SMART WATER MANAGEMENT

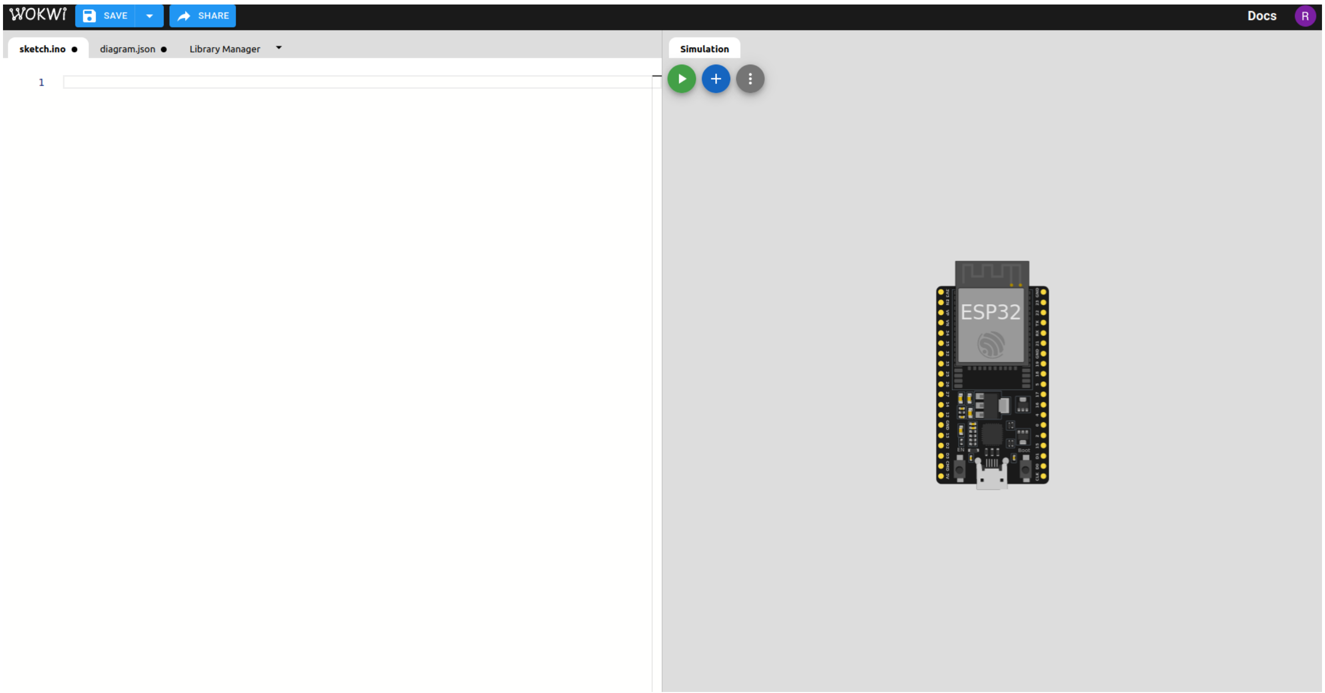
Configuring IOT sensors to measure Water Level and analyse whether the water is contaminated or not using two sensors:

1) Ultrasonic Sensor (HC-SR04): For measuring the water level.

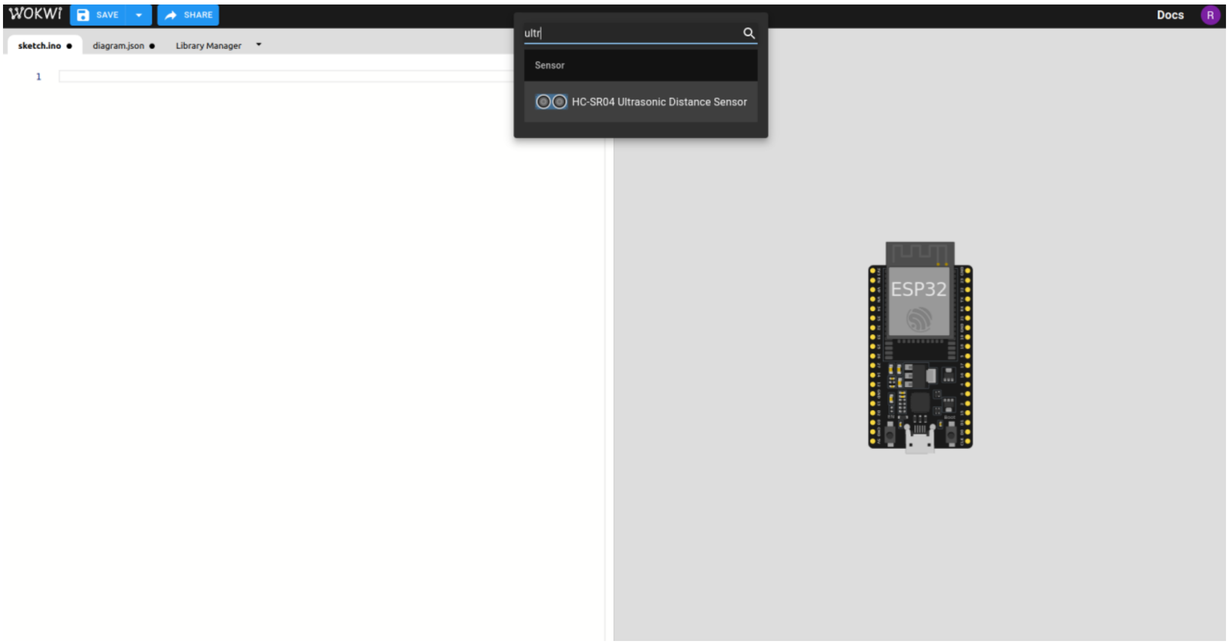
2) Potentiometer for pH sensor: For measuring pH value and tell whether the water is contaminated or not.

