

# WEATHER STATION

**Description:** Weather monitoring using Temperature Sensor and Smoke Sensor.

## Components Required:

1. Arduino UNO
2. WIFI module
3. Smoke Sensor
4. Temperature Sensor (LM35)
5. Power Supply
6. Connectors
7. ThingSpeak (IOT analytic platform service)

## Creating Channel in ThingSpeak

**Step 01:** Create ThingSpeak Account. The URL is given below.

URL: <https://thingspeak.com/>

## Step 02:

Click On New Channel.

My Channels

New Channel

Search by tag

Name	Created	Updated
Smart-Irrigation <a href="#">Private</a> <a href="#">Public</a> <a href="#">Settings</a> <a href="#">Sharing</a> <a href="#">API Keys</a> <a href="#">Data Import / Export</a>	2022-09-22	2022-09-22 05:28
Weather Monitoring <a href="#">Private</a> <a href="#">Public</a> <a href="#">Settings</a> <a href="#">Sharing</a> <a href="#">API Keys</a> <a href="#">Data Import / Export</a>	2022-09-27	2022-09-27 05:59

### Step 03: Providing Details

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy AG

## New Channel

Enter the Name, and Description.

Choose the number of fields required.

Name

Description

Field 1  ☒

Field 2  ☐

Field 3  ☐

Field 4  ☐

Field 5  ☐

Field 6  ☐

Field 7  ☐

Field 8  ☐

Metadata

Tags

## Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

### Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.

### Step 04: Save changes

### CODE:

```
//WEATHER MONITORING PROJECT.01
```

```
#include <SoftwareSerial.h>
```

```
#define RX 8 // it works as a UART
```

```
#define TX 9 // so Rx is connected to Pin no 9 in Arduino Uno, Tx is connected to pin no 8 in UNO
```

```
SoftwareSerial esp8266(RX,TX);
```

```
String inputString = "";
```

```
boolean RX_ST_Flag = false;
```

```
boolean stringComplete = false;
```

```
String AP = "Nokia"; // CHANGE ME HOTSPOT NAME
```

```
String PASS = "12345678"; // CHANGE ME HOTSPOT NAME password
```

```
String API = "5AYF6UISFWRT03F9"; // CHANGE ME
```

```
String HOST = "api.thingspeak.com";
```

```
//String HOST = "184.106.153.149";
```

```
String PORT = "80";
```

```
String field1 = "field1";  
String field2 = "field2";  
String field3 = "field3"; //Extra Field is taken inCase if we use another sensor and forget to mention, we can use this
```

```
int countTrueCommand;  
int countTimeCommand;  
boolean found = false;
```

```
void setup()  
{  
    delay(500);  
    Serial.begin(9600);  
    delay(500);  
  
    Serial.println("WEATHER MONITORING");  
    delay(500);  
  
    esp8266.begin(9600);  
    sendCommand("AT",5,"OK");  
    sendCommand("AT+CWMODE=1",5,"OK");  
    sendCommand("AT+CWJAP=\"\"+ AP +\"\", \"\"+ PASS +\"\",20,\"OK");  
    delay(500);  
}
```

```
void loop()  
{  
  
    int Temperature = analogRead(A0);  
    int Temperature_Temp = (( Temperature/1024.0 ) * 5000 ) / 10;  
    Serial.print( "Temperature = " );  
    Serial.println( Temperature_Temp );  
    delay(2000);  
  
    int Smoke_Sensor = analogRead(A1);
```

```

int Smoke_Sensor_Temp = ( ( Smoke_Sensor/1024.0 ) * 5000 ) / 10;
Serial.print( "Smoke_Sensor = " );
Serial.println( Smoke_Sensor_Temp );
delay(2000);

// String getData = "GET /update?api_key="+ API +"&"+ field1
+"="+String(Solar_Volt_Tx);+"&"+ field2 +"="+String(Wind_Volt_Tx);

String getData = "GET /update?api_key="+ API +"&"+ field1
+"="+String(Temperature_Temp)+"&"+ field2 +"="+String(Smoke_Sensor_Temp);

sendCommand("AT+CIPMUX=1",5,"OK");

sendCommand("AT+CIPSTART=0,\"TCP\", \"\"+ HOST +"\", "+ PORT,15,"OK");

sendCommand("AT+CIPSEND=0,\" "+String(getData.length()+4),4,">");

esp8266.println(getData);

delay(1500);

countTrueCommand++;

sendCommand("AT+CIPCLOSE=0",5,"OK");

delay(5000);

}

