

SPRINT-4

| | |
|--------------|---|
| Date | 15 November 2022 |
| Team ID | PNT2022TMID48307 |
| Project Name | Industry Specific Intelligence Fire Management System |

```
#include <WiFi.h>
#include <Wire.h>
#include <SPI.h>
#include "ThingSpeak.h"
#include <WiFiClient.h>          unsigned long myChannelNumber = 2;  const
char * myWriteAPIKey =
"25V40ZAPI6KIZFGY";
int LED_PIN = 32;
// the current reading from the input pin    int
BUZZER_PIN= 12;
const int  mq2 = 4;
int  value = 0;

//Flame int flame_sensor_pin = 10 ;
// initializing pin 10 as the sensor digital output pin int  flame_pin = HIGH ; //
current state of sensor

char      ssid[]      =
"BOOMIKA;
charpass[]=
"BOOMIKA";
    WiFiClient client;
#define PIN_LM35 39
#define ADC_VREF_mV 3300.0
#define ADC_RESOLUTION 4096.0
#define RELAY_PIN  17
```

```
#define RELAY_PIN1 27
```

```
void setup()
```

```
{
```

```
  Serial.begin(115200);
```

```
  pinMode(RELAY_PIN, OUTPUT);
```

```
  pinMode(RELAY_PIN1, OUTPUT);
```

```
  Serial.print("Connecting to ");
```

```
  Serial.println(ssid);  WiFi.begin(ssid, pass);
```

```
  int wifi_ctr = 0;
```

```
  while (WiFi.status() != WL_CONNECTED)
```

```
  {
```

```
    delay(1000);
```

```
  Serial.print(".");
```

```
  }
```

```
  Serial.println("WiFi connected");
```

```
  ThingSpeak.begin(client);  pinMode(LED_PIN, OUTPUT);  pinMode(mq2, INPUT);
```

```
  pinMode ( flame_sensor_pin , INPUT );
```

```
  // declaring sensor pin as input pin for Arduino  pinMode(BUZZER_PIN, OUTPUT);
```

```
}
```

```
void temperature()
```

```
{
```

```
  int adcVal = analogRead(PIN_LM35);  float
```

```
  milliVolt = adcVal *
```

```
  (ADC_VREF_mV / ADC_RESOLUTION);
```

```
  float tempC = milliVolt /10;
```

```
  Serial.print("Temperature: ");
```

```
  Serial.print(tempC);  Serial.print("°C");  if(tempC > 60)
```

```
  {
```

```
    Serial.println("Alert");
```

```
    digitalWrite(BUZZER_PIN, HIGH);
```

```
  // turn on
```

```
  } else
```

```

    {
        digitalWrite(BUZZER_PIN, LOW);
    // turn on
    }
    int x = ThingSpeak.writeField(myChannelNumber,1, tempC, myWriteAPIKey);  }

void GasSensors()
{
    //mq2

    int gassensorAnalogmq2 = analogRead(mq2);
    Serial.print("mq2 Gas Sensor: ");
    Serial.print(gassensorAnalogmq2);
    Serial.print("\t");
    Serial.print("\t");
    Serial.print("\t");

    if (gassensorAnalogmq2 > 1500)
    {
        Serial.println("mq2Gas");    Serial.println("Alert");
        digitalWrite(RELAY_PIN1, HIGH);
        // turn on fan 10 seconds delay(100);
    } else
    {
        Serial.println("No mq2Gas");
        digitalWrite(RELAY_PIN1,
        LOW);
        // turn off fan 10 seconds delay(100);

    }

    int a = ThingSpeak.writeField(myChannelNumber,4, gassensorAnalogmq2,
    myWriteAPIKey);

}

