Report on smart weather monitoring by GROUP 7

GROUP MEMBERS-

1. Prashasti Singh Gautam 19BEE10028
2. Tekumalla Lakshmi Sowjanya 19BEE10023
3. Samiksha Dash 19BEE10009
4. Prithviraj Mitra 19BEE10015

CONTENTS

INTRODUCTION

This project simulates a “Basic Weather Monitoring System” using a temperature sensor , gas sensor ,ultrasonic sensor and potentiometer wind sensor on the IoT Simulation Software, TinkerCAD. The data obtained from the sensors is also visualised on the Thingspeak dashboard.

SYSTEM DESIGN

The fundamental hardware components are:

1.arduino uno R3

2. ultrasonic distance sensor

3.temparature sensor [TMP36]

4.PIR sensor

5.potentiometer

6.wifi module

HARDWARE COMPONENTS

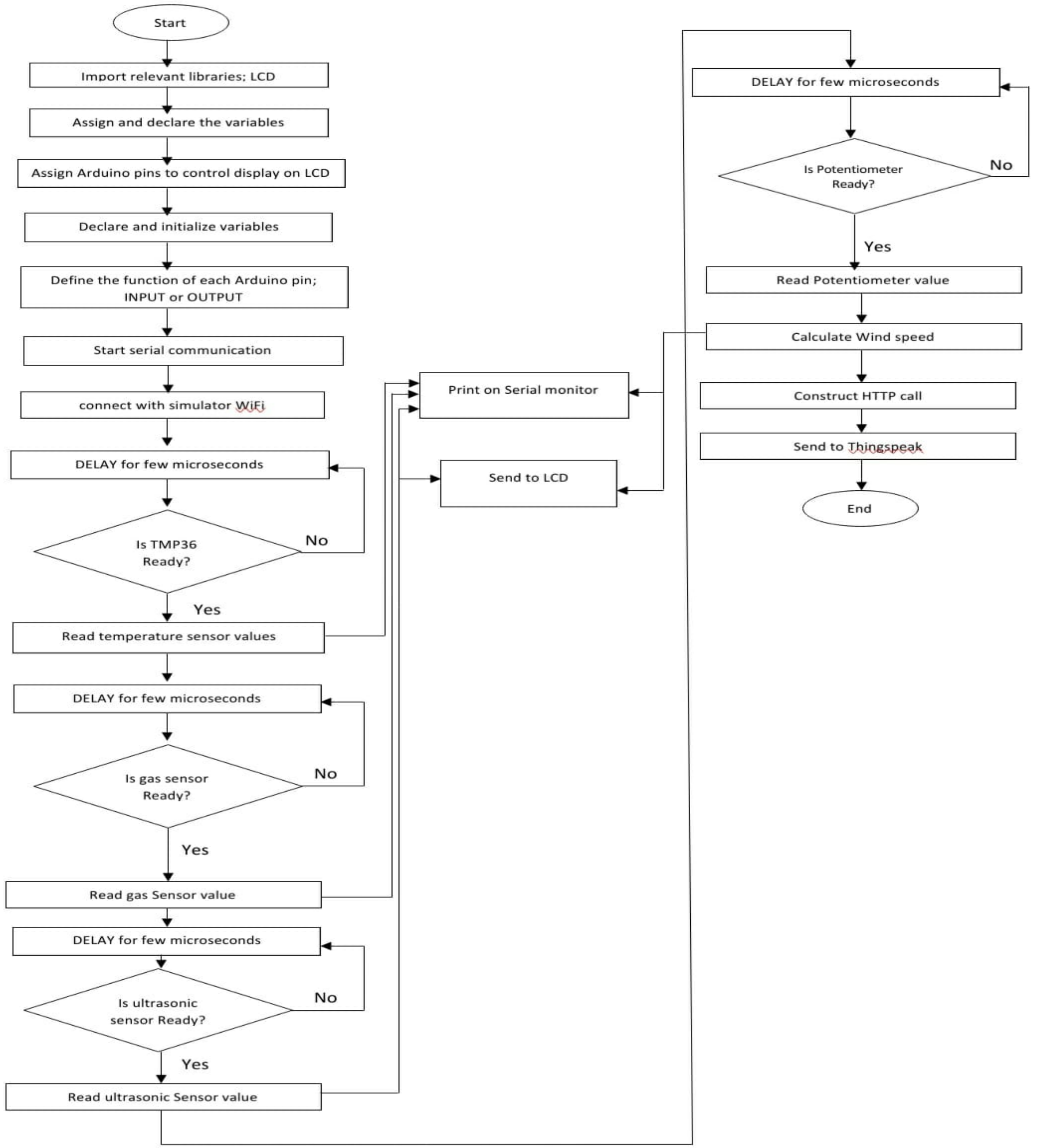
Components used for Design

S.No.	Part type	Properties
1.	TMP36	Temperature sensor
2.	Microcontroller	Arduino Uno R3
3.	ESP8266	WiFi Module
4.	MQ2 Gas sensor	Gas sensor
5.	Piezo	Buzzer
6.	250 K ohm Potentiometer	Potentiometer
7.	Meter	Voltage multimeter
8.	Ultrasonic distance sensor	Distance sensor
9.	Monitor	LCD; characters 16×2
10.	Resistors	1 K ohm, 325 K ohm

DESCRIPTION

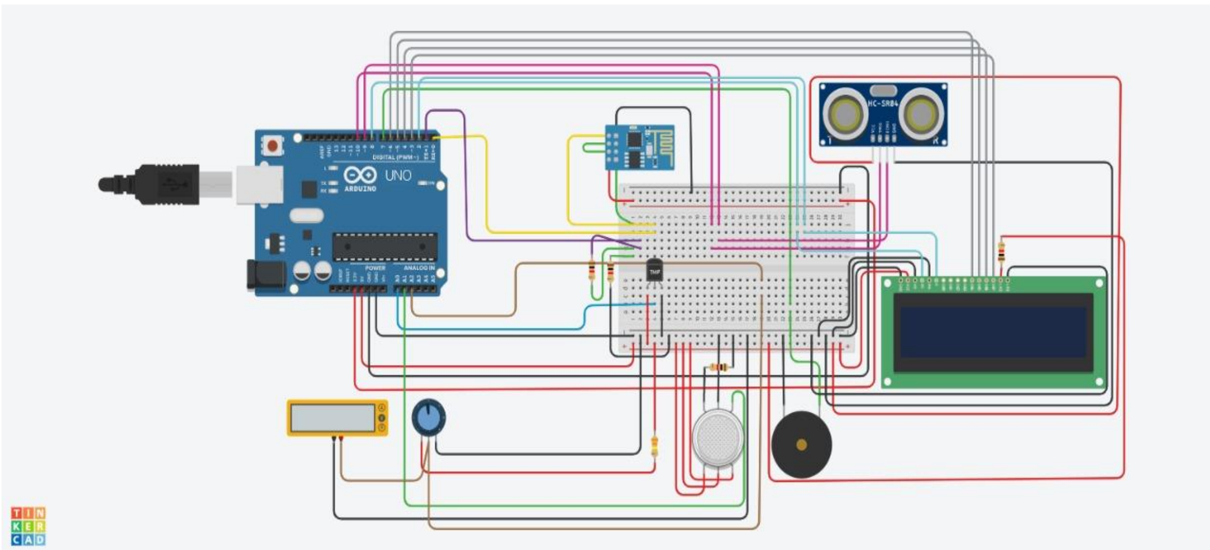
In the smart weather monitoring system, we used four sensors. These sensors are connected to the Arduino and Wi-Fi module is connected to work through the wireless communication. Temperature sensor is connected to detects the maximum and minimum temperature. A gas sensor, that detects the impurification levels in air. If the density of impurities is increases more than 200 then the buzzer gets on which we connected at the gas sensor [MQ2]. To calculate the wind speed, mostly an anemometer used. In an anemometer, at the end of the pole in downward part there is a potentiometer connected which is basically responsible to calculate the wind speed. The cups of the anemometer rotate at the same speed at which wind flows. So, potentiometer detect the voltage which generates due to the rotation of anemometer and with this voltage we calculate the value of speed of wind. To calculate that how much rain falls in the area, we used an ultrasonic sensor. The sensor is mounted over the water to determine the distance to the water. The sensor transmits a sound pulse that reflects from the surface of the water and measures the time it takes for the echo to return. We also connect a LCD to the Arduino to show the values of the windspeed and rainfall. By using the HTTP call, the data from thinkercad import to the thingspeak and get the plots of the sensors data. We also analyze the system through MATLAB to get the minimum and maximum change in data in past 24 hours.

