SMART WATER FOUNTAIN

Name : Subramani.G

Reg.No : 950421106021

Department : Electronics and communication engineering

College name: Dr.G.U.Pope college of engineering

College code: 9504

Course name: IOT

PROJECT OBJECTIVE

An IoT based smart water fountain refers to a network of interconnected devices equipped with sensors designed to control waterflow and detect malfunctions. The fountain is equipped with sensors that monitor various aspects of its operation, such as water level, water quality (e.g., pH and turbidity), temperature. Data collected from the fountain can be analyzed to provide insights into water usage, maintenance needs, and overall performance. This information can be valuable for optimizing the fountain's operation. The primary objective is to provide real-time information about water fountain status to residents through a public platform.

IOT DEVICES SETUP

Smart water fountain was simulated using IOT devices in wokwi simulator.

COMPONENTS USED

1. Raspberry Pi
2. water pump.
3. Relay module
4. An ultrasonic sensor (HC SR04)
5. Wokwi virtual components

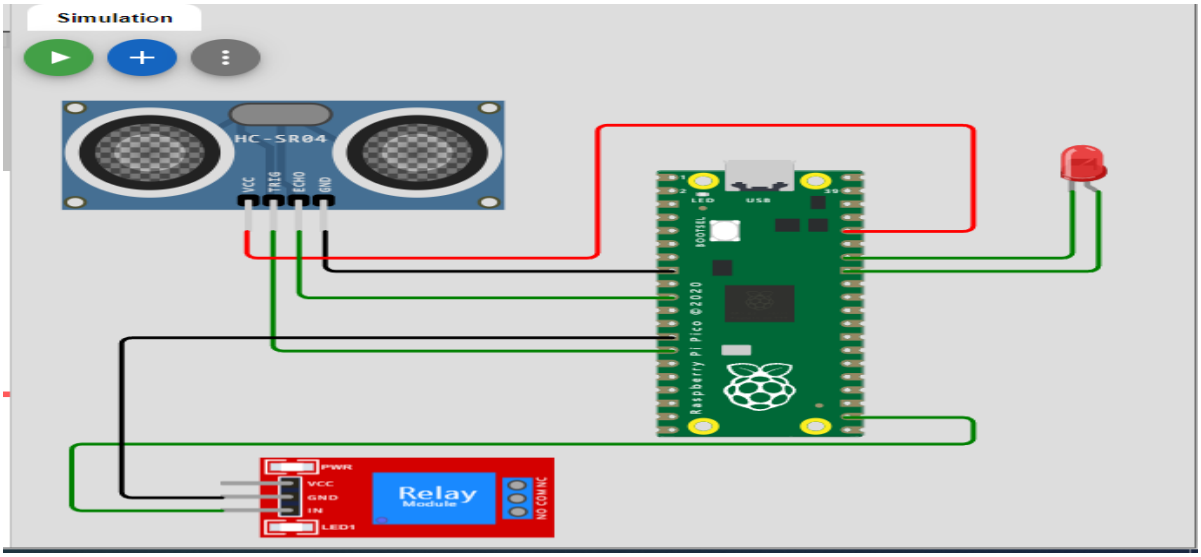
CONNECTIONS

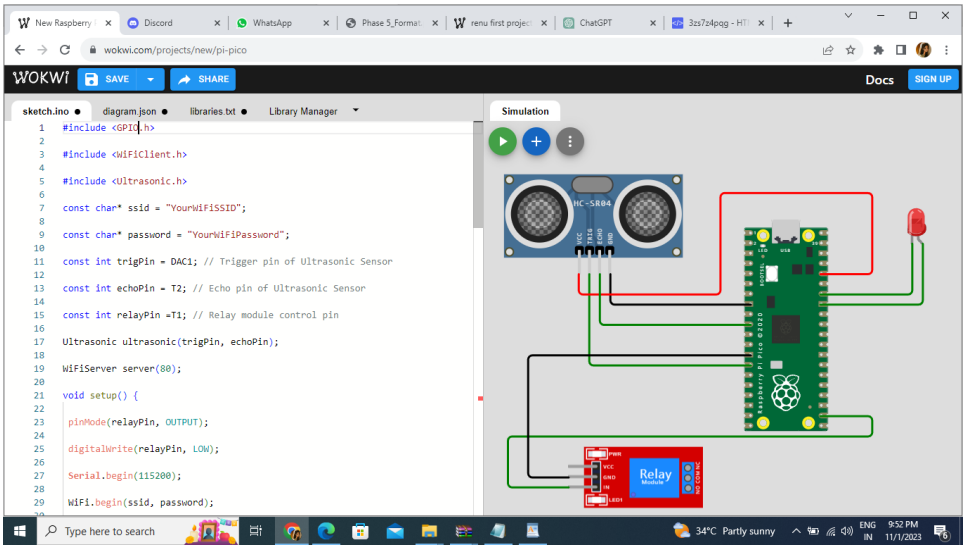
1)Rely module is connected to the Raspberry pi gpio pin 17.

2)Ultrasonic sensor’s trigger pin is connected to raspberry pi gpio pin 10 and the Echo pin is connected to pin 7.

