# User Manual

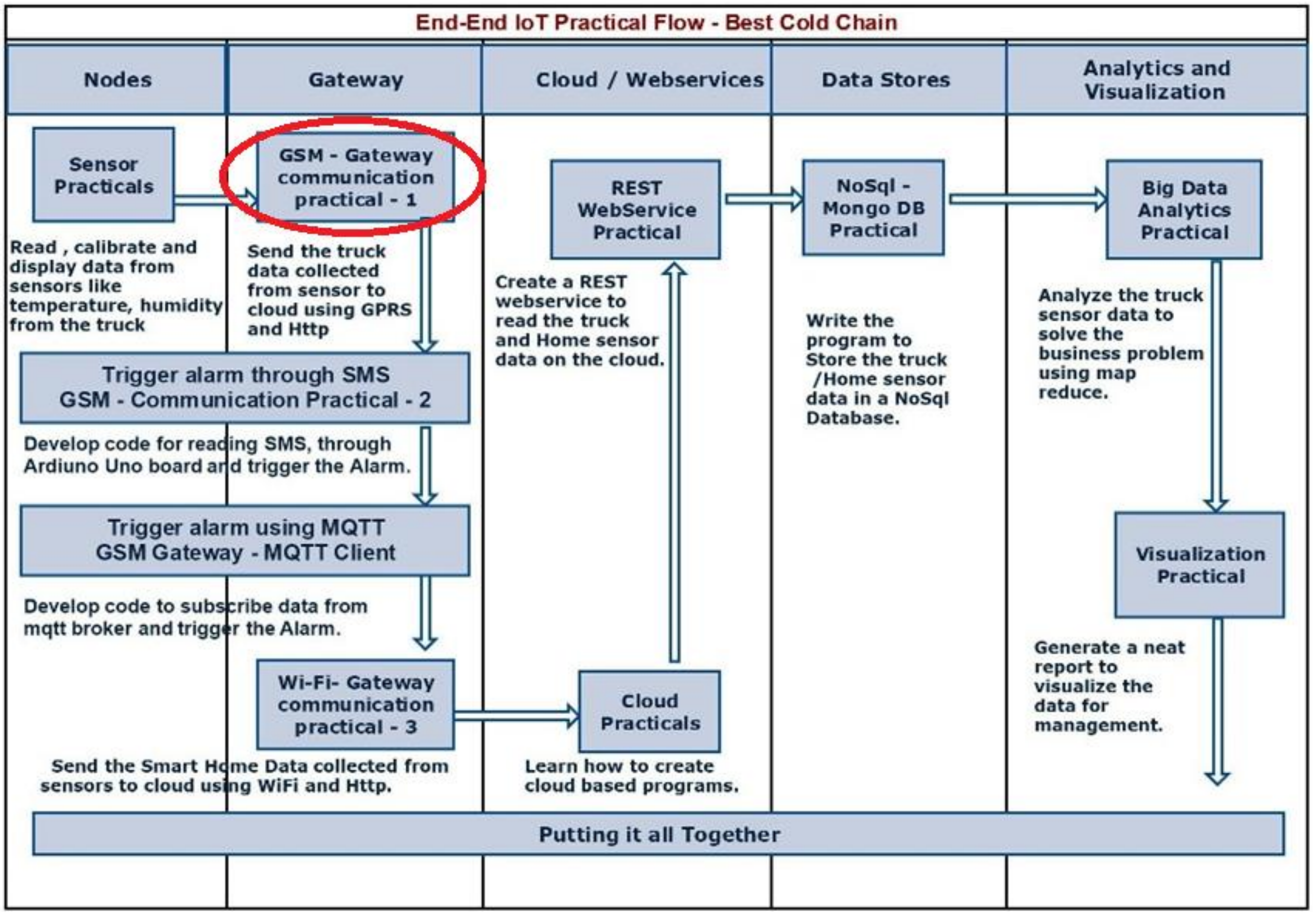
## Communication Practical – 1

## Send sensor data of Best Cold Chain to cloud using GPRS and HTTP

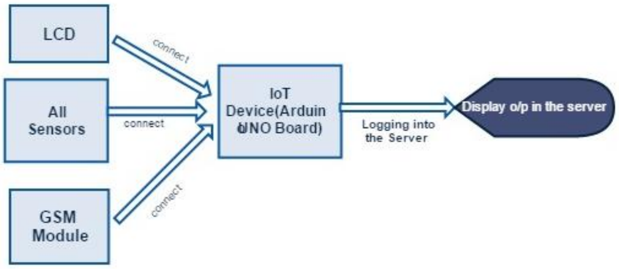
**Practical's Objective:**

To send sensor data of Best Cold Chain to cloud using GPRS and HTTP

1. **End-End IoT Flow Diagram:**

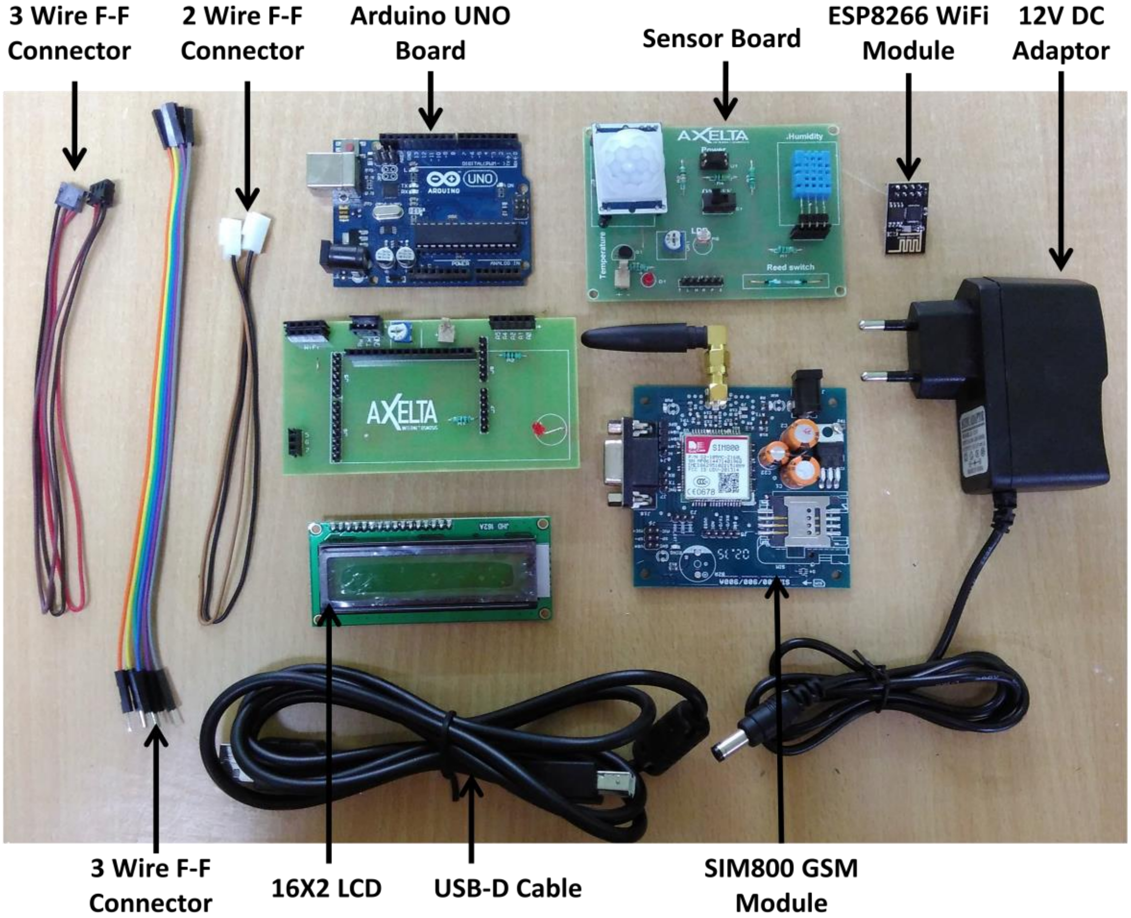

End-End IoT Practical Flow - Best Cold Chain
Nodes
Gateway
Cloud / Webservices
Data Stores
Analytics and Visualization
Sensor Practicals
Read , calibrate and display data from sensors like temperature, humidity from the truck
GSM - Gateway communication practical - 1
Send the truck data collected from sensor to cloud using GPRS and Http
Trigger alarm through SMS GSM - Communication Practical - 2
Develop code for reading SMS, through Ardiuno Uno board and trigger the Alarm.
Trigger alarm using MQTT GSM Gateway - MQTT Client
Develop code to subscribe data from mqtt broker and trigger the Alarm.
Wi-Fi- Gateway communication practical - 3
Send the Smart Home Data collected from sensors to cloud using WiFi and Http.
REST WebService Practical
Create a REST webservice to read the truck and Home sensor data on the cloud.
Cloud Practicals
Learn how to create cloud based programs.
NoSql - Mongo DB Practical
Write the program to Store the truck /Home sensor data in a NoSql Database.
Big Data Analytics Practical
Analyze the truck sensor data to solve the business problem using map reduce.
Visualization Practical
Generate a neat report to visualize the data for management.
Putting it all Together

## 2. Sensors Data Posting Using JSON Hardware Flow Diagram:



| communication Practical - 1 | | | |
|---|---|---|---|
| Step1 | Step2 | Step3 | Step4 |
| Connect All Sensor to Arduino UNO Board | Connecting GSM Module to Arduino UNO Board | 1) Now connect USB cable from Arduino UNO board to PC.<br>2) Connect Adapter to GSM Modem. | Programming to the Arduino UNO Board |

