INTERNET OF THINGS

EE-325

PROJECT REPORT

PROJECT TOPIC: Internet of Things (IoT) based Weather Monitoring System.

TEAM MEMBERS:

Name: Tarun Arora
Roll No.: 2K18/EP/085

Name: Harsh Panchal
Roll No.: 2K18/EP/085

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to our professor, Prof. M.M. Tripathi for their exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by them time to time shall carry me a long way in the journey of life on which we are about to embark.

Thank you

ABSTRACT

In this report, we introduce a solution for monitoring the weather conditions at a particular place and make the information visible anywhere in the world. The technology behind this is Internet of Things (IoT), which is an advanced and efficient solution for connecting the things to the internet and to connect the entire world of things in a network. Here things might be whatever like electronic gadgets, sensors and automotive electronic equipment. The system deals with monitoring and controlling the environmental conditions like temperature, relative humidity, light intensity and CO level with sensors and send the information to Google Sheets. The data updated from the implemented system can be accessible in the internet from anywhere in the world.

CONTENTS

INTRODUCTION

Present innovations in technology mainly focus on controlling and monitoring of different activities. These are increasingly emerging to reach the human needs. Most of this technology is focused on efficient monitoring and controlling different activities. An efficient environmental monitoring system is required to monitor and assess the conditions in case of exceeding the prescribed level of parameters (e.g., noise, CO and radiation levels).

Initially the sensor devices are deployed in environment to detect the parameters (e.g., Temperature, Humidity, Pressure, LDR, noise, CO and radiation levels etc.) while the data acquisition, computation and controlling action (e.g., the variations in the noise and CO levels with respect to the specified levels). The main aim of the this report is to design and implement an efficient monitoring system through which the required parameters are monitored remotely using internet and the data gathered from the sensors are stored in the cloud using Google Sheets.

The solution also provides an intelligent remote monitoring for a particular area of interest. In this report we also present a trending results of collected or sensed data with respect to the normal or specified ranges of particular parameters. The embedded system is an integration of sensor devices, wireless communication which enables the user to remotely access the various parameters and store the data in cloud.

SYSTEM ARCHITECTURE

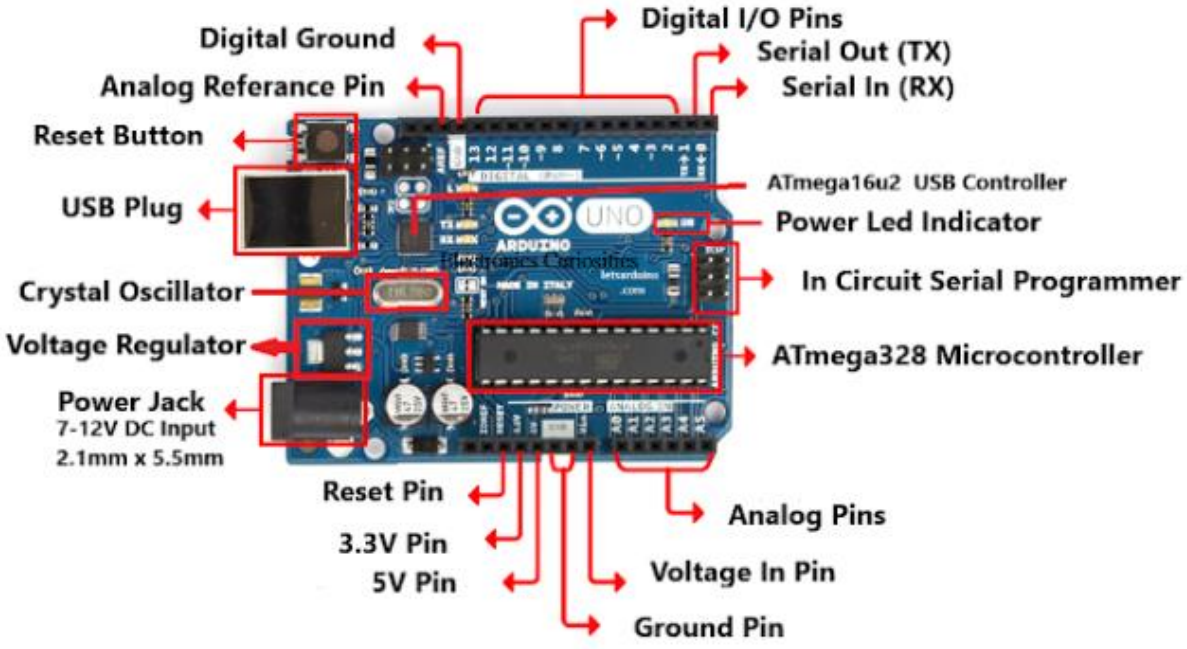
The implemented system consists of a microcontroller (ATmega328) as a main processing unit for the entire system and all the sensor and devices can be connected with the microcontroller. The sensors can be operated by the microcontroller to retrieve the data from them and it processes the analysis with the sensor data and updates it to the internet through Wi-Fi module connected to it.