3)The power supply is properly given and the relay module is connected to water pump.

CIRCUIT





PYTHON SCRIPT

```
#include <GPIO.h>
#include <WiFiClient.h>
#include <Ultrasonic.h>
const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";
const int trigPin = 7; // Trigger pin of Ultrasonic Sensor
const int echoPin = 10; // Echo pin of Ultrasonic Sensor
```

```
const int relayPin =17; // Relay module control pin
Ultrasonic ultrasonic(trigPin, echoPin);
WiFiServer server(80);
void setup() {
 pinMode(relayPin, OUTPUT);
 digitalWrite(relayPin, LOW);
 Serial.begin(115200);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
 delay(1000);
 Serial.println("Connecting to WiFi...");
 }
 server.begin();
}
void loop() {
 WiFiClient client = server.available();
 if (client) {
 String request = client.readStringUntil('\r');
 if (request.indexOf("/on") != -1) {
 digitalWrite(relayPin, HIGH); // Turn the pump on
 delay(2000); // Run the pump for 2 seconds
 digitalWrite(relayPin, LOW); // Turn the pump off
 }
 client.flush();
 }
 // Check water level
 float distance = ultrasonic.read();
 if (distance < 10) {
 // Water is low, update the web interface
```

```
 // You can send an HTML response to the client here
 }
}
```

IOT PLATFORM

Here ,we used thinkspeak platform to aggregate, visulaize and analyze the live data streams in the cloud.