### Hardware requirements:



**Software Requirement:**

1. Arduino IDE
2. DHT-11 Library

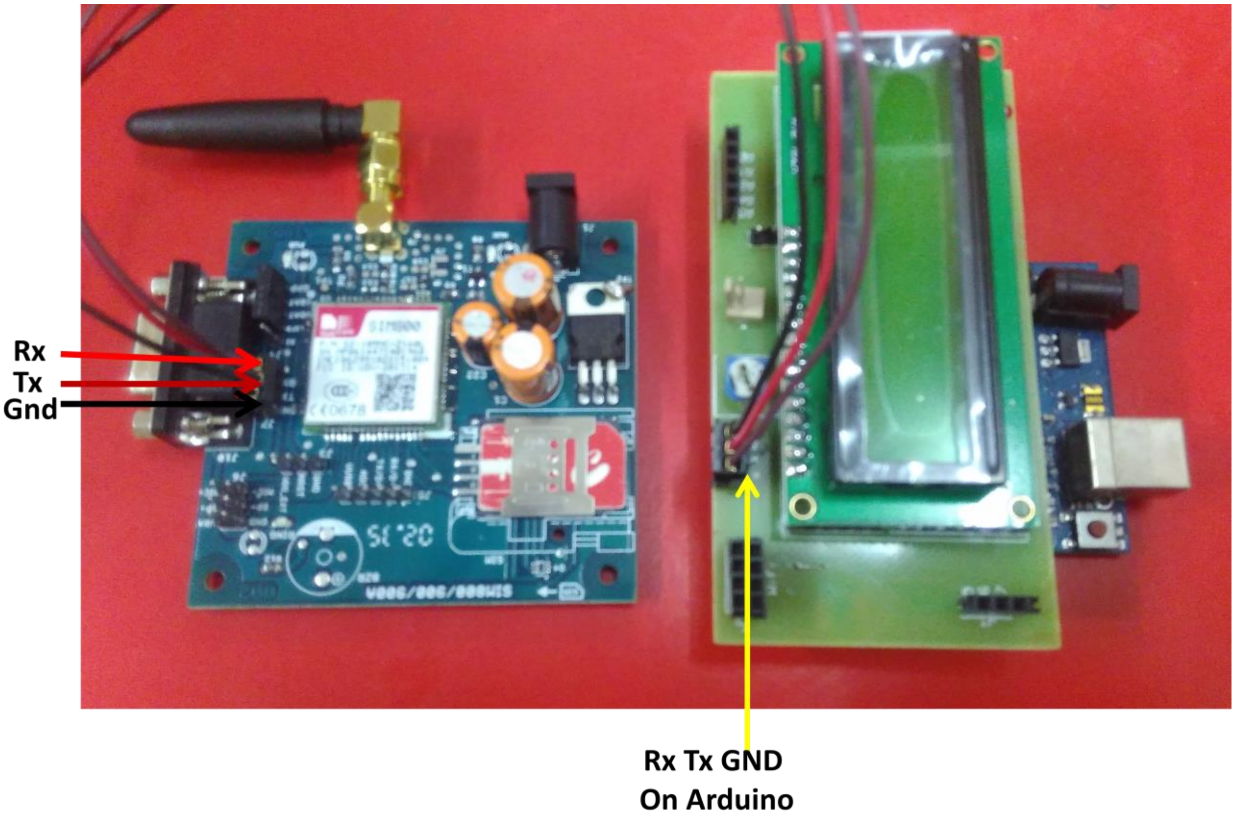
**Arduino UNO Board Connections with Sensor Board:**

Don’t change the sensor connections. They will also remain same.

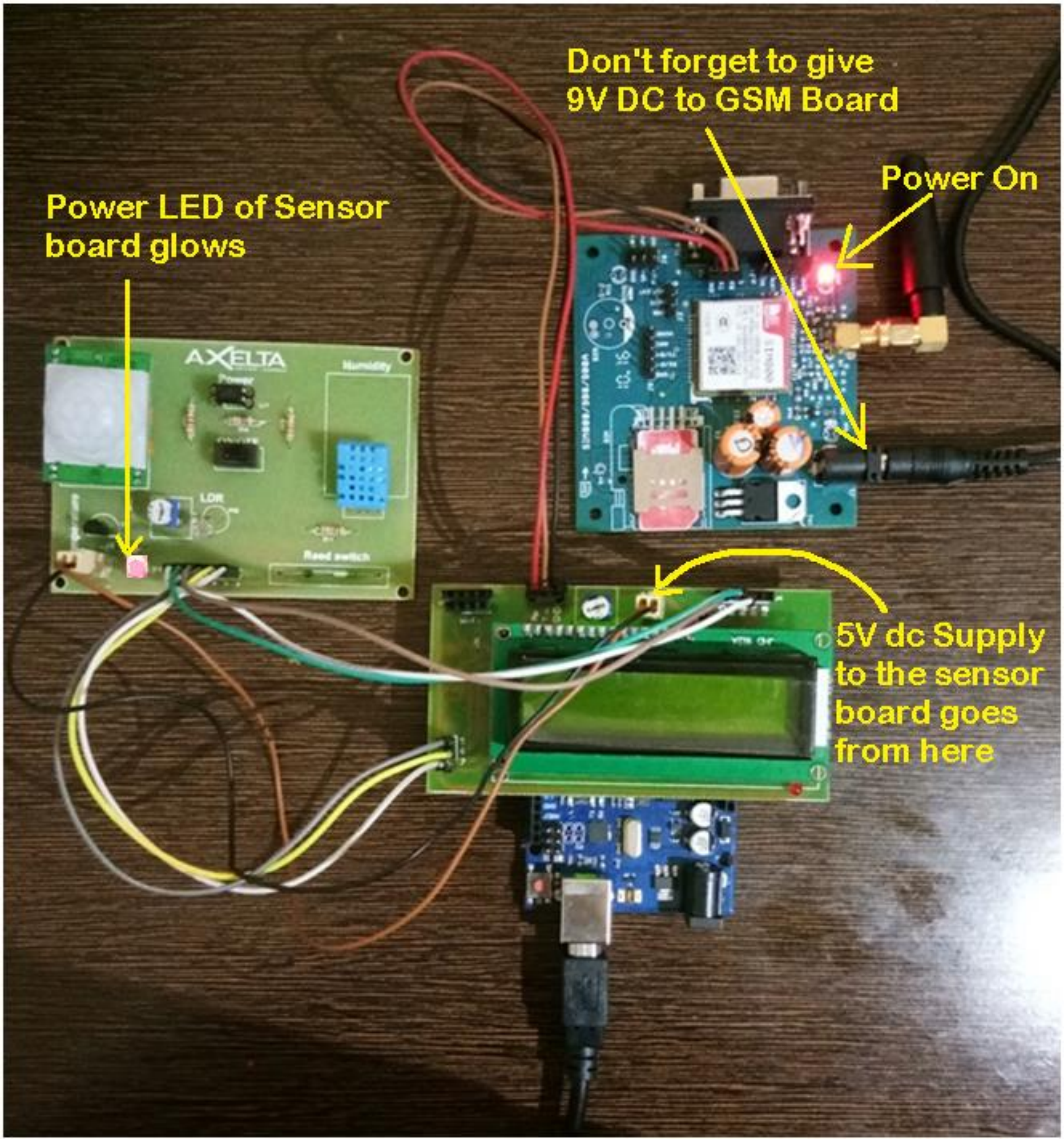
**Connecting GSM Modem to Arduino UNO Board**