FLOWCHART



Start
Import relevant libraries; LCD
Assign and declare the variables
Assign Arduino pins to control display on LCD
Declare and initialize variables
Define the function of each Arduino pin; INPUT or OUTPUT
Start serial communication
connect with simulator WiFi
DELAY for few microseconds
Is TMP36 Ready?
No
Yes
Read temperature sensor values
DELAY for few microseconds
Is gas sensor Ready?
No
Yes
Read gas Sensor value
DELAY for few microseconds
Is ultrasonic sensor Ready?
No
Yes
Read ultrasonic Sensor value
Print on Serial monitor
Send to LCD
DELAY for few microseconds
Is Potentiometer Ready?
No
Yes
Read Potentiometer value
Calculate Wind speed
Construct HTTP call
Send to Thingspeak
End

ARDUINO SIMULATION



ARDUINO CODE

```
//Smart weather reporting system

#include <LiquidCrystal.h>


//temperature sensor variables

float temp_vout;

float temp;

float voltage;
```

```
//gas sensor variables
int gas_sensor_port = A1;
int gas_sensor_value = 0;

//rainfall measurement variables
float rain;
const int triggerPin = 10;
const int echoPin = 9;
long duration;

//wind speed measurement variables
float V_wind = 0;
float Windspeedfloat;
int Windspeedint;
//LCD
LiquidCrystal lcd(8, 2, 6, 5, 4, 3); //Parameters: (rs, enable, d4, d5, d6, d7)

//WIFI module variables
String ssid     = "Simulator Wifi";  // SSID to connect to
String password = "";   //virtual wifi has no password
String host     = "api.thingspeak.com"; // Open Weather Map API
const int httpPort   = 80;
String url     = "/update?api_key=O1WB76S0Z5G24Y0O&field1="; //ThingSpeak Channel API Key


//setting up the wifi module
void setupESP8266(void)
{
  // Start our ESP8266 Serial Communication
```

```
  Serial.begin(115200);   // Baud rate
  Serial.println("AT");   // Serial connection on Tx / Rx port to ESP8266
  delay(10);              // Wait a little for the ESP to respond

  if (Serial.find("OK"))
    Serial.println("ESP8266 OK!!!");  // Connect to Simulator Wifi

  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\""); //AT+CWLAP – list nearby available
WiFi networks
  delay(10);                         // Wait a little for the ESP to respond

  if (Serial.find("OK"))
    Serial.println("Connected to WiFi!!!");     // Open TCP connection to the host:

  //ESP8266 connects to the server as a TCP client.

  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
  delay(50);                                // Wait a little for the ESP to respond

  if (Serial.find("OK"))
   Serial.println("ESP8266 Connected to server!!!") ;
}

//Sends data to Thingspeak
void send_data(void)
{
  // Construct HTTP call
  String httpPacket = "GET " + url + String(temp) + "&field2=" + String(gas_sensor_value) + "&field3=" +
String(rain) + "&field4=" + String(Windspeedfloat) + " HTTP/1.1\r\nHost: " + host + "\r\n\r\n";
```

```
  int length = httpPacket.length();
  // Send our message length
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10);       // Wait a little for the ESP to respond if (!Serial.find(">")) return -1;
  // Send our http request
  Serial.print(httpPacket);
  delay(10);      // Wait a little for the ESP to respond
  if (Serial.find("SEND OK\r\n"))
    Serial.println("ESP8266 sends data to the server");
}

void setup()
{
  pinMode(A1, INPUT);     //gas sensor analog input
  pinMode(7, OUTPUT);     //gas sensor digital output
  pinMode(A0, INPUT);     //temperature sensor analog input
  pinMode(A2, INPUT);     //potentiometer analog input
  setupESP8266();
  lcd.begin(16,2);
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);


}

void loop()
{

  //for temperature sensor
```

```
temp_vout = analogRead(A0);
voltage = temp_vout * 0.0048828125; //convert analog value between 0 to 1023 with 5000mV/5V ADC
temp = (voltage - 0.5) * 100.0;
Serial.println("Current temperature: " + String(temp));
//----------------------------------------------------------------------------------------------------
//for gas sensor
gas_sensor_value = analogRead(gas_sensor_port);
Serial.println("Gas sensor value: " + String(gas_sensor_value));
if (gas_sensor_value > 200)
{
  tone(7,523,1000);
}

//-------------------------------------------------------------------------------------------------------
//for rainfall measurement
digitalWrite(triggerPin, LOW);
delayMicroseconds(2);
// Sets the trigger pin to HIGH state for 10 microseconds
digitalWrite(triggerPin, HIGH);
delayMicroseconds(10);
digitalWrite(triggerPin, LOW);
// Reads the echo pin, and returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
rain = 0.01723 * duration;
delay(10); // Delay a little bit to improve simulation performance

lcd.setCursor(0,0); // Sets the location at which subsequent text written to the LCD will be displayed
lcd.print("Rainfall: "); // Prints string "Rainfall" on the LCD
lcd.print(rain); // Prints the distance value from the sensor
```

