Project

SMART WATER FOUNTAINS

Introduction:

Welcome to the world of smart water fountains! In this presentation, we will explore how these innovative fountains are revolutionizing hydration. Discover the potential of these intelligent devices to improve access to clean drinking water and promote healthy lifestyles. Get ready to dive into the future of hydration!

1.1 Project description:

***** Current water fountains lack smart features and fail to meet modern expectations. Many are outdated, unhygienic, and inconvenient. By revolutionizing hydration through smart water fountains, we can address these shortcomings and provide a better drinking experience for everyone.

***** Smart water fountains come packed with exciting features such as real-time water quality monitoring, touchless operation, integrated hydration tracking, and customizable water preferences. These advancements ensure a safe, convenient, and personalized drinking experience for users.

*****Smart water fountains play a crucial role in promoting sustainability. By encouraging the use of reusable bottles and reducing plastic waste, these fountains contribute to a greener environment. Additionally, they can be powered by renewable energy sources to minimize their carbon footprint.

***** With smart water fountains, the hydration experience is taken to new heights. Users can enjoy customizable water temperature, flavor infusions, and even interactive games while hydrating. These features make drinking water more exciting, encouraging people to stay hydrated throughout the day.

*****Smart water fountains are designed with accessibility in mind. They include features like wheelchair accessibility, multilingual interfaces, and audio instructions to ensure that everyone can easily access clean drinking water. These inclusive designs promote equal opportunities for hydration.

*****The data collected by smart water fountains provides valuable insights into hydration patterns and usage trends. Analyzing this data can help organizations and communities make informed decisions to improve water infrastructure, optimize maintenance, and promote healthier habits.

***** Implementing smart water fountains requires collaboration between technology companies, government agencies, and public institutions. By working together, we can ensure the widespread adoption of these innovative devices, creating a network of smart water fountains that benefits communities worldwide.

***** The future of smart water fountains holds endless possibilities. With advancements in artificial intelligence and Internet of Things, we can expect even more intelligent features like automatic bottle refills, personalized hydration reminders, and real-time water consumption data. Get ready for a future where staying hydrated is easier than ever!

2.1 Project implementation:

Select Sensors:

Choose appropriate sensors to monitor water level, quality, and temperature. For example, ultrasonic sensors for level, pH sensors for quality, and temperature sensors for temperature

Microcontroller or IoT Device:

Select a microcontroller or IoT device like Arduino, Raspberry Pi, or specialized IoT platforms with Wi-Fi, Bluetooth, or cellular connectivity.

Connectivity:

Connect the IoT device to the internet. You can use Wi-Fi, Ethernet, cellular, or LoRaWAN, depending on the deployment location and requirements.

Sensor Integration:

Connect and interface the selected sensors with the IoT device, ensuring data can be collected and transmitted.

DataProcessing:

Implement software to process sensor data. You can use programming languages like Python or C/C++ for this. Analyze data to check water quality, temperature, and level.

User Interface:

Create a user interface, which can be a web application, mobile app, or a dedicated device with a screen. This interface should display water-related information in a user-friendly manner.

Cloud Integration:

Send data to the cloud for storage and further analysis. Cloud platforms like AWS, Azure, or Google Cloud can be used for this purpose.

Data Analysis and Alerts: Set up alerts for abnormal water conditions. Implement algorithms to detect issues and send notifications or take automated actions.

Security:

Implement strong security measures to protect data and the device itself. Use encryption, authentication, and secure communication protocols.

Energy Management:

Optimize power usage by using low-power modes for sensors and IoT devices. Consider solar panels or battery backups for remote locations.

Remote Control:

If necessary, implement remote control features to adjust water flow or other fountain settings.

Maintenance and Updates:

Plan for firmware updates and regular maintenance to ensure the system's long-term reliability.

Testing and Deployment:

Thoroughly test the system in different scenarios before deploying it in the field. Ensure it can handle various environmental conditions.

Regulatory Compliance:

Ensure your smart water fountain complies with any local or regional regulations, especially if used in public spaces.