- The GSM Modem has a 3-wire connector soldered to it.
- Connect this **3-wire connector to the LCD Shield** which is placed on Arduino UNO Board. It is a one to one placement connector, which can be connected only in one way.
- Insert a **SIM card in the GSM modem** and make sure it has GPRS activated and sufficient recharge/money in it.
- Connect **12V Adapter to GSM Modem** & power the board then wait for signal (We can find on board signal by checking the Blinking LED on GSM board which shifts its frequency from fast pace to slow pace) .

**Make sure to connect to J7 on board & black wire of the connector goes to GND on both sides**


Rx
Tx
Gnd
Rx Tx GND
On Arduino

**Final Image after giving 9V Power Supply to the GSM Board.**

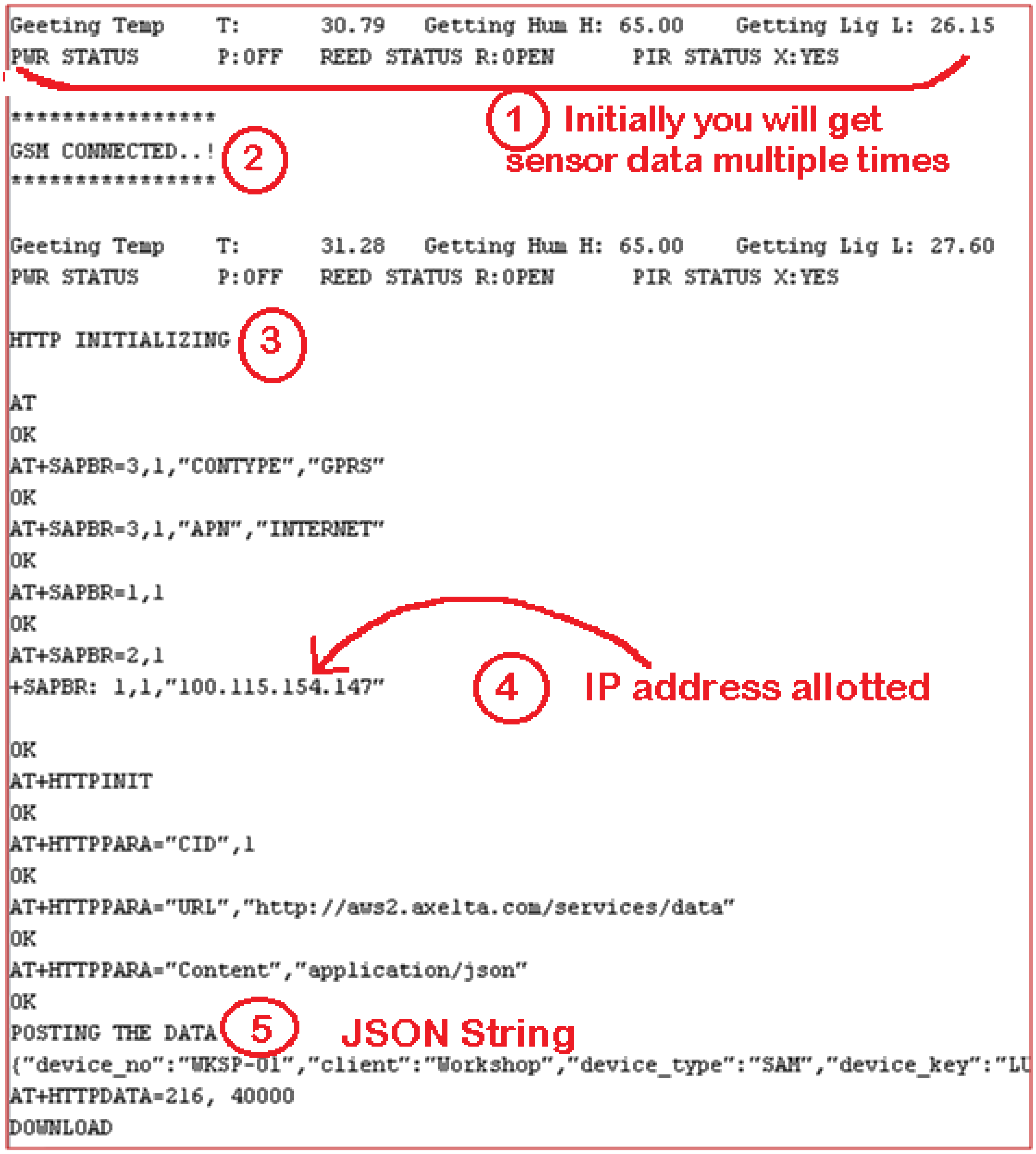


Don't forget to give 9V DC to GSM Board
Power On
Power LED of Sensor board glows
5V dc Supply to the sensor board goes from here
AXELTA
Humidity
Power
LDR
Reed switch