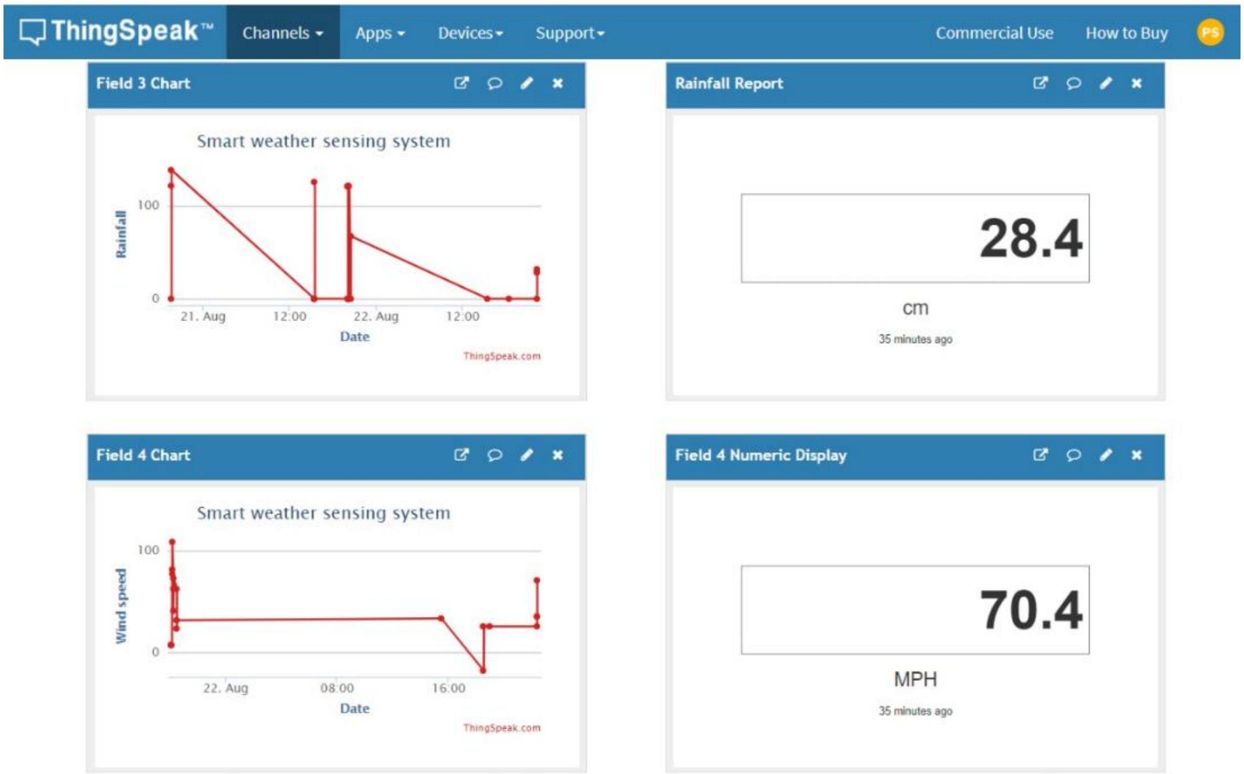
```
Serial.println("Rainfall: " + String(rain));
delay(10);

//--------------------------------------------------------------------------------------------------
//for wind speed measurement
float V_wind = analogRead(A2) * (5.0 / 1023.0);
// Voltage converted to MPH
Windspeedint = (V_wind - 0.4) * 10 * 2.025 * 2.237;     // For LCD screen output
Windspeedfloat = (V_wind - 0.4) * 10 * 2.025 * 2.237; // For Serial monitor output
//wind speed LCD output
lcd.setCursor(0,1);       // adjust cursor
lcd.print("Wind speed");
lcd.print(" ");
if (V_wind < 0.4)
{
  lcd.print("0");
}
else
{
  lcd.print(Windspeedint);
}
lcd.print("MPH");
//wind speed serial monitor output
Serial.print("Wind Speed: ");
if (Windspeedfloat <= 0)
{
  Serial.print("0.0");}
else{
  Serial.print(Windspeedfloat);}// Output Wind speed value
```

