SMART IRRIGATION SYSTEM

OBJECTIVES:-

- Develop a cost-effective and efficient smart irrigation system leveraging STM32F401CCU6 microcontroller technology, aimed at optimizing water usage in agriculture while promoting sustainable farming practices.
- Integrate advanced sensor technologies for real-time monitoring of soil moisture, temperature, and humidity, enabling precise irrigation adjustments tailored to specific plant needs and environmental conditions.
- Implement automated watering algorithms based on sensor data analysis and predefined thresholds, reducing manual intervention and labour requirements while ensuring optimal plant growth and yield.
- Design an intuitive user interface with LED feedback indicators for easy system monitoring and operation, enhancing accessibility and usability for farmers and agricultural workers.

ABSTRACT:-

In the face of growing environmental concerns and the urgent need for sustainable agricultural practices, the development of efficient irrigation systems has become a critical necessity. This project introduces a Smart Irrigation System designed to optimize water usage and enhance plant growth by leveraging advanced sensor technology and microcontroller programming. At the heart of the system lies the STM32F401CCU6 microcontroller, chosen for its reliability, efficiency, and performance in handling real-time data processing.

The system utilizes soil moisture sensors (HW080 and HW103) and a DHT11 temperature and humidity sensor to monitor environmental conditions. These sensors provide precise, real-time data, enabling the system to make informed decisions about the necessity of irrigation. A key feature of this smart system is its ability to autonomously activate a water pump through a relay module when the sensor readings fall outside predefined threshold values, indicating that plants require water. This automated process ensures that plants

receive an optimal amount of water at the right time, preventing both under and overwatering and promoting healthier plant growth.

Additionally, the system incorporates a visual feedback mechanism through a series of LEDs (red, orange, and green), which signal the system's status regarding the plants' watering needs, thereby providing a user-friendly interface. The implementation of the system is facilitated by the Arduino IDE, allowing for straightforward programming and customization to meet various agricultural needs.

By automating the irrigation process based on precise sensor data, this Smart Irrigation System not only conserves water but also reduces labor costs and enhances the overall efficiency of agricultural practices. It represents a significant step forward in the integration of technology into sustainable farming, promising a future where water resources are used judiciously to support the growing demands of agriculture.

