**Deliverable II: Progress report plus code explanation**

**Authors : M.Faiq 18L2137**

**Arbab Hamd Rizwan 18L-0756**

**Zeerak Ahmed 18L-0949**

**Hunzalah Malik 18L-1139**

**Abstract:**

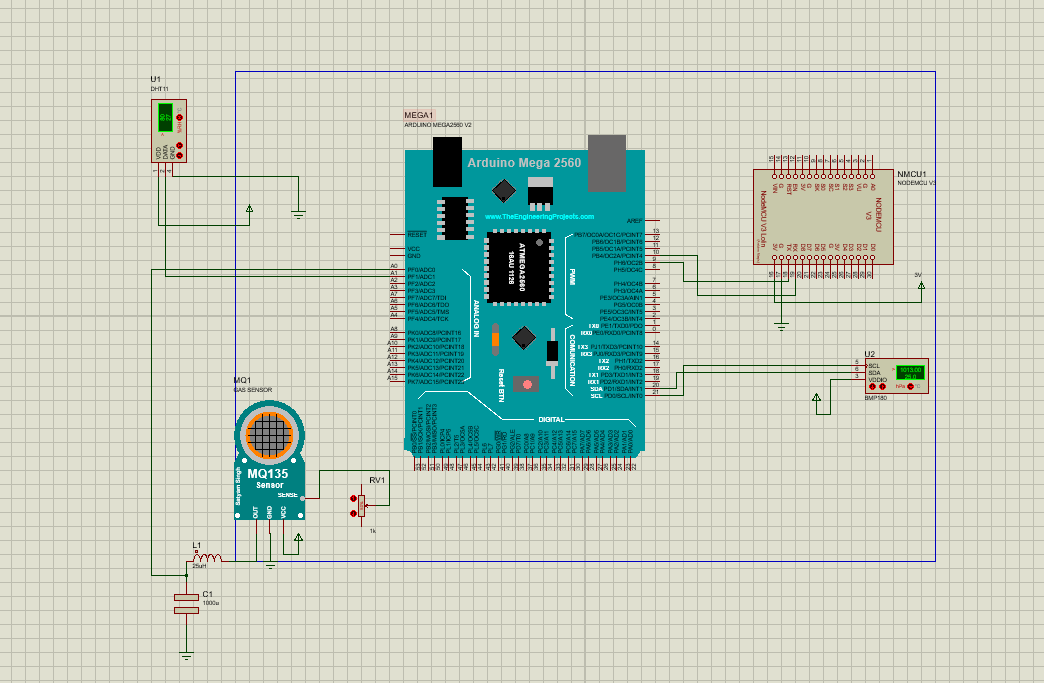
In this project, we have almost implemented a weather station that takes into account following measurable quantities and provides analytics on a website:

1. Air Quality
2. Temperature and humidity
3. Air pressure using a barometric sensor
4. The altitude at which these measurements are taken

We used proteus to make a virtual circuit diagram of the project and used arduino mega along with mq135 gas sensor to measure air quality, bmp180 pressure,altitude sensor and dht11 humidity and temperature sensor. The data captured is sent to a cloud service(things peak).

**Proteus Sketch:**

Below shown is a proteus sketch which is to be used as a circuit diagram for our project. Note however that the wifi module; node mcu, is a wrapper around ESP8266 and isn’t a simulatable component in proteus. And there exist no other implementations of this component in the current proteus installations.

****

**Code Explanation:**

First we include the required libraries as follows :

**#include <SoftwareSerial.h>**

**#include <dht.h>**

**#include <MQ135.h>**

**#include <SFE\_BMP180.h>**

**#include <math.h>**

Software serial is used for esp8266 wifi modules and any data we want to send to our cloud, will be sent by its created object;however, for this deliverable we won’t be showing its usage and will be left for the 3rd deliverable. The rest are fairly straight forward.

**dht DHT;**

**MQ135 mq = MQ135(A0);**

**SFE\_BMP180 bmp180;**

**float Po = 1013.0;**

**float valAirQuality = 1;**

**float valTemperature = 1;**

**float valHumidity = 1;**

**double valPressure = 1;**

**double valAltitude = 1;**

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

Above are the declarations and their suitable initialisations. Note however that we didn’t initialize any analog pins for DHT and bmp180. DHT uses the pin during its function call hence it isn’t needed in the initialisation. Bmp180 however connects with SCL and SDA which are specific pin inputs labeled on the board.