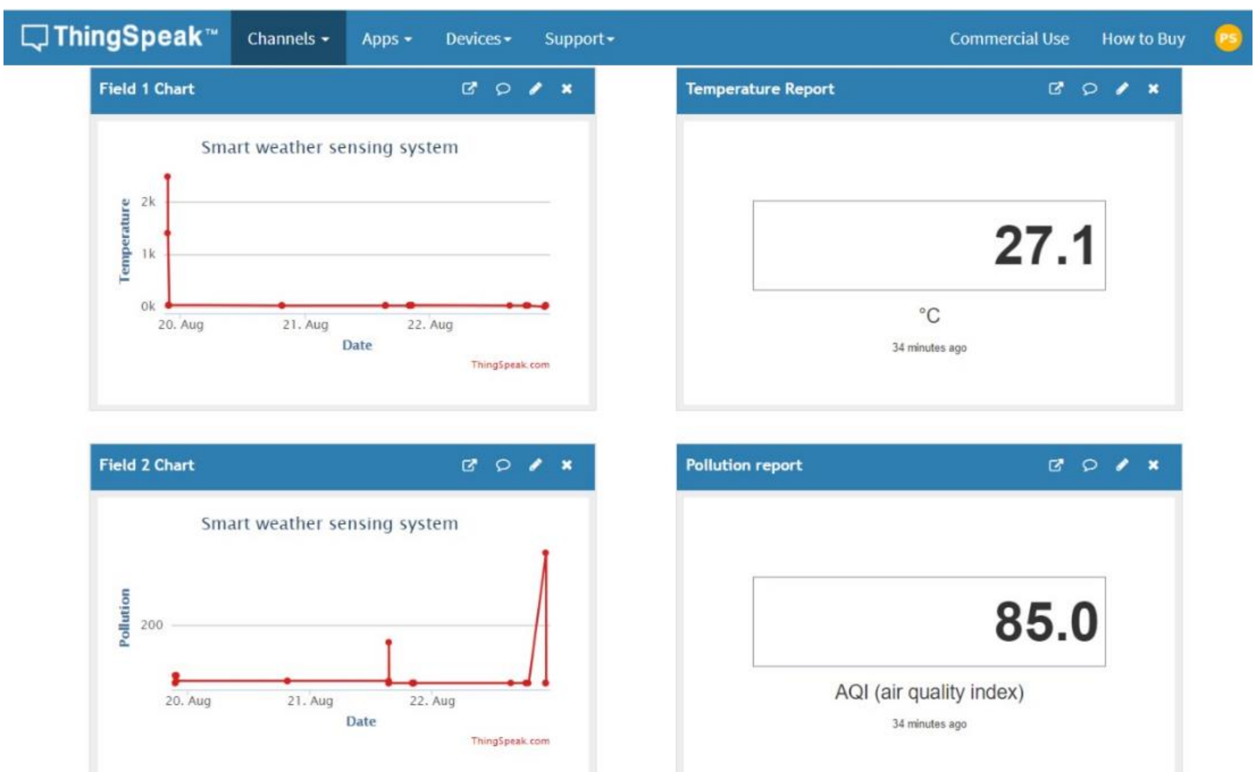
```
  Serial.println(" MPH");
  delay(100);

  Serial.print("Anemometer Voltage: ");
  if (V_wind > 2){
   Serial.println("Out of range!");
  }
  else if (V_wind < 0.4)
  {
   Serial.println("Out of range!");
  }
  else{
   Serial.print(V_wind);
   Serial.println(" V");}

  //send data to thingspeak
  send_data();
  delay(1000);     // delay changed for faster analytics
}
```

SENDING DATA TO THINGSPEAK

The whole data which we get through tinkercad is sent to the ThingSpeak through HTTP call and we get the plot of the sensor values.

