P.T. Sarvajanik College of Science

**Department of Physics, MTB Campus,
Jawaharlal Nehru Marg,
Athwalines, Surat - 395001**

Documentation for the IoT based

Air Quality Monitoring Station

- Project Supervisor -

Prof. Kileen J. Mahajan
Associate professor, Department of Physics

- Project Members -

Vishalkumar Gohel
Alumni

Krupamaya Panda
Alumni

Vishal Warule
Alumni

Index

1. Abstract

In this documentation we have tried explain the construction and working of different parts of an IoT-based Air Quality Monitoring Station which is installed on the campus of Sir P.T. Sarvajanik College of Science, which provides real-time meteorological data of the surroundings using sensors connected to ThinkSpeak server through Wi-Fi modules. The whole system is divided into two main units; first unit comprises wind speed & wind direction, wind gust, and total rainfall of the last 24 hours. The other unit comprises values of ambient temperature, humidity, dew point, heat index, and pressure with real-time PM (**particulate matter of size ranging from 1 micron to 10 microns**) concentration in the air. Also, this real-time data can also be seen on the College's website. Here (hyperlink).

1.1. Miscellaneous Components and Equipment

- General purpose PCB
- Shield for Arduino Uno
- Shield for Node-MCU
- Rainbow wires
- Passive components (Resistor)
- Adapters for modules
- Glue gun
- Soldering iron
- Berg pins (Male and Female)

1.2. Online resources

- Thingspeak.com is web-based application for real-time data visualization for IoT based systems.

1.3. Software resources

- Arduino IDE (It is a programming environment based on C/C++ languages, that allows us to communicate with the micro-controller)
- Visual Studio Code (For constructing the webpage, which will be showing the data gathered from Thingspeak channel).

1.4. System Housing

- Stevenson screen

2. Descriptions of Products used

2.1. Micro-Controllers

2.1.1. Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. We used Arduino Uno for its compatibility in coding and interfacing with various other Wi-Fi modules and Sensor modules.

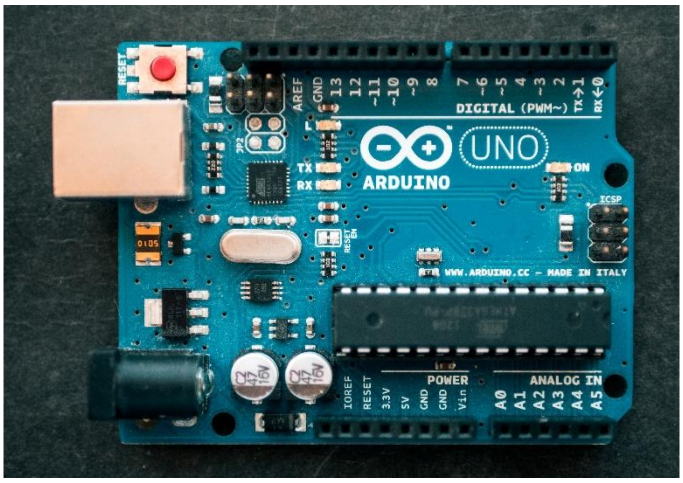


Figure 1
Arduino Uno

2.1.2. Node-MCU (version 0.9)

It Is a micro-controller with built-in Wi-Fi module that helps it to connect to the internet. It has total of 30 pins with GPIO pins, PWM, Digital, Analogue (one), and other pins for serial communication such as RX, TX and I2C pins. It solely used for processing and transmitting real time data gathered from two sensors, BME-280 and PMS1003, to the internet.

Figure 2
Node-MCU V0.9

2.2. Sensors and modules

2.2.1. ESP-01 Wi-Fi module

The ESP-01 ESP8266 Serial Wi-Fi Wireless Transceiver Module is a first version of wireless modules used to for establishing communication between the server (online) and hardware (Micro-controller), to have wireless data transmission by using hotspot of a nearby device (Router, Mobile and PC). In our project we used an ESP-01 module with Arduino Uno via a Bidirectional logic-level converter to transmit weather data wirelessly to the online server (ThingSpeak) for the further visualization. For the further details follow the information provided in the datasheet.

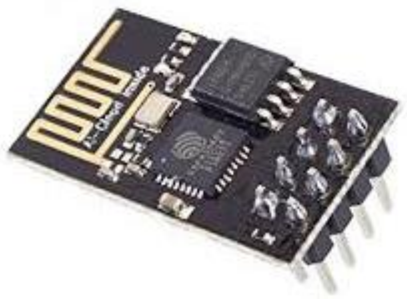


Figure 3
ESP-01 Wi-Fi Module

2.2.2. BME280 Sensor module

A sensor which is used for measuring relative humidity, barometric pressure and ambient temperature especially developed for mobile application and lower power consumption. The product we used for this system came with a I2C interface module which can be interfaced to any micro-controller via 4 pins. Which is SDA, SCL pins for serial communication and V_{cc} and GND pins to provide power respectively. The reason for selecting product was its accuracy in measurement of parameters in acceptable range. For the further details follow the information provided in the datasheet.

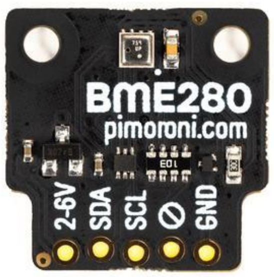


Figure 4
Pressure Sensor module

2.2.3. **PMS1003 Sensor**

It is a digital and universal particle concentration sensor, which can be used to obtain concentration of particles suspended in the air, and then to get output through a digital interface with an external device. PMS1003 has inbuilt optical system consist of photodetector and laser, then for data processing in-built micro-controller, from which we can gather measured data through USART serial communication to an Arduino or any other micro-controller. It can be interfaced Arduino, with an JST adapter, and has 8 pins from which, however, we used only four of them, such as RX, TX, V_{cc} and GND with our Node-MCU. We used this sensor for its accuracy range and reasonable price. For the further details follow the information provided in the datasheet.

Figure 5
Particulate matter sensor

2.2.4. **Bidirectional Logic Level Converter**

It is a small device with four channels (four different terminals), that safely steps down 5V signals to 3.3V and steps up 3.3V to 5V at the same time. It allows to establish serial communication between two devices operating at two different voltages on their terminals. In our case we used for communication between Arduino and ESP-01.

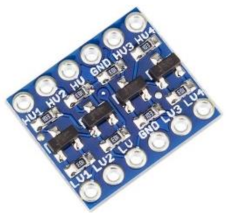


Figure 6
Bi-directional logic level converter

2.3. Sparkfun Weather Meter kit

To measure weather data such as wind speed, wind direction and rain fall amount, and for measurement of these, we used a kit which comes with a set of readymade Anemometer, Wind Vane and Rain Gauge respectively. These instruments produce measured data in form of digital signal, which is later need to be processed into numeric values through programming done in an Arduino board. All these instruments can be connected to a micro-controller via a RJ-11 adapter, since it comes with RJ-11 pins.

