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 flamesensor()
{  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();
flamesensor();
}
```

DIAGRAM. JSON

DIAGRAM.JSON

sketch.ino | diagram.json | libraries.txt | Library Manager

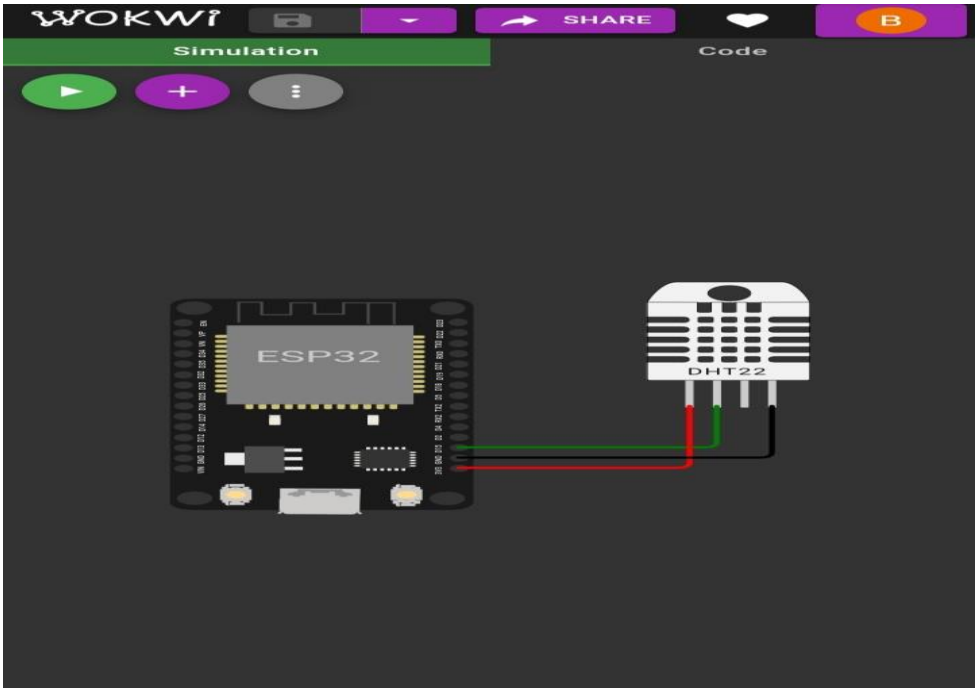
```
{
"version":1,
    "authour":"PNT2022TMID48307",
"editor":"wokwi",
"parts":[
{"type":"wokwi-esp32-devkit-
v1","id":"esp","top":-16.32,"left":-
0.52,"attrs":{},
{
"type":"wokwi-dhd22",
"id":"dhd1",
"top":-30.22,
"left":165.89
"attrs":{"temperature":"59.3"}
}
],
"connections":[
["esp:TX0","$serialMonitor:RX","",[]],
["esp:RX0","$serialMonitor:TX","",[]],
["dht1:SDA","esp:D15","green",["v0"]],
["dht1:VCC","esp:3v3","red",["v0"]],
["dht1:GND","esp:GND.1","black",["v0"]],
]
}
```

LIBRARIES

sketch.ino | diagram.json | **libraries.txt** | Library Manager