The graphs are given below-


ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Field 3 Chart
Smart weather sensing system
Rainfall
100
0
21. Aug
12:00
22. Aug
12:00
Date
ThingSpeak.com
Rainfall Report
28.4
cm
35 minutes ago
Field 4 Chart
Smart weather sensing system
Wind speed
100
0
22. Aug
08:00
16:00
Date
ThingSpeak.com
Field 4 Numeric Display
70.4
MPH
35 minutes ago


ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Field 1 Chart
Smart weather sensing system
Temperature
2k
1k
0k
20. Aug
21. Aug
22. Aug
Date
ThingSpeak.com
Temperature Report
27.1
°C
34 minutes ago
Field 2 Chart
Smart weather sensing system
Pollution
200
20. Aug
21. Aug
22. Aug
Date
ThingSpeak.com
Pollution report
85.0
AQI (air quality index)
34 minutes ago

MATLAB ANALYSIS

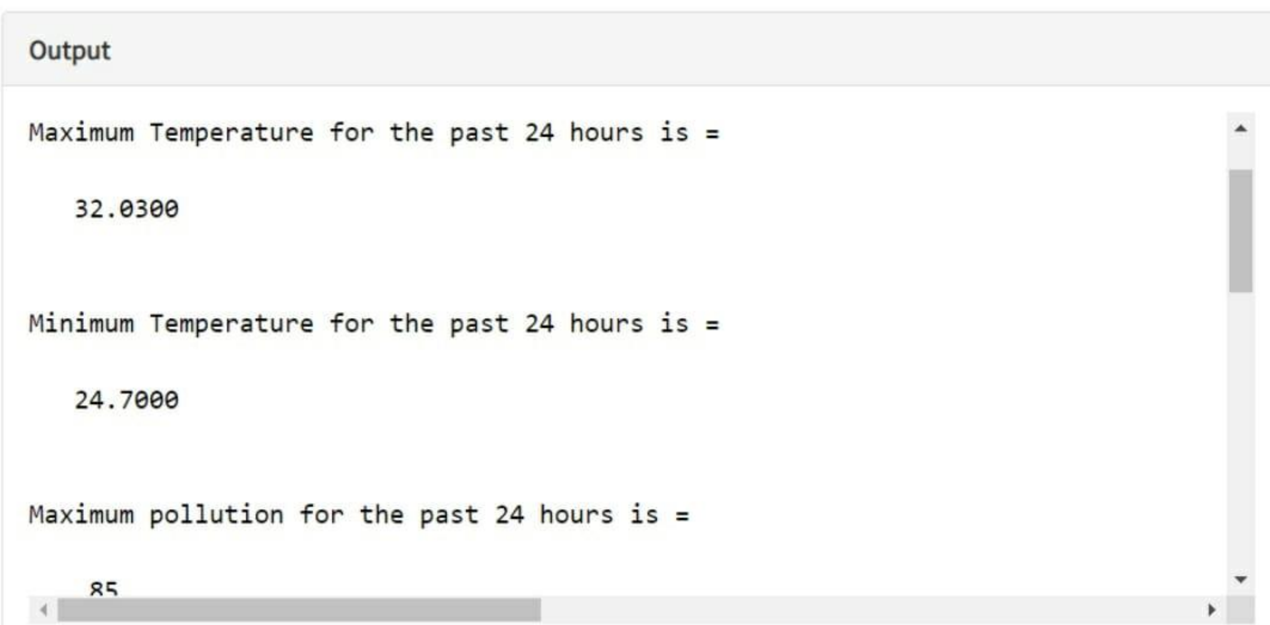
```
% Enter your MATLAB Code below
% Read data from a ThingSpeak channel over the past 24 hours
% to calculate the high and low datas and write to another channel.

% Channel ID to read data from
readChannelID = 1480637;
% Field ID
TemperatureFieldID = 1;
gas_sensorFieldID = 2;
rainfall_FieldID = 3;
wind_FieldID = 4;

% Channel Read API Key
% If your channel is private, then enter the read API Key between the '' below:
readAPIKey = 'Q90LQOIGATHEW7SP';


[tempF,timeStamp1] =
thingSpeakRead(readChannelID,'Fields',TemperatureFieldID, ...
```

```
                                  'numDays',1,'ReadKey',readAPIKey);
[pollutionF,timeStamp2] =
thingSpeakRead(readChannelID,'Fields',gas_sensorFieldID, ...
                                  'numDays',1,'ReadKey',readAPIKey);
[rainfallF,timeStamp3] = thingSpeakRead(readChannelID,'Fields',rainfall_FieldID,
...
                                  'numDays',1,'ReadKey',readAPIKey);
[windF,timeStamp4] = thingSpeakRead(readChannelID,'Fields',wind_FieldID, ...
                                  'numDays',1,'ReadKey',readAPIKey);

% Calculate the maximum and minimum temperatures
[maxTempF,maxTempIndex] = max(tempF);
[minTempF,minTempIndex] = min(tempF);

% Calculate the maximum and minimum Pollution
[maxpollutionF,maxpollutionIndex] = max(pollutionF);
[minpollutionF,minpollutionIndex] = min(pollutionF);

% Calculate the maximum and minimum rainfall
[maxrainfallF,maxrainfallIndex] = max(rainfallF);
[minrainfallF,minrainfallIndex] = min(rainfallF);

% Calculate the maximum and minimum rainfall
[maxwindF,maxwindIndex] = max(windF);
[minwindF,minwindIndex] = min(windF);
```

```
% Select the timestamps at which the maximum and minimum temperatures
were measured

timeMaxTemp = timeStamp1(maxTempIndex);

