Smart water System

Phase 2: innovation

Definition: Smart water system allow the collection, treatment, distribution and recycling of water. This Smart Irrigation System, often deployed underground ,can leak ,freeze ,or breakdown .This systems are widely deployed on infrastructures now a days

Project: Smart Irrigation System

Objective: To efficiently manage and conserve water resources for agricultural irrigation.

Components and Working:

Sensors: Soil moisture sensors are placed in the field to measure soil moisture levels at various locations.

Microcontroller: An Arduino or Raspberry Pi is used to interface with the soil moisture sensors and collect data.

Communication Module: A Wi-Fi module is used to transmit sensor data to a central control unit.

Central Control Unit: A cloud-based platform or a local server collects and processes the soil moisture data.

User Interface: A mobile app or web application allows farmers to monitor soil moisture levels and control irrigation remotely.

Control Actions: Based on the soil moisture data, the system can automatically turn on or off irrigation systems, such as sprinklers or drip lines.

Alerts and Notifications: The system sends alerts to the farmer's mobile app when soil moisture levels drop below a certain threshold or when irrigation is initiated or completed.

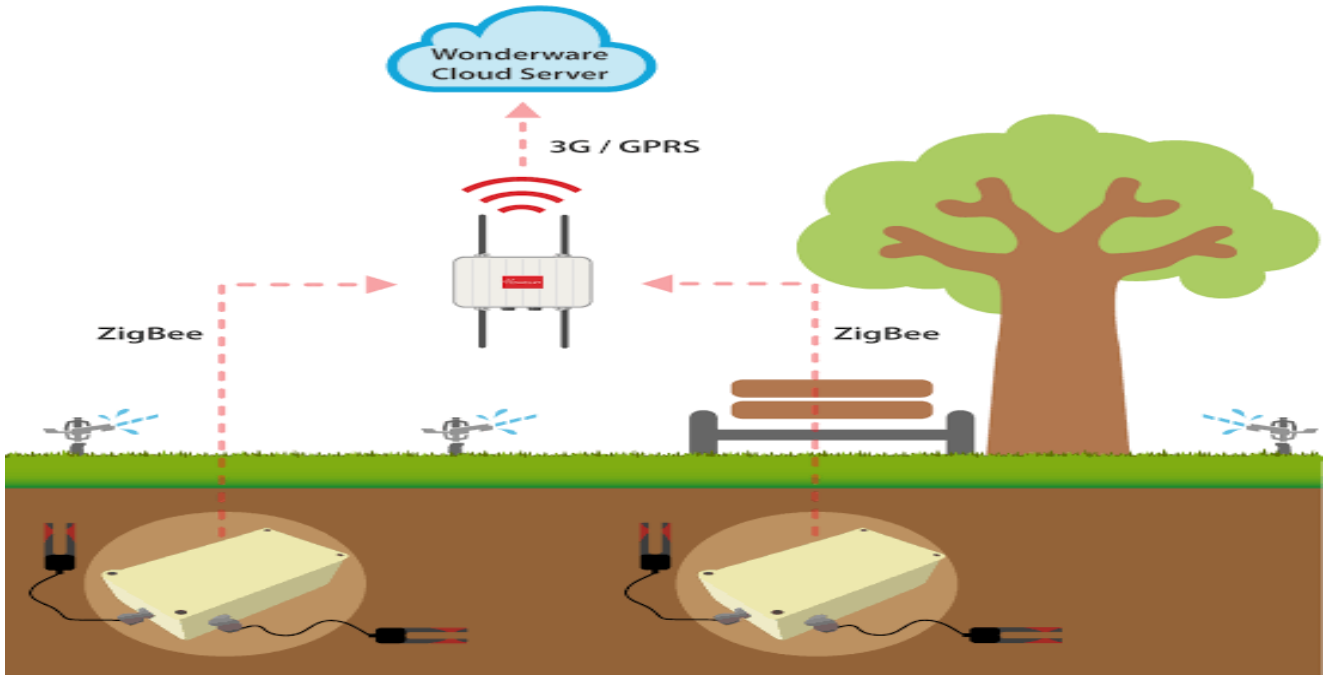
Analytics and Optimization: The system uses historical data to optimize irrigation schedules, ensuring that crops receive the right amount of water, thus conserving water resources.