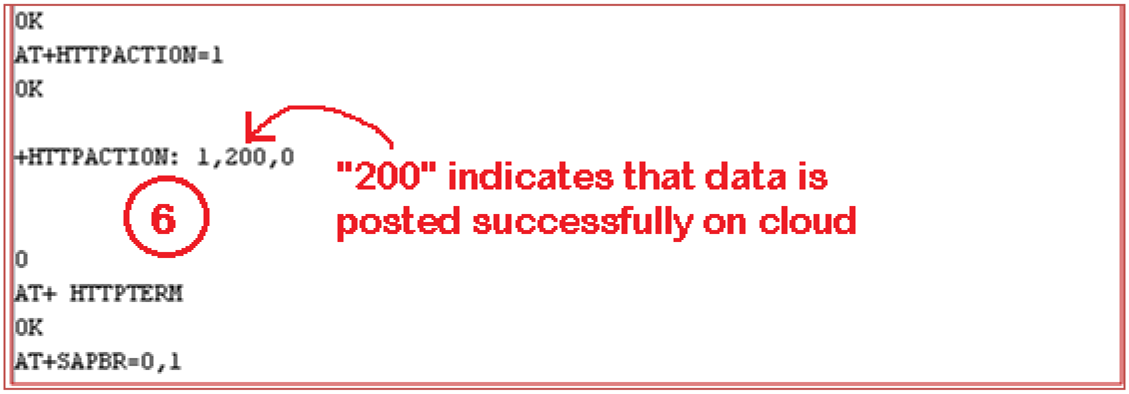
## Programming:

1. **Open Arduino IDE &** Create a new sketch.
2. You can find the “**Communication practical1_code.txt**” by the link: https://drive.google.com/file/d/0Bz7GE98wyjOxdURVUmNGZjdrNkE/view
3. Copy, paste & upload the code in Arduino board & open the serial monitor

If you set correct **baud rate as 9600** You will get the output as given below :



```
Geeting Temp    T:      30.79   Getting Hum H:  65.00     Getting Lig L: 26.15
PWR STATUS      P:OFF   REED STATUS R:OPEN      PIR STATUS X:YES

****************
GSM CONNECTED..!
****************

Geeting Temp    T:      31.28   Getting Hum H:  65.00     Getting Lig L: 27.60
PWR STATUS      P:OFF   REED STATUS R:OPEN      PIR STATUS X:YES

HTTP INITIALIZING

AT
OK
AT+SAPBR=3,1,"CONTYPE","GPRS"
OK
AT+SAPBR=3,1,"APN","INTERNET"
OK
AT+SAPBR=1,1
OK
AT+SAPBR=2,1
+SAPBR: 1,1,"100.115.154.147"

OK
AT+HTTPINIT
OK
AT+HTTPPARA="CID",1
OK
AT+HTTPPARA="URL","http://aws2.axelta.com/services/data"
OK
AT+HTTPPARA="Content","application/json"
OK
POSTING THE DATA
{"device_no":"WKSP-01","client":"Workshop","device_type":"SAM","device_key":"LU
AT+HTTPDATA=216, 40000
DOWNLOAD
```

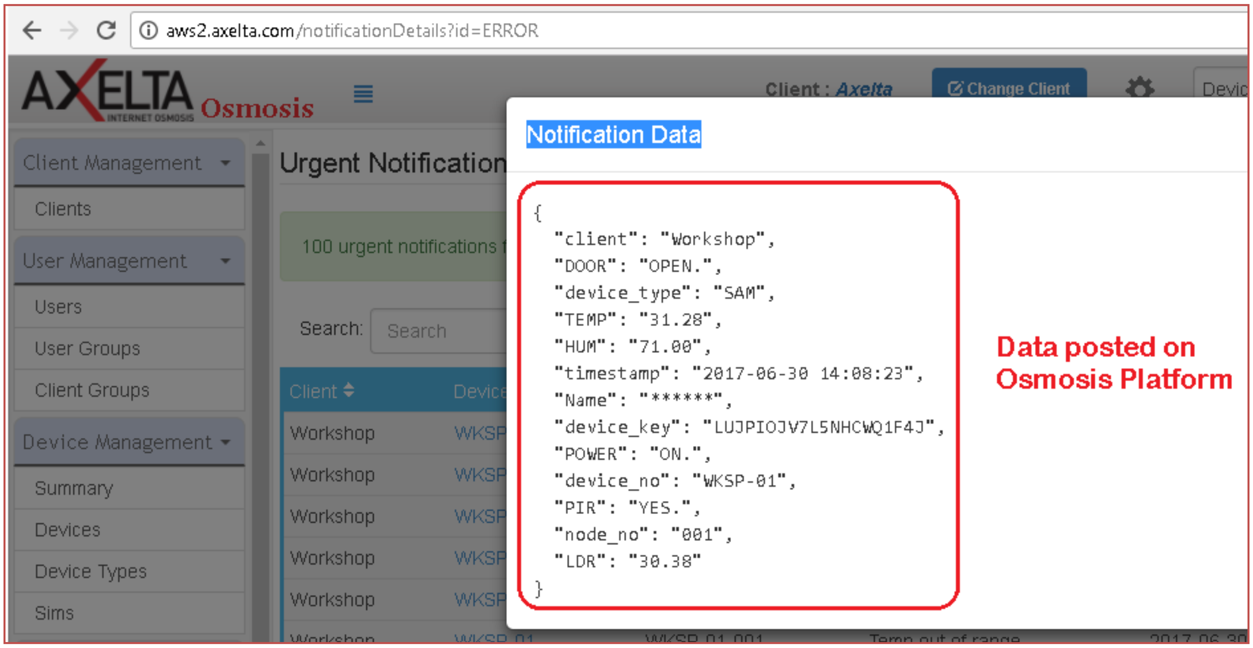


```
OK
AT+HTTPACTION=1
OK

+HTTPACTION: 1,200,0

0
AT+ HTTPTERM
OK
AT+SAPBR=0,1
```