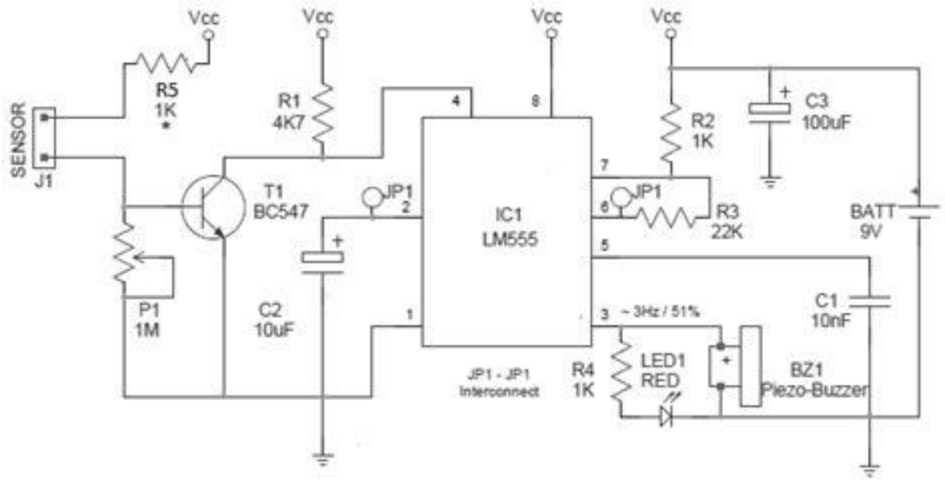
Documentation:

Document the entire system, including hardware components, software code, and data analysis algorithms for future reference and troubleshooting.

Scale and Expand:

Consider scalability and expandability in case you want to monitor multiple fountains or add more features in the future.

3.1 Circuit diagram:



Source code:

4.1 Front End :

```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto server url

WiFiClient espClient;
PubSubClient client(espClient);
```

```
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }}
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("Connected");
      client.publish("/ThinkIOT/Publish", "Welcome");
      client.subscribe("/ThinkIOT/Subscribe");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }}
}

void setup() {
  pinMode(LED,OUTPUT);
  digitalWrite(LED,HIGH);
  pinMode(MOTOR,OUTPUT);
  digitalWrite(MOTOR,LOW);
  Serial.begin(115200);
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
   pinMode(2, OUTPUT);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  dht.setup(DHT_PIN, DHTesp::DHT22);
}

void loop() {
  // Start a new measurement:
  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  // Read the result:
```

