# Experiment 7

**Aim:** Write a program to interface smoke sensor with Arduino and give an alert message when smoke is detected.

## Description:

Smoke in the atmosphere can be detected using the MQ-2 Gas sensor.

| Pin No. | Pin Name | Pin Purpose |
|---|---|---|
| 1. | VCC | Need +5V Voltage |
| 2. | GND | Need to be grounded |
| 3. | Digital Out | No Connection (NC) |
| 4. | Analog out | Analog Voltage out |


The voltage provided by this sensor changes in direct proportion to the amount of smoke in the atmosphere.

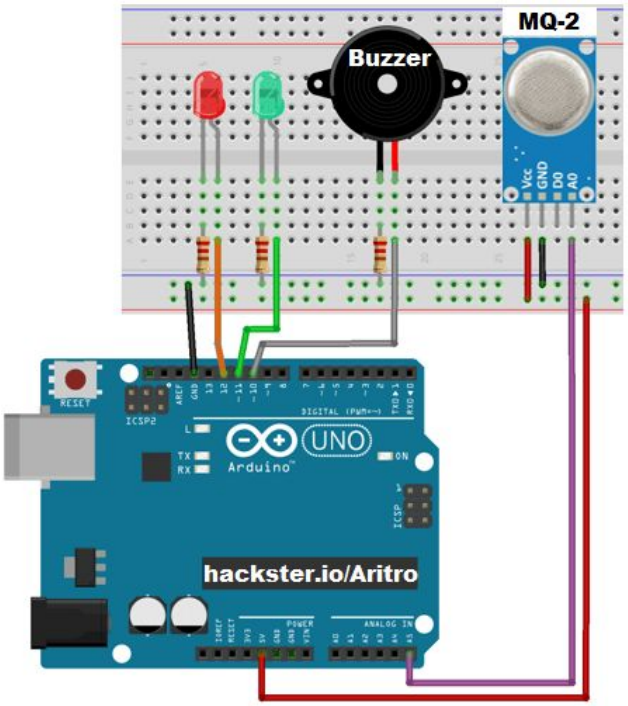
Thus, it can be used to detect smoke levels above a certain threshold and ring an alarm.

## Program:

```
int LED1 = 12;
int LED2 = 11;
int buzzer = 10;
int smokeA0 = A5;
int sensorThreshold = 400;

void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(smokeA0, INPUT);
  Serial.begin(9600);
}

void loop() {
  int analogSensor = analogRead(smokeA0);

  Serial.print("Pin A0: ");
  Serial.println(analogSensor);
  // Checks if it has reached the threshold value
  if (analogSensor > sensorThreshold)
  {
```

```
23      digitalWrite(LED1, HIGH);
24      digitalWrite(LED2, LOW);
25      tone(buzzer, 1000, 200);
26    }
27    else
28    {
29      digitalWrite(LED1, LOW);
30      digitalWrite(LED2, HIGH);
31      noTone(buzzer);
32    }
33    delay(100);
```

## Output:




## Conclusion:

Thus, we have successfully developed an IoT system that can detect and warn fire hazards.

# Experiment 9

**Aim:** Write a program to send and receive messages using MQTT protocol.

## Description:

MQTT stands for Message Queue Telemetry Transport.
It was created in 1999 by IBM designed specifically towards low power, bandwidth and low computational messaging for things such as embedded devices .

MQTT consists of three entities:

**MQTT Publisher:** This entity transmits the data to an MQTT broker. Each datapoint (such as temperature, humidity) is called a topic.

**MQTT Subscriber:** This entity receives the data. It subscribes to the data it requires and obtains them.

**MQTT Broker:** This entity mediates the transfer between the Publisher and Subscriber. It takes care of things such as queueing the data until subscriber gathers it, catering multiple subscribers etc.

### Libraries and Code Used:

`ArduinoMqttClient.h`: Library that contains classes and methods to utilize MQTT in the Arduino Programming environment.

Methods:

`connect(broker, port)`: Initiates MQTT connection to given broker and port.

`ConnectError()`: Displays any errors that occur during MQTT connection.

`poll()`: Used to make the MQTT connection alive.

`beginMessage(topic)`: Picks given topic to transmit message.

`print()`: Message to be transmitted.

`endMessage()`: Ends the message.

`onMessage(func)`: Invokes given function on receiving MQTT message.

`subscribe(topic)`: Subscribes to given topic.