```

```

void flaresensor()
{ flame_pin = digitalRead ( flame_sensor_pin ) ;
// reading from the sensor if
(flame_pin == LOW )
// applying condition
{
Serial.println ( " ALERT: FLAME IS DETECTED" ) ;
digitalWrite (BUZZER_PIN, HIGH ) ;
// if state is high, then turn high the BUZZER
} else
{
Serial.println ( " NO FLAME DETECTED " ) ;
digitalWrite (BUZZER_PIN , LOW ) ;
// otherwise turn it low
} int value = digitalRead(flame_sensor_pin);
// read the analog value from sensor

    if (value ==LOW)
    {
Serial.print("FLAME");
digitalWrite(RELAY_PIN, HIGH);
    } else
    {
        Serial.print("NO FLAME");
digitalWrite(RELAY_PIN, LOW);
    }

} void loop() {
temperature();
GasSensors();
flaresensor();
}

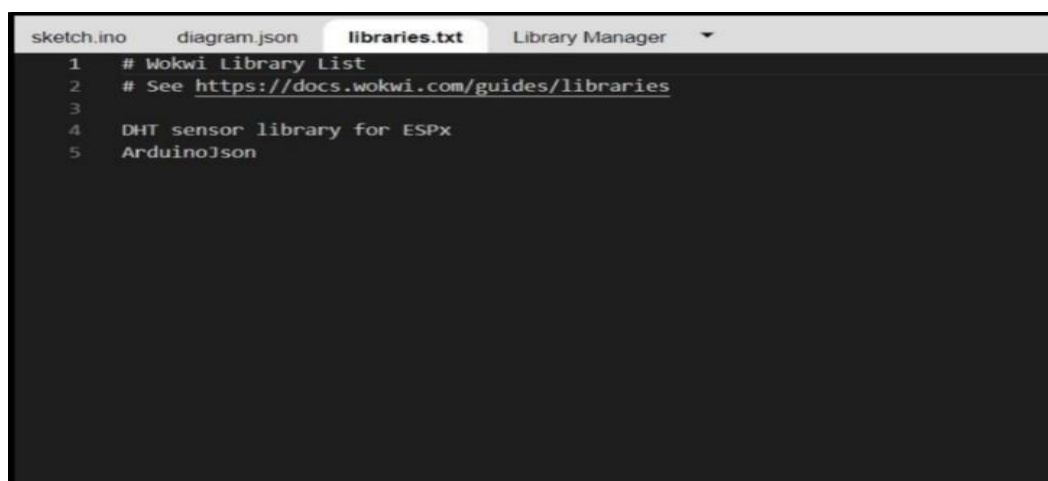
```