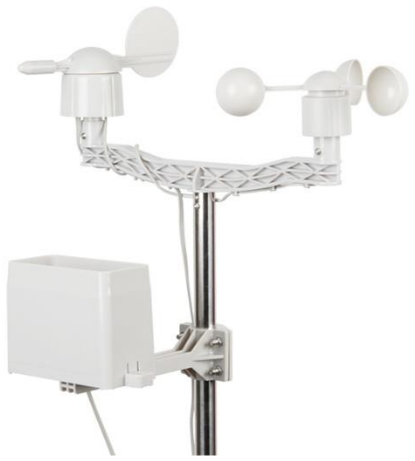


Figure 7
Weather Meter Kit

2.4. Power supply

We used SMPS (Switching Mode Power Supply) for Node-MCU and an Arduino Adapter for Arduino Uno board.

2.5. Enclosure

The whole controlling system of UNIT-A and UNIT-B is kept inside a wooden Stevenson screen installed over the roof of the main building of the institute, Sir P.T. Sarvajanik College of Science.

3. Construction

3.1. UNIT-A

It consists of a Sparkfun Weather meter kit for measurements, an Arduino Uno for data processing, a bidirectional logic level converter for interfacing Uno and ESP-01 module, and an ES8266-01 Wi-Fi module for data transmission through internet. This whole unit is powered by an adapter connected to DC plug on Arduino Uno board.

Circuit Diagram

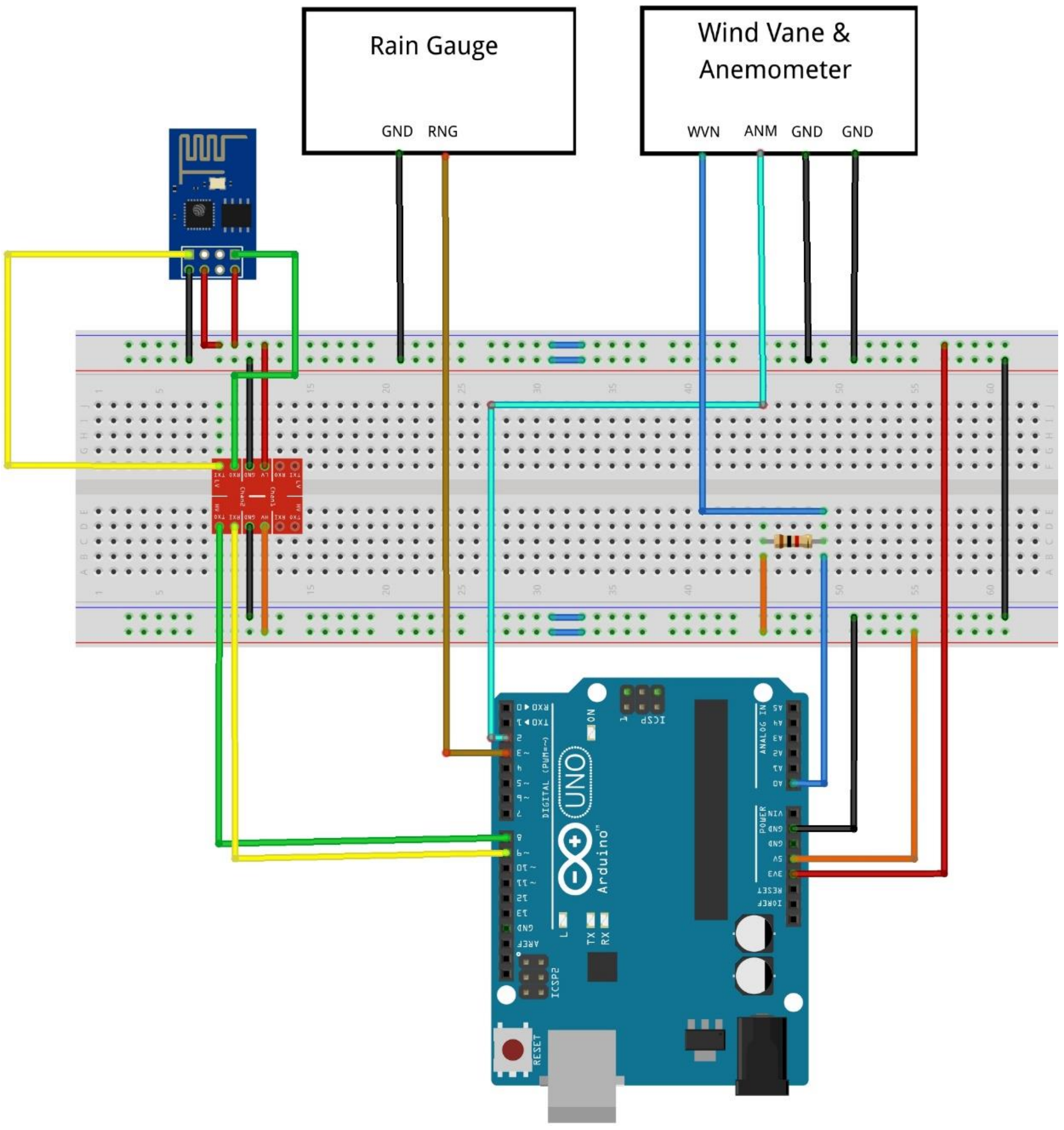


Figure 8
Unit A breadboard schematic

3.1.1. RJ-11 Adapter

Our weather meter kit had pinouts in the form of RJ-11, so to connect it to the Arduino we used a small plastic box with RJ-11 sockets and rainbow wires as output.

3.1.2. Arduino Shield

In UNIT-A, we had to construct our own shield for mounting Arduino-Uno board and other modules. For starters, in a general-purpose PCB we soldered male and female berg pins in order to attach Uno board, and other modules of this unit.

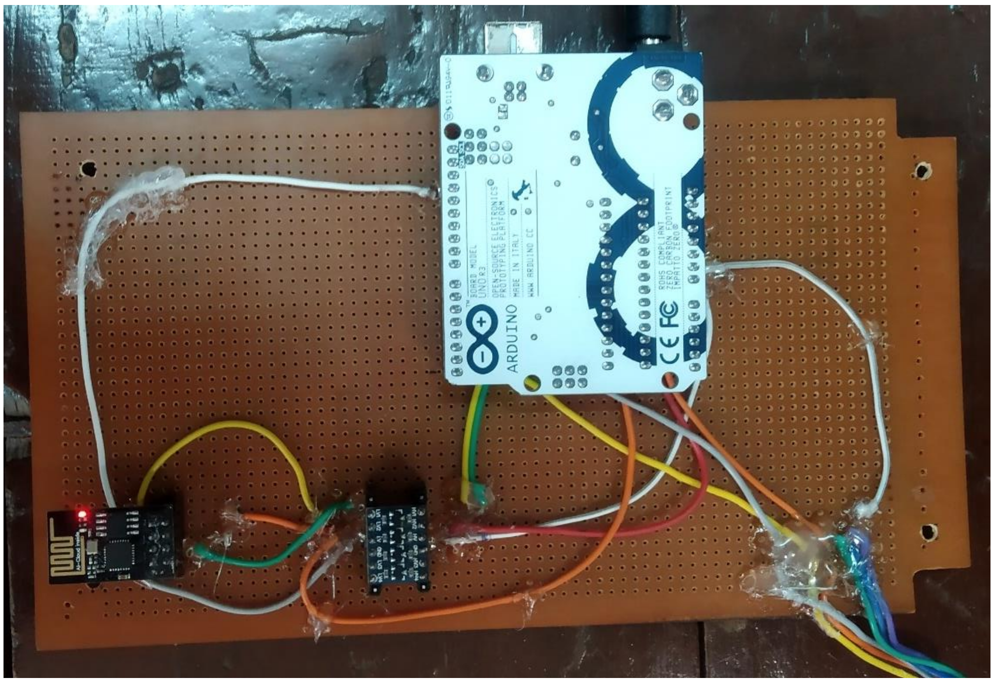


Figure 9
Arduino Shield from General purpose PCB

3.2. UNIT-B

It consists of a Node-MCU for processing and two sensors. Here we used Node-MCU v0.9. This whole unit is powered by an SMPS (Switching Mode Power Supply). Furthermore, for measurement of ambient temperature, relative humidity and barometric pressure, we used pimoroni's BME-280 sensor with an I2C interface module and PMS-1003 from Plantower, which is an optical sensor for measurements of pollutants in the air.