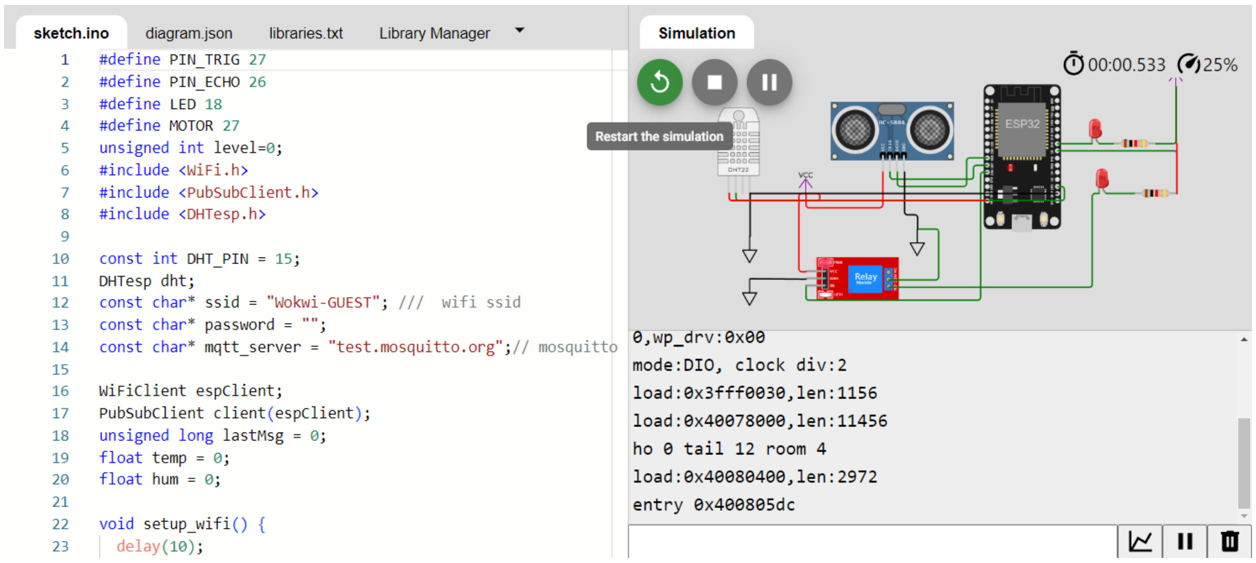
```
int duration = pulseIn(PIN_ECHO, HIGH);
Serial.print("Distance in CM: ");
Serial.println(duration / 58);
Serial.print("Distance in inches: ");
Serial.println(duration / 148);

level=duration/58;

if(level<100)
{
  digitalWrite(LED,HIGH);
}

else
{
  digitalWrite(LED,LOW);
}

if (!client.connected()) {
  reconnect();
}
client.loop();

unsigned long now = millis();
if (now - lastMsg > 2000) { //perintah publish data
  lastMsg = now;
  TempAndHumidity  data = dht.getTempAndHumidity();

  String temp = String(data.temperature, 2);
  client.publish("/Thinkitive/temp", temp.c_str()); // publish temp topic /ThinkIOT/temp
  String hum = String(data.humidity, 1);
  client.publish("/Thinkitive/hum", hum.c_str());   // publish hum topic /ThinkIOT/hum

  Serial.print("Temperature: ");
  Serial.println(temp);
  Serial.print("Humidity: ");
```

```
    Serial.println(hum);
  }

  delay(1000);
}
```

4.2 Back end :

```
{
  "version": 1,
  "author": "Kausik T",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -43.3, "left": 71.8, "attrs": {} },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic1",
      "top": -17.7,
      "left": -138.5,
      "attrs": { "distance": "322" }
    },
    { "type": "wokwi-gnd", "id": "gnd1", "top": 153.6, "left": -29.4, "attrs": {} },
    { "type": "wokwi-vcc", "id": "vcc1", "top": 77.56, "left": -182.4, "attrs": {} },
    { "type": "wokwi-relay-module", "id": "relay1", "top": 192.2, "left": -172.8, "attrs": {} },
    { "type": "wokwi-gnd", "id": "gnd2", "top": 220.8, "left": -259.8, "attrs": {} },
    { "type": "wokwi-dht22", "id": "dht1", "top": -9.3, "left": -293.4, "attrs": {} },
    { "type": "wokwi-gnd", "id": "gnd3", "top": 163.2, "left": -259.8, "attrs": {} },
    { "type": "wokwi-led", "id": "led1", "top": 63.6, "left": 215, "attrs": { "color": "red" } },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": 99.95,
      "left": 278.4,
      "attrs": { "value": "1000" }
    },
    { "type": "wokwi-vcc", "id": "vcc2", "top": -66.44, "left": 326.4, "attrs": {} },
    { "type": "wokwi-led", "id": "led2", "top": -3.6, "left": 205.4, "attrs": { "color": "red" } },
    {
      "type": "wokwi-resistor",
      "id": "r2",
      "top": 32.75,
      "left": 249.6,
      "attrs": { "value": "1000" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "ultrasonic1:TRIG", "esp:D27", "green", [ "v9.6", "h114.8", "v-9.6" ] ],
    [ "ultrasonic1:ECHO", "esp:D26", "green", [ "v19.2", "h95.2", "v-38.4" ] ],
    [ "ultrasonic1:GND", "gnd1:GND", "black", [ "v57.6", "h18" ] ],
    [ "vcc1:VCC", "ultrasonic1:VCC", "red", [ "v19.2", "h-28.8" ] ],
    [ "relay1:VCC", "vcc1:VCC", "red", [ "h-9.6", "v-105.6" ] ],
    [ "gnd2:GND", "relay1:GND", "black", [ "v0" ] ],
    [ "relay1:IN", "esp:D14", "green", [ "h0", "v19", "h240", "v-172.8" ] ],
    [ "dht1:GND", "gnd3:GND", "black", [ "v0" ] ],
    [ "dht1:VCC", "vcc1:VCC", "red", [ "v9.6", "h124.8", "v-9.6" ] ],
    [ "relay1:COM", "gnd1:GND", "green", [ "h68.4", "v-68.6", "h-28.8" ] ],
    [ "led1:A", "r1:1", "green", [ "v0" ] ],
    [ "r1:2", "esp:D5", "green", [ "v0" ] ],
    [ "relay1:NO", "led1:C", "green", [ "h279.6", "v-126.6" ] ],
    [ "vcc2:VCC", "r1:2", "red", [ "v76.8", "h1.2" ] ],
    [ "dht1:GND", "esp:GND.1", "black", [ "v0" ] ],
    [ "dht1:SDA", "esp:D15", "green", [ "v9.6", "h451.3", "v-19.2" ] ],
    [ "dht1:VCC", "esp:VIN", "red", [ "v0" ] ],
    [ "led2:A", "r2:1", "green", [ "v0", "h19.2" ] ],
    [ "esp:D18", "led2:C", "green", [ "h47.7" ] ],
    [ "r2:2", "vcc2:VCC", "green", [ "v0", "h27.6", "v0", "h0", "v0", "h0", "v0", "h0", "v0" ] ]
  ],
  "dependencies": {}
}
```