COMPONENTS USED IN THE PROJECT

The following components are used:

1.) Arduino Uno
2.) Wifi Module (ESP8266)
3.) Humidity and Temperature sensor (DHT11)
4.) LDR/Solar Cell, 5V
5.) Breadboard
6.) Jumper Wires
7.) CO sensor (MQ7 and MQ9)

Arduino UNO :



Arduino is an open-source physical computing platform based on a simple micro-controller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs.

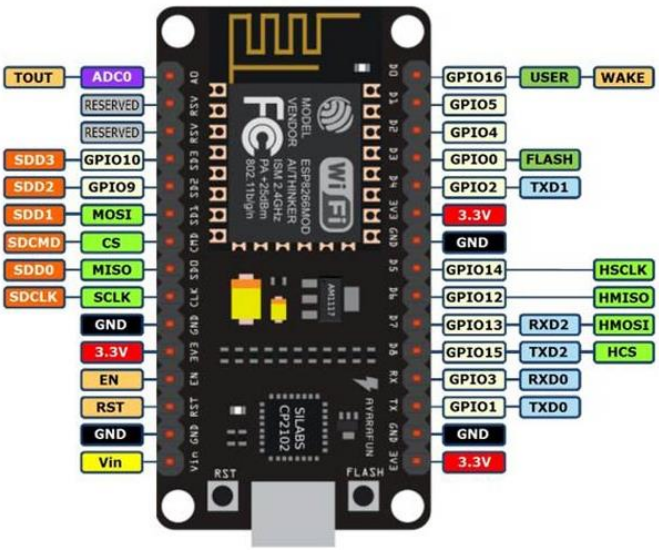
The open-source IDE can be downloaded for free. The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

Pin Details for Arduino UNO are as follows:

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.

AREF	AREF	To provide reference voltage for input voltage.

Wi-Fi Module (ESP8266):



Here we used ESP8266 Wi-Fi module which is having TCP/IP protocol stack integrated on chip. So that it can provide any microcontroller to get connected with Wi-Fi network. ESP8266 is a pre programmed SOC and any microcontroller has to communicate with it through UART interface. It works with a supply voltage of 3.3v. The pin description for the ESP8266 module is as follows:

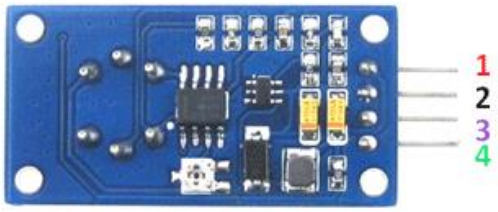
Pin Category	**Name**	**Description**
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply

Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

CO Sensor :

MQ7:

MQ9:



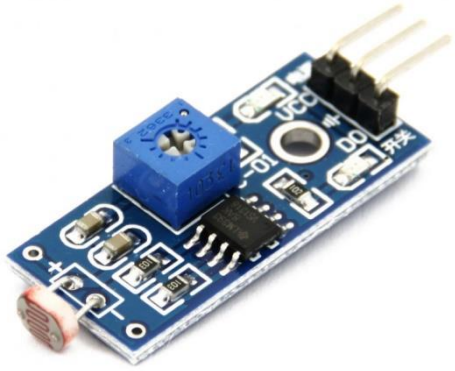
Pin No.	Pin Name
1	Vcc(+5V)
2	Ground
3	Digital Out
4	Analog out

A **gas sensor** is a device which detects the presence or concentration of gases in the atmosphere. Based on the concentration of the gas the sensor produces a corresponding potential difference by changing the resistance of the material inside the sensor, which can be measured as output voltage.

Based on this voltage value the type and concentration of the gas can be estimated.

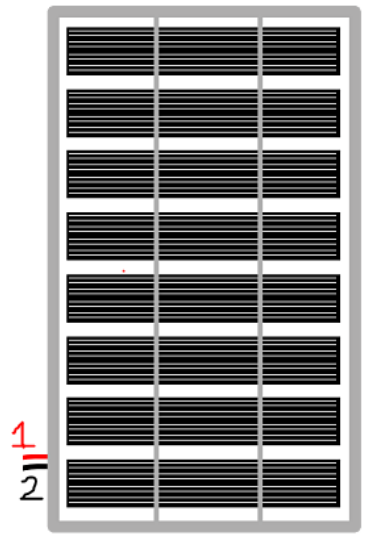
The MQ-7 can detect CO-gas concentrations anywhere from 20 to 2000ppm.This sensor has a high sensitivity and fast response time. MQ-9 sensor in addition to detection of Carbon Monoxide can also detect other flammable gases.

LDR Light-Dependent Resistor :



LDR sensor module is used to detect the intensity of light. It is associated with both analog output pin and digital output pin labelled as AO and DO respectively on the board. When there is light, the resistance of LDR will become low according to the intensity of light. The greater the intensity of light, the lower the resistance of LDR. The sensor has a potentiometer knob that can be adjusted to change the sensitivity of LDR towards light.

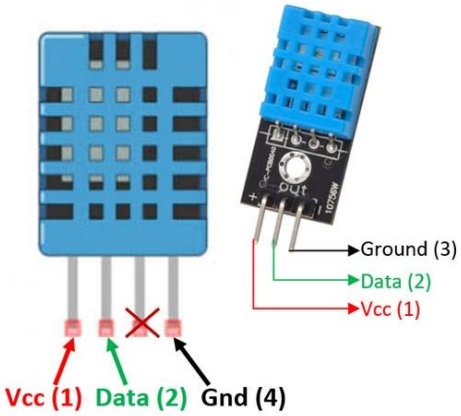
Solar Cell, 5V:



A solar cell, or photovoltaic cell, is an electrical device that converts the energy of light directly into electricity by the photovoltaic effect, which is a physical and chemical phenomenon.

In the above diagram, Pin 1 is connected to a voltage source while Pin 2 is connected to ground.

Temperature and Humidity Sensor :



DHT11 is a Humidity and Temperature Sensor, which generates calibrated digital output. DHT11 can be interface with any microcontroller like Arduino, Raspberry Pi, etc. and get instantaneous results. DHT11 is a low cost humidity and temperature sensor which provides high reliability and long term stability.

This module makes is easy to connect the DHT11 sensor to an Arduino or microcontroller as includes the pull up resistor required to use the sensor. Only three connections are required to be made to use the sensor - Vcc, Gnd and Output.