Circuit Diagram

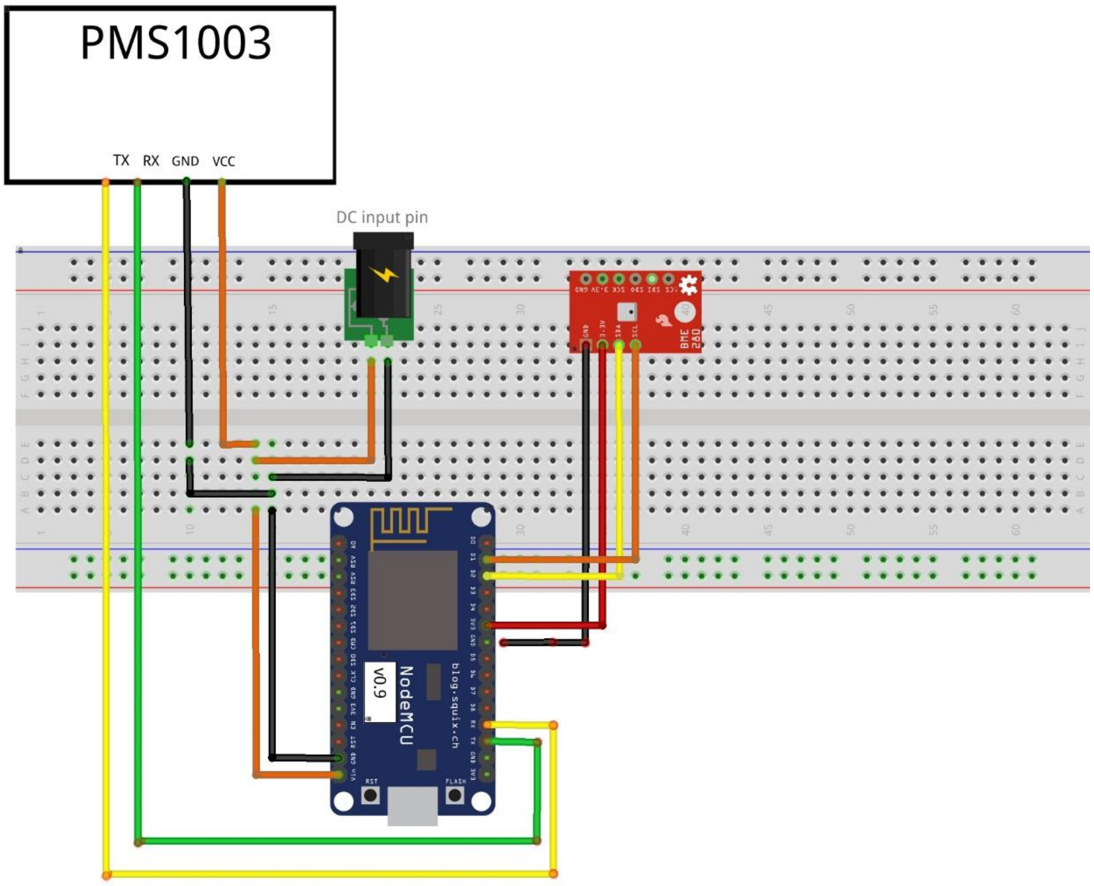


Figure 10
Unit B Breadboard schematic

3.2.1. Adapters

In this unit, PMS-1003 had 1.25mm JST pinouts. So, in order to connect it to Arduino without any JST adapter plate, we constructed our own adapter by soldering wires and a JST connector on a general-purpose PCB. Similarly, for BME-280, we soldered berg pins on a small piece of general-purpose PCB.

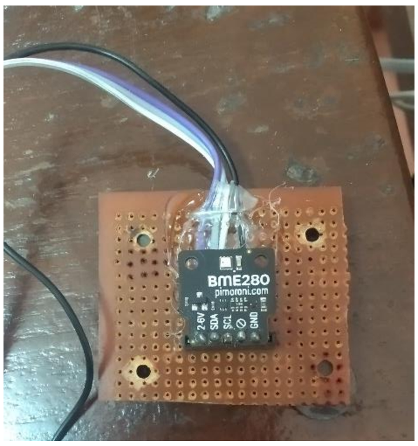


Figure 11
BME280 Adapter

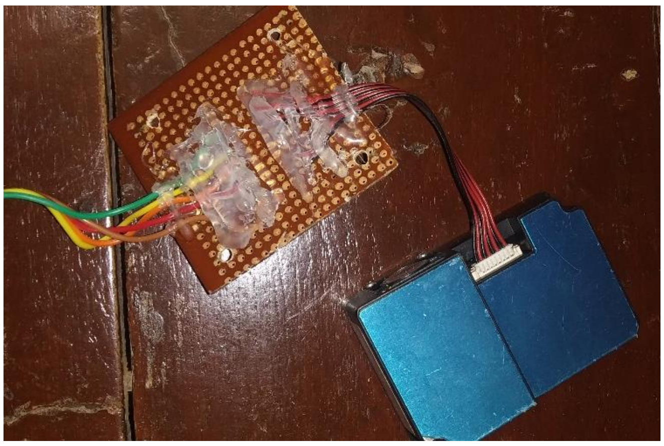


Figure 12
PMS1003 Adapter

3.2.2. Node-MCU Shield

In UNIT-B, similarly, we had to construct our own shield for mounting Node-MCU and other modules. For starters, in a general-purpose PCB we soldered male and female berg pins in order to attach microcontroller, and other connections of this unit.