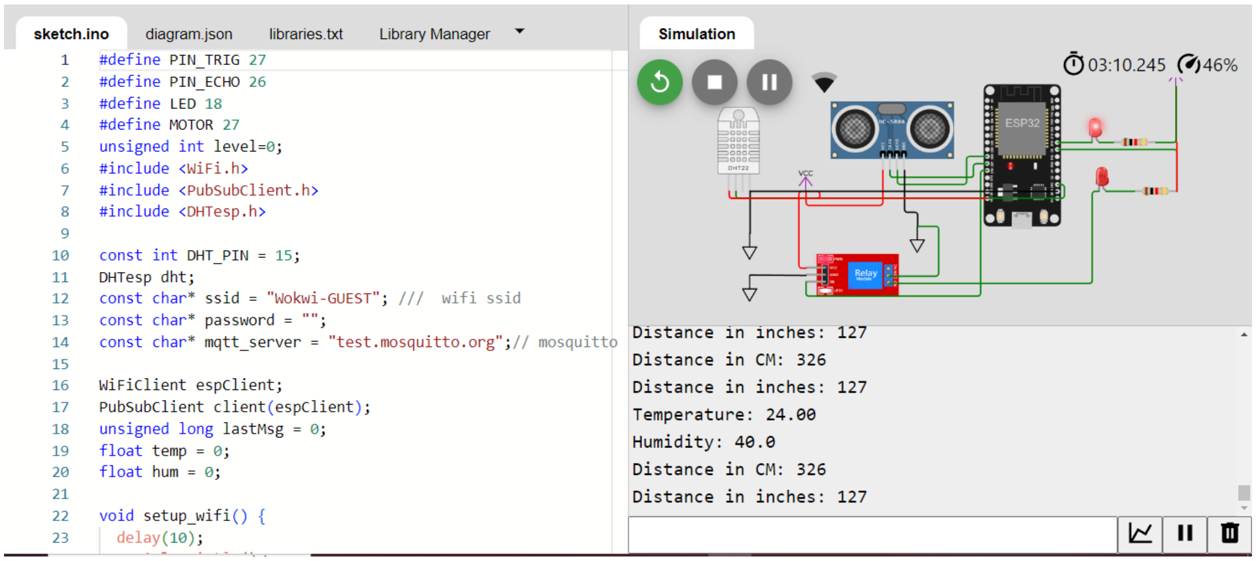
4.3 Sample output:



```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
```

```
0,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
```