```

```

void sendCommand( String command, int maxTime, char readReplay[] )
{
    Serial.print(countTrueCommand);
    Serial.print(". at command => ");
    Serial.print(command);
    Serial.print(" ");
    while(countTimeCommand < (maxTime*1))
    {
        esp8266.println(command);//at+cipsend
        if(esp8266.find(readReplay))//ok
        {
            found = true;
            break;
        }
    }
}

```

```

        countTimeCommand++;
    }

    if(found == true)
    {
        Serial.println("OYI");
        countTrueCommand++;
        countTimeCommand = 0;
    }

    if(found == false)
    {
        Serial.println("Fail");
        countTrueCommand = 0;
        countTimeCommand = 0;
    }

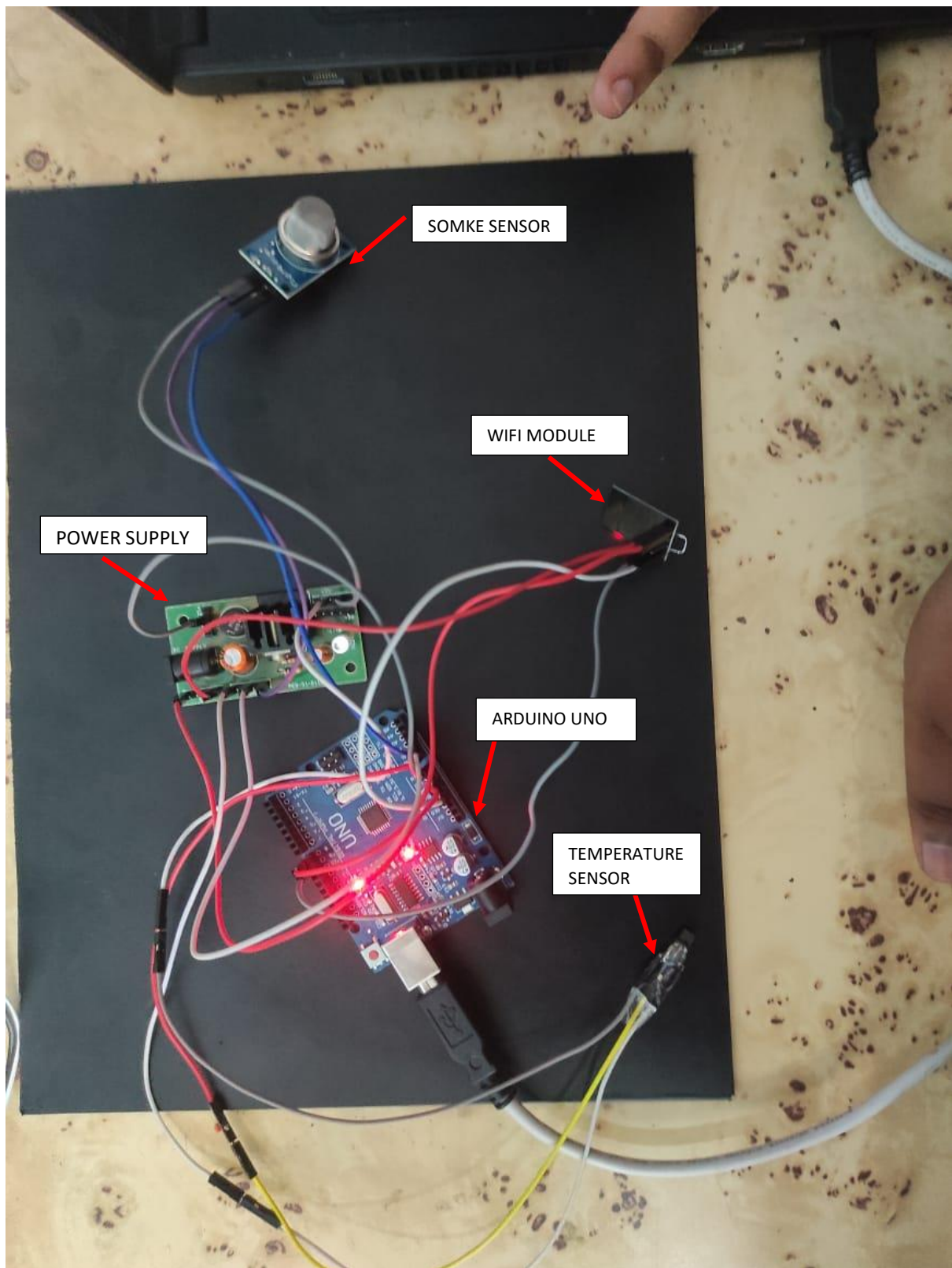
    found = false;
}

void serialEvent()
{
    while (Serial.available())
    {
        char inChar = (char)Serial.read();    // get the new byte:
        inputString += inChar;                // add it to the inputString:
        if(inChar == '#')
        {
            RX_ST_Flag = true;
        }
    }
}

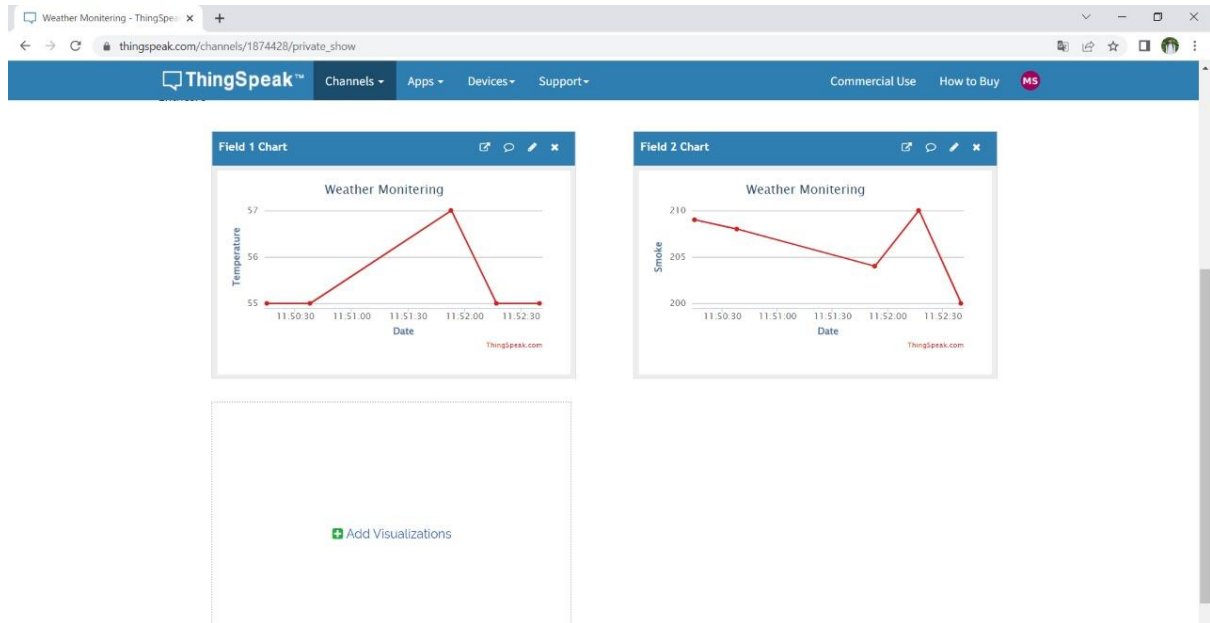
```

**NOTE:** Change the API key, String AP (Hotspot Name), String PASS (Hotspot password)

## CIRCUIT CONNECTION



## RESULT SNAPSHOT:



**Conclusion:** The above snapshots show the Real-Time Monitoring of Temperature and Smoke.