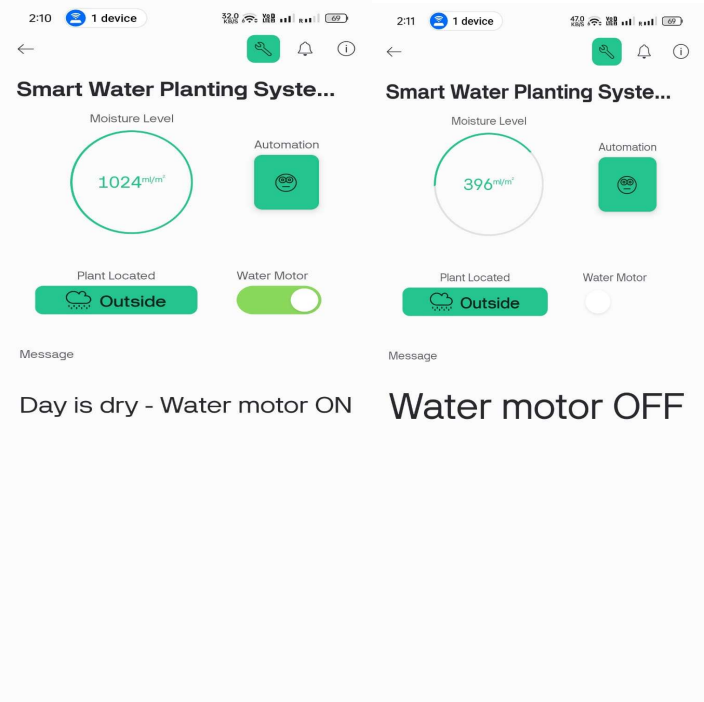

Fig. 5. Showing non-automation mode in Blynk App

In conclusion, the system effectively automated plant care, optimized water use, and provided an efficient remote control solution.

5. Comparison to existing solutions

Existing Smart Irrigation Systems	Prototype Designed
Automation based on soil moisture, but limited weather integration.	Automation based on both soil moisture and weather data, remote control via mobile app.
May save water but doesn't adjust for real-time weather.	Optimizes water use by considering both soil moisture and weather forecasts, reducing water waste.
Often expensive and complex.	Cost-effective, efficient, and accessible for home users.

6. Conclusion

The smart plant watering system effectively automates plant care, optimizes water usage, and provides remote control through a mobile app. By integrating soil moisture sensors and real-time weather data, it ensures efficient irrigation while conserving water.

Compared to traditional methods and existing smart irrigation systems, it offers greater precision, cost-effectiveness, and ease of use.

7. Learning Outcomes

The development and implementation of the smart plant watering system provided valuable insights and skills, including:

1. **System Design and Integration**: Gained experience in designing and integrating hardware components (e.g., NodeMCU, soil moisture sensors, and water pumps) with software systems, including the integration of real-time weather data for optimized decision-making.
2. **Mobile Application Development**: Learned to develop and implement a user-friendly mobile application for remote control, enhancing user experience and enabling real-time monitoring.
3. **Water Conservation Techniques**: Gained knowledge in optimizing irrigation practices by integrating environmental data (e.g., weather forecasts) and using automation to conserve water.
4. **Project Management**: Acquired skills in managing a multidisciplinary project, balancing hardware and software requirements, and delivering a working system that met all objectives.

8. References

1. Arduino documentation for NodeMCU setup and configuration.
2. OpenWeather API documentation for accessing weather predictions.
3. Technical manuals for soil moisture sensor and relay module specifications.