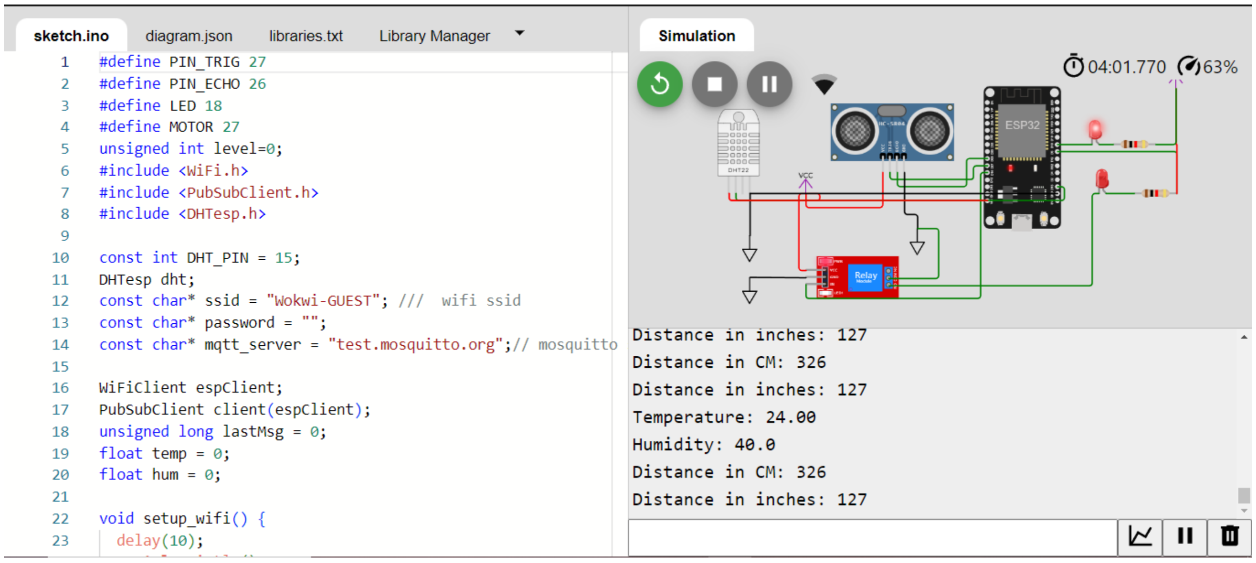
Fig:4.3.1



```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
```

```
Distance in inches: 127
Distance in CM: 326
Distance in inches: 127
Temperature: 24.00
Humidity: 40.0
Distance in CM: 326
Distance in inches: 127
```