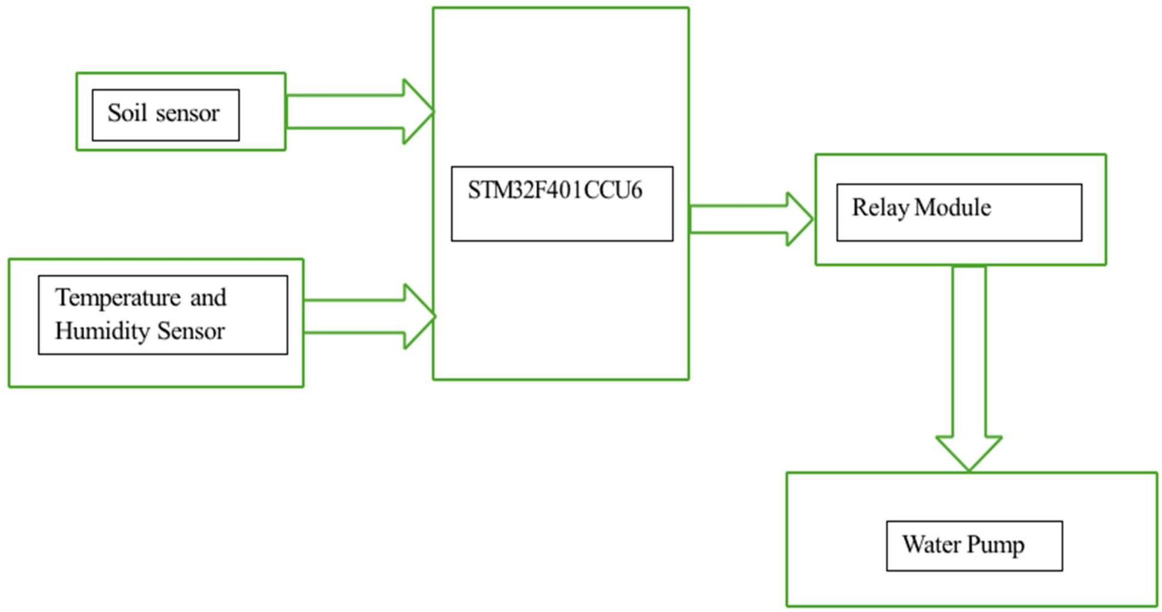
HARDWARE REQUIRMENTS:-

- **STM32F401CCU6 microcontroller:** Chosen for its reliability, efficiency, and performance in handling real-time data processing.
- **Soil moisture sensors (HW080 and HW103)**: Used to monitor soil moisture levels in real-time.
- **DHT11 temperature and humidity sensor:** Provides data on environmental temperature and humidity.
- **Relay module:** Controls the activation of the water pump based on sensor readings.
- **Water pump:** Delivers water to the plants as needed.
- **LEDs and resistors:** Used to create visual feedback indicators for system status monitoring.
- **HW battery**: Power source for the system.
- **Water pump pipe:** Transports water from the pump to the plants.
- **Jumper wires:** Connect various components together.
- **Breadboard:** Provides a platform for prototyping and connecting electronic components.

SOFTWARE REQUIREMENTS:-

- **Arduino IDE:** Used for programming the STM32F401CCU6 microcontroller and implementing the irrigation system's logic.