6

"200" indicates that data is posted successfully on cloud

We can see your data on Axelta's Osmosis IOT platform as follows :



```
{
  "client": "Workshop",
  "DOOR": "OPEN.",
  "device_type": "SAM",
  "TEMP": "31.28",
  "HUM": "71.00",
  "timestamp": "2017-06-30 14:08:23",
  "Name": "******",
  "device_key": "LUJPIOJV7L5NHCWQ1F4J",
  "POWER": "ON.",
  "device_no": "WKSP-01",
  "PIR": "YES.",
  "node_no": "001",
  "LDR": "30.38"
}
```

Data posted on Osmosis Platform

## Step by Step Explanation for ‘C’ program in Arduino.

/*Here we are Sending data to web server using GSM/GPRS modem and we are interfacing GSM modem using UART */

**Step1**: Create a new sketch

**Step2:** Include the header file **“LiquidCrystal.h”,** to use the functions related to LCD and Include the header file "**dht11.h**", to use the Humidity related function.

```
#include<dht11.h>
```

```
#include<LiquidCrystal.h>
```

**Step3**: Include the header file **“SoftwareSerial.h”,** to use the functions related to UART.

```
#include <SoftwareSerial.h>
```

**Step4**: Make the global declaration of the UART pins with the microcontroller pins of the Arduino board, i.e., RX, TX with pins 3,4 using “SoftwareSerial”.

```
SoftwareSerial GSM_Serial(3,4);
```

**Step5**: Make the global declaration of the LCD pins with the microcontroller pins of the Arduino board, i.e., RS, E, D4, D5, D6, D7, with pins 8, 9, 10, 11, 12, 13 using “LiquidCrystal”.

```
LiquidCrystallcd(8,9,10,11,12,13);
```

**Step6**: Initialize variables as integers with the pin numbers to which the digital sensors are connected.

```
int R=5;                    // REED SWITCH
int P=6;                    // POWER
int X=7;                    // PIR
```

**Step7**: Initialize variables as integers with the Analog pin numbers to which the analog sensors are connected.

```
int T= A2;                  // TEMPERATURE - A2 has already been defined as Analog Pin 2 in arduino lib
int H=A4;                   //HUMIDITY
int L=A5;                   //LIGHT
```

**Step8**: Initialize a global variable “CNT” as integer with a value as 4;

```
int CNT=4;
```

**Step9:**Initialize strings to store GSM input & Response.

```
String gsm_input="";
String RES_input="";
```

**Step11**: Initialize few global variables **CELSIUS, HUM, LIGHT** as float.

```
float CELSIUS, HUM, LIGHT;
```

**Step12:**Initialize variables as integers to use them in functions.

```
inti=1;
intStart_chck=0;
```

**Step13**: Write a **“setup”** function and initialize the UART communication with 9600 baud rate and the LCD as 16 columns with 2 rows with the help of **“lcd.begin”** function.

```
void setup()
   {
Serial.begin(9600);
GSM_Serial.begin(9600);
lcd.begin(16,2);
pinMode(P, INPUT);
pinMode(R, INPUT);
pinMode(X, INPUT);
   }
```

**Step14:**Create a function to locate a given search string in a given base string

14.1 Initialize a Boolean **“find_string”**search with two strings

14.2 Initialise a variable to find the length of base string

14.3 Iterate the base string from begining to end minus Substring

14.4 Now check for the matches between extracted Substring and Search string

14.5 If both match then return with a true

14.6 If there are no matches return false

```
booleanfind_string(String base, String search)
{
Int len = search.length(); // find the length of the base string
for(int m = 0; m<((base.length()-len)+1);m++)// Iterate from the beginning of the base string till the end minus length of the substring
    {
if(base.substring(m,(m+len))==search) // Check if the extracted Substring Matches the Search String
    {
return true; // if it matches exit the function with a true value
    }
    }
return false; // if the above loop did not find any matches, control would come here and return a false value
}
```

**Step15:**Create a function to locate a given search Character in a given base string and return its position

15.1 Initialise a Boolean **“find_char_loc”**search with a string and a character

15.2 Iterate the base string from begining to end minus Substring

15.3 Now check if the character Matches the Search character

15.4 If match exist return the function with the current location

```
booleanfind_char_loc(String base, char search)
{
for(int m = 0; m <base.length();m++)// Iterate from the beginning of the base string till the end minus length of the substring
     {
if(base[m]==search) // Check if the character Matches the Search character
       {
return m; // if it matches exit the function with the current location value
       }
     }
}
```