1. You include necessary libraries for Wi-Fi communication and HTTP requests using the ESP8266.

2. Set your Wi-Fi credentials (SSID and password) and ThingSpeak API key and channel ID.

3. The `setup` function initializes Wi-Fi and sets up the relay pin as an output.

4. The `loop` function continuously reads the water level sensor and controls the water pump based on the water level.

5. It also sends the water level data to ThingSpeak using an HTTP GET request.

coding

```
include <Arduino.h>

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>


// Define Wi-Fi credentials

const char* ssid = "YourWiFiSSID";

const char* password = "YourWiFiPassword";


// ThingSpeak settings

const String server = "api.thingspeak.com";
```

```
const String apiKey = "YourAPIKey";
const String channelID = "YourChannelID";

// Define the pins for water level sensor and relay module
const int waterLevelSensorPin = A0;  // Analog pin for water level sensor
const int relayPin = 2;              // Digital pin for controlling the relay

// Define water level threshold (adjust as needed)
const int waterLevelThreshold = 500; // Example threshold value

void setup() {
  // Initialize serial communication for debugging (optional)
  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");

  pinMode(relayPin, OUTPUT);
}

void loop() {
  // Read the water level sensor
  int waterLevel = analogRead(waterLevelSensorPin);
```

```
// Check if the water level is below the threshold
if (waterLevel < waterLevelThreshold) {
  // Turn on the water pump (activate the relay)
  digitalWrite(relayPin, HIGH);
  Serial.println("Water Pump ON");
} else {
  // Turn off the water pump
  digitalWrite(relayPin, LOW);
  Serial.println("Water Pump OFF");
}

// Send data to ThingSpeak
if (WiFi.status() == WL_CONNECTED) {
  HTTPClient http;
  String url = "http://" + server + "/update?api_key=" + apiKey + "&field1=" + String(waterLevel);

  http.begin(url);
  int httpCode = http.GET();

  if (httpCode > 0) {
    Serial.println("Data sent to ThingSpeak");
  } else {
    Serial.println("Error sending data to ThingSpeak");
  }
  http.end();
}

// Add a delay between readings
delay(60000); // Delay for 1 minute
```

}

USER INTERFACE

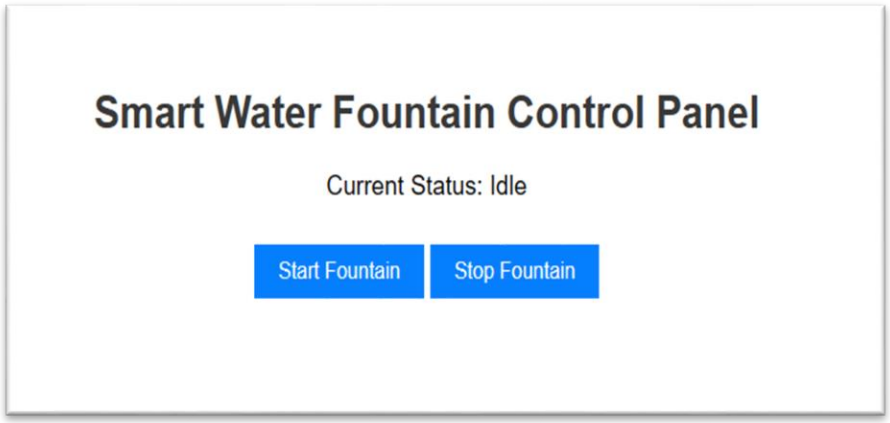
User interface or mobile interface which is created using html coding.

Html coding:

```
<!DOCTYPE html>
<html>
<head>
  <title>Smart Water Fountain Control</title>
  <style>
    /* Add your CSS styles here */
    body {
      font-family: Arial, sans-serif;
    }
    .container {
      max-width: 400px;
      margin: 0 auto;
      padding: 20px;
      text-align: center;
    }
    .button {
      padding: 10px 20px;
      font-size: 16px;
      background-color: #007bff;
      color: #fff;
      border: none;
      cursor: pointer;
    }
  </style>
</head>
```

```
<body>
  <div class="container">
    <h1>Smart Water Fountain Control</h1>
    <p>Status: <span id="status">Idle</span></p>

    <button class="button" id="startButton">Start Fountain</button>
    <button class="button" id="stopButton">Stop Fountain</button>

    <script>
      // Add your JavaScript code here
      const statusElement = document.getElementById('status');
      const startButton = document.getElementById('startButton');
      const stopButton = document.getElementById('stopButton');

      // Function to start the fountain
      function startFountain() {
        // You would send a command to your smart water fountain here
        // For this example, we'll just update the status
        statusElement.innerText = 'Fountain is running';
      }

      // Function to stop the fountain
      function stopFountain() {
        // You would send a command to stop the fountain here
        // For this example, we'll just update the status
        statusElement.innerText = 'Fountain stopped';
      }

      // Add event listeners to the buttons
```

```
        startButton.addEventListener('click', startFountain);
        stopButton.addEventListener('click', stopFountain);
    </script>
  </div>
</body>
</html>
```



CONCLUSION

Thus,IOT based smart water fountain was designed, simulated, implemented using various platform such as wokwi,thinkspeak,html etc