BLOCK DIAGRAM:-



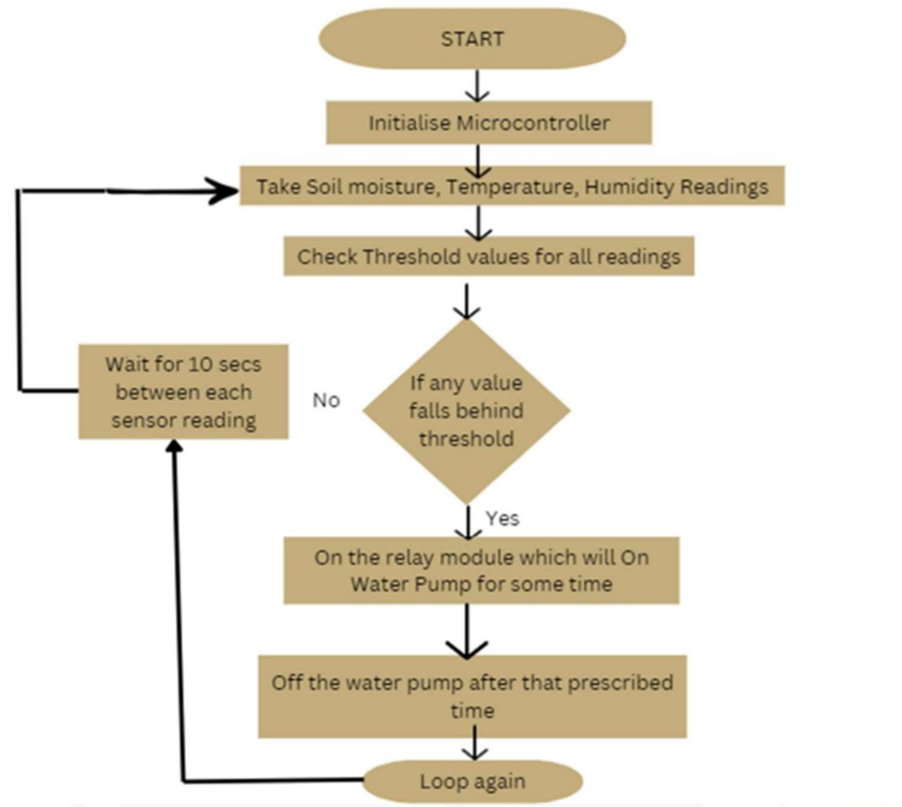
ALGORITHM:-

1. Start: Begin the algorithm.

2. Initialize system components: Set up the microcontroller, sensors, relay module, and LED indicators.

3. Loop:
   a. Read sensor data: Gather data from soil moisture, temperature, and humidity sensors.
   b. Analyse sensor data:
      i. Check if soil moisture is below a predefined threshold.
      ii. Check if soil temperature is above a predefined threshold.
      iii. Check if humidity is below a predefined threshold.
   c. If conditions are met:
      Activate the water pump via the relay module.
      Turn on the red LED indicator to signal watering is needed.
   d. If conditions are not met:
      Deactivate the water pump.
      Turn on the green LED indicator to signal no watering is needed.
   e. Wait: Pause for a predefined interval before repeating the loop.
4. End: Finish the algorithm.

FLOW CHART:-



PROGRAM:-

```
#include <Arduino.h>

#define DHT11_PIN PB9

#define RELAY_PIN PB6

#define SOIL_SENSOR_POWER_PIN PA7

#define SOIL_SENSOR_ANALOG_PIN PA0

#define RED_LED PA3

#define ORANGE_LED PA5

#define GREEN_LED PA9

uint8_t RHI, RHD, TCI, TCD, SUM;

uint32_t pMillis, cMillis;

float tCelsius = 0;

float RH = 0;

uint16_t readValue;
```

```
const float TEMP_THRESHOLD = 25.0;              // Threshold temperature in Celsius
const float HUMIDITY_THRESHOLD = 50.0;          // Threshold humidity in percentage
const uint16_t SOIL_MOISTURE_THRESHOLD = 900;
const unsigned long RELAY_ON_TIME = 5000;
const unsigned long SENSOR_CHECK_INTERVAL = 10000;
const unsigned long GREEN_LED_ON_TIME = 5000;
unsigned long previousSensorCheckMillis = 0;
unsigned long greenLedOnTime = 0;
void microDelay(uint16_t delay) {
  delayMicroseconds(delay);
}
uint8_t DHT11_Start(void) {
  uint8_t Response = 0;
  pinMode(DHT11_PIN, OUTPUT);
  digitalWrite(DHT11_PIN, LOW);
  delay(20);
  digitalWrite(DHT11_PIN, HIGH);
  microDelay(30);
  pinMode(DHT11_PIN, INPUT_PULLUP);
  microDelay(40);
  if (!digitalRead(DHT11_PIN)) {
    microDelay(80);
    if (digitalRead(DHT11_PIN)) Response = 1;
  }
  pMillis = millis();
  cMillis = millis();
  while (digitalRead(DHT11_PIN) && pMillis + 2 > cMillis) {
    cMillis = millis();
  }
  return Response;
```

```
}
uint8_t DHT11_Read(void) {
  uint8_t a, b = 0; // Ensure b is initialized to 0
  for (a = 0; a < 8; a++) {
    pMillis = millis();
    cMillis = millis();
    while (!digitalRead(DHT11_PIN) && pMillis + 2 > cMillis) {
      cMillis = millis();
    }
    delayMicroseconds(40); // Wait for the data bit to be ready
    if (digitalRead(DHT11_PIN))
      b |= (1 << (7 - a));
    else
      b &= ~(1 << (7 - a));
    pMillis = millis();
    cMillis = millis();
    while (digitalRead(DHT11_PIN) && pMillis + 2 > cMillis) {
      cMillis = millis();
    }
  }
  return b;
}
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // For indicating sensor read status
  pinMode(RELAY_PIN, OUTPUT);       // For controlling the relay
  digitalWrite(RELAY_PIN, LOW);     // Initially turn off the relay
  pinMode(DHT11_PIN, OUTPUT);
  pinMode(SOIL_SENSOR_POWER_PIN, OUTPUT);
  pinMode(RED_LED, OUTPUT);         // Red LED
  pinMode(ORANGE_LED, OUTPUT);      // Orange LED
```

```
  pinMode(GREEN_LED, OUTPUT);       // Green LED
  digitalWrite(SOIL_SENSOR_POWER_PIN, LOW); // Initialize
SOIL_SENSOR_POWER_PIN
  Serial.begin(9600);
}
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousSensorCheckMillis >= SENSOR_CHECK_INTERVAL) {
    previousSensorCheckMillis = currentMillis; // save the last time you read the sensors
    if (DHT11_Start()) {
      RHI = DHT11_Read();
      RHD = DHT11_Read();
      TCI = DHT11_Read();
      TCD = DHT11_Read();
      SUM = DHT11_Read();
      if ((RHI + RHD + TCI + TCD) == SUM) {
        tCelsius = TCI + TCD / 10.0;
        RH = RHI + RHD / 10.0;
        digitalWrite(SOIL_SENSOR_POWER_PIN, HIGH);
        delay(10); // Wait for stabilization
        readValue = analogRead(SOIL_SENSOR_ANALOG_PIN);
        digitalWrite(SOIL_SENSOR_POWER_PIN, LOW);
        Serial.print("Temperature: ");
        Serial.print(tCelsius);
        Serial.print("°C, Humidity: ");
        Serial.print(RH);
        Serial.print("%, Soil Moisture: ");
        Serial.println(readValue);
        if (tCelsius > TEMP_THRESHOLD || RH < HUMIDITY_THRESHOLD ||
readValue > SOIL_MOISTURE_THRESHOLD) {
          digitalWrite(RED_LED, HIGH); // Red LED on
```

```
        delay(3000);
        digitalWrite(RED_LED, LOW); // Wait for 3 seconds
        digitalWrite(RELAY_PIN, HIGH); // Water pump ON
        digitalWrite(ORANGE_LED, HIGH); // Orange LED on
        Serial.println("Water pump is on");
        delay(RELAY_ON_TIME); // Wait for RELAY_ON_TIME milliseconds
        digitalWrite(RELAY_PIN, LOW); // Water pump OFF
        digitalWrite(ORANGE_LED, LOW); // Orange LED off
        Serial.println("Water pump is off");
      } else {
        digitalWrite(GREEN_LED, HIGH); // Green LED on
        greenLedOnTime = currentMillis; // save the time when green LED was turned on
      }
    } else {
      Serial.println("Checksum failed, data invalid.");
    }
    digitalWrite(LED_BUILTIN, HIGH); // Turn on built-in LED to indicate successful
read
    delay(100); // Short delay to visually distinguish LED status change
    digitalWrite(LED_BUILTIN, LOW);
    if (currentMillis - greenLedOnTime >= GREEN_LED_ON_TIME) {
      digitalWrite(GREEN_LED, LOW); // Green LED off
    }
  }
 }
}
```