`read()`: Returns one byte of message from selected topic.

## Program:

Publisher:

```
#include <WiFi.h>
#include <ArduinoMqttClient.h>
#include <DHTesp.h>

const char ssid[] = "Wokwi-GUEST";
const char pass[] = "";

WiFiClient wifiClient; // provided by WiFi.h
MqttClient mqttClient(wifiClient); // by ArduinoMqttClient.h
DHTesp dht; // by DHTesp.h

const char broker[] = "test.mosquitto.org"; // broker's url
const int port = 1883; // broker's port

const char topic0[] = "c5tempValue";
const char topic1[] = "c5humValue";

const int dhtPin = 19;
const int led = 22;

void setup(){
Serial.begin(9600);
pinMode(led, OUTPUT);

pinMode(dhtPin, INPUT);
dht.setup(dhtPin, DHTesp::DHT22);

Serial.print("Connecting to WiFi:");
WiFi.begin(ssid, pass, 6); // 6 means WiFi Channel 6
while ( WiFi.status() != WL_CONNECTED){
Serial.print('.'); // prints dots till it connects to wifi
delay(2000);
}
Serial.println("Connected!");

Serial.println("Connecting to MQTT Broker...");
if( !mqttClient.connect(broker, port)){
Serial.print("Connection to broker failed!: ");
Serial.println(mqttClient.connectError());
}
else
Serial.println("Connected to MQTT Broker!");
}

void loop(){
mqttClient.poll();

int temp = dht.getTemperature();
int hum = dht.getHumidity();

mqttClient.beginMessage(topic0);
mqttClient.print(temp);
mqttClient.endMessage();

mqttClient.beginMessage(topic1);
```

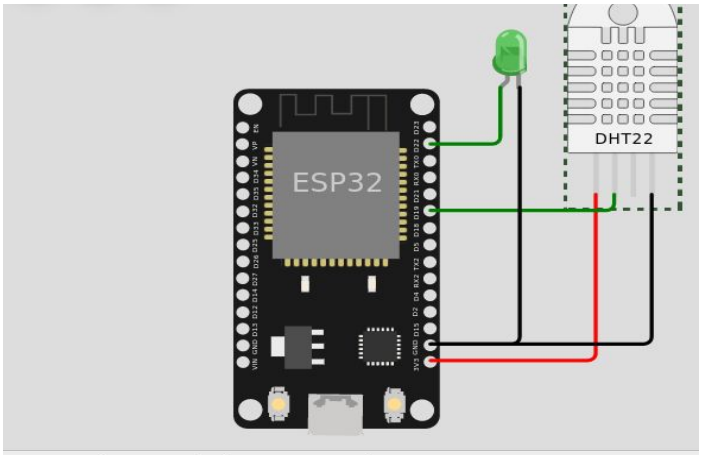
```
mqttClient.print(hum);
mqttClient.endMessage();

Serial.printf("Sent values: %d'C %d%% \n", temp, hum);
delay(2000);

digitalWrite(led, HIGH);
delay(600);
digitalWrite(led, LOW);
}
```

Subscriber:

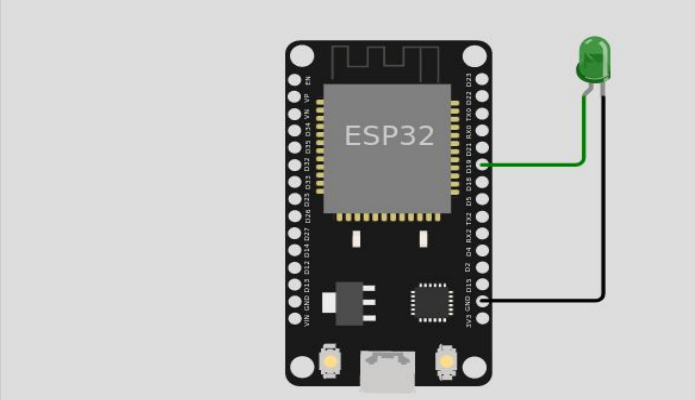
```
#include <WiFi.h>
#include <ArduinoMqttClient.h>

const char ssid[] = "Wokwi-GUEST";
const char pass[] = "";

WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

const char broker[] = "test.mosquitto.org";
const int port = 1883;

const char topic0[] = "c5tempValue";
const char topic1[] = "c5humValue";

const int led = 19;
void setup(){
  Serial.begin(9600);
  pinMode(led, OUTPUT);

  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, pass, 6);
  while ( WiFi.status() != WL_CONNECTED){
    Serial.print('.');
    delay(2427);
  }
  Serial.println("Connected!");

  if ( !mqttClient.connect(broker, port)){
    Serial.print("Error connecting to broker: ");
    Serial.println(mqttClient.connectError());
  }
  else
    Serial.println("Connected to Broker!");


  mqttClient.onMessage(onMqttMessage);
  mqttClient.subscribe(topic0);
  mqttClient.subscribe(topic1);
  Serial.printf("Subscribed to: %s  %s \n", topic0, topic1);
}

void loop(){
  mqttClient.poll();
}
```

```
void onMqttMessage(int messageSize) {
  Serial.printf("Topic %s has sent: ",mqttClient.messageTopic());

  while (mqttClient.available())
    Serial.print((char)mqttClient.read());
  Serial.println();
  digitalWrite(led, HIGH);
  delay(600);
  digitalWrite(led, LOW);
}
```

