

CS3242 - Micro-controllers and Applications

Environment Monitoring IoT device

S.K Ekanayake

170158F

Content

1. Scope of the project	3
2. Special features.....	3
3. High level design diagram	4
4. List of components and their costs	5
5. Schematic diagram	6
6. Fault recovery option implemented	7
7. Project circuit photo	7
8. How the device work	
a) Join to any Wi-Fi through configuration setup	9
b) Sensor data real-time upload to the google excel sheet in every 15min....	10
c) Sending sensor data to local web server using CAP protocol.	11
d) SD card saved data	11
9. Pseudo code algorithm	12
10. Used libraries for the project	13
11. Full source code	
a) Environment_Monitoring_System.ino code	14
b) PageIndex.h code	21
c) Google Sheet Script code	23
12. References.....	24

Scope of the project

The Environment Monitoring IoT device is capable of measuring the following parameters.

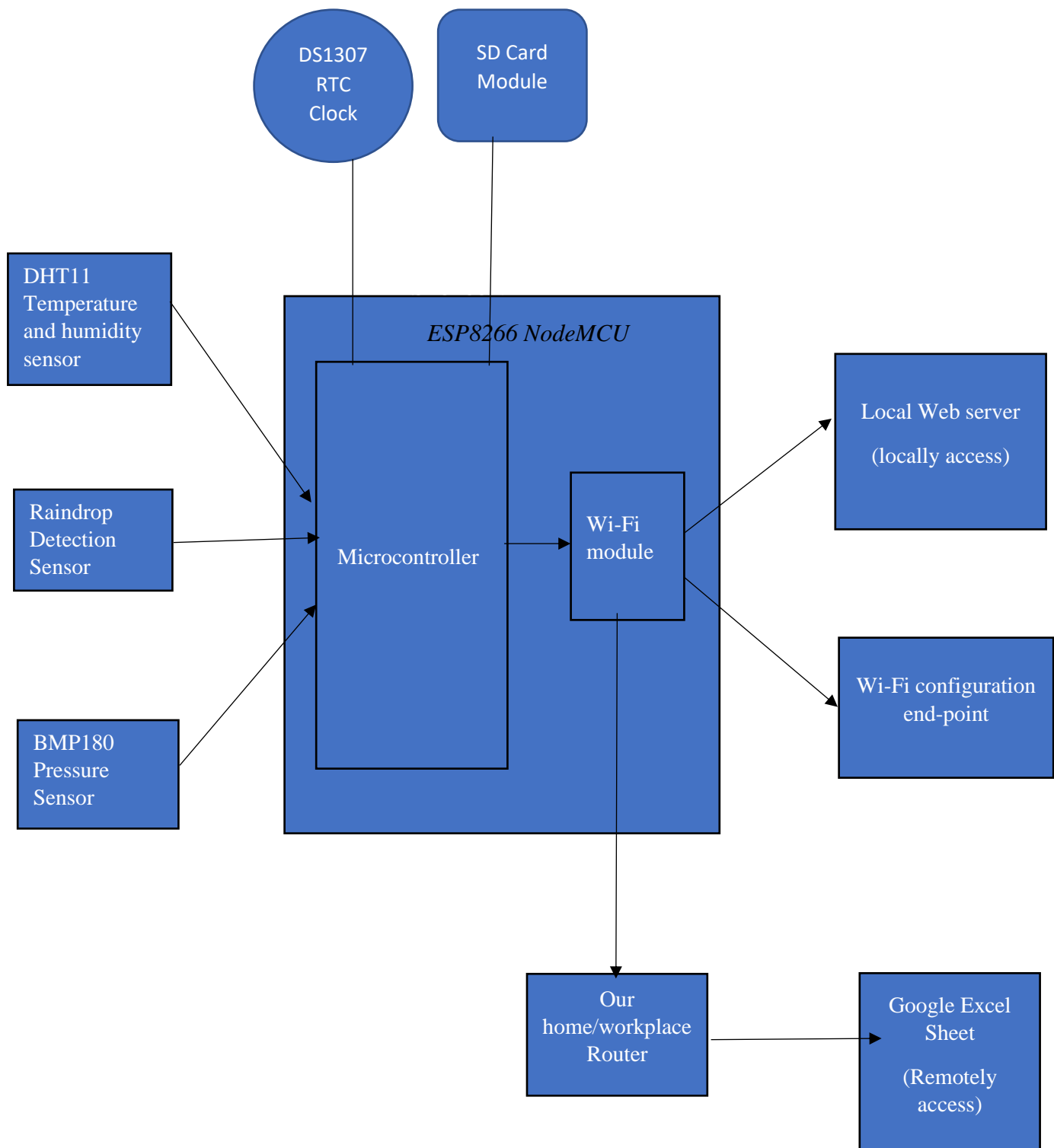
1. Temperature of the environment (in $^{\circ}\text{C}$)
2. Relative humidity of the environment (in %)
3. Rainfall of the environment (in %)
4. Air pressure (in Pa)
5. Amplitude (in m)

The device collects data every 2 seconds via the sensors. Then the data real-time sending to the google excel sheet (which can access remotely) every 15 minutes when devices is ON and connected to the Wi-Fi. These data are formatted to have the CAP format and sent to the web server which is locally connected through the Node MCU network. The data gets updated real time in the local web server and can be monitored via an interface.

Special features

1. Power saving – Buck converter ensure that power saving. Which is done by Buck converters are highly efficient (often higher than 90%), making them useful for tasks such as converting a computer's main supply voltage (often 12 V) down to lower voltages needed by USB.
2. Different types of power inputs
 - i. Using DC power socket – variable voltage input can be given (in range of 7V to 35V)
 - ii. Micro USB port – USB power input(5V)
3. Inbuild storage memory – using SD card to store data as backup of sensor read data.
4. Real time clock – To make sure which data is related which date and time.
5. Can connect to any Wi-Fi router – using network configuration UI