DIAGRAM. JSON



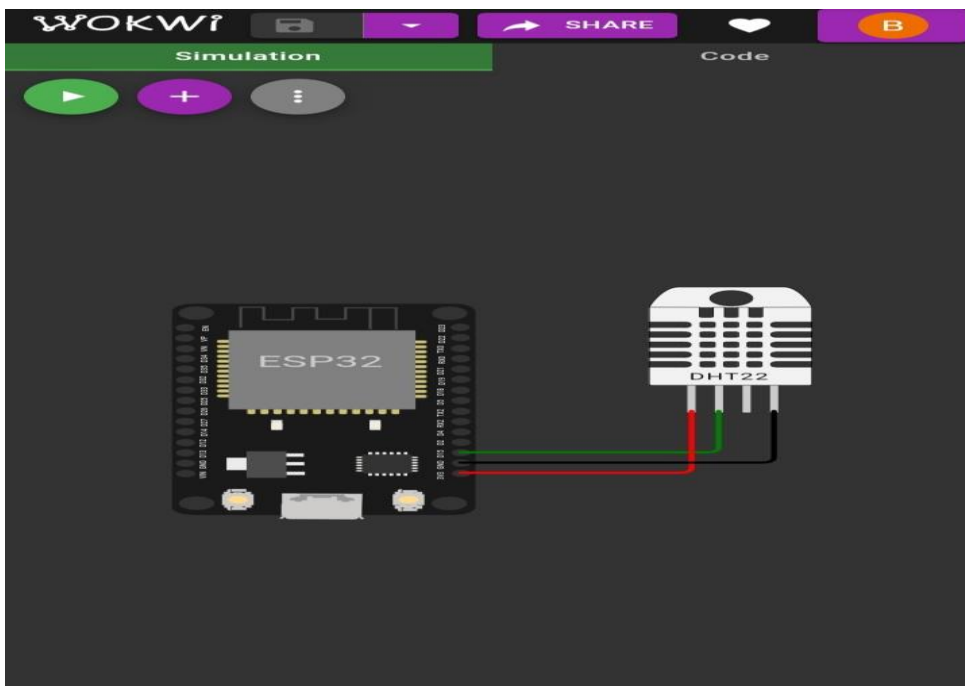
```
{
  "version": 1,
  "author": "PNT2022IMID48307",
  "editor": "wokwi",
  "parts": [
    {
      "type": "wokwi-esp32-devkit-v1",
      "id": "esp",
      "top": -16.32,
      "left": -0.62,
      "attrs": {}
    },
    {
      "type": "wokwi-dht22",
      "id": "dht1",
      "top": -30.22,
      "left": 165.89,
      "attrs": {
        "temperature": "59.3"
      }
    }
  ],
  "connections": [
    [
      "esp:TX0",
      "serialMonitor:RX",
      "",
      []
    ],
    [
      "esp:RX0",
      "serialMonitor:TX",
      "",
      []
    ],
    [
      "dht1:SDA",
      "esp:D15",
      "green",
      ["v0"]
    ],
    [
      "dht1:VCC",
      "esp:3v3",
      "red",
      ["v0"]
    ],
    [
      "dht1:GND",
      "esp:GND.1",
      "black",
      ["v0"]
    ]
  ]
}
```

LIBRARIES



```
1 # Wokwi Library List
2 # See https://docs.wokwi.com/guides/libraries
3
4 DHT sensor library for ESPx
5 ArduinoJson
```

CIRCUIT



WOKWI

SAVE

SHARE

Docs

sketch.ino

diagram.json

libraries.txt

Library Manager

Simulation

```

1  #include "DHTesp.h"
2  #include <stdlib.h>
3  #include <time.h>
4
5  const int DHT_PIN = 15;
6
7  bool is_exhaust_fan_on = false;
8  bool is_sprinkler_on = false;
9
10 float temperature = 0;
11
12 int gas_ppm = 0;
13 int flame = 0;
14 int flow = 0;
15
16 String flame_status = "";
17 String accident_status = "";
18 String sprinkler_status = "";
19
20 DHTesp dhtSensor;
21
22
23 void setup() {
24   Serial.begin(999000);
25
26   /* sensor pin setups */
27   dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
28   /*if real gas sensor is used make sure the sen
29   /*
30   // Here random values for readings and stdout
31   // working of the devices as physical or sim
32   // available.
33   */
34 }
35
36 void loop() {
37
38   TempAndHumidity data = dhtSensor.getTempAndHu
39
40   //setting a random seed
41   srand(time(0));
42
43   //initial variable activities like declaring
44   temperature = data.temperature;
45   gas_ppm = rand()%1000;
46   int flamerange = rand()%1024;
47   flame = map(flamerange,0,1024,0,1024);
48   int flamerange = map(flamerange,0,1024,0,3);
49   int flow = ((rand()%100)>50?1:0);
50
51   //set a flame status based on how close it is.
52   switch (flamerange) {
53     case 2: // A fire closer than 1.5 feet away
54       flame_status = "Close Fire";
55       break;
56     case 1: // A fire between 1-3 feet away.
57       flame_status = "Distant Fire";
58       break;
59     case 0: // No fire detected.
60       flame_status = "No Fire";
61       break;
62   }
63
64   //toggle the fan according to gas in ppm in th
65   if(gas_ppm > 100){
66     is_exhaust_fan_on = true;
67   }
68   else{
69     is_exhaust_fan_on = false;
70   }
71
72   //find the accident status 'cause fake alert m
73   if(temperature < 40 && flamerange ==2){
74     accident_status = "need auditing";
75     is_sprinkler_on = false;
76   }
77   else if(temperature < 40 && flamerange ==0){
78     accident_status = "not found";
79     is_sprinkler_on = false;
80   }
81   else if(temperature > 50 && flamerange == 1){
82     is_sprinkler_on = true;
83     accident_status = "moderate";
84   }
85   else if(temperature > 55 && flamerange == 2){
86     is_sprinkler_on = true;
87     accident_status = "severe";
88   }
89   else{
90     is_sprinkler_on = false;
91     accident_status = "none";
92   }
93
94   //send the sprinkler status
95   if(is_sprinkler_on){
96     if(flow){
97       sprinkler_status = "working";
98     }
99   }
100   else{