Configuring these sensors:

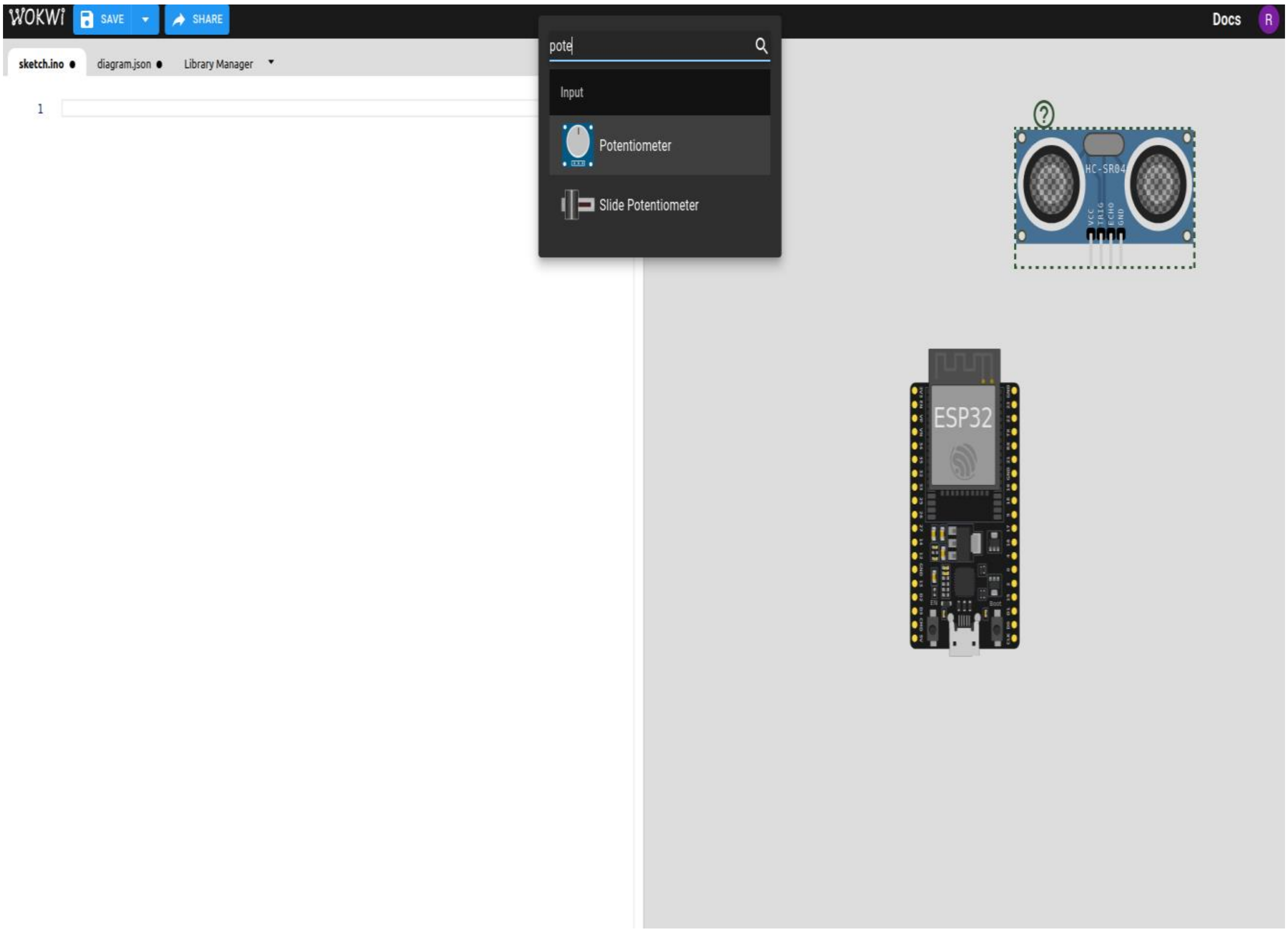
1)Using Virtual Environment for this project which is WOKWI .In this we have ESP32.So we can use that:



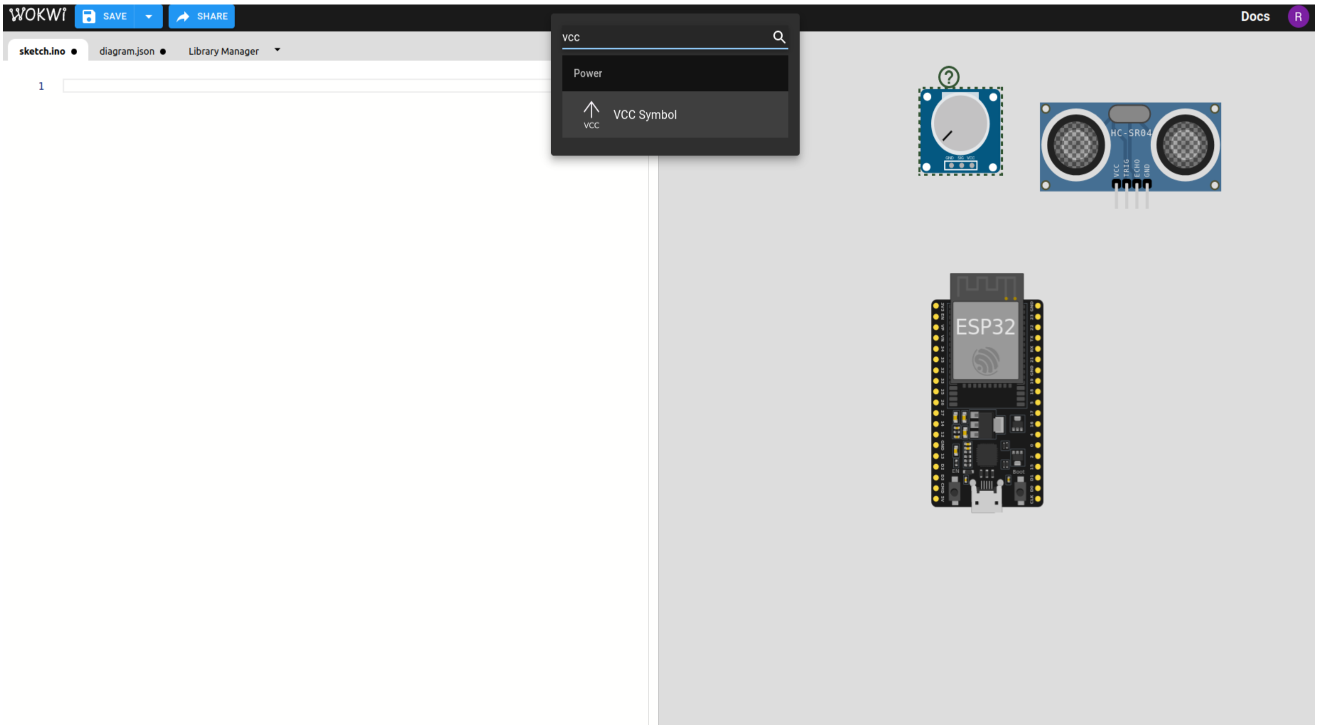
2) Using Ultrasonic Sensor:

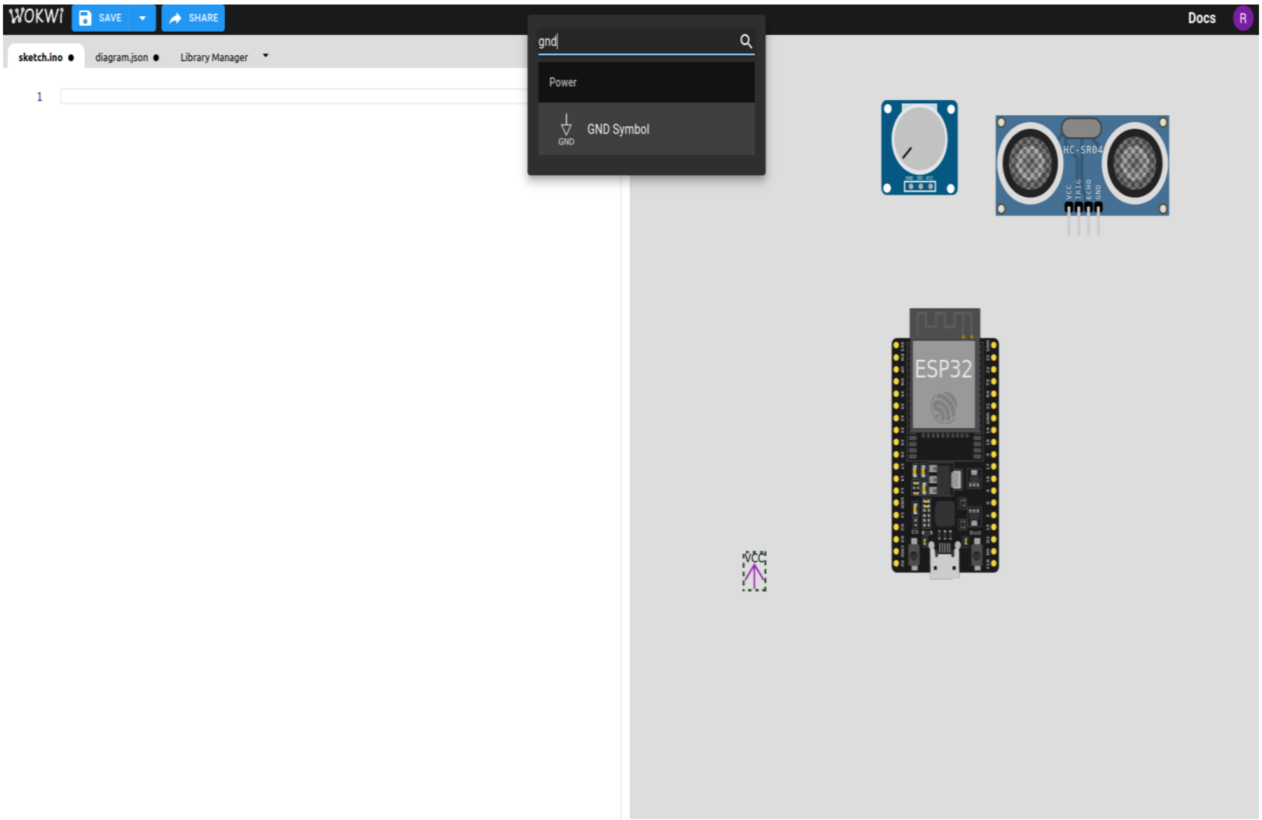


3) Using Potentiometer for measuring pH value:

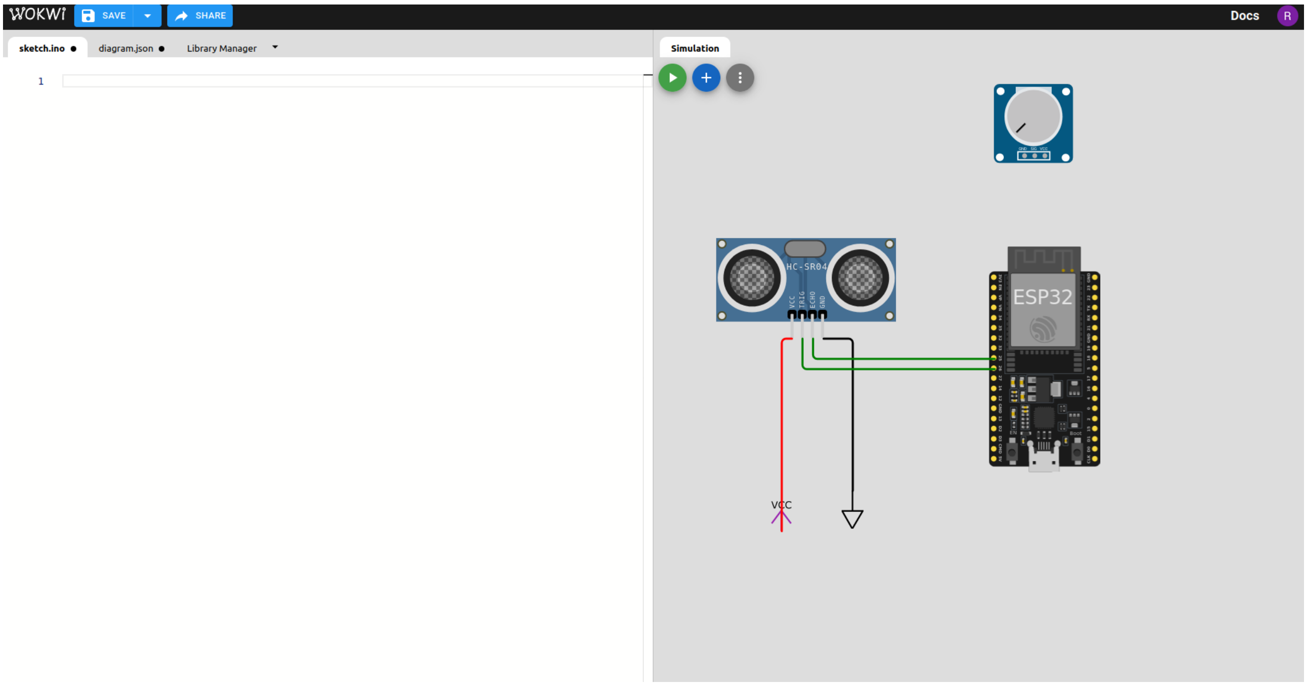


1) Using VCC and GND:

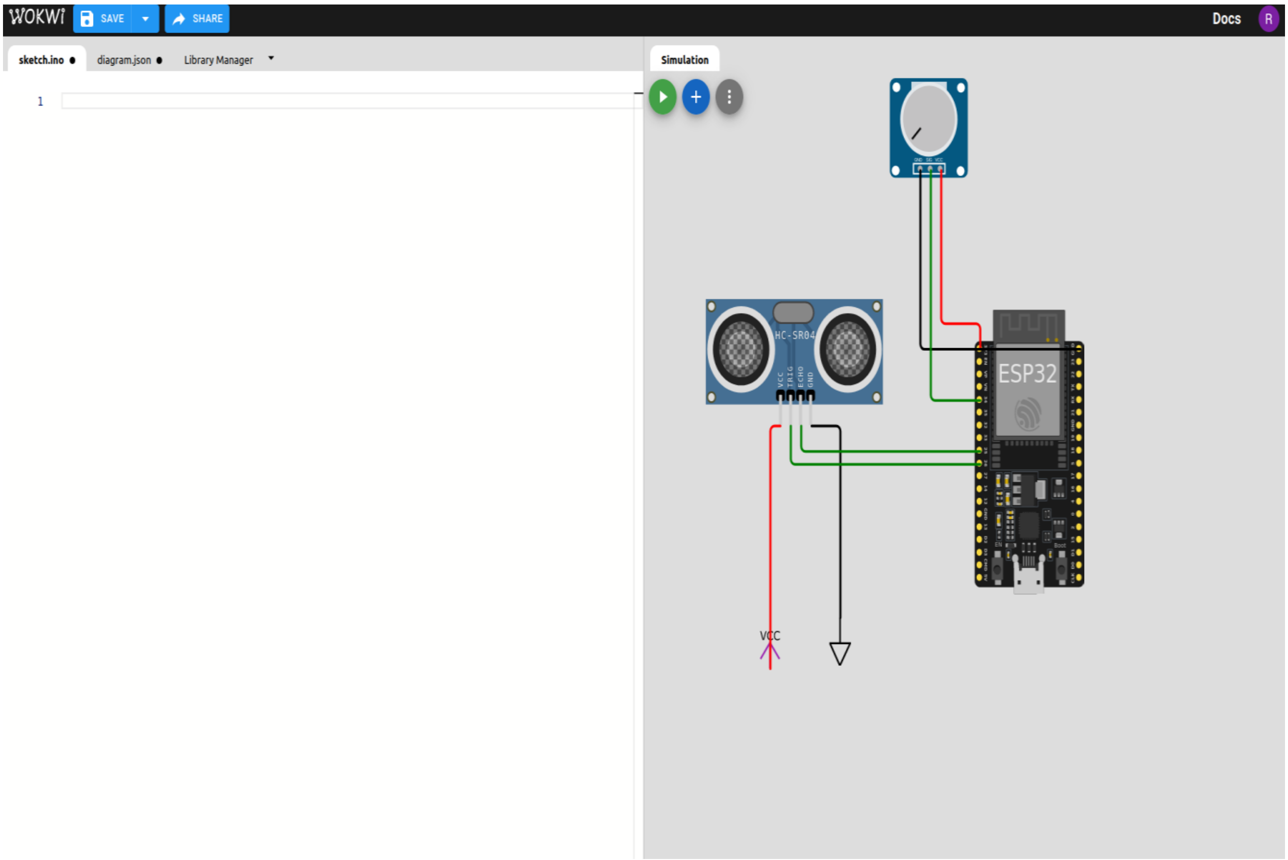




5) Connecting Ultrasonic Sensor to ESP32:



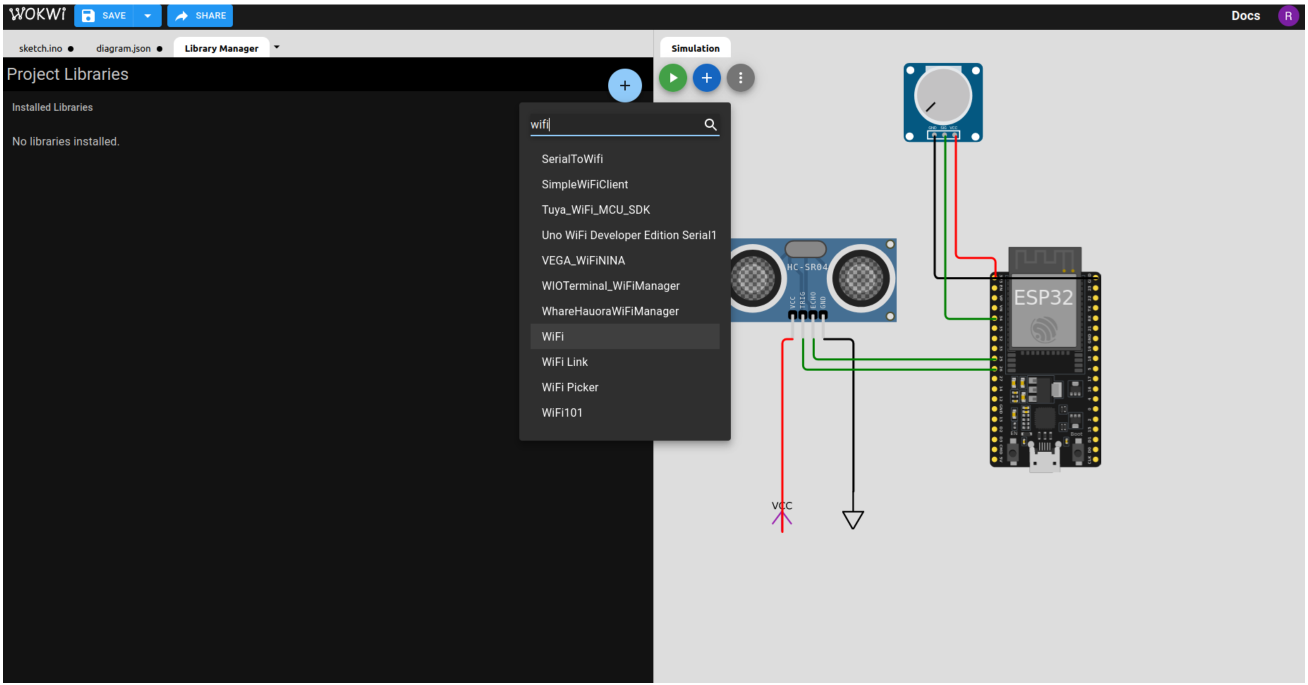
6) Connecting Potentiometer to ESP32:

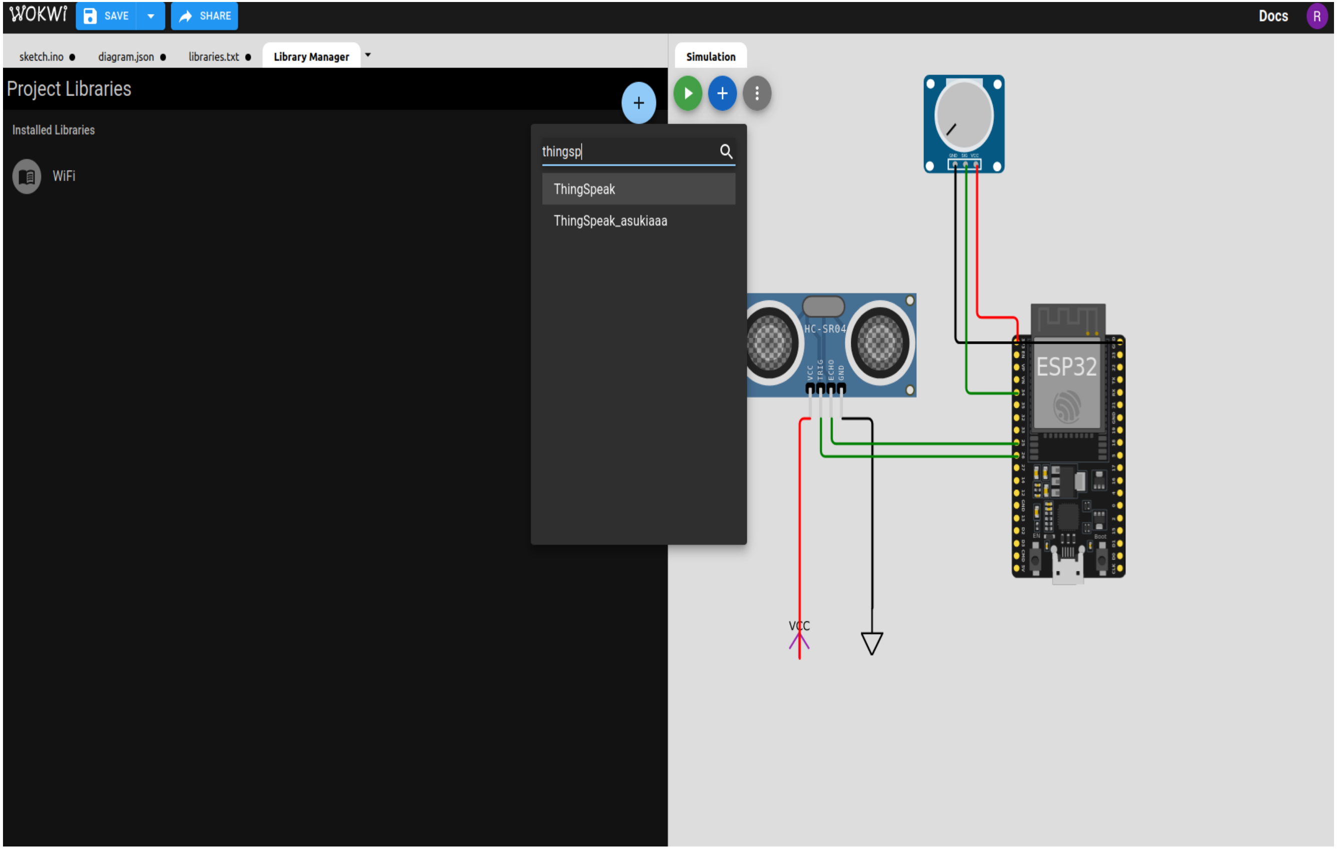


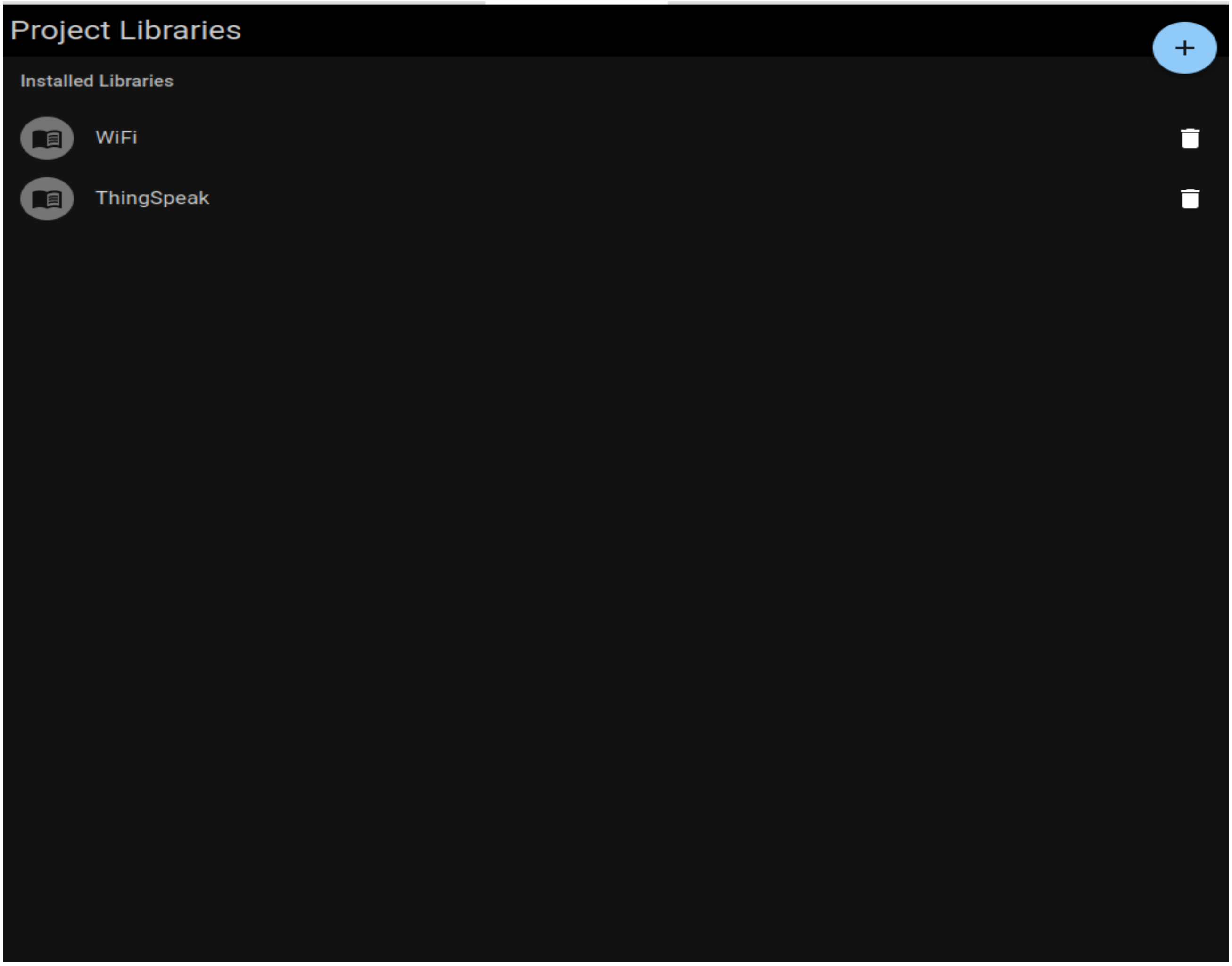
7)Libraries Needed:

a)WiFi

b)ThingSpeak







Project Libraries

Installed Libraries

WiFi

ThingSpeak

8) Program for measurement of water , measuring pH value to find whether the water is contaminated or not and sharing data to a data sharing platform named ThingSpeak:

File - sketch.ino :

```
#include <Wire.h>
#include <WiFi.h>
#include <ThingSpeak.h>


const char *ssid = "Wokwi-GUEST";
const char *password = "";


#define TRIG_PIN 26
#define ECHO_PIN 25
#define POTENTIOMETER_PIN A0
const float contaminationThreshold = 7.0;
```

```
unsigned long channelID = 2327527; // Use Your ThingSpeak Channel ID
const char *writeAPIKey = "3K1OPPG28L2BIA3E"; // Use Your ThingSpeak Write API Key


WiFiClient client;


void setup() {
Serial.begin(115200);
connectToWiFi();
ThingSpeak.begin(client);
pinMode(TRIG_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
}


void loop() {
float distance = readUltrasonicDistance();
float pHValue = analogRead(POTENTIOMETER_PIN) / 100.0;
Serial.print("Ultrasonic Distance: ");
Serial.print(distance);
Serial.println(" cm");


Serial.print("pH Value: ");
Serial.println(pHValue);


if (pHValue < contaminationThreshold) {
Serial.println("Water is contaminated!");
} else {
Serial.println("Water is clean.");
}


ThingSpeak.setField(1, distance);
ThingSpeak.setField(2, pHValue);


int updateSuccess = ThingSpeak.writeFields(channelID, writeAPIKey);


if (updateSuccess) {