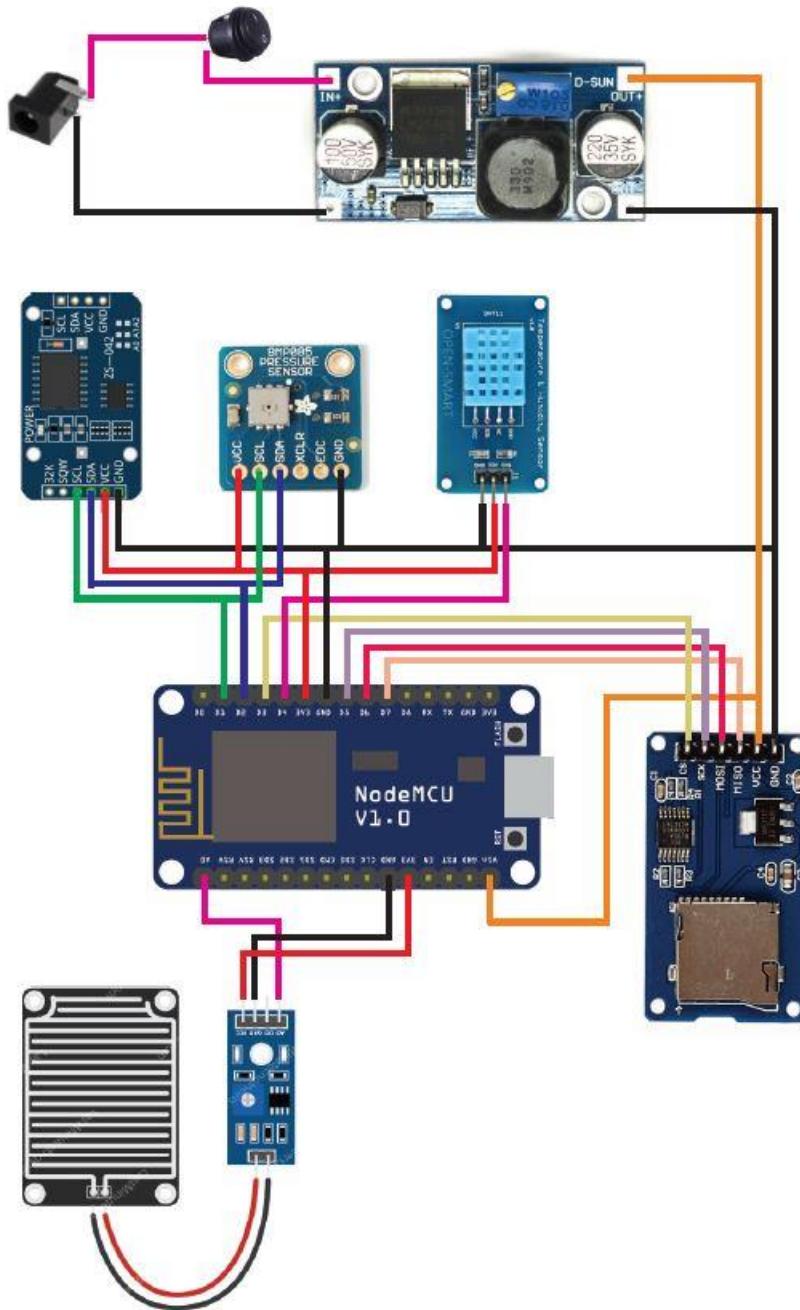
High level design diagram



List of components and their cost

	<i>Components</i>	<i>price (Rs.)</i>
I.	ESP8266 (NodeMCU)	650
II.	DHT11	200
III.	BMP180	330
IV.	DS1307 clock RTC module	180
V.	Micro SD card module	160
VI.	Raindrop Detection sensor	190
VII.	DC-DC Buck Converter	250
VIII.	CR2032 Watch Clock battery	150
IX.	DC-005 DC power socket	40
X.	Switch	20
XI.	Voltage regulator	20
XII.	Jumping Wires	200
XIII.	Dot board	60