**Step16**:

16.1 Start a **“gsminit”** function.

16.2 Using **“GSM_Serial.println”** function, send the command “AT”

16.3 Read the GSM Response

16.4 Print on LCD
16.5 Using “**GSM_Serial.println**” function, send the command “ATE0” for making the ECHO mode off.
16.6 Write 1 seconds delay.
16.7 Read GSM Response
16.8 Print on LCD.

```
voidgsminit()             //GSM Initialization
{
GSM_Serial.println("AT");
lcd.clear();
lcd.print("AT");
DisplayGSMResponse();
GSM_Serial.println("ATE0");
lcd.clear();
lcd.print("ATE0");
    //delay(1000);
DisplayGSMResponse();
}
```

**Step17:**GSM Response

17.1 Start a **“DisplayGSMResponse”**function
17.2 Check for Serial Available
17.3 Initialize a String **“gsm_input”** to store GSM response available on Serial.
17.4 when Serial Avalable read the data in to variable as long as the data available and remove extra spaces in between.
17.5 print the output on LCD and on serial.

```
voidDisplayGSMResponse()
{
if(GSM_Serial.available())
    {
    String gsm_input="";
while(GSM_Serial.available())
    {
gsm_input+= (char)GSM_Serial.read();
    }
gsm_input.trim();
lcd.setCursor(0,1);
lcd.print(gsm_input);
delay(1000);
Serial.println(gsm_input);
    }
}
```

**Step18:**GSM Connection check

18.1 Start a **“GSM_Check”**function

18.2 Using **“GSM_Serial.println”** function, send the command “AT”

18.3 Initialize the string to read the data from the GSM Response

18.4 Use **“find_String”**to search GSM response with “OK”

18.5 Use **“find_Char_loc”** to find the location of response.

18.6 Now if response matches with “OK” print “GSM Connected” or print “GSM NOT CONNECTED"

```
voidGSM_Check()
{
GSM_Serial.println("AT");
if(GSM_Serial.available())
    {
    String gsm_input="";
while(GSM_Serial.available()) // read the data into a variable as long as the
    {
gsm_input+= (char)GSM_Serial.read();
    }
if(find_string(gsm_input,"O"))
    {
intloc = find_char_loc(gsm_input,'O');
    String response = gsm_input.substring(loc);
Serial.println(response);
if(response="K")
    {
lcd.clear();
lcd.print("GSM CONNECTED..!");
Serial.println("GSM CONNECTED..!");
Start_chck=1;
i=1;
    }
    }
    }
if(i==0)
    {
lcd.clear();
lcd.print("GSM NOT");
lcd.setCursor(0,1);
lcd.print("CONNECTED..!!");
Serial.println("GSM  NOT CONNECTED..!");
    }
}
```

**Step19**: HTTP initialization

19.1 Start a**"httpinit"** function.

19.2 Using "GSM_Serial.println" function, send the command "**AT+SAPBR=3,1,CONTYPE,GPRS"**, i.e., make a barrier connection, set the parameters of barriers and the type of connection is of GPRS.

19.3 Write 2 seconds delay for each GSM AT command and print to LCD.

19.4 Using "GSM_Serial.println" function, send the command **"AT+SAPBR=3,1, APN,INTERNET",** i.e., set the Access Point Name as INTERNET.

19.5 Using "GSM_Serial.println" function, send the command **"AT+SAPBR=1,1,",** i.e., make the barrier connection and make the barrier open.

19.6 Using "GSM_Serial.println" function, send the command **"AT+SAPBR=2,1,",** i.e., make the barrier connection and send a query.

19.7 Using "GSM_Serial.println" function, send the command **"AT+HTTPINIT",** i.e., before using HTTP service, HTTPINIT should be executed to initialize the HTTP stack firstly.

19.8 Using "GSM_Serial.println" function, send the command **"AT+HTTPPARA=CID,1",** i.e., it is used for bearer profile identifier.

19.9 Using "GSM_Serial.println" function, send the command **"AT+HTTPPARA=URL,http://aws.axelta.com/services/data"**, i.e., specify the URL to which the data should be posted.

19.10 Using "GSM_Serial.println" function, send the command **"AT+HTTPPARA=Content,application/json"** i.e., to specify the content/data which is being posted is of JSON format.

```
voidhttpinit()
  {
Serial.println("HTTP INITIALIZING");
GSM_Serial.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
lcd.clear();
lcd.print("AT+SAPBR=GPRS");
delay(2000);
DisplayGSMResponse();
GSM_Serial.println("AT+SAPBR=3,1,\"APN\",\"INTERNET\"");
lcd.clear();
lcd.print("AT+SAPBR=APN");
delay(5000);
DisplayGSMResponse();
GSM_Serial.println("AT+SAPBR=1,1");
lcd.clear();
lcd.print("AT+SAPBR=1,1");
delay(2000);
DisplayGSMResponse();
GSM_Serial.println("AT+SAPBR=2,1");
lcd.clear();
lcd.print("AT+SAPBR=2,1");
delay(2000);
DisplayGSMResponse();
GSM_Serial.println("AT+HTTPINIT");
lcd.clear();
lcd.print("AT+HTTPINIT");
delay(2000);
DisplayGSMResponse();
GSM_Serial.println("AT+HTTPPARA=\"CID\",1");
lcd.clear();
lcd.print("AT+HTTPPARA");
delay(2000);
DisplayGSMResponse();
GSM_Serial.println("AT+HTTPPARA=\"URL\",\"http://aws.axelta.com/services/data\"");
lcd.clear();
lcd.print("AT+HTTPPARA=URL");
delay(2000);
DisplayGSMResponse();
}
```

**Step20**:

20.1 Start a**"httppost"** function.

20.2 Initialize a String "data" and arrange data in JSON format.

20.3 Using “GSM_Serial.println” function, send the command **“AT+HTTPDATA=JSON Count,40000”**, i.e., the character count of the data to be sent and the duration in milliseconds.