Serial.println("ThingSpeak update successful");
} else {
Serial.println("Error updating ThingSpeak");
}


delay(2000);
}


void connectToWiFi() {
Serial.print("Connecting to WiFi");
WiFi.begin(ssid, password);


while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.print(".");
}


Serial.println("\nConnected to WiFi");
}


float readUltrasonicDistance() {
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);


return pulseIn(ECHO_PIN, HIGH) * 0.0343 / 2;
}
```

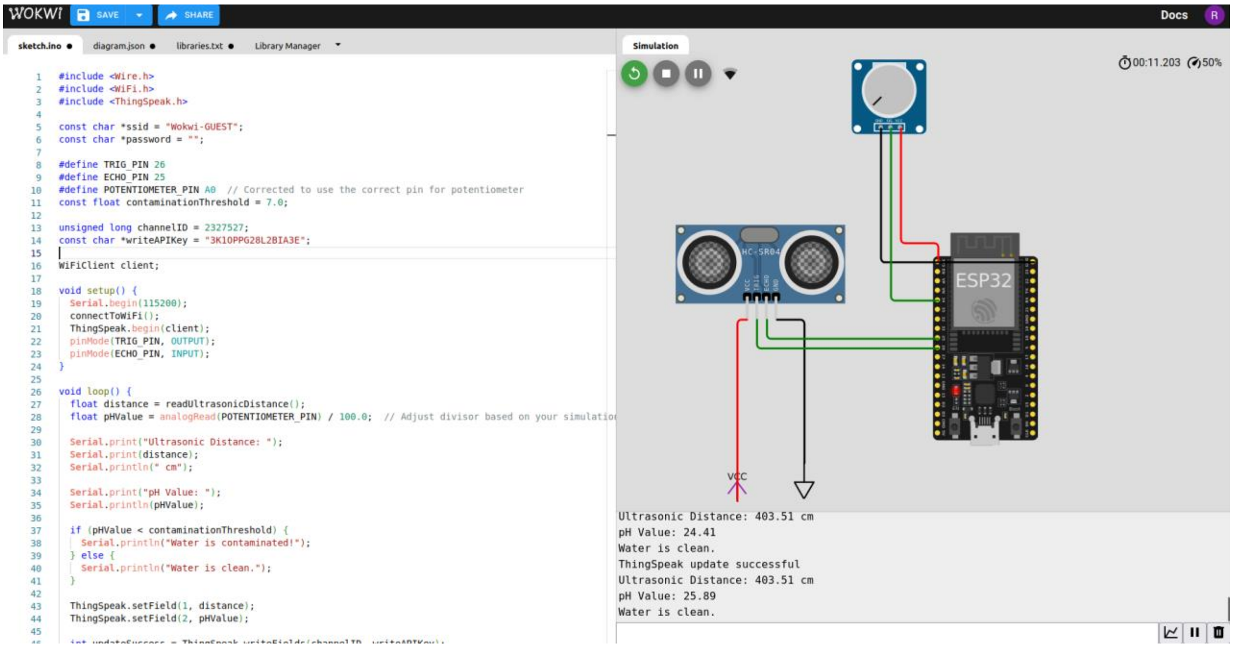
File – diagram.json:

```
{
"version": 1,
"author": "Rohit Kanna",
"editor": "wokwi",
"parts": [
{ "type": "board-esp32-devkit-c-v4", "id": "esp", "top": -9.6, "left": 62.44, "attrs": {} },
{ "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -17.7, "left": -196.1, "attrs": {} },
{ "type": "wokwi-potentiometer", "id": "pot1", "top": -183.7, "left": -19.4, "attrs": {} },
{ "type": "wokwi-vcc", "id": "vcc1", "top": 231.16, "left": -144, "attrs": {} },
{ "type": "wokwi-gnd", "id": "gnd1", "top": 220.8, "left": -77.4, "attrs": {} }
],
"connections": [
[ "esp:TX", "$serialMonitor:RX", "", [] ],
[ "esp:RX", "$serialMonitor:TX", "", [] ],
[ "vcc1:VCC", "ultrasonic1:VCC", "red", [ "v0" ] ],
[ "gnd1:GND", "ultrasonic1:GND", "black", [ "v0" ] ],
[ "ultrasonic1:TRIG", "esp:26", "green", [ "v0" ] ],
[ "ultrasonic1:ECHO", "esp:25", "green", [ "v0" ] ],
[ "pot1:VCC", "esp:3V3", "red", [ "v115.2", "h-48.8" ] ],
[ "pot1:GND", "esp:GND.2", "black", [ "v0" ] ],
[ "pot1:SIG", "esp:34", "green", [ "v0" ] ]
],
"dependencies": {}
}
```