Schematic Diagram

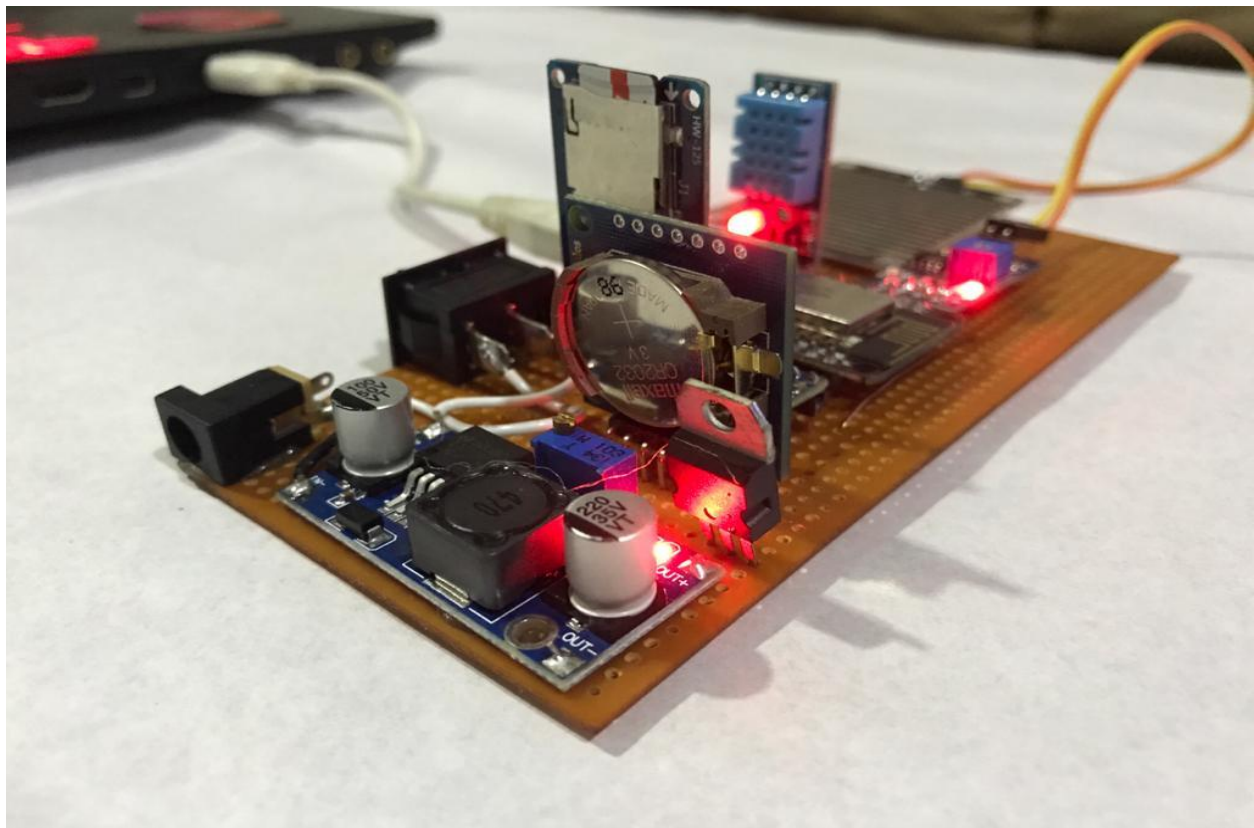


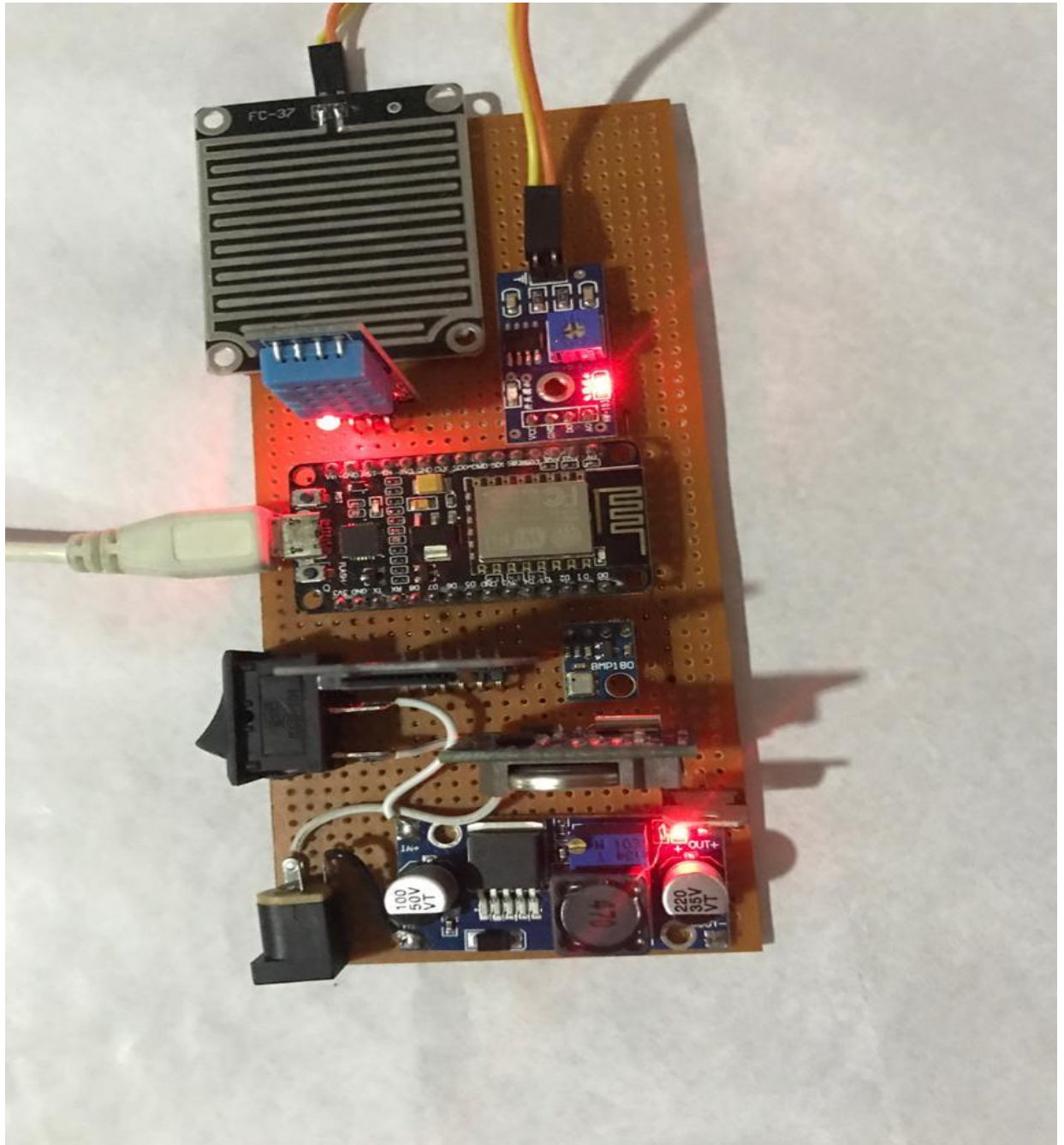
Fault recovery option implemented

- I. Save the sensor read data into a SD card
- II. Make temporary array and sending data when the connection stabilishes again

When every 15 minutes data is sending to the google excel sheet. When data sending to excel sheet, I used SD card module to store all data in a micro SD card. So, if connection is lost but the data is safely kept in the SD card. In addition, in coding I implemented arrays which are collect sensor data only if there is no Wi-Fi connection. When the internet connection stabilishes again all the data in the temp arrays are send again to the excel sheet. For update the sheet with real sensor read time, I used DS1307 clock RTC module. That will help to recovery the data in fault with correct sensor read time.