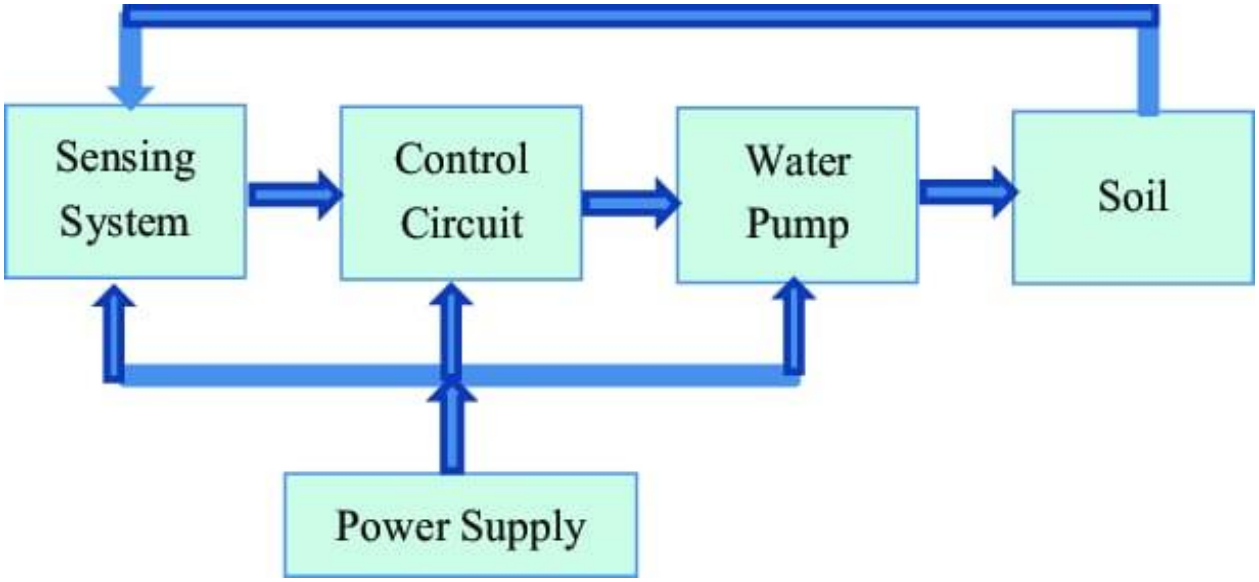
Security: The system is secured with user authentication to prevent unauthorized access.

How It Works:

- Soil moisture sensors continuously measure soil moisture levels in the field.
- The data is sent to the central control unit via Wi-Fi.
- Farmers can monitor soil moisture levels in real-time through the mobile app.
- The system analyzes data and triggers irrigation when necessary.
- Farmers receive notifications about irrigation activities and can manually control irrigation through the app.
- Over time, the system learns optimal watering schedules based on historical data, reducing water wastage

Block diagram:





Source code:

```
 #include <ESP8266WiFi.h>
#include <WiFiClient.h>

// Wi-Fi credentials
const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";

// Server details
const char* serverAddress = "your_server_ip_or_domain";
const int serverPort = 80;  // HTTP port

// Water level sensor
const int sensorPin = A0;
```

```
// Pump control
const int relayPin = 2;

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Initialize the pump control pin
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);

  // Initialize sensor
  pinMode(sensorPin, INPUT);
```

```
}

void loop() {
  // Read water level sensor data
  int waterLevel = analogRead(sensorPin);

  // Send data to server (you can replace this with your actual data format)
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    if (client.connect(serverAddress, serverPort)) {
      client.print("GET /update?water_level=");
      client.print(waterLevel);
      client.println(" HTTP/1.1");
      client.println("Host: your_server_ip_or_domain");
      client.println("Connection: close");
      client.println();
      client.stop();
    }
  }
```

```
  // Check water level and control the pump (adjust threshold
as needed)
  if (waterLevel < 500) {
    digitalWrite(relayPin, HIGH);  // Turn on the pump


    digitalWrite(relayPin, LOW);   // Turn off the pump
  }


  delay(60000);  // Delay for 1 minute before next reading
}
```