## Output:




```
Connecting to WiFi:.Connected!
Connecting to MQTT Broker...
Connected to MQTT Broker!
Sent values: -6'C   48%
Sent values: -6'C   48%
Sent values: 64'C   48%
```




```
Connecting to WiFi.Connected!
Connected to Broker!
Subscribed to: c5tempValue  c5humValue
Topic c5tempValue has sent: -6
Topic c5humValue has sent: 48
```

**Conclusion:** Thus, we have succesfully transmitted data between two boards using the MQTT protocol, using Mosquitto service as a broker.

# Experiment 10+11

**Aim:** Write programs to send and receive sensor data to the cloud.

## Description:

The word “Cloud” simply means a computer connected to the internet.

In the IoT context, a cloud acts as a centralized entity that coordintes the network of IoT devices, allowing them to work together, while also acting as the storage and processing center.

In this experiment, we will be using an ESP32 microcontroller to send temperature data from different locations to a server.

## Program:

Sender:

```
// Sensor in Hyderabad
// Sender
#include <WiFi.h>
#include <HTTPClient.h>
#include "DHTesp.h"
const int sensorPin=22, led=33;
DHTesp dht;
HTTPClient http;
const String
url="http://packetracer123cisco.000webhostapp.com/db.php?
key=top_secret&action=insert&Loc=Hyderabad&Temp=";

void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
  pinMode(sensorPin, INPUT);
  //initializing the dht instance with the correct model
(DHT22)
  dht.setup(sensorPin,DHTesp::DHT22);
  Serial.print("Connecting to WiFi");
  WiFi.begin("Wokwi-GUEST", "",6); //6 means WiFi channel 6
  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }
  Serial.println(" Connected!");
  Serial.print("Fetching " + url + "... ");
```

```
}
int oldTemp=555; //setting some garbage initial value
int temp;
void loop() {
  int temp=dht.getTemperature();
  delay(3000);
  if (temp==oldTemp) return; //so that data gets updated only
when temp changes
  oldTemp=temp;
  http.begin(url+temp);
  int httpResponseCode = http.GET();
  if (httpResponseCode > 0) {
    Serial.println(http.getString());  //prints response
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
  }
  else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
}
```

Receiver:

```
// Receiver's end

#include <WiFi.h>
#include <HTTPClient.h>
HTTPClient http; //creates an instance called http

const String
url="http://packetracer123cisco.000webhostapp.com/data";
void setup() {
Serial.begin(9600);
Serial.print("Connecting to WiFi");
WiFi.begin("Wokwi-GUEST", "",6); //uses WiFi channel 6,
speeds up connection
while (WiFi.status() != WL_CONNECTED) {
delay(100);
Serial.print(".");
}
Serial.println(" Connected!");
Serial.print("Fetching " + url + "... ");

}
```

```
void loop() {
http.begin(url);
int httpResponseCode = http.GET();
if (httpResponseCode > 0) {
Serial.println("Temp: "+http.getString()); //prints response
}
else {
Serial.print("Error code: ");
Serial.println(httpResponseCode);
}
http.end();
delay(3000);
}
```