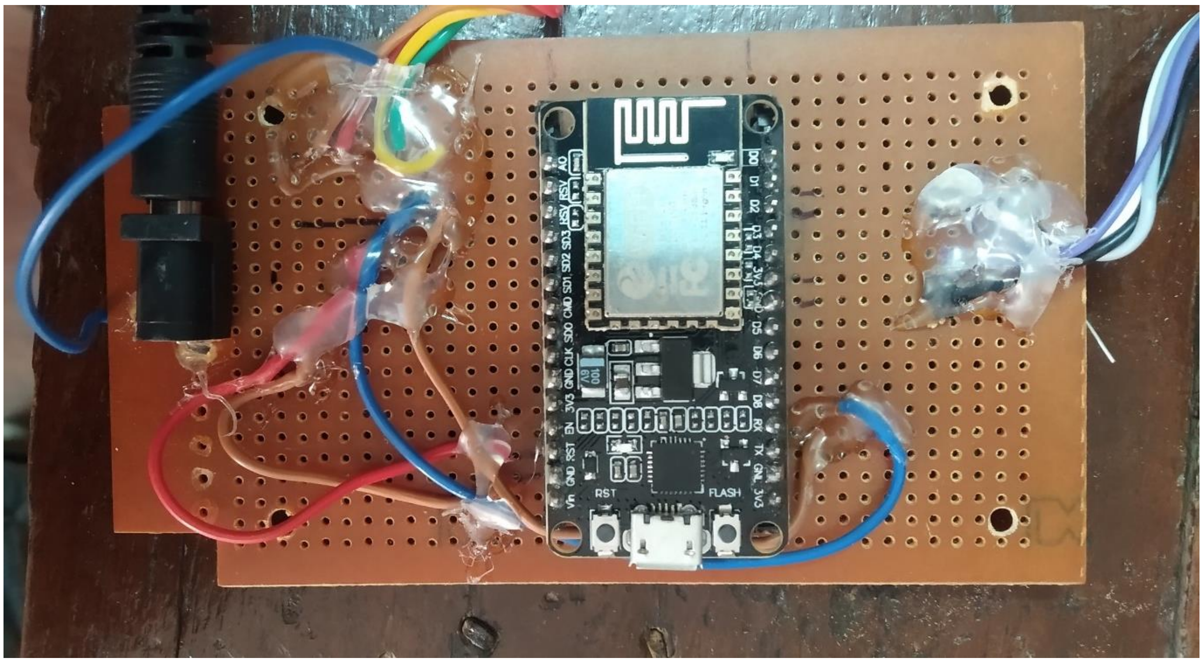


Figure 13
Node-MCU shield from General purpose PCB

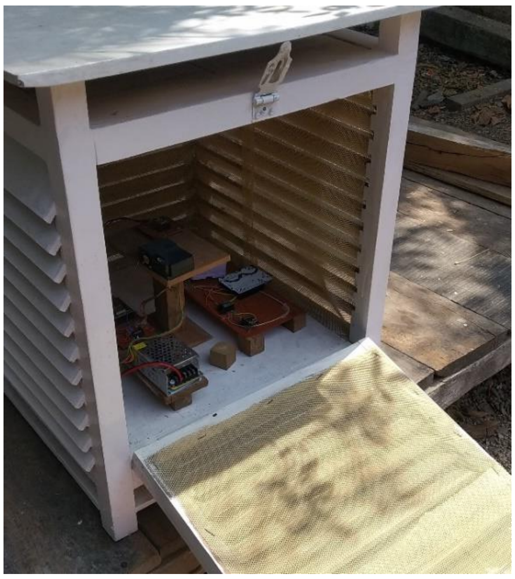
3.3. Stevenson screen

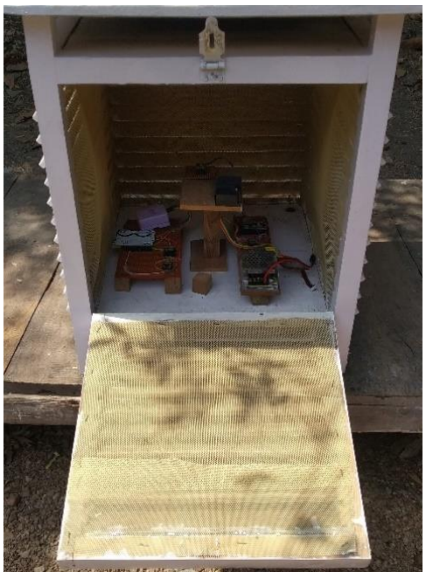
For housing of both units, A and B, we designed our own wooden Stevenson screen (with one sided louvers) at our college workshop. A Stevenson screen a shelter to meteorological instruments against direct heat radiation and rain, while still allowing air to circulate freely around them. Generally, a Stevenson screen is box with double sided lowers on its walls allowing ventilation to the interior, hence, a shelter with direct contact with external environment. It forms part of a standard weather station and holds sensitive instruments that may include thermometer, barometer, etc. However, in our case, we had digital interface.

Figure 14

Figure 15

Figure 16





4. Working

4.1. Unit A

4.1.1. Wind Vane

The flow of air moves the pointer in the direction of blowing wind. The wind vane used in this kit has an analog output, which in response to a particular direction, generates an analog output and the output is noted by the Arduino. Thus, it has different output voltages for different directions. So, from a certain value of voltage, and from the conditional algorithm, we deduce the direction vane pointed.

4.1.2. Anemometer

This is an instrument that measures wind speed, here, as the wind blows, the cups rotate, making the rod spin. Thus, the stronger the wind blows, the faster the rod spins. Inside the centre of revolving cups, a magnet is connected to the spinning rod. With each rotation, when this magnet comes closer to a reed switch it produces a high signal to the Arduino. In Arduino, from the time interval between these signals, it calculates the wind speed in km/h or miles/hour.

4.1.3. Rain gauge

The rain gauge is a self-emptying tipping bucket type. This tipping bucket rain gauge consists of a funnel that collects and channels the precipitation into a small seesaw-like container. After a pre-set amount of precipitation falls, the lever tips, dumping the collected water and sending an electrical signal. Here, there are two magnets on each side of the tipping mechanism, when these magnets come in contact with the hall effect sensors at the bottom of the assembly, it generates a tip. Now, each tip corresponds to 0.2794mm of rain, and which can be recorded with a digital counter or microcontroller interrupt input.

4.1.4. Processing

To process these data in Unit A, after collecting it through RJ-11 adapters, we used an Uno board, here after collecting digital data sets, Uno calculates the following variables: (1) Rain amount (2) Wind Speed (3) Wind direction (4) Wind gust (Previously maximum windspeed). These are then calculated by an inbuilt library and coding we had done in Arduino IDE.

4.1.5. Data transmission

The processed data (i.e. decimal numbers of respective instrument) is given to an ESP-01 module connected to the micro-controller on the shield. Here to operate the module we need to do additional coding in Uno board, such as, which server (ThingSpeak) and which channel to select, Wi-Fi network password and SSID, and to send data sets as strings. The module transmits a set of data packet, which consists of four different decimal numbers, every 20 to 30 seconds to the server.

4.2. UNIT B

4.2.1. BME280

It is a semiconductor-based sensor that comes with a readymade analog to digital converter module for measuring the pressure, humidity and temperature. The sensor has a minute hole on it’s surface by which it reacts to the thermodynamic changes of the surrounding and produces analog signals in an internal circuit. Now, these analog signals from the sensor are subjected to an onboard logic circuit of the sensor module which produces digital signals to be read by the host micro-controller.

4.2.2. PMS1003

In this sensor the data output is taken in the form of digital interface through RX – TX which is connected to the RX-TX pins of Node-MCU. To begin with it is an optical device, where laser scattering principle is used for such sensor, i.e. produce scattering by using laser to radiate suspending particles in the air, then collect scattering light in a certain degree, and finally obtain the curve of scattering light change with time. In the end, equivalent particle diameter and the number of particles with different diameter per unit volume can be calculated by microprocessor based on MIE theory. In a nutshell the PMS1003 sensor shows direct particle concentration on serial monitor, after using a proper library (There are many open source libraries for it in Arduino) and functions in coding.