REAL TIME CONSTRAINTS:-

Real-time constraints refer to the limitations or requirements that must be met to ensure the system functions correctly within a specific timeframe. For this Smart Irrigation System, here are some real-time constraints to consider:

- **Sensor Data Sampling Rate:** Ensure that the system samples sensor data frequently enough to detect changes in soil moisture, temperature, and humidity in a timely manner, typically within seconds or minutes depending on the application.
- **Decision Making Time:** Define a maximum allowable time for the system to analyze sensor data and make irrigation decisions. This time constraint ensures that the system responds promptly to changing environmental conditions, typically within a few seconds.
- **Communication Latency:** If the system communicates with external devices or platforms for remote monitoring or control, minimize communication latency to maintain real-time responsiveness. Aim for low-latency communication protocols and networks to ensure efficient data exchange, typically with response times in the order of milliseconds to seconds.

CONCLUSION:-

The Smart Irrigation System represents a significant advancement in agricultural technology, offering a cost-effective and sustainable solution for optimizing water usage and promoting plant health. By leveraging sensor technology and automation, the system addresses the challenges of traditional irrigation methrads and contributes to more efficient and environmentally friendly farming practices. With its potential to conserve water, reduce labor, and improve crop yield, the Smart Irrigation System holds promise for revolutionizing agriculture and supporting food security in a changing climate.