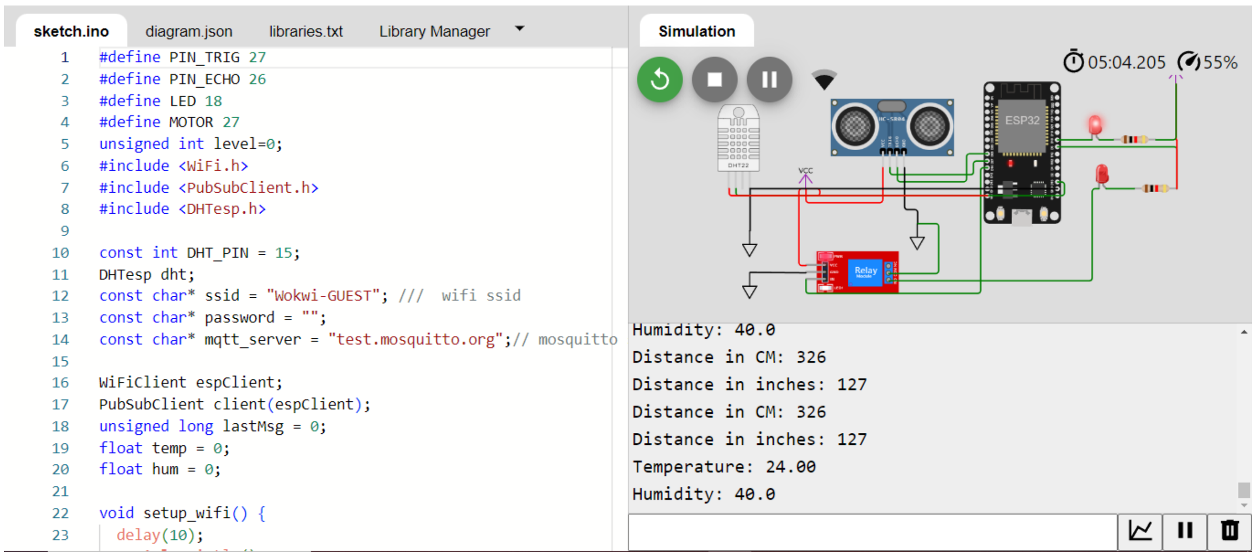
Fig:4.3.2



```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
```

```
Distance in inches: 127
Distance in CM: 326
Distance in inches: 127
Temperature: 24.00
Humidity: 40.0
Distance in CM: 326
Distance in inches: 127
```

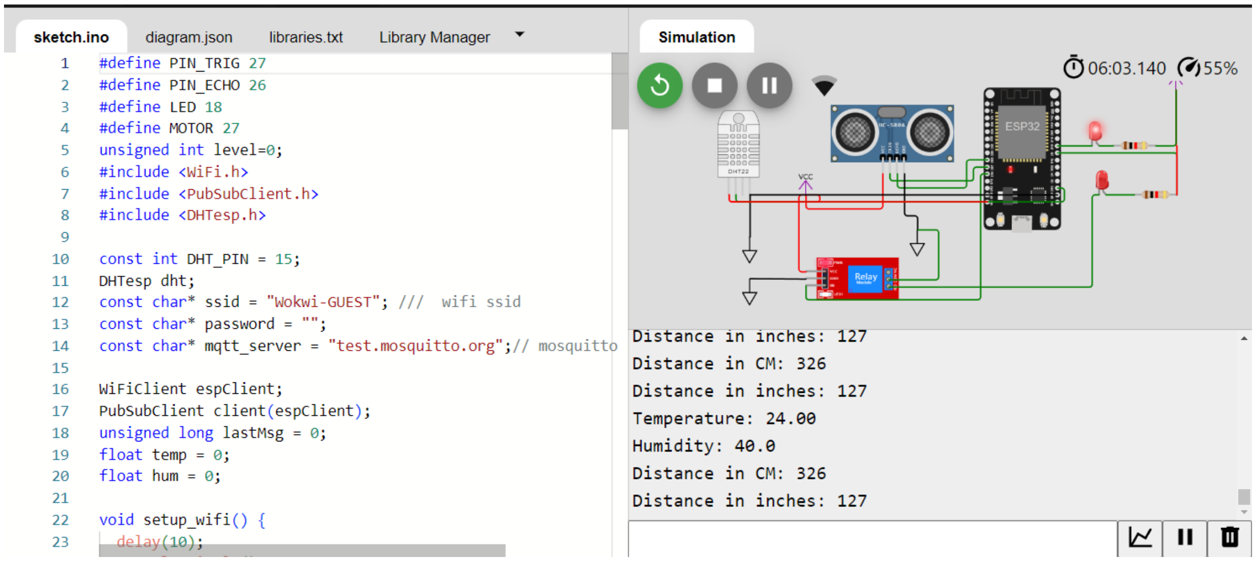
Fig:4.3.3



```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
```

```
Humidity: 40.0
Distance in CM: 326
Distance in inches: 127
Distance in CM: 326
Distance in inches: 127
Temperature: 24.00
Humidity: 40.0
```