4.2.3. Microcontroller and Wi-Fi data transmission

Data taken from sensors (BME280 and PMS1003) is processed in NodeMCU and transmitted through the built-in Wi-Fi. Furthermore, the SoC ESP8266 12-E also works as a microcontroller and as a data transmitter. Also, the Node-MCU has an CPH102 chip on board because of which, it is easy to program through Arduino IDE. However, one also needs to install libraries and drivers in Arduino IDE to communicate with ESP 12-E chip. We may have used NodeMCU in Unit A, but we didn’t used because of the incompatibility between Node-MCU and libraries of Weather meter kit.

5. Testing

5.1. Sensor and Arduino Interfaces

Before starting experiments on sensors and Arduino, we needed to know how we can interface and connect a sensor with a given micro-controller. Which means how to retrieve the digital data of measurements in floating point number or to the electrical circuit. Luckily, there are a lot of open source libraries for these sensors on internet, using which we can not only retrieve just a number but also the unit and the scale of the parameter. Furthermore, by using libraries we can obtain the measured data just by calling a pre-defined function within it, which makes it flexible for data processing within the algorithm. Moreover, it saves time for doing coding right from the scratch.

In our case, we started for looking how to interface the BME-280 with an Arduino Uno or a Node-MCU (Which are, however, are similar up to a certain extent). One can easily find about these libraries on internet and can add them to the lib directory of the device or can install the open-source (readymade) libraries given in the Arduino IDE. In our case we installed the library from the IDE, after which, all we had left to do was to use the inbuilt function of different parameters. (One can actually learn about these functions by visiting the respective GitHub page of the library)

After that, we can look for the circuit diagram by which can connect it to the Arduino. Since we had used popular sensors it was easier to know the circuit diagrams of each one.

Similarly, in case of PMS-1003 and weather meter kit we used the same method to carry out the interface, where we just used a certain function to get the data of the certain variable. Hence, for each type of reading we used different functions.

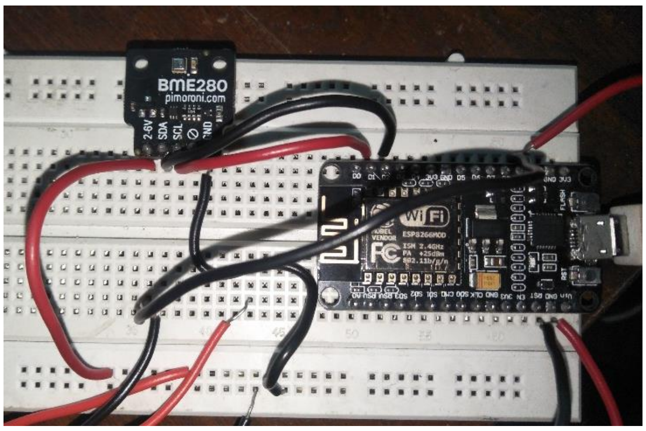
In a nutshell, one should first learn about the interface between the controller and the sensor before going to buy or experimenting in such projects. Also, the circuit connections should be kept in mind as per the project requirements. (However, in many sensor modules available in market, generally, do not need extra components).

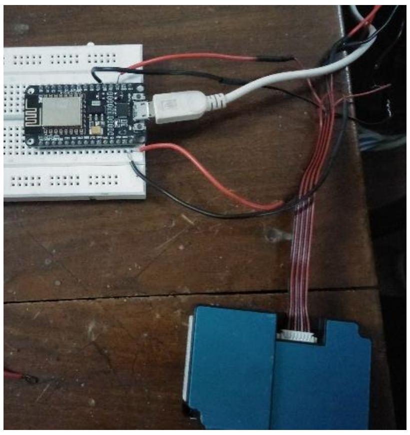
5.2. Communication between sensors and Microcontrollers

The method by which the digital data is transferred between different modules or micro-controllers is known as Serial Communication. During the BME-280 we used I2C communication to connect it with host MCU. In addition to that the PMS-1003 sensor, and ESP-01 Wi-Fi module we used serial communication to establish communication (data transmission) between the host and the module.

5.3. Independent experiments on breadboard

To know the workings and processing of the sensor and Arduino we had experimented on a breadboard to have a temporary circuit, where we did trials and error to attain the proper circuit and coding to get the readings on our serial monitor. Here, we tested each of the sensors separately on the micro-controller and observed the readings.



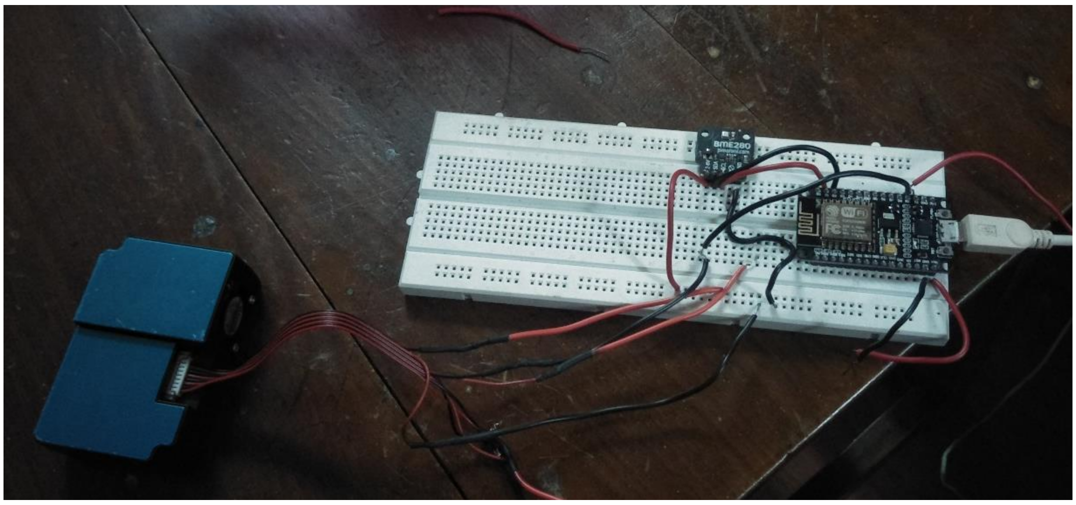


Similarly, for the Sparkfun weather meter kit, we tested each of the instruments separately on the breadboard first before going over to the final PCB. The following images show the experiment of each sensor.

5.4. Experiments for combined system

As we had planned to operate the whole system in two independent units, where each unit had different sensors to be connected. So far, we had only operated each sensor or instrument independently, there were no such difficulties. However, it was a bit problematic when we tried to combine the independent coding of sensors in both UNIT A and UNIT B. Though, we resolved the issues by trial and error after many times, it was due to the errors in syntax and algorithm we might had failed to observe.

The following images show the testing of both the units on breadboard.



5.5. Data transmission

After successfully obtaining the data we were supposed to dump it on our institute's website, in order to do that we used an indirect approach and some changes in our coding. Our aim at first was to send the data over internet to a web-based host (ThingSpeak), and then to import it from there to the college's website. Here, as we have mentioned earlier in section 2, we had Wi-Fi modules (ESP-01 and ESP-12E) for wireless transmission.

6. Programming

6.1. Overview

As stated, the project is divided in two parts. In first part Node-MCU is used as the means of sending the data to the ThingSpeak server. In the first part Arduino-Uno is used as a microcontroller to collect weather data from weather meter kit. Since, it does not have any in-built Wi-Fi module, we needed to provide it with an additional Wi-Fi module which is ESP-01. ESP-01 is programmable through any Arduino device. In the other part, Node-MCU is used which is the combination of a microcontroller and a Wi-Fi module. So, it does not need additional Wi-Fi module to send data. C-programming language is used to program both the microcontrollers with the help of Arduino-IDE.

