## **Smart Irrigation**

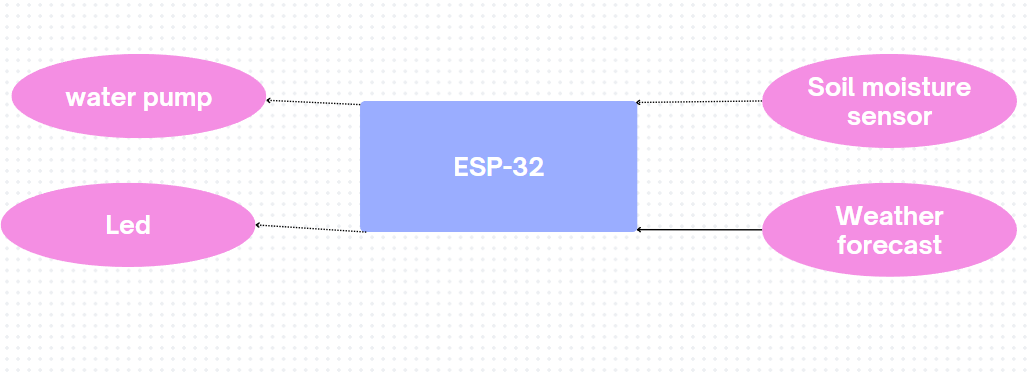
## Description:

This smart irrigation system uses an ESP32 to automate plant watering efficiently. It measures soil moisture with a sensor and checks real-time weather forecasts from Open-Meteo for rain predictions. If the soil is very dry, it waters the plants immediately; if moderately dry, it checks if rain is expected before watering to avoid waste. A relay controls the water pump, and an LED indicates when the pump is active. This setup conserves water, reduces manual work, and ensures optimal soil moisture based on both local conditions and upcoming weather.

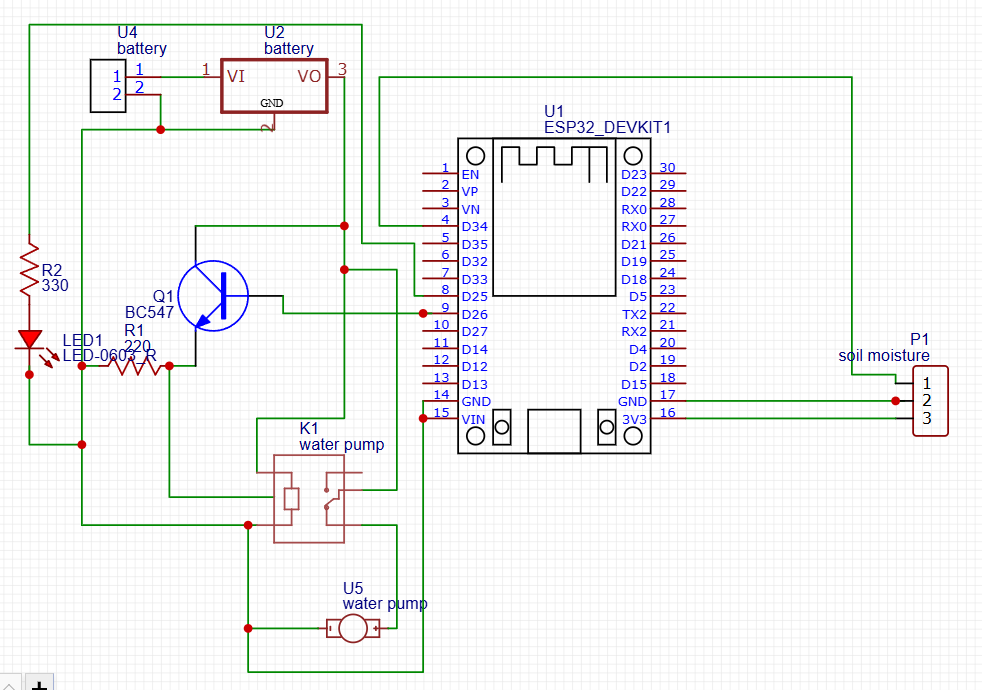
### Materials Used:

|  |  |  |
| --- | --- | --- |
| **S. No** | **Component** | **Description** |
| 1 | esp32 | The microcontroller that processes all the sensors output and then controls the rest |
| 2 | Micro waterpump | A pump used to on and off the flow of water |
| 3 | Soil sensor | A sensor which is used to measure soil moisture |
| 4 | Rocker Switch | A simple switch to turn ON or OFF the system |
| 5 | Power Sorce | A power supply to provide the necessary voltage to the system. |
| 6 | Relay | To provide enough voltage for the pump to function |

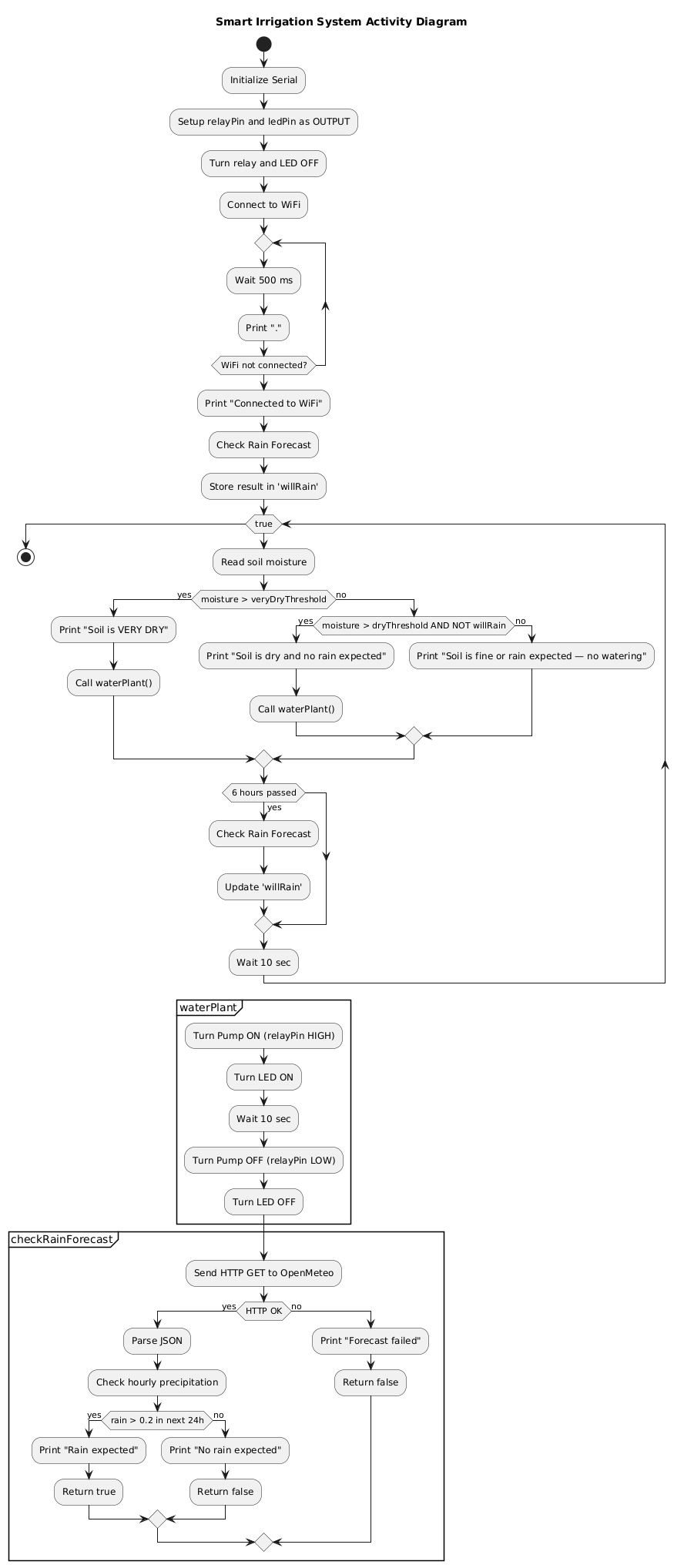
### **Block Diagram:**



### **Circuit Diagram:**



**Flow Chart:**



## **Introduction to the Project:**

The Smart Irrigation System using an ESP32 is designed to automate garden or farm watering by monitoring soil moisture levels and checking the weather forecast for rain. The system reads data from a soil moisture sensor to determine how dry the soil is. It connects to Wi-Fi to fetch rain forecast data periodically. Based on these inputs, it automatically turns a water pump (through a relay) ON or OFF and indicates watering status with an LED. This ensures plants get enough water without wasting it, even when the user is not around.

## **Soil Moisture Sensor Role:**

The soil moisture sensor continuously measures the moisture content in the soil. When the soil is very dry, the sensor sends low moisture values to the ESP32. Depending on how dry it is, the system decides to turn on the water pump to irrigate the plants. This keeps plants healthy by ensuring the roots get enough water.

## **ESP32’s Role:**

The ESP32 acts as the brain of the system. It connects to Wi-Fi, checks online weather forecast data to predict rain, reads soil moisture levels, and controls the relay and LED accordingly. It ensures that the pump runs only when needed and avoids watering if rain is expected soon, saving water. The ESP32 repeats this process automatically at regular intervals without needing manual input.

## **Relay and Pump Control:**

The relay module works like an electronic switch. When the ESP32 decides to water the plants, it energizes the relay, which turns ON the water pump. When watering is complete or unnecessary, the relay is de-energized, and the pump turns OFF. An LED is also turned ON while watering to visually indicate the pump status.