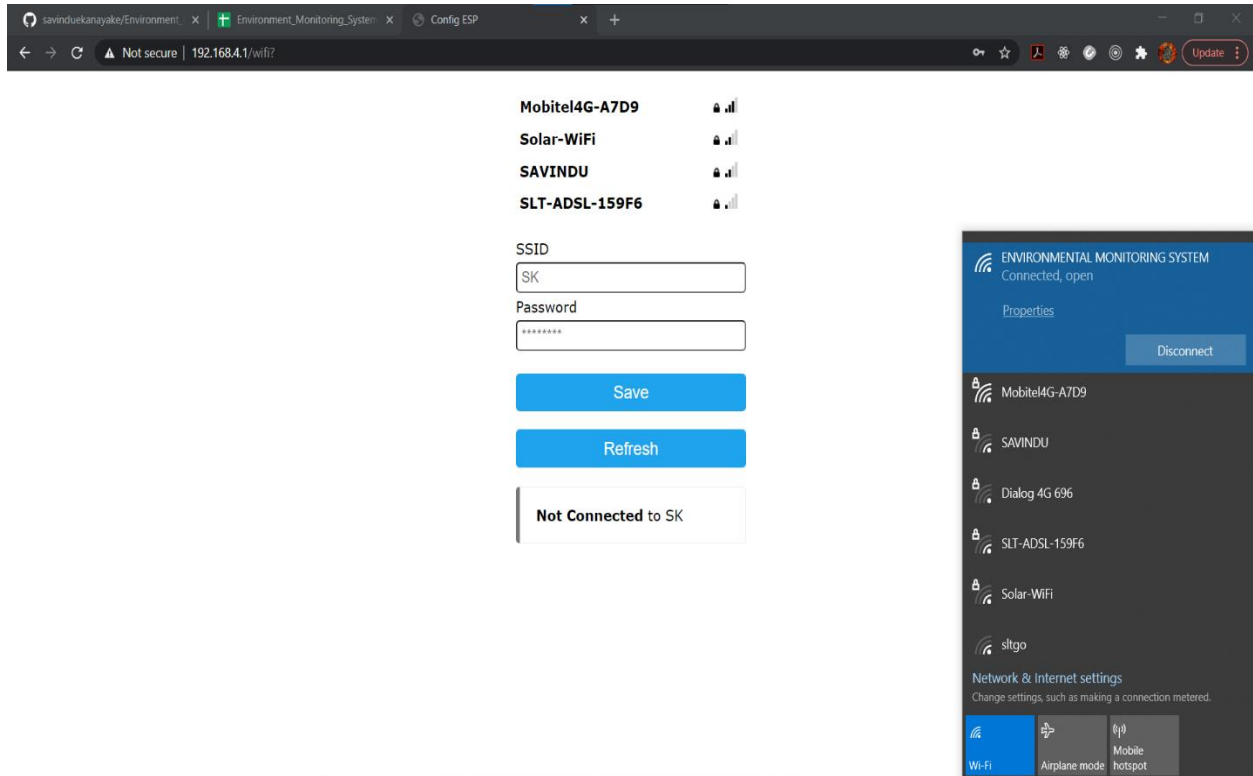
Project circuit photo





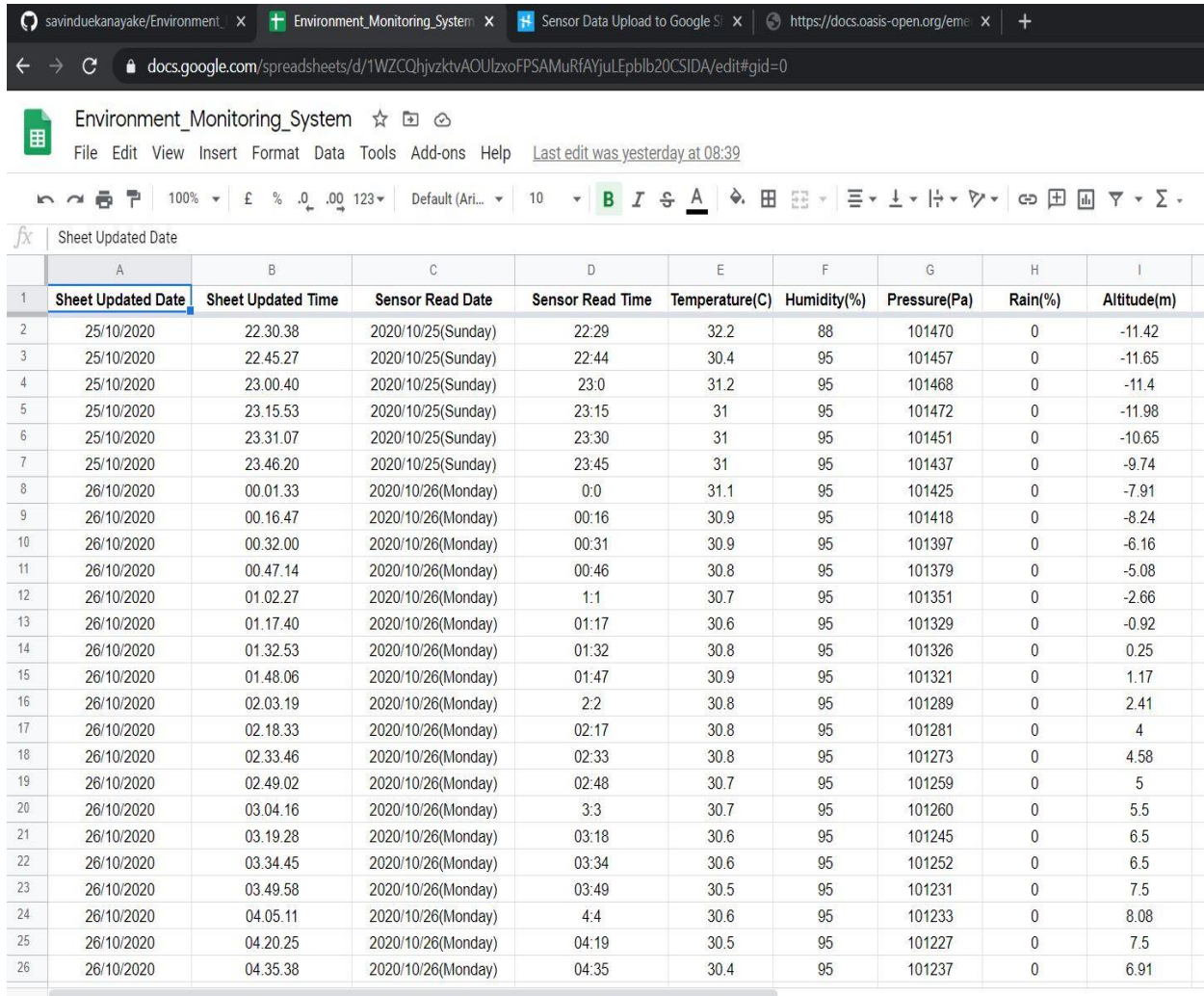
How the Device work

a) Join to any Wi-Fi through configuration setup



- First switch ON the device.
- Search the available Wi-Fi connections using Laptop.
- Then connect to the Wi-Fi 'ENVIRONMENT_MONITORING_SYSTEM'.
- Then the configuration setup will auto popup in the laptop. (if not, we can go to configuration setup using IP- 192.168.4.1)
- Then connect to your Wi-Fi using entering correct credentials to the configuration interface.
- Now you are connected to the internet through the router.

