**PHASE 3 – DEVELOPMENT PART-1**

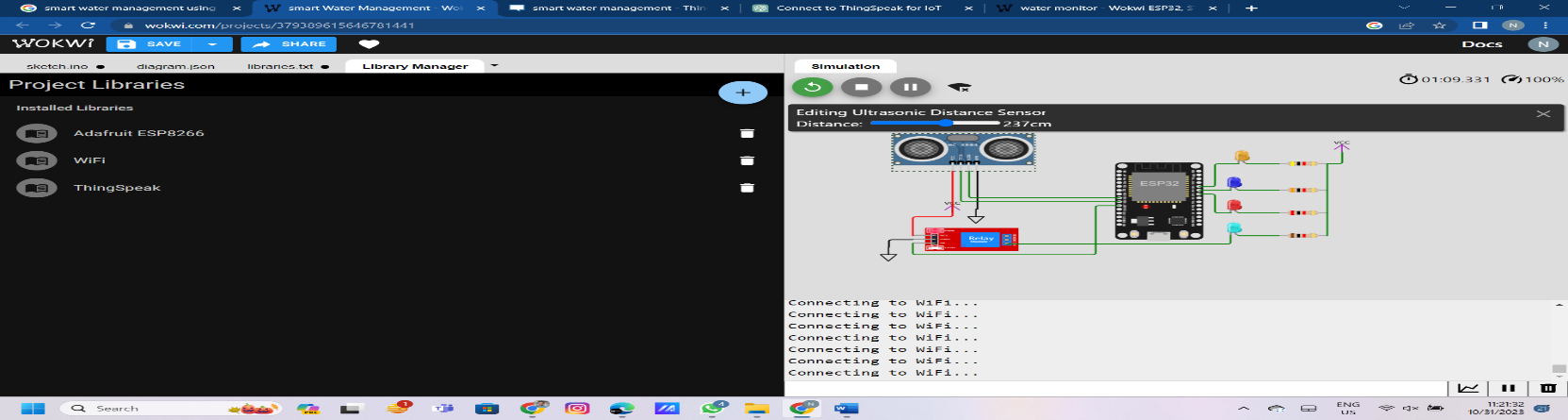
**PROJECT NAME: SMART WATER SYSTEM**

**INTRODUCTION:**

Smart water management is the iot based project which is used to monitor the flow of the water in public tanks and other public places to avoid to wastage of water. Where the circuit diagram is developed in wokwi platform to monitor the flow of water using ultrasonic sensor and the output was displayed in thinkspeak.

**1. Including Necessary Libraries:**

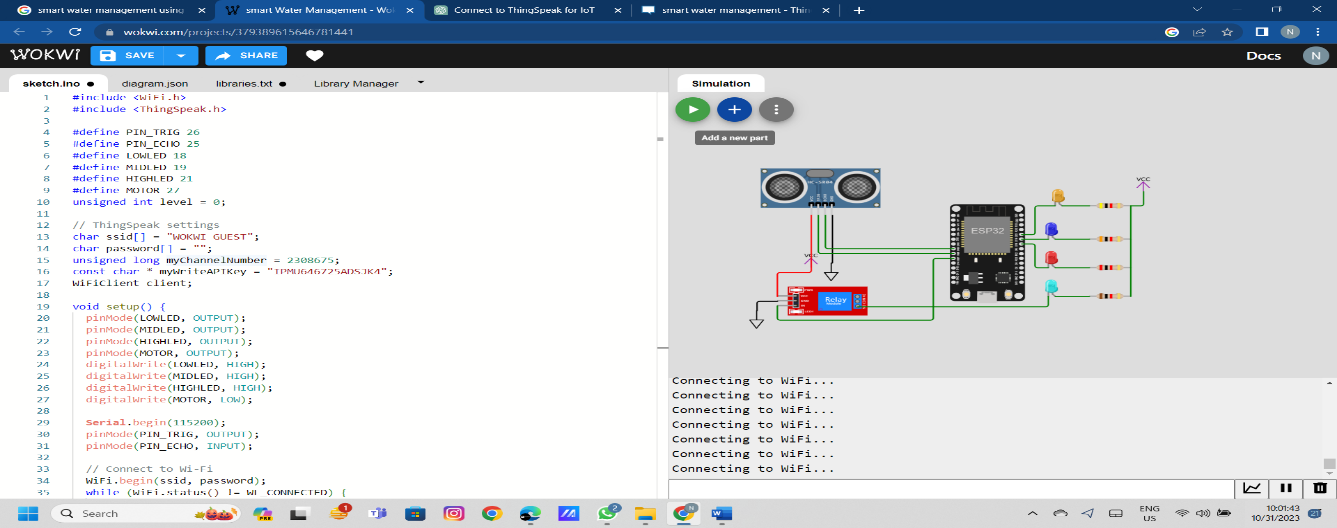
The code starts by including the necessary libraries: ESP8266WiFi.h, WiFiClient.h, and ThingSpeak.h. These libraries are required to connect to Wi-Fi and send data to Thing Speak using an ESP8266 board.