```
# Wokwi Library List
# See https://docs.wokwi.com/guides/libraries

DHT sensor library for ESPx
ArduinoJson
```

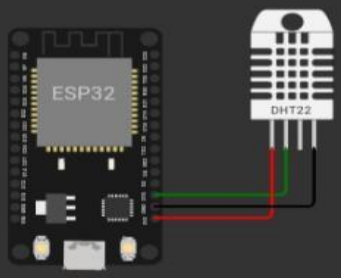
CIRCUIT



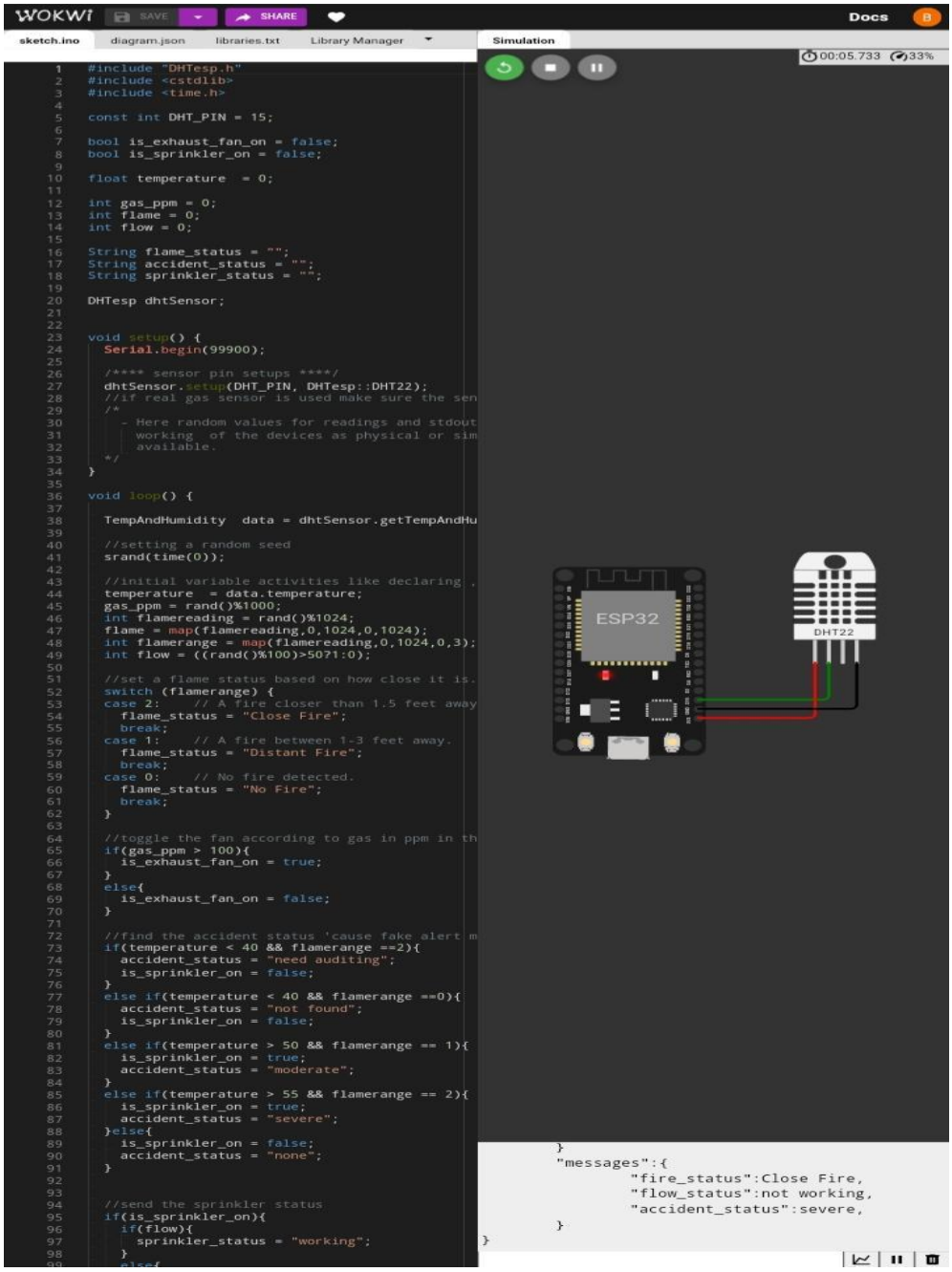
WOKWI SAVE SHARE Docs

sketch.ino | diagram.json | libraries.txt | Library Manager | Simulation

```
#include "DHTesp.h"
#include <cstdlib>
#include <time.h>

const int DHT_PIN = 15;

bool is_exhaust_fan_on = false;
bool is_sprinkler_on = false;

float temperature  = 0;

int gas_ppm = 0;
int flame = 0;
int flow = 0;

String flame_status = "";
String accident_status = "";
String sprinkler_status = "";

DHTesp dhtSensor;


void setup() {
  Serial.begin(99900);