6.2. Unit-A (Arduino-Uno with Wi-Fi module ESP-01)

In this part, a weather meter kit is connected with Arduino-Uno in order to get the values such as wind-speed, wind-direction, wind-gust and total rainfall of the last hours. To achieve the detection of different data from weather meter kit on Arduino-Uno we used the following libraries,

- ADS Weather
- Software Serial

ADS Weather library helped us gathering all the data provided by the weather meter kit into Arduino-Uno. The other library called Software Serial library is used to establish the serial connection between ESP-01 with Arduino-Uno. Through this communication we could program ESP-01 with the help of Arduino-Uno. Also, we could send the data from Arduino to ESP-01 and then to ThingSpeak server.

6.3. Unit-B (Node-MCU)

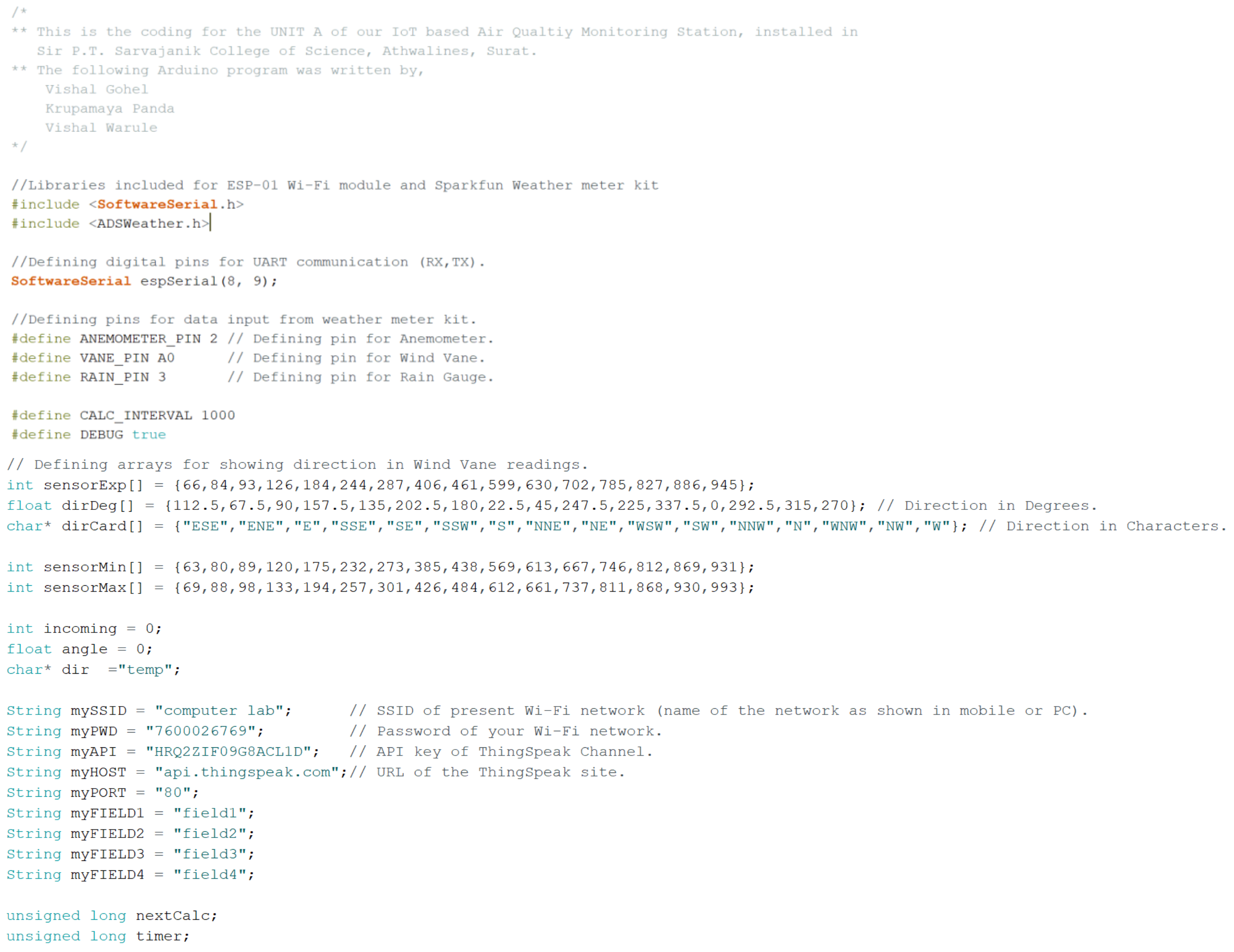
In this part, sensors and modules such as BME-280 and PMS-1003 were connected to Node-MCU. From BME-280, we gathered the data of temperature, humidity and pressure, and using these values, dew point and heat index were calculated, and from PMS-1003, we could get abundance of pollutants in the environment. To collect all the data from two different modules, the major libraries used are as follows,

- PMS
- Adafruit Sensor
- Adafruit BME-280
- ESP8266Wifi

PMS library helped us with collecting the data of particulate matters (PM) of different diameters of 2.5µm, 1µm and 10µm from PMS-1003 sensor. All the programs and calculations for collecting the data of BME-280 and calculating additional data were provided by the libraries, Adafruit Sensor and Adafruit BME-280. These data and calculations were sent to the ThingSpeak server with the help of ESP8266WiFi library.

