© July 2025| IJIRT | Volume 12 Issue 2 | ISSN: 2349-6002

IoT-Based Smart Irrigation System for Efficient Water Management

Akshay Pawar, Trapti Harinkhere, Ruchi Thakre, Ashvin Kalambe, Dr.Rajesh Nigam (Guide)

Abstract- Water scarcity is a growing global concern, especially in agriculture. Traditional irrigation methods lead to excessive water use, harming crop yield and depleting resources. This paper presents a smart irrigation system using IoT technology, which integrates sensors, microcontrollers (ESP32), and cloud platforms to automate and optimize watering based on real-time soil moisture data. The system features remote monitoring and control via mobile or web dashboards, reducing manual intervention and water waste. We also review existing literature on smart irrigation systems, compare different approaches, and provide detailed circuit diagrams, sample code, and pictures. The proposed design demonstrates improved water efficiency, scalability, and adaptability.

INTRODUCTION

Water conservation in agriculture is essential as traditional irrigation systems often rely on fixed schedules rather than real-time soil conditions. IoT-based smart irrigation addresses this inefficiency by monitoring environmental data and automating the watering process. The project described herein utilizes an ESP32 microcontroller, a capacitive soil moisture sensor, a relay module, and a DC water pump. The system integrates with cloud services (e.g., Blynk or ThingSpeak) for real-time data visualization and control, allowing users to adjust watering parameters remotely.

LITERATURE REVIEW

We examined seven key research papers on smart irrigation systems using IoT, focusing on architecture, communication protocols, and sensor integration. The following table summarizes the comparison:

No.	Paper Title	Platform Used	Communication	Sensors Used	Control Logic	Mobile Integration
1	IoT Based Smart Irrigation System	ESP8266 + Blynk	Wi-Fi	Soil moisture, DHT11	Threshold-based	Yes
2	Wireless Sensor Network for Irrigation	ZigBee	ZigBee	Soil moisture	Rule-based	No
3	GSM Based Smart Irrigation	Arduino + GSM	GSM	Moisture, Temp	Manual + Auto	SMS
4	Cloud Based Agri-IoT	ESP32 + AWS	MQTT	Soil, Rain	AI-enabled	Yes
5	Bluetooth Based Irrigation System	Arduino + Bluetooth	Bluetooth	Moisture	Manual	Yes (BT)
6	Real-Time Irrigation Using ML	Raspberry Pi	Wi-Fi	Soil, Temp	ML Model	Yes
7	Sensor-based Water Pumping System	Arduino UNO	Wi-Fi	Soil	Timer/Threshold	No

Key Insights:

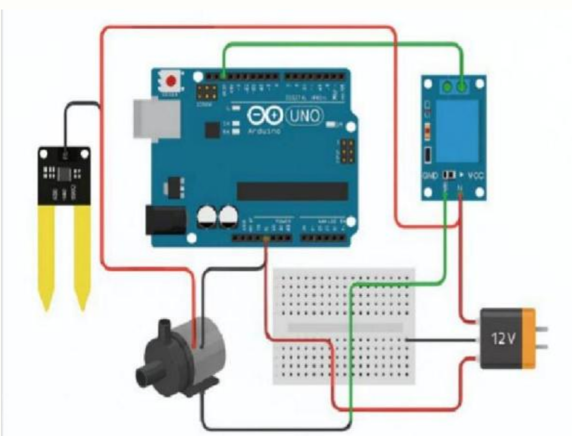
- Wi-Fi-based systems with mobile integration (Papers 1, 4) are most accessible.
- Paper 6 suggests potential in ML-based prediction models.
- Paper 3's GSM model works well in areas with poor Wi-Fi but lacks real-time control.
- ESP32 (Paper 4) is increasingly popular for its built-in Wi-Fi and multiple ADC pins.

Methodology System Architecture The system uses:

- ESP32 Microcontroller: Handles data processing and internet connectivity.
- Soil Moisture Sensor: Detects the water content in soil.
- Relay Module: Controls the DC water pump based on logic.
- Blynk Cloud: Displays sensor data and allows remote pump control.
- Mobile/Web Dashboard: Visualizes data and issues manual commands.