  /**** sensor pin setups ****/
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  //if real gas sensor is used make sure the sen
  /*
    - Here random values for readings and stdout
      working  of the devices as physical or sim
      available.
  */
}

void loop() {

  TempAndHumidity  data = dhtSensor.getTempAndHu

  //setting a random seed
  srand(time(0));

  //initial variable activities like declaring ,
  temperature  = data.temperature;
  gas_ppm = rand()%1000;
  int flamereading = rand()%1024;
  flame = map(flamereading,0,1024,0,1024);
  int flamerange = map(flamereading,0,1024,0,3);
  int flow = ((rand()%100)>50?1:0);

  //set a flame status based on how close it is.
  switch (flamerange) {
  case 2:    // A fire closer than 1.5 feet away
    flame_status = "Close Fire";
    break;
  case 1:    // A fire between 1-3 feet away.
    flame_status = "Distant Fire";
    break;
  case 0:    // No fire detected.
    flame_status = "No Fire";
    break;
  }

  //toggle the fan according to gas in ppm in th
  if(gas_ppm > 100){
    is_exhaust_fan_on = true;
  }
  else{
    is_exhaust_fan_on = false;
  }

  //find the accident status 'cause fake alert m
  if(temperature < 40 && flamerange ==2){
    accident_status = "need auditing";
    is_sprinkler_on = false;
  }
  else if(temperature < 40 && flamerange ==0){
    accident_status = "not found";
    is_sprinkler_on = false;
  }
  else if(temperature > 50 && flamerange == 1){
    is_sprinkler_on = true;
    accident_status = "moderate";
  }
  else if(temperature > 55 && flamerange == 2){
    is_sprinkler_on = true;
    accident_status = "severe";
  }else{
    is_sprinkler_on = false;
    accident_status = "none";
  }


  //send the sprinkler status
  if(is_sprinkler_on){
    if(flow){
      sprinkler_status = "working";
    }
    else{
```



OUTPUT




```
#include "DHTesp.h"
#include <cstdlib>
#include <time.h>

const int DHT_PIN = 15;

bool is_exhaust_fan_on = false;
bool is_sprinkler_on = false;

float temperature  = 0;

int gas_ppm = 0;
int flame = 0;
int flow = 0;

String flame_status = "";
String accident_status = "";
String sprinkler_status = "";

DHTesp dhtSensor;


void setup() {
  Serial.begin(99900);

  /**** sensor pin setups ****/
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  //if real gas sensor is used make sure the sen
  /*
    - Here random values for readings and stdout
      working  of the devices as physical or sim
      available.
  */
}

void loop() {

  TempAndHumidity  data = dhtSensor.getTempAndHu

  //setting a random seed
  srand(time(0));

  //initial variable activities like declaring ,
  temperature  = data.temperature;
  gas_ppm = rand()%1000;
  int flamereading = rand()%1024;
  flame = map(flamereading,0,1024,0,1024);
  int flamerange = map(flamereading,0,1024,0,3);
  int flow = ((rand()%100)>5071:0);

  //set a flame status based on how close it is.
  switch (flamerange) {
  case 2:    // A fire closer than 1.5 feet away
    flame_status = "Close Fire";
    break;
  case 1:    // A fire between 1-3 feet away.
    flame_status = "Distant Fire";
    break;
  case 0:    // No fire detected.
    flame_status = "No Fire";
    break;
  }

  //toggle the fan according to gas in ppm in th
  if(gas_ppm > 100){
    is_exhaust_fan_on = true;
  }
  else{
    is_exhaust_fan_on = false;
  }

  //find the accident status 'cause fake alert m
  if(temperature < 40 && flamerange ==2){
    accident_status = "need auditing";
    is_sprinkler_on = false;
  }
  else if(temperature < 40 && flamerange ==0){
    accident_status = "not found";
    is_sprinkler_on = false;
  }
  else if(temperature > 50 && flamerange == 1){
    is_sprinkler_on = true;
    accident_status = "moderate";
  }
  else if(temperature > 55 && flamerange == 2){
    is_sprinkler_on = true;
    accident_status = "severe";
  }else{
    is_sprinkler_on = false;
    accident_status = "none";
  }


  //send the sprinkler status
  if(is_sprinkler_on){
    if(flow){
      sprinkler_status = "working";
    }
```