6.4. Codes

6.4.1. UNIT A Codes



```
/*
** This is the coding for the UNIT A of our IoT based Air Qualtiy Monitoring Station, installed in
   Sir P.T. Sarvajanik College of Science, Athwalines, Surat.
** The following Arduino program was written by,
    Vishal Gohel
    Krupamaya Panda
    Vishal Warule
*/

//Libraries included for ESP-01 Wi-Fi module and Sparkfun Weather meter kit
#include <SoftwareSerial.h>
#include <ADSWeather.h>

//Defining digital pins for UART communication (RX,TX).
SoftwareSerial espSerial(8, 9);

//Defining pins for data input from weather meter kit.
#define ANEMOMETER_PIN 2 // Defining pin for Anemometer.
#define VANE_PIN A0      // Defining pin for Wind Vane.
#define RAIN_PIN 3       // Defining pin for Rain Gauge.

#define CALC_INTERVAL 1000
#define DEBUG true

// Defining arrays for showing direction in Wind Vane readings.
int sensorExp[] = {66,84,93,126,184,244,287,406,461,599,630,702,785,827,886,945};
float dirDeg[] = {112.5,67.5,90,157.5,135,202.5,180,22.5,45,247.5,225,337.5,0,292.5,315,270}; // Direction in Degrees.
char* dirCard[] = {"ESE","ENE","E","SSE","SE","SSW","S","NNE","NE","WSW","SW","NNW","N","WNW","NW","W"}; // Direction in Characters.

int sensorMin[] = {63,80,89,120,175,232,273,385,438,569,613,667,746,812,869,931};
int sensorMax[] = {69,88,98,133,194,257,301,426,484,612,661,737,811,868,930,993};

int incoming = 0;
float angle = 0;
char* dir  ="temp";

String mySSID = "computer lab";      // SSID of present Wi-Fi network (name of the network as shown in mobile or PC).
String myPWD = "7600026769";         // Password of your Wi-Fi network.
String myAPI = "HRQ2ZIF09G8ACL1D";   // API key of ThingSpeak Channel.
String myHOST = "api.thingspeak.com";// URL of the ThingSpeak site.
String myPORT = "80";
String myFIELD1 = "field1";
String myFIELD2 = "field2";
String myFIELD3 = "field3";
String myFIELD4 = "field4";

unsigned long nextCalc;
unsigned long timer;
```

```
int windDir;
int windSpeed;
int rainAmmount;

ADSWeather ws1(RAIN_PIN, VANE_PIN, ANEMOMETER_PIN); //This should configure all pins correctly


void setup()
{
  Serial.begin(9600); // For communication between Arduino Board and your PC/device.
  espSerial.begin(115200); // For communication of ESP-01 module with PC/device.

  espData("AT+RST", 1000, DEBUG);
  espData("AT+CWMODEqqqqqqqQqqwq=1", 1000, DEBUG);
  espData("AT+CWJAP=\""+ mySSID +"\",\""+ myPWD +"\"", 1000, DEBUG);

//ws1.countRain is the ISR for the rain gauge.
  attachInterrupt(digitalPinToInterrupt(RAIN_PIN), ws1.countRain, FALLING);

//ws1.countAnemometer is the ISR for the anemometer.
  attachInterrupt(digitalPinToInterrupt(ANEMOMETER_PIN), ws1.countAnemometer, FALLING);
  nextCalc = millis() + CALC_INTERVAL;

  delay(1000);
}



  void loop()
  {
    timer = millis();
    int rainAmmount;
    float windSpeed;
    int windGust;

    ws1.update(); //Call this every cycle in your main loop to update all the sensor values

   if(timer > nextCalc)
    {
      nextCalc = timer + CALC_INTERVAL;
      rainAmmount = ws1.getRain();
      windSpeed = ws1.getWindSpeed();
      windGust = ws1.getWindGust();
      // Use of inbuilt functions in the 'AWSWeather' library.

  //    This will print the calculated wind speed data on Serial Monitor.

      // Here windSpeed / 10 will give the interger component of the wind speed
      // Here windSpeed % 10 will give the fractional component of the wind speed
      Serial.print("Wind speed: ");
      Serial.print(windSpeed / 100*1.609);
      Serial.print('.');
      Serial.print(windSpeed % 10);

  //    This will print the calculated wind gust data on Serial Monitor.
      Serial.print("Gusting at: ");
      Serial.print(windGust / 100*1.609); // To  get answer in km/h.
      Serial.print('.');
      Serial.print(windGust % 10);
      Serial.println("");

  //    This will print the calculated wind direction data on Serial Monitor.
      Serial.print("Wind Direction: ");
      Serial.print(ws1.getWindDirection()); // To get the numerical data of direction in degrees.
      Serial.println("");


  //    This will print the calculated data of total rain happened in last one day, on Serial Monitor.
      Serial.print("Total Rain: ");
      Serial.println((float) rainAmmount / 1000); // To data in units of inches.

    }

  // For calculating direction from the measured voltage in Wind Vane.
  incoming = analogRead(VANE_PIN);
  for(int i=0; i<=15; i++) {
      if(incoming >= sensorMin[i] && incoming <= sensorMax[i]) {
         dir = dirCard[i];
         angle = dirDeg[i];
```

```
          break;
        }
      }

    // For printing direction in Characters.
       Serial.print(dir);
       Serial.print(angle);
       Serial.print(":\n");


       String sendData = "GET /update?api_key="+ myAPI +"&"+ myFIELD1 +"="+String((float) rainAmmount / 1000)+
       "&"+ myFIELD2 +"="+String((windSpeed/100)*1.609)+"&"+ myFIELD3 +"="+String(angle)+"&"+ myFIELD4 +"="+String(windGust/100);
       espData("AT+CIPMUX=1", 1000, DEBUG);
       espData("AT+CIPSTART=0,\"TCP\",\""+ myHOST +"\","+ myPORT, 1000, DEBUG);
       espData("AT+CIPSEND=0," +String(sendData.length()+4),1000,DEBUG);
       espSerial.find(">");
       espSerial.println(sendData);
       espData("AT+CIPCLOSE=0",1000,DEBUG);

       delay(500);

    }


     // Function for communication with ESP-01 module.
     String espData(String command, const int timeout, boolean debug)
     {
       Serial.print("AT Command ==> ");
       Serial.print(command);
       Serial.println("     ");

       String response = "";
       espSerial.println(command);
       long int time = millis();
       while ( (time + timeout) > millis())
       {
         while (espSerial.available())
         {
           char c = espSerial.read();
           response += c;
         }
       }
       if (debug)
       {
         //Serial.print(response);
       }
       return response;
     }
```

6.4.2. UNIT B Codes

```
/*
** This is the coding for the UNIT A of our IoT based Air Qualtiy Monitoring Station, installed in
   Sir P.T. Sarvajanik College of Science, Athwalines, Surat.
** The following Arduino program was written by,
    Vishal Gohel
    Krupamaya Panda
    Vishal Warule
*/

#include "PMS.h" // Calling library of PMS1003.
#include<ESP8266WiFi.h> //Calling library of ESP8266(ESP-12E)
#include <Arduino.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h> // Calling libraries in Arduino for sensor interface.
#include <Adafruit_BME280.h> // Calling library for BME280 sensor

Adafruit_BME280 bme;


PMS pms(Serial);
PMS::DATA data;


String apiKey = "QF253KQPF9YSH2CX";         // API key of ThingSpeak Channel.
const char* ssid = "computer lab";          // SSID/Given name of your present Wi-Fi network as shown in PC/Device.
const char* password = "12345****54";       // Password of your Wi-Fi network.
const char* server = "api.thingspeak.com";  // URL of the ThingSpeak site.
```

```
WiFiClient client;


void setup()
{
  Serial.begin(9600); // For communication between Node-MCU Board and your PC/device.
  if (!bme.begin(0x76)) {
        Serial.println("Could not find a valid BME280 sensor, check wiring!");
        while (1);
    }

    Serial.println("-- Default Test --");
    int delayTime = 1000;
    Serial.println();

    delay(100); // let sensor boot up

       Serial.println("Connecting to ");
       Serial.println(ssid);
       WiFi.begin(ssid, password);

      while (WiFi.status() != WL_CONNECTED)
     {
            delay(500);
            Serial.print(".");
     }
      Serial.println("");
      Serial.println("WiFi connected");



           // Printing the ESP IP address
           Serial.println(WiFi.localIP());

     }

     void loop()
     {
       //Defining in-built funcions of PMS1003 and BME280 libraries.
       float rH = bme.readHumidity();
       float temp = bme.readTemperature();
       float pres = bme.readPressure()/100.0F;
       float dP = bme.readTemperature() - ((100-rH)/5);
       float hI = 0.5 * (bme.readTemperature() + 61.0 + ((bme.readTemperature()-68.0)*1.2) + (rH*0.094));

       if (isnan(rH) || isnan(temp) || isnan(pres) || isnan(dP) || isnan(hI))
       {
         Serial.println("Failed to read from BME sensor!");
         return;
       }


     if (pms.read(data)){
       if (client.connect(server,80))   //   "184.106.153.149" or api.thingspeak.com
         {


            String postStr = apiKey;
            postStr +="&field1=";
            postStr += String(temp);
            postStr +="&field2=";
            postStr += String(pres);
            postStr +="&field3=";
            postStr += String(rH);
            postStr +="&field4=";
            postStr += String(dP);
            postStr +="&field5=";
            postStr += String(hI);
            postStr +="&field6=";
            postStr += String(data.PM_AE_UG_1_0);
            postStr +="&field7=";
            postStr += String(data.PM_AE_UG_2_5);
            postStr +="&field8=";
            postStr += String(data.PM_AE_UG_10_0);
            postStr += "\r\n\r\n";
            postStr += "\r\n\r\n";
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
    Serial.print("Temperature: ");
    Serial.print(temp);
    Serial.print(" degrees Celcius,\nHumidity: ");
    Serial.print(rH);
    Serial.println("%");
    Serial.print("Pressure: ");
    Serial.print(pres);
    Serial.println("hPA");
    Serial.print("Dew Point: ");
    Serial.print(dP);
    Serial.println("*C");
    Serial.print("Heat Index: ");
    Serial.print(hI);
    Serial.println("*C");
    Serial.println("Sent to ThingSpeak");
 }
    Serial.println("Waiting...");

  // thingspeak needs minimum 15 sec delay between updates, i've set it to 30 seconds
  delay(1000);
  client.stop();
}
    Serial.println("Waiting...");
}
```