timeMinTemp = timeStamp1(minTempIndex);


% Select the timestamps at which the maximum and minimum pollution were
measured

timeMaxpollution = timeStamp2(maxpollutionIndex);

timeMinpollution = timeStamp2(minpollutionIndex);


% Select the timestamps at which the maximum and minimum rainfall were
measured

timeMaxrainfall = timeStamp3(maxrainfallIndex);

timeMinrainfall = timeStamp3(minrainfallIndex);


% Select the timestamps at which the maximum and minimum wind were
measured

timeMaxwind = timeStamp4(maxwindIndex);

timeMinwind = timeStamp4(minwindIndex);


display(maxTempF,'Maximum Temperature for the past 24 hours is');

display(minTempF,'Minimum Temperature for the past 24 hours is');


display(maxpollutionF,'Maximum pollution for the past 24 hours is');

display(minpollutionF,'Minimum pollution for the past 24 hours is');
```

```
display(maxrainfallF,'Maximum rainfall for the past 24 hours is');
display(minrainfallF,'Minimum rainfall for the past 24 hours is');

display(maxwindF,'Maximum wind for the past 24 hours is');
display(minwindF,'Minimum wind for the past 24 hours is');



% Replace the [] with channel ID to write data to:
writeChannelID = [1480637];
% Enter the Write API Key between the '' below:
writeAPIKey = 'O1WB76S0Z5G24Y0O';
```

MATLAB OUTPUTS



Output

```
Maximum pollution for the past 24 hours is =

    85


Minimum pollution for the past 24 hours is =

    85


Maximum rainfall for the past 24 hours is =

  121.2100
```

Output

```
Maximum rainfall for the past 24 hours is =

  121.2100


Minimum rainfall for the past 24 hours is =

     0


Maximum wind for the past 24 hours is =

  108.0800
```

Output

```
Maximum wind for the past 24 hours is =

  108.0800


Minimum wind for the past 24 hours is =

  -18.1100
```

CONCLUSION

In this paper we show that how we use IOT technology to convert the physical parameters of environment into scalable data. This paper demonstrates Design and Implementation of Weather Monitoring & Controlling System used for controlling the devices as well as monitoring the environmental parameters. Embedded controlled sensor networks have proven themselves to be a reliable solution in providing remote control and sensing for environmental monitoring systems. The sensors have been integrated with the system to monitor and compute the level of existence of Wind, gas, temperature and rain fall in atmosphere using information and communication technologies. The sensors can upload the data in Lab view using serial communication.