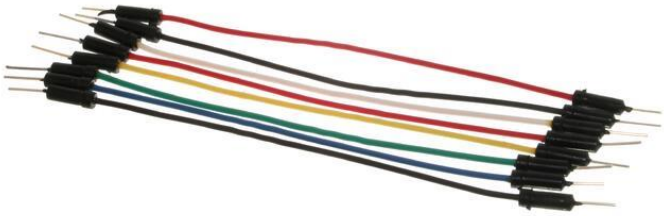
Pin Description for DHT11 is as follows:

No:	Pin Name	Description
For DHT11 Sensor		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit
For DHT11 Sensor module		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

BREADBOARD :

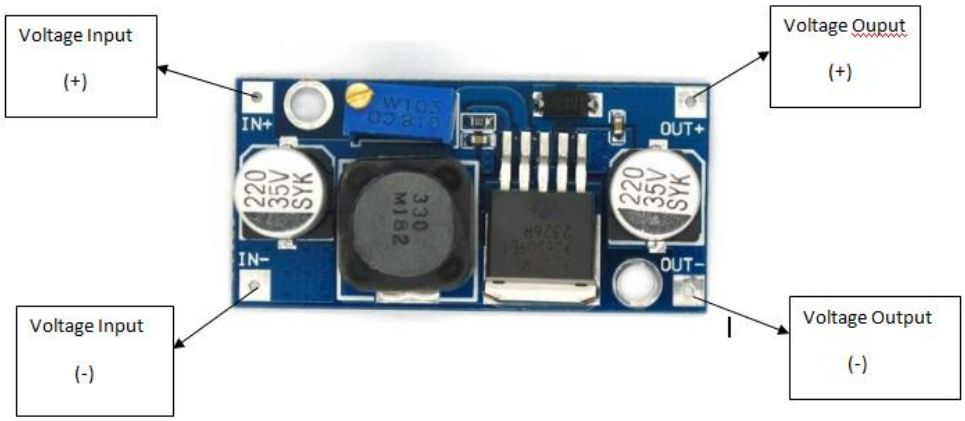
A breadboard is used to make up temporary circuits for testing or to try out an idea. No soldering is required so it is easy to change connections and replace components. Parts are not damaged and can be re-used afterwards.

Jumper Wires:



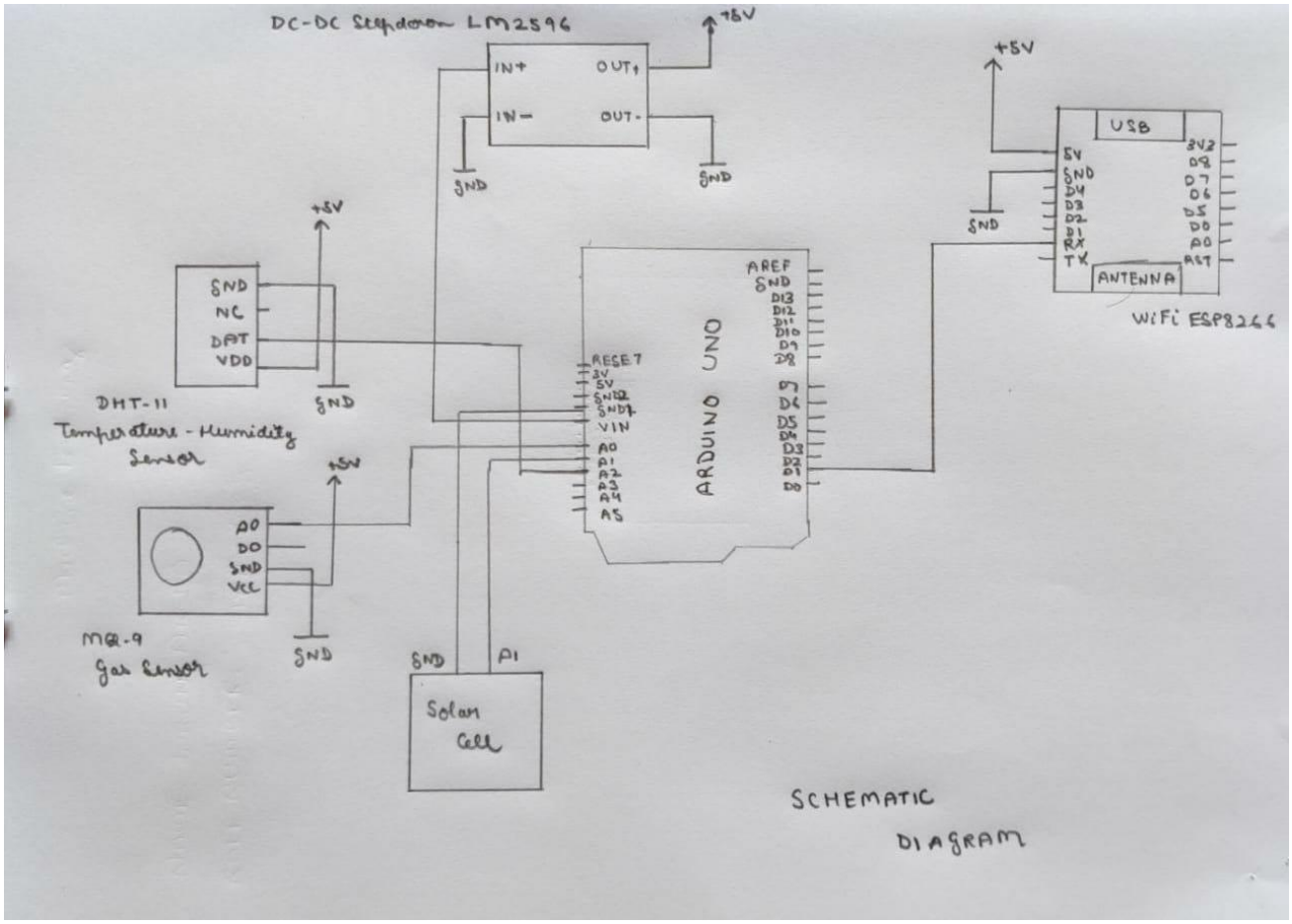
Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed.

DC step down converter, LM2596:



The **LM2596** is a commonly used popular **step-down switching regulator IC**. The adjustable version can take in input voltage from 4.5V to 40V and convert it to variable voltage sourcing upto of 3A of continues current. Because of its high current capability is commonly used in power modules to power/control heavy loads.

Schematic Diagram



Software Used and Programming the Arduino UNO and Wifi module ESP8266

Arduino IDE:

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click.

Arduino Programs are written in C/C++. The two important functions that are an essential part of the code are:

- setup() – this function run once at the beginning of the program and can be used to intitialise certain variables and settings.
- loop() – this function is called repeatedly till the board is turned off.

Code to program Arduino UNO for the project

Functions used in the code given below:

1. Gas_read(): This function reads the MQ-9 sensor's data and take the average of 100 values reported and print it using Serial.print() command which can be read further to transfer the data to the wifi module, which can further push the data to Google sheets.

2. read_Solar(): This function calculates the voltage output from the solar cell and prints it similarly to that in Gas_read().