**void setup() {**

**Serial.begin(9600); // start serial monitor , baud rate 9600**

**Serial.println("Weather Monitor System v1.0 Initialized");**

**esp8266.begin(115200); // start Esp8266 object**

**bool success = bmp180.begin();**

**if(success==1)**

**{**

**Serial.println("Baromatric Sensor Active!");**

**}**

**}**

Above is the setup function and for the reasons mentioned, we didn’t need to initialize input modes of the pins from which the data will be taken. Like the serial port, we need to initialize the software serial object used for the wifi module to a specific baudrate. For esp8266, it's 115200.

**void loop() {**

**int rzero = (int) mq.getRZero();**

**Serial.print("Rzero is :");**

**Serial.println(rzero);**

**valAirQuality = getAirQuality();**

**valTemperature = getTemp();**

**valPressure = pressureVal();**

**valAltitude = getAltitude();**

**Serial.println(valTemperature);**

**Serial.println(valPressure);**

**Serial.println(valAltitude);**

**Serial.println(valAirQuality);**

**}**

The code above is fairly straightforward; however, note that using the gas sensor, we make a RZero function call. Documentation for the usage of this library recommends taking this calibration reading to know whether the sensor is working fine or not. Even though the details are still very vague and purely beyond the scope of our understanding as a computer science student.

**int getAirQuality(){**

**return analogRead(mq.getPPM());**

**}**

**float getTemp()**

**{**

**DHT.read11(A1);**

**return DHT.temperature;**

**}**

**float getHumidity() //change as per your case**

**{**

**DHT.read11(A1);**

**return DHT.humidity;**

**}**

The function implementations above are fairly trivial and hence don’t need any kind of explanation.

**double getAltitude()**

**{**

**boolean status = bmp180.startTemperature();**

**double T,P;**

**double alt;**

**if (status != 0) {**

**delay(1000);**

**status = bmp180.getTemperature(T);**

**if (status != 0) {**

**status = bmp180.startPressure(3);**

**if (status != 0) {**

**delay(status);**

**status = bmp180.getPressure(P, T);**

**if(status!=0)**

**{**

**alt = bmp180.altitude(P,Po);**

**return alt;**

**}**

**else**

**{**

**return 0;**

**}**

**}**

**else**

**{**

**return 0;**

**}**

**}**

**else {**

**return 0;**

**}**

**}**

**else {**

**return 0;**

**}**

**}**

**double pressureVal()**

**{**

**boolean status = bmp180.startTemperature();**

**double T,P;**

**if (status != 0) {**

**delay(1000);**

**status = bmp180.getTemperature(T);**

**if (status != 0) {**

**status = bmp180.startPressure(3);**

**if (status != 0) {**

**delay(status);**

**status = bmp180.getPressure(P, T);**

**if(status!=0)**

**{**

**return (P);**

**}**

**else**

**{**

**return 0;**

**}**

**}**

**else**

**{**

**return 0;**

**}**

**}**

**else {**

**return 0;**

**}**

**}**

**else {**

**return 0;**

**}**

**}**

Pressure and altitude functions we believe need some explanation which are using the bmp180 pressure and altitude sensor. However, it should be understood that the sensor works in a sequential manner and takes a range of atmospheric readings before proceeding to next ones. Each start call returns the wait time before the next reading can be taken which should be followed by a delay call using the returned wait signal. In case, it isn’t possible to take the requested reading, a zero is returned. After the start call is made, we finally take the measurement and initiate the next start call according to the sequence. The rest is the same as the process described above done in a sequential manner.

Here is the typical sequence taken by bmp180 sensor:

Temperature > Pressure > Altitude

**Conclusion:**

We are most likely done with implementing the main logic of this project and all that’s left is sending these measurements to our website for further processing by using a ThingSpeak api call. For now, we have achieved the following milestones:

1. Implementation of logic
2. Arduino Sketch implementation
3. Circuit Design using proteus