File – libraries.txt:

```
# Wokwi Library List

# See https://docs.wokwi.com/guides/libraries

WiFi

ThingSpeak
```

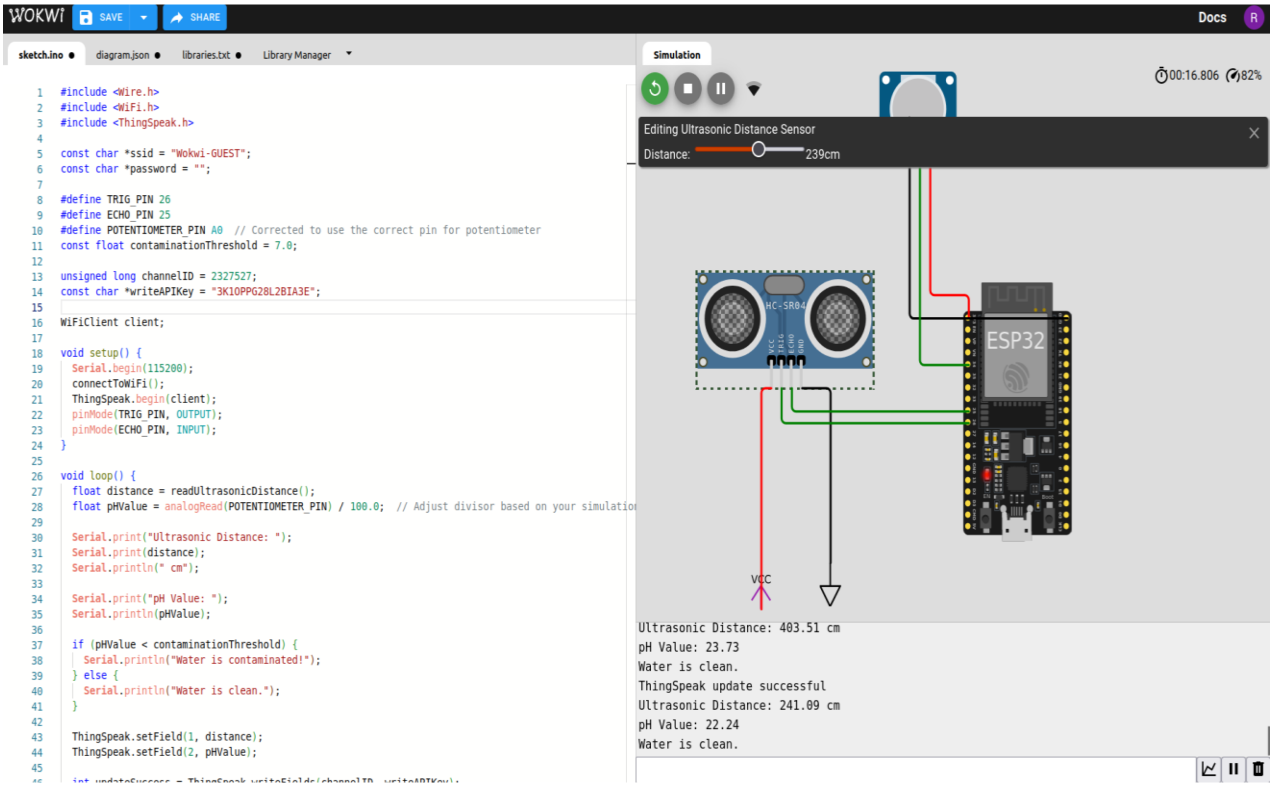
Output:



```
#include <Wire.h>
#include <WiFi.h>
#include <ThingSpeak.h>

const char *ssid = "Wokwi-GUEST";
const char *password = "";

#define TRIG_PIN 26
#define ECHO_PIN 25
#define POTENTIOMETER_PIN A0  // Corrected to use the correct pin for potentiometer
const float contaminationThreshold = 7.0;

unsigned long channelID = 2327527;
const char *writeAPIKey = "3K1OPPG28L2BIA3E";

WiFiClient client;

void setup() {
  Serial.begin(115200);
  connectToWiFi();
  ThingSpeak.begin(client);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  float distance = readUltrasonicDistance();
  float pHValue = analogRead(POTENTIOMETER_PIN) / 100.0;  // Adjust divisor based on your simulatio

  Serial.print("Ultrasonic Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  Serial.print("pH Value: ");
  Serial.println(pHValue);

  if (pHValue < contaminationThreshold) {
    Serial.println("Water is contaminated!");
  } else {
    Serial.println("Water is clean.");
  }

  ThingSpeak.setField(1, distance);
  ThingSpeak.setField(2, pHValue);

  int updateSuccess = ThingSpeak.writeFields(channelID, writeAPIKey);
```

```
Ultrasonic Distance: 403.51 cm
pH Value: 24.41
Water is clean.
ThingSpeak update successful
Ultrasonic Distance: 403.51 cm
pH Value: 25.89
Water is clean.
```