3. temp_read(): This function reads the value from DHT sensor and store the value of temperature and humidity in two different variable and then print it using Serial.print() command which can be read further to transfer the data to the wifi module, which can further push the data to Google sheets.

```
int gas = A0;
int sensorValue;
```

```
int analog_value;
float input_voltage = 0.0;

#include "DHT.h"
#define DHTPIN A2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
float h,t;

void Gas_read()
{
	//Average
	for(int x = 0 ; x < 100 ; x++)
	{
		sensorValue = sensorValue + analogRead(gas);
	}
	sensorValue = sensorValue/100.0;
	Serial.print("<C")	;Serial.print(sensorValue);Serial.print(">");
}

void read_solar()
{
	analog_value = analogRead(A1);
	input_voltage = (analog_value * 5.0) / 1024.0;


	if (input_voltage < 0.1)
	{
		input_voltage=0.0;
	}
	Serial.print("<D");Serial.print(input_voltage);Serial.print(">");
}


void temp_read()
{
	h = dht.readHumidity();
	// Read temperature as Celsius (the default)
	t = dht.readTemperature();
	Serial.print("<A");Serial.print(t);Serial.print(">");
	Serial.print("<B");Serial.print(h);Serial.print(">");
}
```

```
void setup()
{
        Serial.begin(9600);
        dht.begin();
}

void loop()
{
        temp_read();
        Gas_read();
         read_solar();
         Serial.println("<E>");
        delay(2000);
}
```

Code to program Wifi-Module, ESP-8266 for the project

Using the Wifi-Module, ESP-8266, we will program it in a way to connect it to a wifi-network, for the project we have connected it to a mobile's hotspot named 'Tarun' and stored in a variable named 'ssid' and password is stored in a variable named 'password'.

After successfully connecting to the Wifi Network, it reads from the serial using Serial.read() command and then check for the letter in the serial line. For the project we have set:

- **'A' -> Temperature**
- **'B' -> Humidity**
- **'C' -> reading from gas sensor**
- **'D' -> Voltage from Solar cell**

The code for the ESP-8266 is as follows:

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

const char* ssid = "Tarun";    //your wifi network
const char* password = "123456789";     //your wifi pasword
```

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

WiFiClientSecure client;

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 = "AKfycbxUXAdux83G4-2aaSfJnjPu3n_pP5K78dPls5_a_sQCI_IMkMxs";
//spreadsheet id


float temp=25.4, humi=52.0, volt1=2.3;
int gas1=100;
char cmd_arr1[20];
int cmd_count1;

void setup()
{
          Serial.begin(9600);
          Serial.println();
          Serial.print("connecting to ");
          Serial.println(ssid);
          WiFi.mode(WIFI_STA);
          WiFi.begin(ssid, password);
           while (WiFi.status() != WL_CONNECTED) {
                    delay(500);
                    Serial.print(".");
             }
          Serial.println("");
          Serial.println("WiFi connected");
           Serial.println("IP address: ");
          Serial.println(WiFi.localIP());

          client.setInsecure();
}

void serial_get_command()
{
          char inchar=0;
          if(Serial.available() > 0)
          {
                    inchar = Serial.read();
                      if(inchar == '<')
```

```
                {
                        cmd_count1=0;
                        while(inchar != '>' && cmd_count1<20)
                         {
                                if(Serial.available() > 0)
                                {
                                        inchar = Serial.read();
                                        cmd_arr1[cmd_count1++] = inchar;
                                         cmd_arr1[cmd_count1] = '\0';
                                }
                           }
                        if(inchar == '>')
                        {
                                 cmd_count1--;
                                cmd_arr1[cmd_count1] = '\0';
                        if(cmd_arr1[0]=='A'){cmd_arr1[0]='0';temp = atof(cmd_arr1);}
                        else if(cmd_arr1[0]=='B'){cmd_arr1[0]='0';humi = atof(cmd_arr1);}
                        else if(cmd_arr1[0]=='C'){cmd_arr1[0]='0';gas1 = atof(cmd_arr1);}
                         else if(cmd_arr1[0]=='D'){cmd_arr1[0]='0';volt1 = atof(cmd_arr1);}
                         else if(cmd_arr1[0]=='E'){cmd_arr1[0]='0';sendData(temp, humi, gas1,
volt1);}
                        }
                }
        }
}


void loop()
{
        serial_get_command();

}

void sendData(int tem, int hum , int gas, int volt)
{
        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");
  }
  String string_temperature =  String(tem, DEC);
String string_humidity =  String(hum, DEC);
String string_Gas =  String(gas, DEC);
String string_Volt =  String(volt, DEC);
  String url = "/macros/s/" + GAS_ID + "/exec?temperature=" + string_temperature +
"&humidity=" + string_humidity + "&gas=" + string_Gas + "&volt=" + string_Volt;
//Serial.print
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 successfull!");
}
else
{
        Serial.println("esp8266/Arduino CI has failed");
 }
Serial.println("reply was:");
Serial.println("==========");
Serial.println(line);
Serial.println("==========");
Serial.println("closing connection");
}
```