b) Sensor data real-time upload to the google excel sheet in every 15min

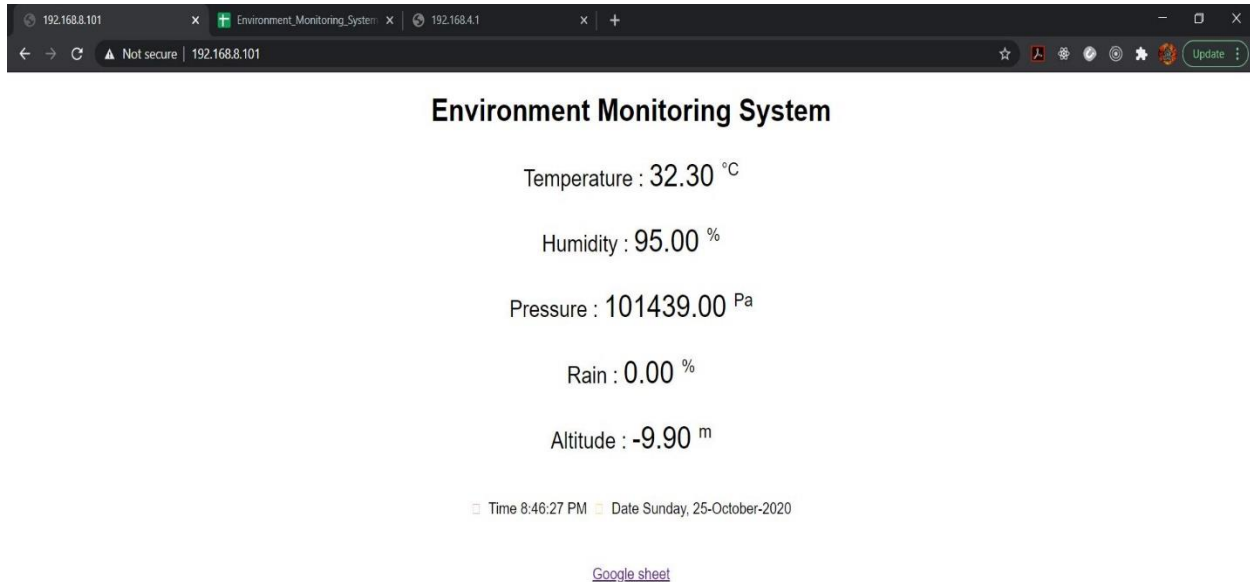


	A	B	C	D	E	F	G	H	I
1	Sheet Updated Date	Sheet Updated Time	Sensor Read Date	Sensor Read Time	Temperature(C)	Humidity(%)	Pressure(Pa)	Rain(%)	Altitude(m)
2	25/10/2020	22.30.38	2020/10/25(Sunday)	22:29	32.2	88	101470	0	-11.42
3	25/10/2020	22.45.27	2020/10/25(Sunday)	22:44	30.4	95	101457	0	-11.65
4	25/10/2020	23.00.40	2020/10/25(Sunday)	23:0	31.2	95	101468	0	-11.4
5	25/10/2020	23.15.53	2020/10/25(Sunday)	23:15	31	95	101472	0	-11.98
6	25/10/2020	23.31.07	2020/10/25(Sunday)	23:30	31	95	101451	0	-10.65
7	25/10/2020	23.46.20	2020/10/25(Sunday)	23:45	31	95	101437	0	-9.74
8	26/10/2020	00.01.33	2020/10/26(Monday)	0:0	31.1	95	101425	0	-7.91
9	26/10/2020	00.16.47	2020/10/26(Monday)	00:16	30.9	95	101418	0	-8.24
10	26/10/2020	00.32.00	2020/10/26(Monday)	00:31	30.9	95	101397	0	-6.16
11	26/10/2020	00.47.14	2020/10/26(Monday)	00:46	30.8	95	101379	0	-5.08
12	26/10/2020	01.02.27	2020/10/26(Monday)	1:1	30.7	95	101351	0	-2.66
13	26/10/2020	01.17.40	2020/10/26(Monday)	01:17	30.6	95	101329	0	-0.92
14	26/10/2020	01.32.53	2020/10/26(Monday)	01:32	30.8	95	101326	0	0.25
15	26/10/2020	01.48.06	2020/10/26(Monday)	01:47	30.9	95	101321	0	1.17
16	26/10/2020	02.03.19	2020/10/26(Monday)	2:2	30.8	95	101289	0	2.41
17	26/10/2020	02.18.33	2020/10/26(Monday)	02:17	30.8	95	101281	0	4
18	26/10/2020	02.33.46	2020/10/26(Monday)	02:33	30.8	95	101273	0	4.58
19	26/10/2020	02.49.02	2020/10/26(Monday)	02:48	30.7	95	101259	0	5
20	26/10/2020	03.04.16	2020/10/26(Monday)	3:3	30.7	95	101260	0	5.5
21	26/10/2020	03.19.28	2020/10/26(Monday)	03:18	30.6	95	101245	0	6.5
22	26/10/2020	03.34.45	2020/10/26(Monday)	03:34	30.6	95	101252	0	6.5
23	26/10/2020	03.49.58	2020/10/26(Monday)	03:49	30.5	95	101231	0	7.5
24	26/10/2020	04.05.11	2020/10/26(Monday)	4:4	30.6	95	101233	0	8.08
25	26/10/2020	04.20.25	2020/10/26(Monday)	04:19	30.5	95	101227	0	7.5
26	26/10/2020	04.35.38	2020/10/26(Monday)	04:35	30.4	95	101237	0	6.91

- Then IoT device is sending sensor data to google excel sheet in every 15min.
- This Google Excel Sheet can be remotely access by anywhere using the link.
- Link -

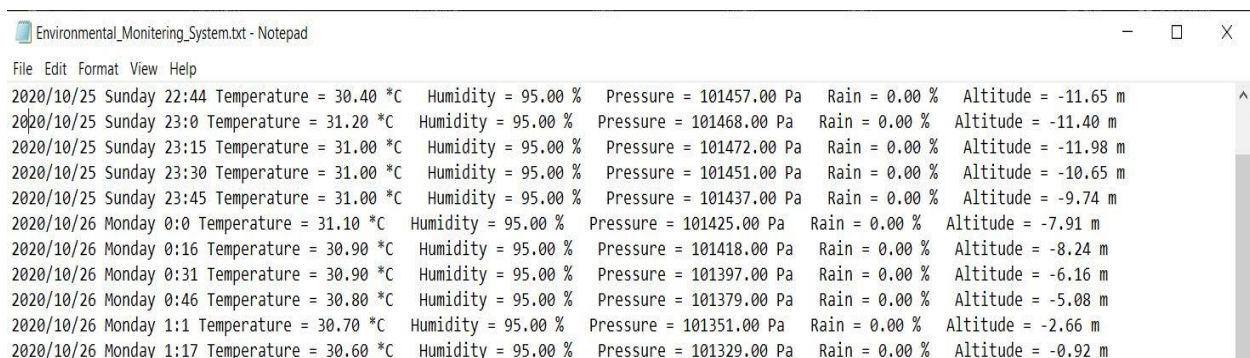
<https://docs.google.com/spreadsheets/d/1WZCQhjvzktvAOUlzxoFPSAMuRfAYjuLEpblb20CSIDA/edit#gid=0>

c) Sending sensor data to local web server using CAP protocol.



- Latest values of sensor data are display in a locally available web server
- For that need to connect to the Node MCU Wi-Fi using IP 192.168.8.1(dynamic IP)
- So, entering Ip 192.168.4.1 also redirected to this web page.
- For this NodeMCU sending data using Common Alerting Protocol (CAP) to this locally available server.
- Then the server read the data from CAP body and display in this page.

d) Sensor data also securely save in the SD card as a backup



Algorithm used for device and server (Pseudo code)

Device

function setup():

- connect to NodeMCU Wi-Fi module
- configuration your home Wi-Fi using credintails
- initialize DHT11, Rainfall and BMP180 sensors

function loop():

- time1 =millis()
- time2 = millis()
- while((time2-time1) < 900000): // this for loop collects data for 15 minutes.
 - time2 = millis();
 - get sensor readings and update the 5 lists
 - delay 2000ms
- calculate average of each parameter and save them to 5 different variables
- calculate standard deviation of each parameter
- sending calculate data to Google Excel Sheet
- format the sensor read data to CAP format
- send CAP formatted data to Local Web Server
- save the sensor collected data into SD card

Server backend

function connection():

- connect to local Wi-Fi

function getData():

- try:
 - fetch all data in the Parameters
 - render fetched data to the Page
- except any error:
 - print the error

Google excel sheet

```
function doGet(e):
  if(e.parameter == 'undefined'):
    print('No Params')
  else:
    rawData = [];
    get the timestamp of current time
    for( param in e.parameter):          // fetch all data in the Parameters
      value <= stripQuotes(e.parameter[param])
      switch case:
        rawData[column] <= added that value
        // value added to the right column using switch case
    write the values to the new row in Excel Sheet
function stripQuotes(value):
  value.replace(/^[“”][“”]$/g, "")
```

Used libraries for the project

- ESP8266WiFi.h – Connect with router
- ESP8266WebServer.h – Create a web server
- DNSServer.h – Create a domain name system
- WifiManager.h – Enter SSID & password of Wi-Fi without hard coding a credentials
- WiFiClientSecure.h – Keep connection
- Dht11.h – For DHT11 module library
- SPI.h – Start SPI communication with SD card
- SD.h – SD card library
- RTCLib.h – Real time clock library
- Wire.h – To make i2c communication
- Adafruit_BMP085.h – BMP085 barometric sensor library