ARCHITECTURE DIAGRAM:

Circuit Diagram

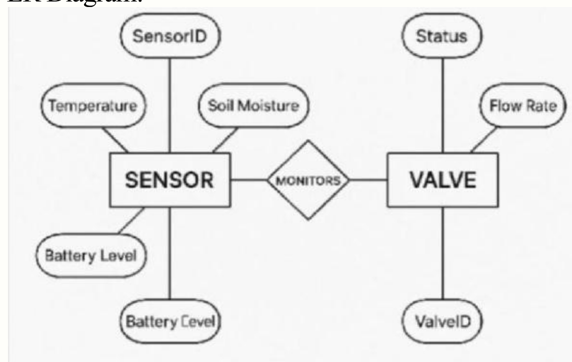


Hardware Connections:

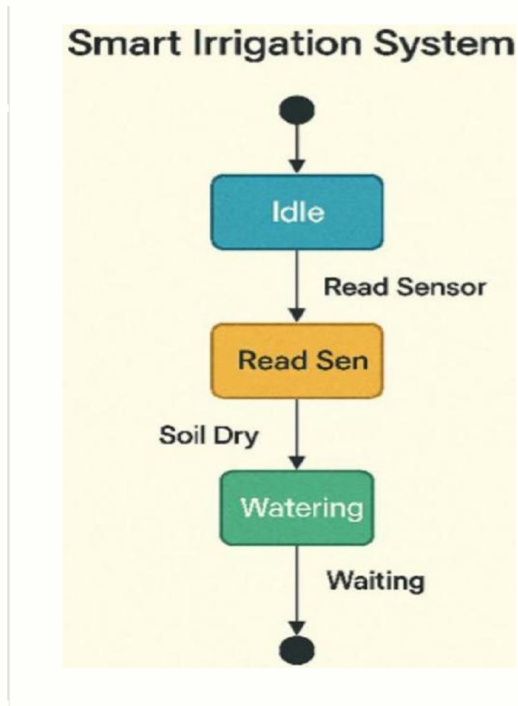
Connections:

- Soil Moisture Sensor: VCC → 3.3V, GND → GND, SIG → GPIO34
- Relay Module: IN → GPIO23, VCC → 5V, GND → GND
- Pump Power: Controlled via Relay switching

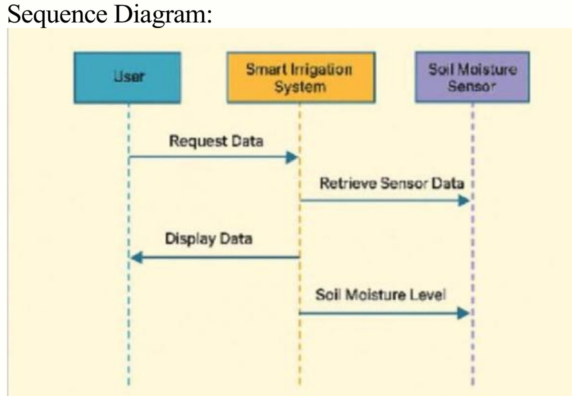
ER Diagram:



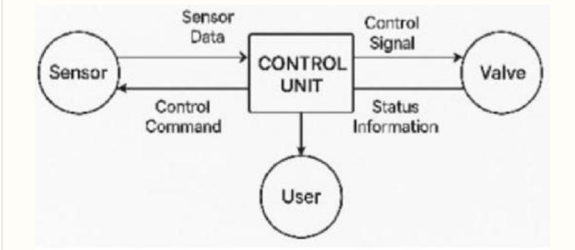
State Chart Diagram:



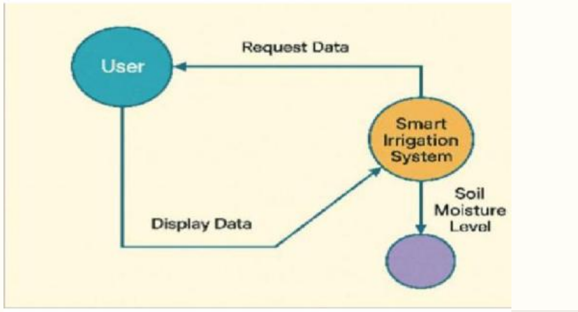
Sequence Diagram:



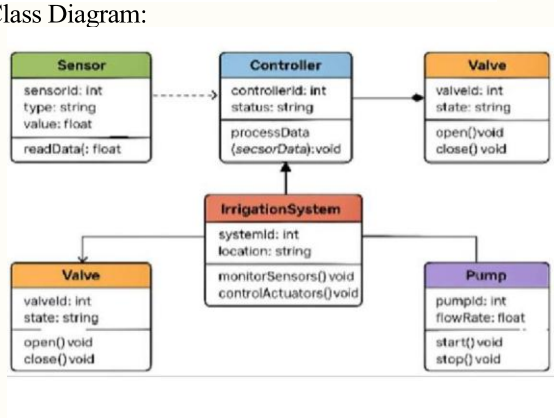
Data Flow Diagram:



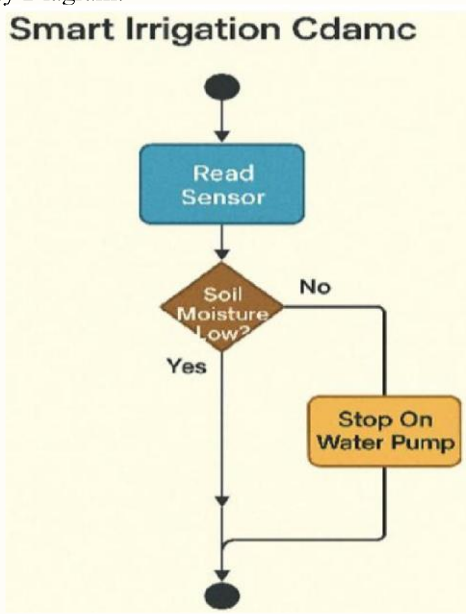
Collaboration Diagram:

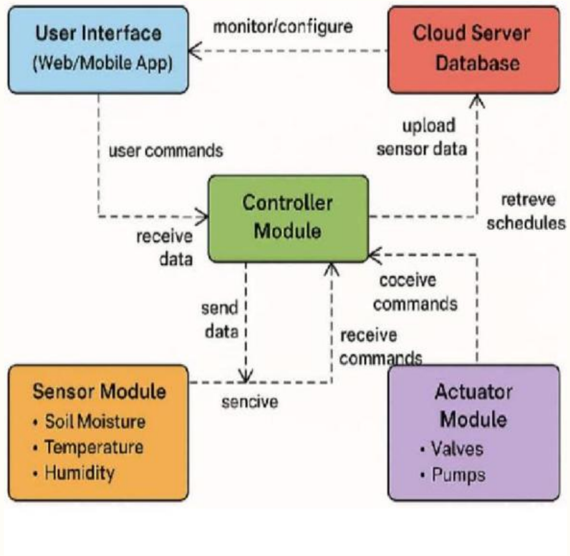


Class Diagram:

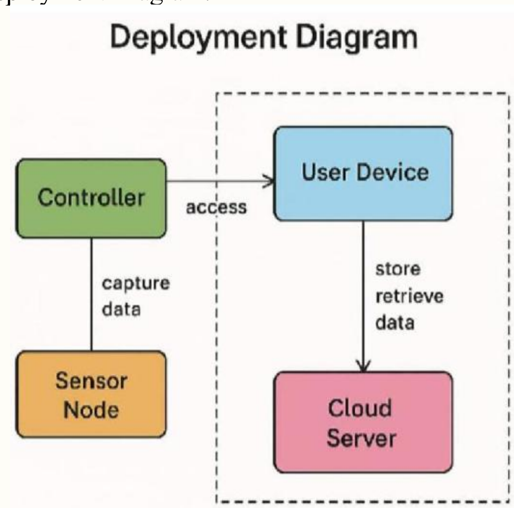


Activity Diagram:

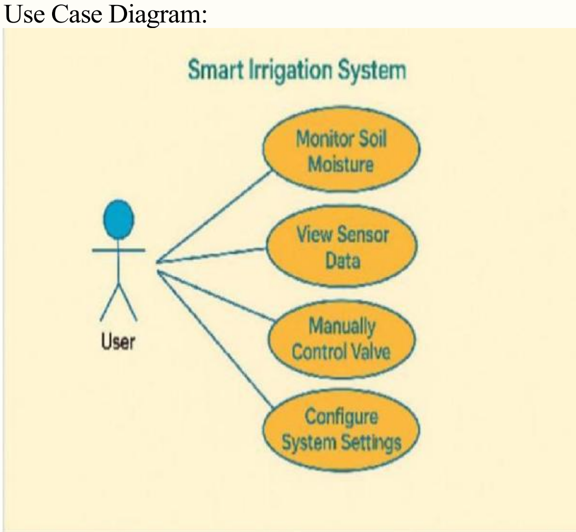




Deployment Diagram:



Use Case Diagram:



Component Diagram:

Code (ESP32 - Arduino IDE)

```
#include <WiFi.h> #include <SPI.h> #include <Wire.h>
#include <Adafruit_GFX.h> #include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128 #define SCREEN_HEIGHT
32 #define OLED_RESET 4 #define SCREEN_ADDRESS
0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
const char* ssid = "IOT_Irrigation";
const char* password = "IOT@12345";
WiFiServer server(80); String header;
const int ledPin16 = 26; const int potPin = 34; int potValue = 0;
unsigned long currentTime = millis(); unsigned long previousTime = 0;
const long timeoutTime = 2000;
// Moisture thresholds
const int thresholdLow = 30;    // Auto ON if below this const int
thresholdHigh = 60;             // Auto OFF if above this
bool manualMode = false;        // Track if user manually controlled bool
pumpState = false;              // Track pump status (true = ON)
void setup() { Serial.begin(115200);
  pinMode(ledPin16, OUTPUT); digitalWrite(ledPin16, LOW);
  WiFi.softAP(ssid, password); Serial.println("IP address:");
   Serial.println(WiFi.softAPIP());
   if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed")); while (true);
    }
   delay(200); display.clearDisplay(); display.setTextSize(1);
   display.setTextColor(SSD1306_WHITE); display.setCursor(0, 10);
   display.println("IP address: "); display.setCursor(0, 20);
   display.println(WiFi.softAPIP()); display.display();
   delay(5000);
   server.begin();
}
void loop() {
   // Read moisture
  potValue = analogRead(potPin);
  potValue = map(potValue, 1400, 4095, 100, 0);
   potValue = constrain(potValue, 0, 100); Serial.print("Moisture: ");
   Serial.println(potValue);
   // Auto control logic if (!manualMode) {
     if (potValue < thresholdLow) { digitalWrite(ledPin16, HIGH); pumpState =
       true;
     } else if (potValue > thresholdHigh) { digitalWrite(ledPin16, LOW);
       pumpState = false;
     }
   }
   // OLED display display.clearDisplay(); display.setTextSize(1);
   display.setTextColor(SSD1306_WHITE); display.setCursor(0, 0);
```

```
  display.print("Moisture: "); display.print(potValue); display.println(" %");
  display.setCursor(0, 15);
  if (manualMode) {
    display.println(pumpState ? "Manual ON" : "Manual OFF");
  } else {
    if (potValue < thresholdLow) { display.println("Auto ON: Dry soil");
    } else if (potValue > thresholdHigh) { display.println("Auto OFF: Soil
      wet");
    } else {
      display.println("In Range: Pump stable");
    }
  }
  display.display();
  // Web client handling
  WiFiClient client = server.available(); if (client) {
    currentTime = millis(); previousTime = currentTime; Serial.println("New
    Client."); String currentLine = "";
    while (client.connected() && currentTime - previousTime <= timeoutTime)
{
      currentTime = millis(); if (client.available()) {
         char c = client.read(); Serial.write(c);
        header += c;
         if (c == '\n') {
           if (currentLine.length() == 0) { client.println("HTTP/1.1 200 OK");
             client.println("Content-type:text/html");
             client.println("Connection: close"); client.println();
             // Handle web control
             if (header.indexOf("GET /16/on") >= 0) { manualMode = true;
               pumpState = true; digitalWrite(ledPin16, HIGH);
               Serial.println("Manual ON");
             } else if (header.indexOf("GET /16/off") >= 0) { manualMode =
                true;
                pumpState = false; digitalWrite(ledPin16, LOW);
                Serial.println("Manual OFF");
             } else if (header.indexOf("GET /auto") >= 0) { manualMode = false;
                Serial.println("Switched to AUTO mode");
             }
             // HTML page
             client.println("<!DOCTYPE html><html><head><meta
name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
             client.println("<style>html { font-family: monospace; text-
align: center;} .button { background-color: green; border: none; color:
white; padding: 16px 40px; font-size: 20px; margin: 2px; cursor: pointer;}
.button2 {background-color: gray;}</style></head><body><h2>IOT
Irrigation</h2>");
             client.println("<p>Moisture: " + String(potValue) + "%</p>");
      client.println("<p>Mode: " + String(manualMode ? "Manual" :
      "Auto") + "</p>");
             client.println("<p>Pump: " + String(pumpState ? "ON" : "OFF") +
```