**2. Defining Pin and Variable Constants:**

The code defines pin numbers for the ultrasonic sensor, LEDs, and a motor, as well as a variable to store the distance measurement (level).

**Circuit diagram:**



**3. Configuring ThingSpeak Settings:**

You need to configure your Wi-Fi network credentials (ssid and password), ThingSpeak channel number (myChannelNumber), and Write API Key (myWriteAPIKey). You should replace the placeholders with your own values.

**4. Setup Function:**

In the setup function:

Pins for LEDs, motor, and ultrasonic sensor are set as output or input.

The LEDs are initially turned on, and the motor is turned off.

Serial communication is initialized for debugging.

Wi-Fi connection to your network is established, and you'll see status messages in the Serial Monitor.

The ThingSpeak client is initialized.

**5. Loop Function:**

The loop function is where the main operation occurs, and it repeats continuously. It triggers the ultrasonic sensor to take a distance measurement by sending a short pulse on PIN\_TRIG.The code measures the time it takes for the echo signal to return and calculates the distance in centimeters and inches. The results are printed to the Serial Monitor.The distance measurement is stored in the level variable. Based on the distance, the code controls the LEDs and motor to indicate the proximity of an object. The distance measurements in centimeters and inches are sent to ThingSpeak using ThingSpeak.setField() and then sent to the ThingSpeak channel using ThingSpeak.writeFields().The HTTP response code is checked, and the status of the data upload is printed to the Serial Monitor. There’s a delay of 1 second before the next measurement.

**6. Debugging and Testing:**

To debug and test the code, you can open the Arduino IDE's Serial Monitor. It will display the distance measurements and the status of data sent to ThingSpeak. This helps you ensure that the system is working as expected.

**7. ThingSpeak Output:**

After successfully uploading data to ThingSpeak, you can log in to your ThingSpeak account and access the data in your specified channel. The channel will display the distance measurements in real-time.

By following this code, you can create a system that measures the distance using an ultrasonic sensor, controls LEDs and a motor based on proximity, and logs the data to ThingSpeak for monitoring and analysis. Make sure to replace the placeholder values with your own Wi-Fi credentials and ThingSpeak details to make it work with your specific setup.

**Code:**

#include <WiFi.h>

#include <ThingSpeak.h>

#define PIN\_TRIG 26

#define PIN\_ECHO 25

#define LOWLED 18

#define MIDLED 19

#define HIGHLED 21

#define MOTOR 27

unsigned int level = 0;

// ThingSpeak settings

char ssid[] = "WOKWI GUEST";

char password[] = "";

unsigned long myChannelNumber = 2325279;

const char \* myWriteAPIKey = "1IHS3OWB1QSUX6XK";

WiFiClient client;

void setup() {

  pinMode(LOWLED, OUTPUT);

  pinMode(MIDLED, OUTPUT);

  pinMode(HIGHLED, OUTPUT);

  pinMode(MOTOR, OUTPUT);

  digitalWrite(LOWLED, HIGH);

  digitalWrite(MIDLED, HIGH);

  digitalWrite(HIGHLED, HIGH);

  digitalWrite(MOTOR, LOW);

**Serial**.begin(115200);

  pinMode(PIN\_TRIG, OUTPUT);

  pinMode(PIN\_ECHO, INPUT);

  // 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 ThingSpeak

  ThingSpeak.begin(client);

}

void loop() {

  // Start a new measurement:

  digitalWrite(PIN\_TRIG, HIGH);

  delayMicroseconds(10);

  digitalWrite(PIN\_TRIG, LOW);

  // Read the result:

  int duration = pulseIn(PIN\_ECHO, HIGH);

**Serial**.print("Distance in CM: ");

**Serial**.println(duration / 58);

**Serial**.print("Distance in inches: ");

**Serial**.println(duration / 148);

  level = (duration / 10);

  if (level < 100) {

    digitalWrite(LOWLED, LOW);

    digitalWrite(MOTOR, HIGH);

    digitalWrite(HIGHLED, HIGH);

    digitalWrite(MIDLED, HIGH);

  } else if ((level > 200) && (level < 400)) {

    digitalWrite(LOWLED, HIGH);

    digitalWrite(HIGHLED, HIGH);

    digitalWrite(MIDLED, LOW);

  } else if (level >= 400) {

    digitalWrite(HIGHLED, LOW);

    digitalWrite(MIDLED, HIGH);

    digitalWrite(LOWLED, HIGH);

    digitalWrite(MOTOR, LOW);

  }

  // Send data to ThingSpeak

  ThingSpeak.setField(1, duration / 58); // Distance in CM

  ThingSpeak.setField(2, duration / 148); // Distance in inches

  int httpCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

  if (httpCode == 200) {

**Serial**.println("Data sent to ThingSpeak successfully");

  } else {

**Serial**.print("Failed to send data to ThingSpeak, HTTP error code: ");

**Serial**.println(httpCode);

  }

  delay(1000); // Delay before the next measurement

}

**Output:**