Full Source code

- GitHub link for full source code and libraries–

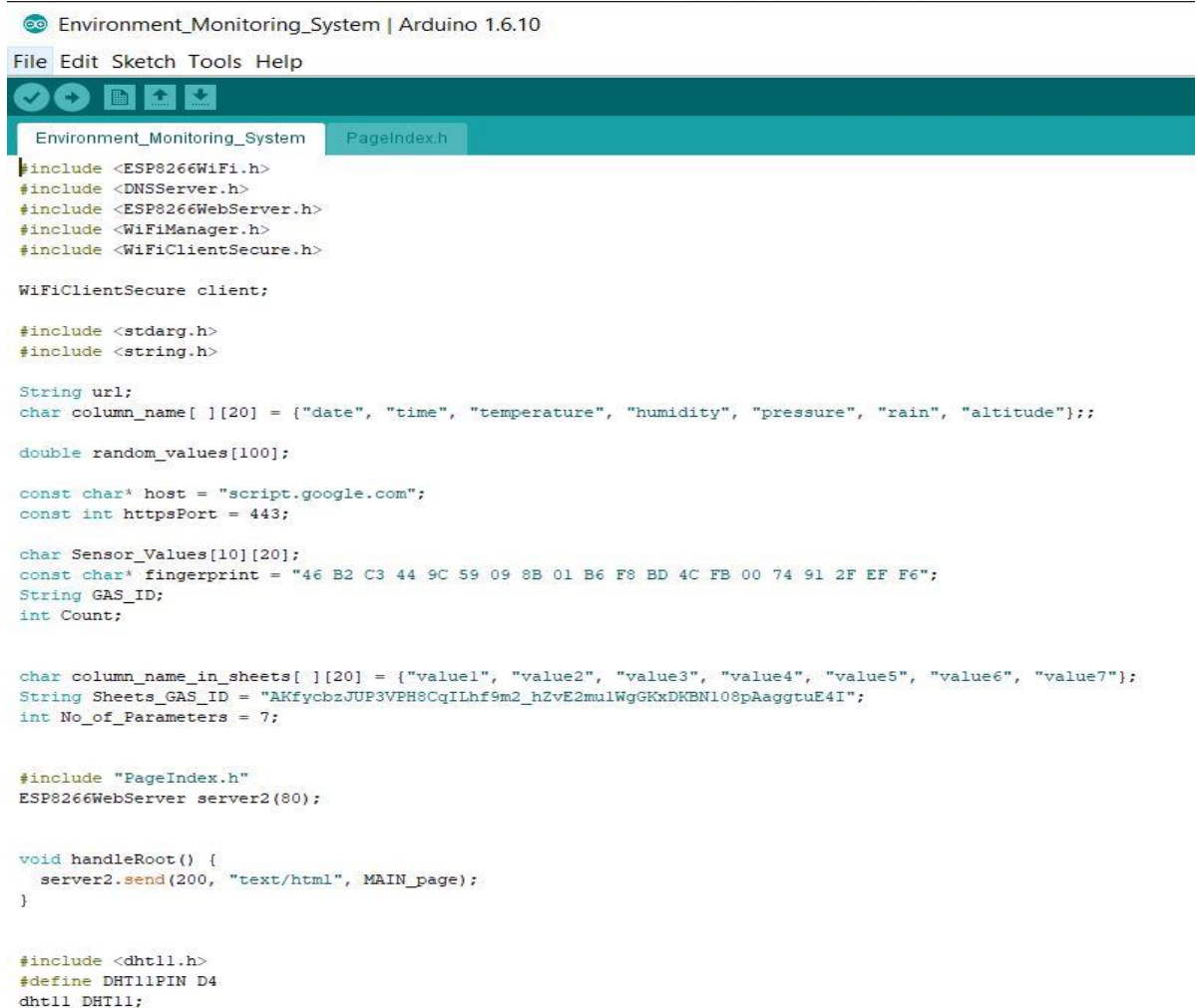
https://github.com/savinduekanayake/Environment_Monitoring_System

- Google sheet which upload sensor data in 15min in real-time –

<https://docs.google.com/spreadsheets/d/1WZCQhjvzktvAOUIzxoFPSAMuRfAYjuLEpblb20CSI DA/edit#gid=0>

❖ Node MCU updated code

1. Environment_Monitoring_System.ino



```
Environment_Monitoring_System | Arduino 1.6.10
File Edit Sketch Tools Help
Environment_Monitoring_System PageIndex.h
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>
#include <WiFiClientSecure.h>

WiFiClientSecure client;

#include <stdarg.h>
#include <string.h>

String url;
char column_name[ ][20] = {"date", "time", "temperature", "humidity", "pressure", "rain", "altitude"};

double random_values[100];

const char* host = "script.google.com";
const int httpsPort = 443;

char Sensor_Values[10][20];
const char* fingerprint = "46 B2 C3 44 9C 59 09 8B 01 B6 F8 BD 4C FB 00 74 91 2F EF F6";
String GAS_ID;
int Count;

char column_name_in_sheets[ ][20] = {"value1", "value2", "value3", "value4", "value5", "value6", "value7"};
String Sheets_GAS_ID = "AKfyCbzJUP3VPH8CqILhf9m2_hZvE2mulWgGKxDKBN108pAaggtuE4I";
int No_of_Parameters = 7;

#include "PageIndex.h"
ESP8266WebServer server2(80);

void handleRoot() {
  server2.send(200, "text/html", MAIN_page);
}

#include <dht11.h>
#define DHT11PIN D4
dht11 DHT11;
```

```

#include <Wire.h>
#include <Adafruit_BMP085.h>
Adafruit_BMP085 bmp;

float temperature;
float humidity;
float pressure;
float rain;
float altitude;

int chk;

float temperaturearray[1000];
float humidityarray[1000];
float pressurearray[1000];
float rainarray[1000];
float altitudearray[1000];

int counter1 = 1;

#include <SPI.h>
#include <SD.h>

File myFile;

const int chipSelect = D3;

double counter = 0;

// Date and time functions using a DS1307 RTC connected via I2C and Wire lib
#include "RTClib.h"

RTC_DS1307 rtc;

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

```

```

int currentMonth;
int currentDay;
int currentHour;
int currentMinutes;
String date;

String currentDate1;
String currentTime1;

long time1;
long time2;
void setup()
{
    Serial.begin(115200);

    pinMode(A0, INPUT); // Rain sensor pin

    while (!bmp.begin())
    {
        Serial.println("Not connected with BMP180/BMP085 sensor, check connections ");
    }

    while (!SD.begin(D3)) {
        Serial.println("SD card not found!");
    }

    if (!rtc.begin()) {
        Serial.println("Couldn't find RTC");
        Serial.flush();
        abort();
    }

    /*uncomment for reset the time*/

    if (!rtc.isrunning()) {
        Serial.println("RTC is NOT running, let's set the time!");
        rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    }
}

```



```

while (!Serial);

WIFI_Connect();

server2.on("/", handleRoot);
server2.on("/readTemperature", handleTemperature);
server2.on("/readHumidity", handleHumidity);
server2.on("/readPressure", handlePressure);
server2.on("/readRain", handleRain);
server2.on("/readAltitude", handleAltitude);

server2.begin();

Google_Sheets_Init(column_name_in_sheets, Sheets_GAS_ID, No_of_Parameters );

}

void loop()
{
    time1 = millis();
    time2 = millis();
    while ((time2 - time1) < 900000) {
        time2 = millis();
        temperature = bmp.readTemperature();

        chk = DHT11.read(DHT11PIN);
        humidity = DHT11.humidity;

        pressure = bmp.readPressure();

        rain = (100 - map(analogRead(A0), 0, 1024, 0, 100));

        altitude = bmp.readAltitude();

        Serial.print("Temperature = ");
        Serial.print(temperature);
        Serial.print(" *C  ");

        Serial.print("Humidity = ");