## **Rain Forecast Checking:**

To make smart watering decisions, the ESP32 periodically connects to an online weather service (e.g., Open Meteo API) to check if it will rain in the next few hours. If rain is expected, the system avoids unnecessary watering even if the soil is slightly dry, preventing overwatering and conserving water. The rain forecast is re-checked every few hours to keep the decision updated.

## **Automatic Watering Logic:**

The system follows simple logic:

* If the soil is **very dry**, it waters immediately, regardless of forecast.
* If the soil is moderately dry and **no rain is expected**, it waters.
* If the soil is moist enough or rain is expected soon, it does not water.  
   This ensures plants get enough water without waste.

## **Manual Monitoring and Debugging:**

Through the serial monitor, users can see real-time messages showing soil moisture readings, whether the pump is ON or OFF, and rain forecast status. This makes it easy to test and monitor the system.

## **Conclusion:**

This project automates irrigation intelligently using real-time soil data and weather predictions. It saves water, reduces the need for manual watering, and promotes healthy plant growth. It demonstrates an affordable IoT-based smart irrigation solution using ESP32, suitable for home gardens, small farms, or greenhouse setups.

### **Code:**

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

// WiFi credentials

const char\* ssid = "POCO X6 Pro 5G\_PeCSNsn\_MI"; // <- Replace with your WiFi SSID

const char\* password = "12345678"; // <- Replace with your WiFi password

// Location: Whitefield, Bangalore

const String latitude = "12.9698";

const String longitude = "77.7499";

const String openMeteoURL = "<https://api.open-meteo.com/v1/forecast?latitude=>" + latitude + "&longitude=" + longitude + "&hourly=precipitation&forecast\_days=1";

// Pins and thresholds

const int soilPin = 34; // Analog pin for soil sensor (ADC1)

const int relayPin = 26; // Relay control pin

const int ledPin = 25; // LED pin to indicate pump status

// Moisture threshold (0 = wet, 4095 = dry)

const int veryDryThreshold = 3200;

const int dryThreshold = 2500;

bool willRain = false;

void setup() {

Serial.begin(9600);