## Server's PHP code:

```
<?php

$css='<style>
        *{
            text-align:center;
        }
        html{
            background-color:beige;
        }
        table, th, td{
        border: 2px solid black;
        border-radius: 3px;
        background-color: #ffda6b;
        padding: 3px;
        margin:auto;
        color:black;
    }
    table{
        padding:6px;
        border-spacing: 6px;
    }
    th{
        background-color: #c4fffb;
    }
    td{
        background-color:#cfffd2;
    }
</style>';

$form="<form method=post
action=https://packetracer123cisco.000webhostapp.com/db.php?
key=top_secret>
<label> Name: <input name=Loc> </label>
<label> Temp: <input name=Temp> </label>
<button type=submit> Submit! </button>
</form>";

$chart="<div id='areachart'></div>
<script type='text/javascript'
src='https://www.gstatic.com/charts/loader.js'></script>
```

```
<script type='text/javascript'>
// Load google charts
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);
// Draw the chart and set the chart values
function drawChart() {
  var data = google.visualization.arrayToDataTable([
  ['Time Stamp', 'Celsius']";

if(empty($_GET["key"]) || $_GET["key"]!="top_secret")
    die("Invalid Key.");

$servername = "localhost";
$username = "id19581966_littledb"; // Capitals littleDB dont work..
$password = "littlesDB@272951";
$db = "id19581966_iot";
$table= "temp";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $db);
// Check connection
if (!$conn)
    die("Connection failed: " . mysqli_connect_error());
        }
        else
            die("Empty Table.");
        break;

    default:
        echo "Invalid action.";
}
    mysqli_close($conn);
?>
switch ($_GET["action"]){
    case "insert":
        $temp= empty($_GET["Temp"]) ? die("Temperature value empty") :
$_GET["Temp"];
        $loc = empty($_GET["Loc"])  ? die("Location value empty") :
$_GET["Loc"];
        $sql = "insert into $table (Loc,Temp) values('$loc', '$temp')";
        if (mysqli_query($conn, $sql))
             echo "Inserted Loc: $loc and Temp: $temp";
        else
            echo "Error: " . mysqli_error($conn);
        break;

    case "select":
        $sql= "select * from $table";
        $result=mysqli_query($conn,$sql);
        echo "<h1> Temperature Table </h1><table border=1> <tr>
<th>Timestamp</th> <th>Loc</th> <th>Temp</th> </tr>";
        if (mysqli_num_rows($result)){
            while ($row = mysqli_fetch_assoc($result))
                echo "<tr><td>".$row["timeStamp"]."</td> <td>".
$row["Loc"]."</td> <td>".$row["Temp"]."</td></tr>";
            echo "</table>";
        }
        else
            echo "Empty Table.";
        break;
    case "chart":
```

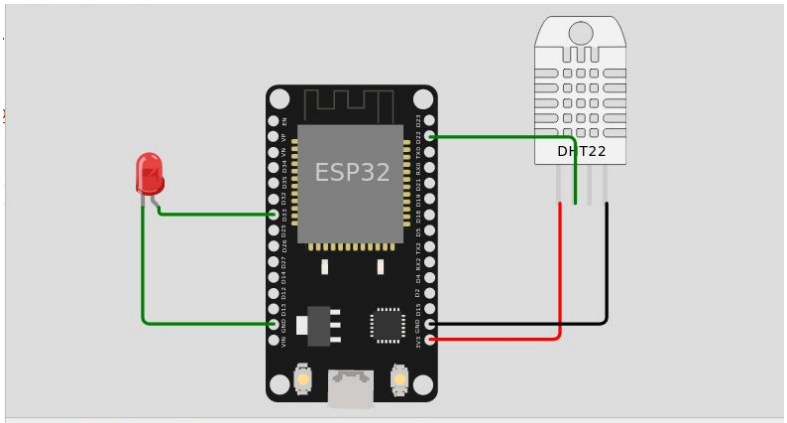
```
        $loc= empty($_GET["Loc"]) ? die("Location value empty") :
$_GET["Loc"];
        $sql="select * from $table";
        $result=mysqli_query($conn,$sql);
        if (mysqli_num_rows($result)){
            echo $chart;
            while ($row = mysqli_fetch_assoc($result))
                if($row["Loc"]==$_GET["Loc"])
                    echo ",['{$row["timeStamp"]}',{$row["Temp"]}]";
            echo "]); var options = {'title':'Temperature Analytics @ ".
$_GET["Loc"] . "', 'width':550, 'height':400};
            var chart = new
google.visualization.AreaChart(document.getElementById('areachart'));
            chart.draw(data, options);}</script>";
        }
        else
            die("Empty Table.");
        break;

    default:
        echo "Invalid action.";
}
    mysqli_close($conn);
?>
```