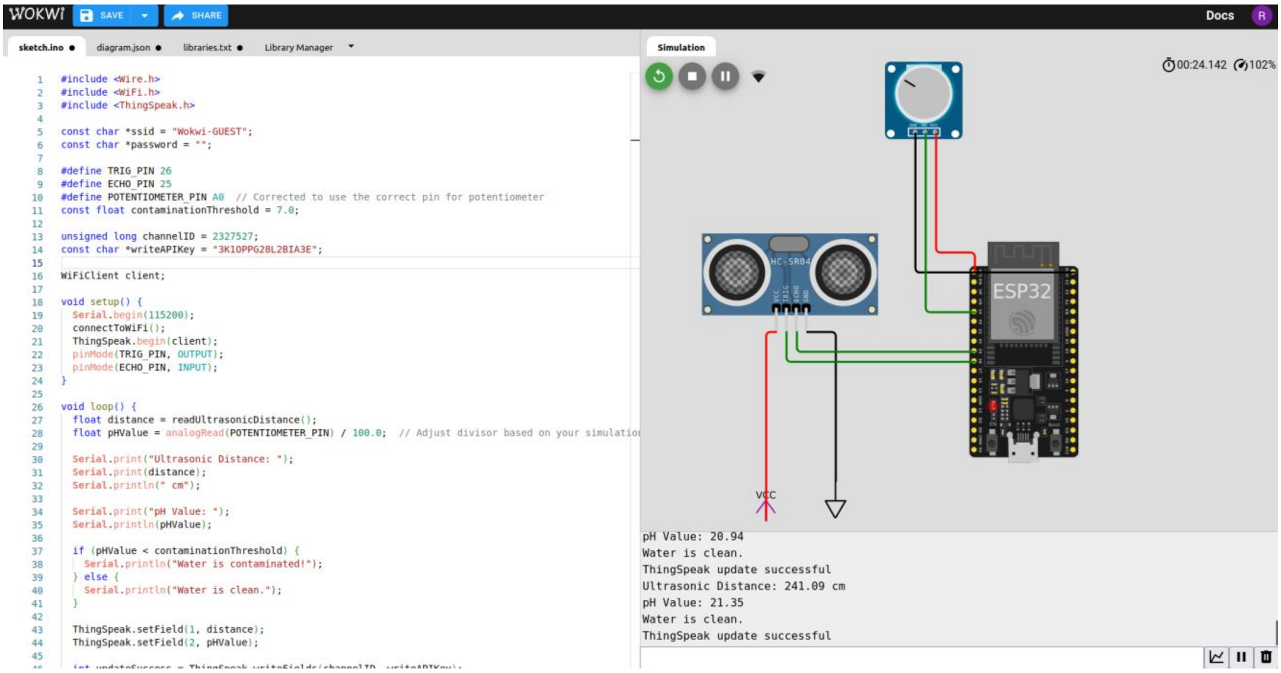
By Adjusting Distance in Ultrasonic Sensor:



```
#include <Wire.h>
#include <WiFi.h>
#include <ThingSpeak.h>

const char *ssid = "Wokwi-GUEST";
const char *password = "";

#define TRIG_PIN 26
#define ECHO_PIN 25
#define POTENTIOMETER_PIN A0  // Corrected to use the correct pin for potentiometer
const float contaminationThreshold = 7.0;

unsigned long channelID = 2327527;
const char *writeAPIKey = "3K1OPPG28L2BIA3E";

WiFiClient client;

void setup() {
  Serial.begin(115200);
  connectToWiFi();
  ThingSpeak.begin(client);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  float distance = readUltrasonicDistance();
  float pHValue = analogRead(POTENTIOMETER_PIN) / 100.0;  // Adjust divisor based on your simulatio

  Serial.print("Ultrasonic Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  Serial.print("pH Value: ");
  Serial.println(pHValue);

  if (pHValue < contaminationThreshold) {
    Serial.println("Water is contaminated!");
  } else {
    Serial.println("Water is clean.");
  }

  ThingSpeak.setField(1, distance);
  ThingSpeak.setField(2, pHValue);

  int updateSuccess = ThingSpeak.writeFields(channelID, writeAPIKey);
```

```
Ultrasonic Distance: 403.51 cm
pH Value: 23.73
Water is clean.
ThingSpeak update successful
Ultrasonic Distance: 241.09 cm
pH Value: 22.24
Water is clean.
```

By Adjusting Potentiometer:



```
#include <Wire.h>
#include <WiFi.h>
#include <ThingSpeak.h>

const char *ssid = "Wokwi-GUEST";
const char *password = "";

#define TRIG_PIN 26
#define ECHO_PIN 25
#define POTENTIOMETER_PIN A0  // Corrected to use the correct pin for potentiometer
const float contaminationThreshold = 7.0;

unsigned long channelID = 2327527;
const char *writeAPIKey = "3K1OPPG28L2BIA3E";

WiFiClient client;

void setup() {
  Serial.begin(115200);
  connectToWiFi();
  ThingSpeak.begin(client);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  float distance = readUltrasonicDistance();
  float pHValue = analogRead(POTENTIOMETER_PIN) / 100.0;  // Adjust divisor based on your simulatio

  Serial.print("Ultrasonic Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  Serial.print("pH Value: ");
  Serial.println(pHValue);

  if (pHValue < contaminationThreshold) {
    Serial.println("Water is contaminated!");
  } else {
    Serial.println("Water is clean.");
  }

  ThingSpeak.setField(1, distance);
  ThingSpeak.setField(2, pHValue);

  int updateSuccess = ThingSpeak.writeFields(channelID, writeAPIKey);
```

```
pH Value: 20.94
Water is clean.
ThingSpeak update successful
Ultrasonic Distance: 241.09 cm
pH Value: 21.35
Water is clean.
ThingSpeak update successful
```