```

```

    Serial.print("Temperature = ");
    Serial.print(temperature);
    Serial.print(" *C  ");

    Serial.print("Humidity = ");
    Serial.print(humidity);
    Serial.print(" %  ");

    Serial.print("Pressure = ");
    Serial.print(pressure);
    Serial.print(" Pa  ");

    Serial.print("Rain = ");
    Serial.print(rain);
    Serial.print(" %  ");

    Serial.print("Altitude = ");
    Serial.print(altitude);
    Serial.println(" m  ");

    for (int x = 0; x < 50; x++) {
        handleTemperature();
        handleHumidity();
        handlePressure();
        handleRain();
        handleAltitude();
        server2.handleClient();
        delay(100);
    }
    delay(1);
    counter++;
}

getTime ();
myFile = SD.open("Environmental_Monitoring_System.txt", FILE_WRITE);
if (myFile) {
    Serial.println("Writing to SD card");
    myFile.println(String(currentYear) + "/" + String(currentMonth) + "/" + String(currentDay) + " " + date + " " + String(currentHour) + ":" + String(currentMinutes) + " " + "Temperature = " + String(temperature) + " *C  " + "Humidity = " + String(humidity) + " %  " +
    myFile.close();
}
}

```



```

if (WiFi.status() != WL_CONNECTED)
{
    temperaturearray[counter1];
    humidityarray[counter1];
    pressurearray[counter1];
    rainarray[counter1];
    altitudearray[counter1];
    Serial.print("no wifi");
    counter1++;
}
else {
    if (counter1 > 1) {
        for (int x = 1; x <= counter1; x++) {
            Data_to_Sheets(No_of_Parameters, 0, 0, temperaturearray[x], humidityarray[x], pressurearray[x], rainarray[x], altitudearray[x]);
            delay(500);
        }
        Data_to_Sheets(No_of_Parameters, 0, 0, temperature, humidity, pressure, rain, altitude);
    }
    counter1 = 1;
}
}
}

```

```

void handleTemperature() {
    String Temperature_Value = String(temperature);

    server2.send(200, "text/plain", Temperature_Value);
}

```

```

void handleHumidity() {
    String Humidity_Value = String(humidity);

```

```

<
Environment_Monitoring_System $ PageIndex.h
void handleTemperature() {
    String Temperature_Value = String(temperature);

    server2.send(200, "text/plain", Temperature_Value);
}

void handleHumidity() {
    String Humidity_Value = String(humidity);

    server2.send(200, "text/plain", Humidity_Value);
}

void handlePressure() {
    String Pressure_Value = String(pressure);

    server2.send(200, "text/plain", Pressure_Value);
}

void handleRain() {
    String Rain_Value = String(rain);

    server2.send(200, "text/plain", Rain_Value);
}

void handleAltitude() {
    String Altitude_Value = String(altitude);

    server2.send(200, "text/plain", Altitude_Value);
}

```

```

Environment_Monitoring_System$ PageIndex.h
void Google_Sheets_Init(char test[ ][20], String sheets_gas_id, int param_size)
{
    GAS_ID = sheets_gas_id;
    Count = param_size;

    for (int i = 0; i < Count; i++)
    {
        for (int j = 0; j < 20; j++)
        {
            column_name[i][j] = test[i][j];
        }
    }

    for (int i = 0; i < Count; i++)
    {
        Serial.print("column_name= ");
        Serial.println(column_name[i]);
    }
}

void Data_to_Sheets(int num, ...)
{
    va_list lst;
    va_start(lst, num);

    for (int i = 0; i < num; i++)
    {
        random_values[i] = va_arg(lst, double);
    }
    va_end(lst);

    float_to_string();
    Send_Data();
}

```

```

Environment_Monitoring_System$ PageIndex.h

void float_to_string()
{
    currentDate = String(currentYear) + "/" + String(currentMonth) + "/" + String(currentDay) + "(" + date + ")";
    currentTime = String(currentHour) + ":" + String(currentMinutes);
    currentDate.toCharArray(Sensor_Values[0], 20);
    currentTime.toCharArray(Sensor_Values[1], 20);
    for (int j = 2; j < Count; j++)
    {
        sprintf(Sensor_Values[j], "%.02f", random_values[j - 1]);

        Serial.print("Sensor Values : ");
        Serial.println(Sensor_Values[j]);
    }
}

void Send_Data()
{
    sheets_initialization();

    String url = "/macros/s/" + GAS_ID + "/exec?";
    int i = 0;
    while (i != Count)
    {
        if (i == 0)
        {
            url = url + column_name[i] + "=" + Sensor_Values[i];
            i++;
        }
        if (i == Count)
            break;
        url = url + "&" + column_name[i] + "=" + Sensor_Values[i];
        i++;
    }

    Serial.print("requesting URL: ");
    Serial.println(url);

    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
    <

```

```
        i++;
    }

    Serial.print("requesting URL: ");
    Serial.println(url);

    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
        "Host: " + host + "\r\n" +
        "User-Agent: BuildFailureDetectorESP8266\r\n" +
        "Connection: close\r\n\r\n");

    Serial.println("request sent");

    while (client.connected())
    {
        String line = client.readStringUntil('\n');

        if (line == "\r") {
            Serial.println("headers received");
            break;
        }
    }

    String line = client.readStringUntil('\n');

    if (line.startsWith("{\"state\":\"success\"}") {
        Serial.println("esp8266/Arduino CI successful!");
    }

    else {
        Serial.println("esp8266/Arduino CI has failed");
    }

    Serial.println("reply was:");
    Serial.println("=====");
    Serial.println(line);
    Serial.println("=====");
    Serial.println("closing connection");
}
```

```
void sheets_initialization()
{
    client.setInsecure();
    Serial.print("connecting to ");
    Serial.println(host);

    if (!client.connect(host, httpsPort)) {
        Serial.println("connection failed");
        return;
    }

    if (client.verify(fingerprint, host)) {
        Serial.println("certificate matches");
    }

    else {
        Serial.println("certificate doesn't match");
    }
}

void WIFI_Connect()
{
    WiFiManager wifiManager;
    Serial.println("Connecting.....");
    wifiManager.autoConnect("ENVIRONMENTAL MONITORING SYSTEM");
    Serial.println("connected");

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void getTime () {
    DateTime now = rtc.now();

    currentYear = now.year();
    currentMonth = now.month();
    currentDay = now.day();
    currentHour = now.hour();
    currentMinutes = now.minute();
    date = daysOfTheWeek[now.dayOfTheWeek()];

    Serial.print(currentYear);
    Serial.print('/');
    Serial.print(currentMonth);
    Serial.print('/');
    Serial.print(currentDay);
    Serial.print(" ");
    Serial.print(date);
    Serial.print(" ");
    Serial.print(currentHour);
    Serial.print(':');
    Serial.println(currentMinutes);
}
```