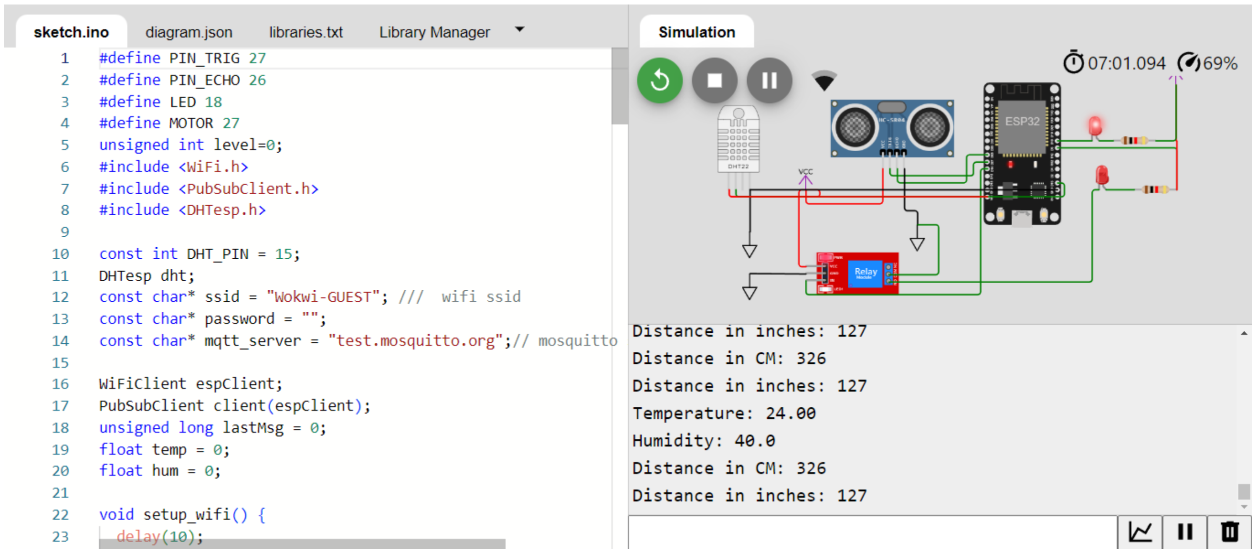
Fig:4.3.4



```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
```

```
Distance in inches: 127
Distance in CM: 326
Distance in inches: 127
Temperature: 24.00
Humidity: 40.0
Distance in CM: 326
Distance in inches: 127
```

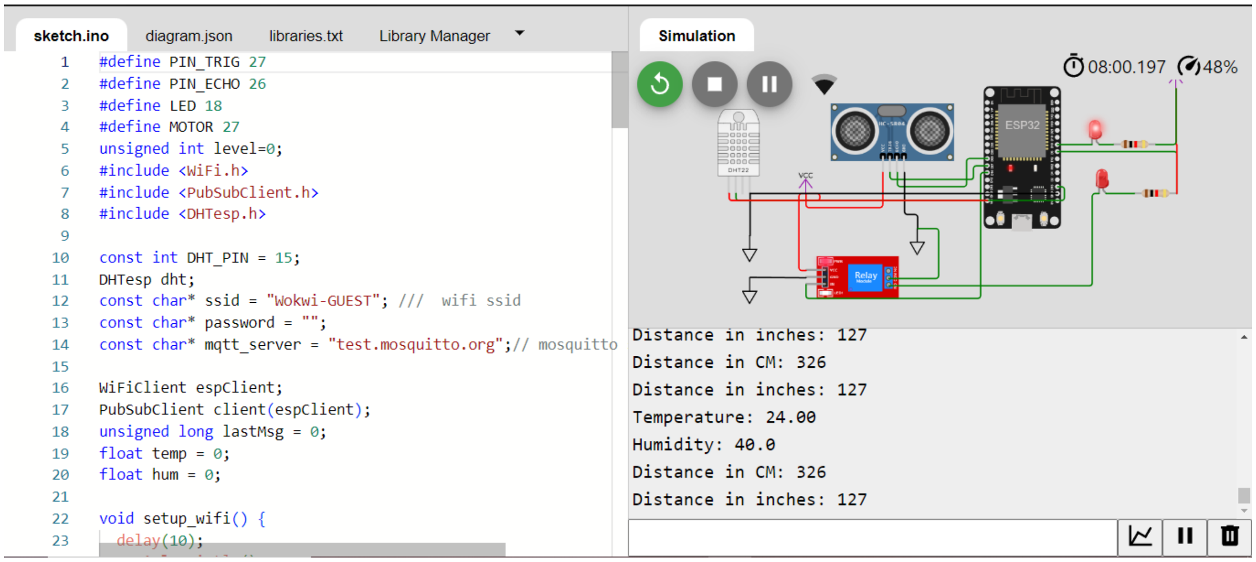
Fig:4.3.5



```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
```

```
Distance in inches: 127
Distance in CM: 326
Distance in inches: 127
Temperature: 24.00
Humidity: 40.0
Distance in CM: 326
Distance in inches: 127
```