7. Data

7.1. Visualization

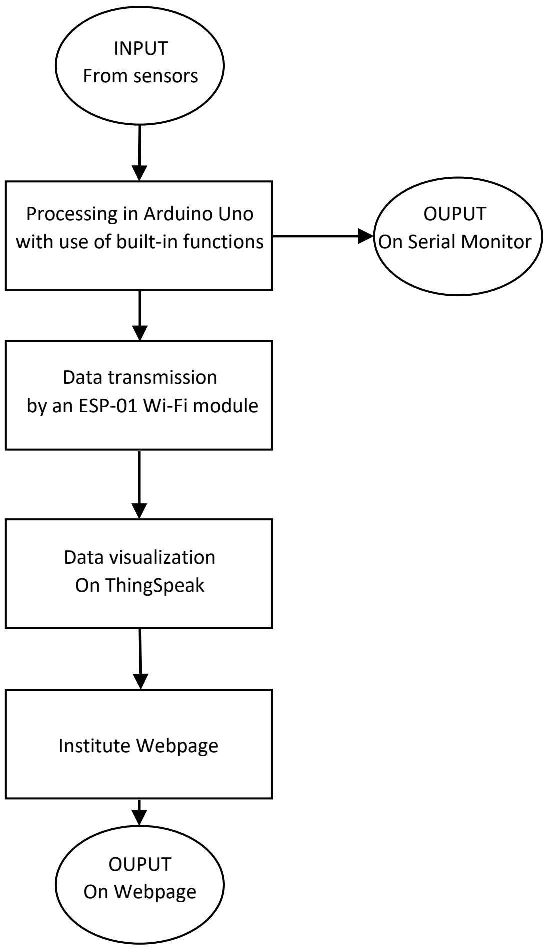
To plot the meteorological variables such as Temperature, Pressure, Humidity, and etc. we used a web-based application. Here, from the received data over the time, the Thingspeak plots a real-time graph with a data update within an interval of 15 seconds. Also, it has many features included to generate numerous kinds of visualizations based on MATLAB's web-based environment. In this project, we used the free license version of the channel, which allowed us to visualize up-to eight different variables per channel to visualize. So far, we have 2 channels with total 12 visualizations of 12 parameters.

7.2. Import and Export

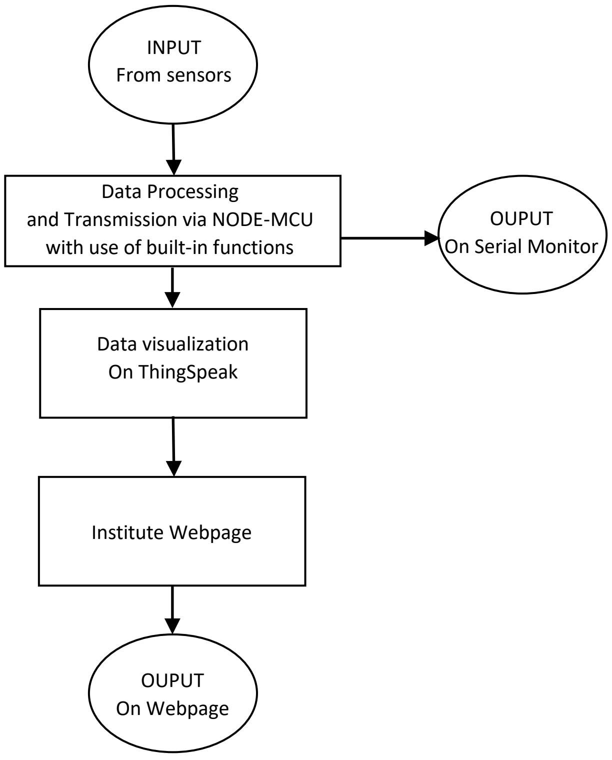
As mentioned earlier in section 5.5, in order to import the data from the host MCUs to the college's website we first exported it to the ThingSpeak channel. It was done by using providing the channel id in the programming of both units A and B. Now, one of the features of the ThingSpeak is it allows to export the data to another address (webpage) as well. So, we included the channel ids in our HTML code. This whole process for data, from sensors to the college's website, takes an approximate time of 30 seconds. To have the data in form of CSV files, users can download files by logging in to the channel.

7.3. Flowchart

7.3.1. UNIT A


INPUT
From sensors
Processing in Arduino Uno with use of built-in functions
OUPUT
On Serial Monitor
Data transmission by an ESP-01 Wi-Fi module
Data visualization On ThingSpeak
Institute Webpage
OUPUT
On Webpage

7.3.2. UNIT B


INPUT
From sensors
Data Processing and Transmission via NODE-MCU with use of built-in functions
OUPUT
On Serial Monitor
Data visualization On ThingSpeak
Institute Webpage
OUPUT
On Webpage

8. Webpage for the meteorological data

To show the meteorological data on our college’s website we created a web-page with help of languages HTML, CSS, JS. The complete look of the webpage has been given in the Figure x. As mentioned in earlier sections, the data is gathered through the export links of our ThingSpeak channels along with real-time data visualization of ambient temperature. The web page is divided into 6 parts, 3 of the weather station data, and other three consisting of ThingSpeak channel links, the following documentation and details of the project members.

The first section in web-page contains the data from **BME-280**, temperature, pressure, heat index, dew point, relative humidity and real-time visualization of ambient temperature. The second section, comprises of AQI data as measured by **PMS1003** sensor, and shows the values **PM 1.0**, **PM 2.5**, **PM 10.0** in units of µg/m^3. The largest among these three values of **PM** concentration is defined as **AQI index** on our webpage. Finally, in the third section the wind speed, direction (in characters), wind gust and total rainfall of last 24 hours is shown.

Section 4 has two buttons, each with link for the corresponding channel’s public view, following which the webpage has documentation and details of makers of this project.