2. PageIndex.h

```
Environment_Monitoring_System $ PageIndex.h
const char MAIN_page[] PROGMEM = R"====={
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" rel="stylesheet">
  <style>
    html {
      font-family: Arial;
      display: inline-block;
      margin: 0px auto;
      text-align: center;
    }
    h1 { font-size: 2.0rem; }
    p { font-size: 2.0rem; }
    .units { font-size: 1.2rem; }
    .dht-labels{
      font-size: 1.5rem;
      vertical-align:middle;
      padding-bottom: 15px;
    }
  </style>
</head>
<body>
  <h1>Environment Monitoring System</h1>
  <p>
    <span class="dht-labels">Temperature : </span>
    <span id="TemperatureValue">0</span>
    <sup class="units">°deg;C</sup>
  </p>
  <p>
    <span class="dht-labels">Humidity : </span>
    <span id="HumidityValue">0</span>
    <sup class="units">%</sup>
  </p>
  <p>
    <span class="dht-labels">Pressure : </span>
    <span id="PressureValue">0</span>
    <sup class="units">Pa</sup>
  </p>
  <p>
    <span class="dht-labels">Rain : </span>
    <span id="RainValue">0</span>
    <sup class="units">%</sup>
  </p>
  <p>
    <span class="dht-labels">Altitude : </span>
    <span id="AltitudeValue">0</span>
    <sup class="units">m</sup>
  </p>
  <p>
    <i class="far fa-clock" style="font-size:1.0rem;color:#e3a8c7;"></i>
    <span style="font-size:1.0rem;">Time </span>
    <span id="time" style="font-size:1.0rem;"></span>
  </p>
  <p>
    <i class="far fa-calendar-alt" style="font-size:1.0rem;color:#f7dc68;"></i>
    <span style="font-size:1.0rem;">Date </span>
    <span id="date" style="font-size:1.0rem;"></span>
  </p>
</body>
<script>
  <style="font-size:1.0rem;color:red;"></i>
  <a href="https://docs.google.com/spreadsheets/d/1WZCQhjvzktvA0U1zxoFPSAMuRfAYjuLEpblb20CSIDA/edit#gid=0" target="_blank" style="font-size:1.0rem;">Google sheet</a>
  </script>
  setInterval(function() {
    // Call a function repetatively with 2 Second interval
    getTemperatureData();
    getHumidityData();
    getPressureData();
    getRainData();
    getAltitudeData();
  }, 2000);

  setInterval(function() {
    // Call a function repetatively with 1 Second interval
    Time_Date();
  }, 1000);

  function getTemperatureData() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("TemperatureValue").innerHTML =
          this.responseText;
      }
    };
    xhttp.open("GET", "readTemperature", true);
    xhttp.send();
  }
}
```

```

    }
    function getHumidityData() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("HumidityValue").innerHTML =
                    this.responseText;
            }
        };
        xhttp.open("GET", "readHumidity", true);
        xhttp.send();
    }
    function getPressureData() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("PressureValue").innerHTML =
                    this.responseText;
            }
        };
        xhttp.open("GET", "readPressure", true);
        xhttp.send();
    }
    function getRainData() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("RainValue").innerHTML =
                    this.responseText;
            }
        };
        xhttp.open("GET", "readRain", true);
        xhttp.send();
    }
    function getAltitudeData() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("AltitudeValue").innerHTML =
                    this.responseText;
            }
        };
        xhttp.open("GET", "readAltitude", true);
        xhttp.send();
    }
}

function Time_Date() {
    var t = new Date();
    document.getElementById("time").innerHTML = t.toLocaleTimeString();
    var d = new Date();
    const dayNames = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];
    const monthNames = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"];
    document.getElementById("date").innerHTML = dayNames[d.getDay()] + ", " + d.getDate() + "-" + monthNames[d.getMonth()] + "-" + d.getFullYear();
}
</script>
</body>
</html>
)=====";
```


❖ Google Sheet Script

Environment_Monitoring_System

File Edit View Run Publish Resources Help

This project is running on our new Apps Script runtime powered by Chrome V8. [More info](#) [Dismiss](#)

Select function

Code.gs

Code.gs

```

1 function doGet(e) {
2   Logger.log( JSON.stringify(e) ); // view parameters
3   var result = 'Ok'; // assume success
4   if (e.parameter ==| 'undefined') {
5     result = 'No Parameters';
6   }
7   else {
8     var sheet_id = '1WZCQhvjvktvA0U1zxoFPSAMuRfAYjuLEpb1b20CSIDA'; // Spreadsheet ID
9     var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet(); // get Active sheet
10    var newRow = sheet.getLastRow() + 1;
11    var rowData = [];
12    d=new Date();
13    rowData[0] = d; // Timestamp in column A
14    rowData[1] = d.toLocaleTimeString(); // Timestamp in column A
15
16    for (var param in e.parameter) {
17      Logger.log('In for loop, param=' + param);
18      var value = stripQuotes(e.parameter[param]);
19      Logger.log(param + ':' + e.parameter[param]);
20      switch (param) {
21        case 'value1': //Parameter 1, It has to be updated in Column in Sheets in the code, otherwise
22          rowData[2] = value; //Value in column A
23          result = 'Written on column A';
24          break;
25        case 'value2': //Parameter 2, It has to be updated in Column in Sheets in the code, otherwise
26          rowData[3] = value; //Value in column B
27          result += ' Written on column B';
28          break;
29        case 'value3': //Parameter 3, It has to be updated in Column in Sheets in the code, otherwise
30          rowData[4] = value; //Value in column C
31          result += ' Written on column C';
32          break;
33        case 'value4': //Parameter 4, It has to be updated in Column in Sheets in the code, otherwise
34          rowData[5] = value; //Value in column d
35          result += ' Written on column D';
36          break;
37        case 'value5': //Parameter 5, It has to be updated in Column in Sheets in the code, otherwise
38          rowData[6] = value; //Value in column e
39          result += ' Written on column E';
40          break;
41        case 'value6': //Parameter 6, It has to be updated in Column in Sheets in the code, otherwise
42          rowData[7] = value; //Value in column f
43          result += ' Written on column F';
44          break;
45        case 'value7': //Parameter 7, It has to be updated in Column in Sheets in the code, otherwise
46          rowData[8] = value; //Value in column g
47          result += ' Written on column G';
48          break;
49        default:
50          result = "unsupported parameter";
51      }
52    }
53    Logger.log(JSON.stringify(rowData));
54    // Write new row below
55    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
56    newRange.setValues([rowData]);
57  }
58  // Return result of operation
59  return ContentService.createTextOutput(result);
60 }
61 function stripQuotes( value ) {
62   return value.replace(/^[\'"]|[\']$/g, "");
63 }
64

```

This project is published

This project is published

References

- CAP Scheme – <https://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2.xsd>
- make a local server from NodeMCU
https://www.youtube.com/watch?v=cx4S_rcMrZc&t=571s
- Google sheet update with NodeMCU - <https://www.hackster.io/thatiotguy/sensor-data-upload-to-google-sheets-through-nodemcu-632358>