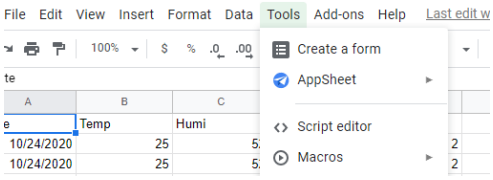
Google Sheets as a Cloud Service

For the project we have used Google sheets as the place to store all our sensors' reading. The reading is stored along with the date of the reading as the first column; followed by Temperature, Humidity, Gas sensor reading, Voltage from solar cell.

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Temp	Humi	Gas	Solar Volt						
2	10/24/2020	25	52	100	2						
3	10/24/2020	25	52	100	2						
4	10/24/2020	33	33	35	2						
5	10/24/2020	33	33	36	2						
6	10/24/2020	33	33	36	2						
7	10/24/2020	33	33	36	2						
8	10/24/2020	33	33	36	2						
9	10/24/2020	33	34	35	2						
10	10/24/2020	33	33	35	2						
11	10/24/2020	33	33	36	2						
12	10/24/2020	33	34	37	2						
13	10/24/2020	33	34	36	2						
14	10/24/2020	33	34	37	2						
15	10/24/2020	33	36	35	2						
16	10/24/2020	33	36	35	2						
17	10/24/2020	34	36	35	2						
18	10/24/2020	34	34	35	2						
19	10/24/2020	34	34	35	2						
20	10/24/2020	34	34	36	2						

Pushing the sensors' data to Google Sheets

To push the sensors' data onto the google sheets, we will need to use 'Script Editor' which can be accessed by clicking on 'Tools' -> 'Script Editor'.



Google Apps Script is a rapid application development platform that makes it fast and easy to create business applications that integrate with Google Workspace. One can write code in modern JavaScript and have access to built-in libraries for Google Workspace applications.

The following is the code that we have written in the script editor and deployed it as a web application:

Code to push data onto the google sheets:

```
function doGet(e) {
  Logger.log( JSON.stringify(e) );
  var result = 'Ok';
  if (e.parameter == 'undefined') {
```

```
    result = 'No Parameters';
  }
  else {
    var sheet_id = '1GACfW8z2LpSHv83C4Ro3y5KP8eHbZW4MHJLGQGl2Tb4';   //
Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet();
    var newRow = sheet.getLastRow() + 1;

    var rowData = [];
    rowData[0] = new Date();
    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);
      switch (param) {
        case 'temperature':
          rowData[1] = value;
          result = 'Written on column B';
          break;
        case 'humidity':
          rowData[2] = value;
          result += ' ,Written on column C';
          break;
        case 'gas':
          rowData[3] = value;
          result += ' ,Written on column D';
          break;
        case 'volt':
          rowData[4] = value;
          result += ' ,Written on column E';
          break;
        default:
          result = "unsupported parameter";
      }
    }
    Logger.log(JSON.stringify(rowData));
    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
    newRange.setValues([rowData]);
  }
  return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
  return value.replace(/^["']|['"]$/g, "");
}
```

REFERENCES

https://onlinecourses.nptel.ac.in/noc20_cs66/preview

https://www.arduino.cc/

https://components101.com/

https://www.hackster.io/thatiotguy/sensor-data-upload-to-google-sheets-through-nodemcu-632358