## Outputs:

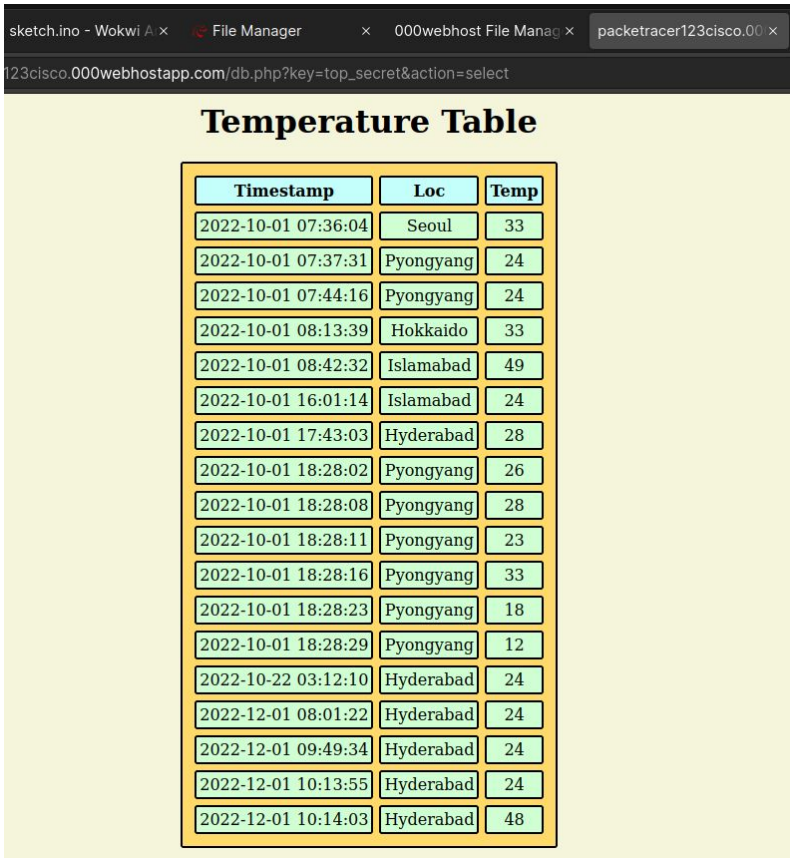
Receiver:


```
Islamabad,2022-10-01 12:06:04,24
Islamabad,2022-10-01 12:06:44,24
Islamabad,2022-10-01 12:11:28,24
Delhi,2022-10-01 22:46:10,33
Hyderabad,2022-10-01 22:49:19,74
Hyderabad,2022-10-01 22:49:35,28
```

Receiver:




```
Connecting to WiFi.......... Connected!
Fetching http://packetracer123cisco.000webhostapp.com
/db.php?key=top_secret&action=insert&Loc=Hyderabad&Temp=... Inserted Loc:
Hyderabad and Temp: 24
Inserted Loc: Hyderabad and Temp: 48
```




| Timestamp | Loc | Temp |
|---|---|---|
| 2022-10-01 07:36:04 | Seoul | 33 |
| 2022-10-01 07:37:31 | Pyongyang | 24 |
| 2022-10-01 07:44:16 | Pyongyang | 24 |
| 2022-10-01 08:13:39 | Hokkaido | 33 |
| 2022-10-01 08:42:32 | Islamabad | 49 |
| 2022-10-01 16:01:14 | Islamabad | 24 |
| 2022-10-01 17:43:03 | Hyderabad | 28 |
| 2022-10-01 18:28:02 | Pyongyang | 26 |
| 2022-10-01 18:28:08 | Pyongyang | 28 |
| 2022-10-01 18:28:11 | Pyongyang | 23 |
| 2022-10-01 18:28:16 | Pyongyang | 33 |
| 2022-10-01 18:28:23 | Pyongyang | 18 |
| 2022-10-01 18:28:29 | Pyongyang | 12 |
| 2022-10-22 03:12:10 | Hyderabad | 24 |
| 2022-12-01 08:01:22 | Hyderabad | 24 |
| 2022-12-01 09:49:34 | Hyderabad | 24 |
| 2022-12-01 10:13:55 | Hyderabad | 24 |
| 2022-12-01 10:14:03 | Hyderabad | 48 |

## Conclusions:

Thus, we have succesfully utilized the cloud to gather temperature data from different locations.