// Pin setup

pinMode(relayPin, OUTPUT);

pinMode(ledPin, OUTPUT);

digitalWrite(relayPin, LOW); // Pump OFF with active-HIGH relay

digitalWrite(ledPin, LOW); // LED OFF

// Connect to WiFi

WiFi.begin(ssid, password);

Serial.print("Connecting to WiFi");

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

delay(500);

Serial.print(".");

}

Serial.println("\nConnected to WiFi");

// Initial forecast check

willRain = checkRainForecast();

}

void loop() {

int moisture = analogRead(soilPin);

Serial.print("Soil Moisture Reading: ");

Serial.println(moisture);

// Watering logic

if (moisture > veryDryThreshold) {

Serial.println("Soil is VERY DRY — watering immediately.");

waterPlant();

} else if (moisture > dryThreshold && !willRain) {

Serial.println("Soil is dry and no rain expected — watering.");

waterPlant();

} else {

Serial.println("Soil is fine or rain expected — no watering.");

}

// Update weather forecast every 6 hours

static unsigned long lastCheck = 0;

if (millis() - lastCheck > 6UL \* 60 \* 60 \* 1000) {

willRain = checkRainForecast();

lastCheck = millis();

}

delay(10000); // Check soil every 1 minute

}

void waterPlant() {

digitalWrite(relayPin, HIGH); // Turn pump ON (active HIGH)

digitalWrite(ledPin, HIGH); // LED ON

delay(10000); // Water for 10 seconds

digitalWrite(relayPin, LOW); // Turn pump OFF

digitalWrite(ledPin, LOW); // LED OFF

}

bool checkRainForecast() {

HTTPClient http;

http.begin(openMeteoURL);

int httpCode = http.GET();

if (httpCode > 0) {

String payload = http.getString();

DynamicJsonDocument doc(16 \* 1024);

DeserializationError error = deserializeJson(doc, payload);

if (error) {

Serial.println("JSON parsing failed");

return false;

}

JsonArray precip = doc["hourly"]["precipitation"];

for (int i = 0; i < 24; i++) {

float rain = precip[i];

if (rain > 0.2) {

Serial.println("Rain expected in the next 24 hours.");

return true;

}

}

Serial.println("No rain expected in the next 24 hours.");

return false;

} else {

Serial.println("Failed to fetch weather forecast.");

return false; // Default to no rain if request fails

}

http.end();

}

### **Future Scope:**

### **1. Automatic Mode Enhancements**

* **Dynamic watering duration:** Adjust pump ON time based on how dry the soil is. Very dry → longer watering; slightly dry → shorter watering.
* **Multiple soil sensors:** If you have a large garden, use multiple sensors for different zones.
* **Multiple relays:** Control more than one pump/valve for multiple sections.

### **2. Real-time Control & Monitoring**

* **Blynk / MQTT / Web Dashboard:** Add a mobile app or web dashboard to:
  + Monitor soil moisture in real-time
  + See rain forecast info

### **3. Energy Efficiency**

* **Deep Sleep:** Put the ESP32 into deep sleep between readings to save power (good for battery/solar setups).
* **Adaptive update rate:** Increase check frequency when soil is dry, decrease when conditions are good.

### **4. Better Weather API**

* **Use more weather data:** Check humidity, temperature, and wind speed to refine watering decisions.
* **Offline fallback:** If weather API fails, use last known data or default rules.

### **5. Safety & Robustness**

* **Water level detection:** Use a float sensor to check water tank level to avoid dry running the pump.
* **Pump health monitoring:** Add current sensing to detect pump malfunction.
* **Watchdog timer:** Reset ESP32 automatically if it hangs.

### **6. Data Logging**

* **Store history:** Save moisture readings and watering events to SD card or send to cloud (Firebase, Google Sheets, InfluxDB).
* **Graph trends:** Visualize moisture trends and forecast vs. actual rainfall.