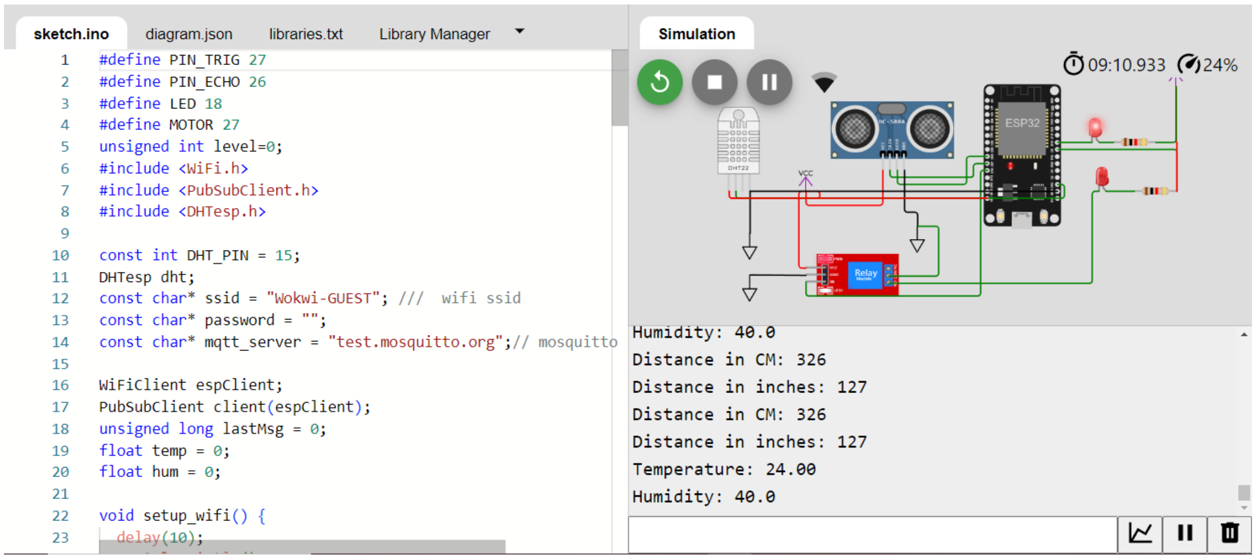
Fig:4.3.6



```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
```

```
Distance in inches: 127
Distance in CM: 326
Distance in inches: 127
Temperature: 24.00
Humidity: 40.0
Distance in CM: 326
Distance in inches: 127
```

Fig4.3.7



```
#define PIN_TRIG 27
#define PIN_ECHO 26
#define LED 18
#define MOTOR 27
unsigned int level=0;
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
```

```
Humidity: 40.0
Distance in CM: 326
Distance in inches: 127
Distance in CM: 326
Distance in inches: 127
Temperature: 24.00
Humidity: 40.0
```

Fig:4.3.8