9. Source Code

Here is the Arduino source code of our project -

```
#define BLYNK_TEMPLATE_ID "TemplateID"

#define BLYNK_TEMPLATE_NAME "Smart Water Planting System"



char auth[] = "blynk_auth_token"; // the auth code that you got on your gmail

char ssid[] = "wifi_ssid"; // username or ssid of your WI-FI

char pass[] = "wifi_password"; // password of your Wi-Fi
```

```
#define relay D2


#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <ESP8266HTTPClient.h>
#include <ArduinoJson.h>


bool automationEnabled = false;
bool plantLocationSet = false;
int soilMoistureLevel = 0;
int moistureThreshold = 600;
unsigned long lastWateringTime = 0;
bool waterMotorState = false;
bool motor = false;


WiFiClient wifiClient;


String apiKey = "Weather_API_KEY";//Weather API Key
String city = "Warangal";
String units = "metric";
String serverPath = "http://api.openweathermap.org/data/2.5/weather?q=" + city + "&appid=" + apiKey + "&units=" + units;


// Returns false if rainy; true if dry day
bool checkWeatherData() {
 // Only fetch weather data if connected to Wi-Fi
 if (WiFi.status() == WL_CONNECTED) {
```

```
HTTPClient http;

// Make the HTTP GET request
http.begin(wifiClient, serverPath);  // Specify the URL
int httpCode = http.GET();           // Send the request

// Check if the request was successful
if (httpCode > 0) {
  String payload = http.getString();  // Get the response payload
  Serial.println("Weather Data: " + payload);  // Print raw JSON data

  // Parse the JSON data
  DynamicJsonDocument doc(1024);
  deserializeJson(doc, payload);

  // Extract relevant data
  float temperature = doc["main"]["temp"];
  float humidity = doc["main"]["humidity"];
  String weatherDescription = doc["weather"][0]["description"];
  int weatherID = doc["weather"][0]["id"];  // Weather condition ID

  // Print weather data
  Serial.println("Temperature: " + String(temperature) + " °C");
  Serial.println("Humidity: " + String(humidity) + " %");
  Serial.println("Weather: " + weatherDescription);

  // Check if the weather condition indicates rain (Weather ID between 200 and 531)
  if (weatherID >= 200 && weatherID <= 531) {
```

```
      Serial.println("Day is rainy no need to give water today");
      return false;
    } else {
      Serial.println("Day is dry");
    }
  } else {
    Serial.println("Error on HTTP request");
  }
  http.end();
 } else {
  Serial.println("Wi-Fi not connected");
 }
 return true;
}

//send message to app
void sendMessage(const String& message) {
 Blynk.virtualWrite(V0, message);
 Serial.println(message);
}

void readSoilMoisture() {
 soilMoistureLevel = analogRead(A0);
 delay(500);
}

void readAndSendSoilMoisture() {
 soilMoistureLevel = analogRead(A0);
```

```
 Blynk.virtualWrite(V1, soilMoistureLevel);
 Serial.println(soilMoistureLevel);
}

void giveWater() {
 if (soilMoistureLevel > moistureThreshold) {
   if (checkWeatherData()) {
     giveWaterDirectly();
   } else {
     sendMessage("Day is rainy - no need to water");
   }
 } else {
   sendMessage("Soil moisture sufficient - motor OFF");
 }
}

void giveWaterDirectly() {

 if (soilMoistureLevel > moistureThreshold) { // Adjust threshold based on calibration
   int cnt = 0;
   if(automationEnabled) {
     sendMessage("(Automation Enabled) Water motor ON");
     Blynk.logEvent("(Automation Enabled) Water motor ON");
   }
   else {
     sendMessage("Water motor ON");
   }
   while(soilMoistureLevel > moistureThreshold && cnt<5) {
```

```
    readAndSendSoilMoisture();
    digitalWrite(relay, LOW); // Turn on the water motor
    waterMotorState = true;
    delay(1000);
    cnt++;
   }
   digitalWrite(relay, HIGH); // Turn off the water motor
   waterMotorState = false;
   sendMessage("Water motor OFF");
 } else {
   sendMessage("Soil moisture sufficient-motor OFF");
 }
}

void checkPeriodicWatering() {
 unsigned long currentTime = millis();
 // Check if 10 minutes (600000 ms) have passed since last watering
 if (automationEnabled && (lastWateringTime == 0 || currentTime - lastWateringTime >=
60000)) {
   lastWateringTime = currentTime;
   if (plantLocationSet) {
    giveWaterDirectly();
   } else {
    giveWater();
   }
 }
}

BLYNK_WRITE(V2) {
```

```
 automationEnabled = param.asInt();
 Serial.print("Automation: ");
 Serial.println(automationEnabled ? "Enabled" : "Disabled");
}

BLYNK_WRITE(V3) {
 plantLocationSet = param.asInt();
 Serial.print("Plant Location: ");
 Serial.println(plantLocationSet ? "Indoor" : "Outdoor");
}

BLYNK_WRITE(V4) {
 bool motor = param.asInt();
 if (motor && automationEnabled) {
   sendMessage("Automation is ON");
 } else if (motor) {
   if (plantLocationSet) {
     giveWaterDirectly();
   } else {
     giveWater();
   }
 }
}

void setup() {
 Serial.begin(115200);
 pinMode (A0,INPUT);
 pinMode(relay, OUTPUT);
```

```
 digitalWrite(relay, HIGH); // Default OFF
 Blynk.begin(auth, ssid, pass, "blynk.cloud", 8080);
 while (!Blynk.connected()) {
   delay(100);
   Serial.print(".");
 }
 Serial.println("\nConnected to Blynk server!");
}
void loop() {
 if (Blynk.connected()) {
   Blynk.run();
 } else {
   Serial.println("Blynk connection lost, retrying...");
   Blynk.connect();
 }
 readSoilMoisture();
 if (automationEnabled) {
   checkPeriodicWatering();
 }
}
```