```

OUTPUT

WOKWI

SAVE SHARE

Docs

sketch.ino diagram.json libraries.txt Library Manager

Simulation

00:05.733 33%

```
1 #include "DHTesp.h"
2 #include <stdlib.h>
3 #include <time.h>
4
5 const int DHT_PIN = 15;
6
7 bool is_exhaust_fan_on = false;
8 bool is_sprinkler_on = false;
9
10 float temperature = 0;
11
12 int gas_ppm = 0;
13 int flame = 0;
14 int flow = 0;
15
16 String flame_status = "";
17 String accident_status = "";
18 String sprinkler_status = "";
19
20 DHTesp dhtSensor;
21
22
23 void setup() {
24   Serial.begin(999000);
25
26   /* sensor pin setups */
27   dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
28   /* if real gas sensor is used make sure the sensor is
29    * working of the devices as physical or simulation
30    * available.
31    */
32 }
33
34 void loop() {
35   TempAndHumidity data = dhtSensor.getTempAndHumidity();
36
37   //setting a random seed
38   srand(time(0));
39
40   //initial variable activities like declaring
41   temperature = data.temperature;
42   gas_ppm = rand()%1000;
43   int flamerange = rand()%1024;
44   flame = map(flamerange, 0, 1024, 0, 1024);
45   int flamerange = map(flamerange, 0, 1024, 0, 3);
46   int flow = ((rand()%100)>50)?1:0;
47
48   //set a flame status based on how close it is.
49   switch (flamerange) {
50     case 2: // A fire closer than 1.5 feet away
51       flame_status = "Close Fire";
52       break;
53     case 1: // A fire between 1-3 feet away.
54       flame_status = "Distant Fire";
55       break;
56     case 0: // No fire detected.
57       flame_status = "No Fire";
58       break;
59   }
60
61   //toggle the fan according to gas in ppm in the room
62   if(gas_ppm > 100){
63     is_exhaust_fan_on = true;
64   }
65   else{
66     is_exhaust_fan_on = false;
67   }
68
69   //find the accident status 'cause fake alert messages
70   if(temperature < 40 && flamerange == 2){
71     accident_status = "need auditing";
72     is_sprinkler_on = false;
73   }
74   else if(temperature < 40 && flamerange == 0){
75     accident_status = "not found";
76     is_sprinkler_on = false;
77   }
78   else if(temperature > 50 && flamerange == 1){
79     is_sprinkler_on = true;
80     accident_status = "moderate";
81   }
82   else if(temperature > 55 && flamerange == 2){
83     is_sprinkler_on = true;
84     accident_status = "severe";
85   }
86   else{
87     is_sprinkler_on = false;
88     accident_status = "none";
89   }
90
91   //send the sprinkler status
92   if(is_sprinkler_on){
93     if(flow){
94       sprinkler_status = "working";
95     }
96     else{
97       sprinkler_status = "not working";
98     }
99   }
100 }
```

ESP32

DHT22

messages: {
 "fire_status": "Close Fire",
 "flow_status": "not working",
 "accident_status": "severe",
}