20.4 Using “GSM_Serial.print” function, send the JSON data with fields of device key, node number, Temperature, Humidity, Light intensity, Power supply sensing, Reed switch/Door sensing, etc.

20.5 Connect to the server before posting the data using HTTP.

20.5 Using “GSM_Serial.println” function, send the command “**AT+HTTPACTION=1**”, i.e., option ‘1’ is for posting the data.

20.6 Initialize a string “RES_input” and read all the data available on serial from “AT+HTTPACTION” response

20.7 Use the **“find_string”** and **“find_char_loc”**to search the response with “200”.

20.8 If the string matches with response print “DATA POSTED” else print “Error in Posting”

```
voidhttppost()
  {
Serial.println("POSTING THE DATA");
    String data;
delay(500);
data+="{\"device_no\":\"WKSP01\",\"client\":\"Workshop\",\"device_type\":\"SAM\",\"device_key\
":\"LUJPIOJV7L5NHCWQ1F4J\",\"node_no\":\"001\",\"Temp\":\"";
data += String(CELSIUS);
    data += "\",\"HUM\":\"";
data += String(HUM);
    data += "\",\"LDR\":\"";
data += String(LIGHT);
    data += "\",\"POWER\":\"";
if(digitalRead(P)==LOW)
    {
data += "ON.";
    }
else
    {
data += "OFF";
    }
    data += "\",\"DOOR\":\"";
if(digitalRead(R)==LOW)
    {
data += "OPEN.";
    }
```

```
else
    {
data += "CLOSE";
    }

data += "\"}";
Serial.println(data);
int size = data.length();
    String initData = "AT+HTTPDATA=";
initData += String(size);
initData += ", 40000";
GSM_Serial.println(initData);
lcd.clear();
lcd.print("AT+HTTPDATA");
delay(500);
DisplayGSMResponse();
GSM_Serial.println(data);
lcd.clear();
lcd.print("Connecting to");
lcd.setCursor(0,1);
lcd.print("Server....!!!");
delay(2000);
GSM_Serial.println("AT+HTTPACTION=1");
delay(10000);
if(GSM_Serial.available())
    {
    String RES_input="";
while(GSM_Serial.available()) // read the data into a variable as long as the
    {
RES_input+= (char)GSM_Serial.read();
    }
lcd.clear();
Serial.println(RES_input);
if(find_string(RES_input,","))
    {
intloc = find_char_loc(RES_input,'1');
    String no = RES_input.substring(loc+1);
    String response=no.substring(1,4);
Serial.println(response);
if(response=="200")
     {
lcd.clear();
lcd.print("RESPONSE:");
lcd.setCursor(10,0);
lcd.print(response);
lcd.setCursor(0,1);
lcd.print("DATA POSTED");
delay(1000);
     }
else
     {
lcd.clear();
lcd.print("Error in Posting");
delay(1000);
       }
     }
    }
  }
```

**Step21**: GSM & HTTP Termiation

21.1 Start a “**gsmterm**” function.

21.2 Using “GSM_Serial.println” function, send the command “**AT+HTTPTERM**”, i.e., to terminate the HTTP connection.

21.3 Using “GSM_Serial.println” function, send the command “**AT+ SAPBR=0,1**”, i.e., close the barrier connection.

```
voidgsmterm()
   {
GSM_Serial.println("AT+ HTTPTERM");
lcd.clear();
lcd.print("AT+HTTPTERM");
delay(1000);
DisplayGSMResponse();
GSM_Serial.println("AT+SAPBR=0,1");
lcd.clear();
lcd.print("AT+SAPBR=0,1");
delay(1000);
DisplayGSMResponse();
    }
```

**Step22**:

22.1 Start a void **“TEMPERATURE**” function and with the help of “analogRead” function read the value from the Analog variable and store it to an integer variable. This gets the Temperature from the sensor and prints the actual value by converting analog voltage to the temp. Refer to data sheet for temp in manual.

22.2 Divide the obtained value with the resolution of the ADC (i.e., 1023.0) and multiply the result with reference milli volts. (5V = 5000mV) and save it to a float variable.

22.3 Divide the above result by 10 because there will be 1°C change for every 10mV of output and save it to another float variable.

22.4 Set the cursor position by column and row numbers using “lcd.setCursor” function.

22.5 Display the value on the LCD using “lcd.print” function.

```
void TEMPERATURE()
    {
Serial.print("Geeting Temp\t");
lcd.clear();
intvalue_temp=analogRead(T);
floatmillivolts_temp=(value_temp/1023.0)*5000;
CELSIUS=millivolts_temp/10;
lcd.setCursor(0,0);
lcd.print("T:");
lcd.print(CELSIUS);
Serial.print("T:\t");
Serial.print(CELSIUS); Serial.print("\t");
    }
```

**Step23**: Write similar functions to read analog Humidity and Light values.

```
void HUMIDITY()
    {
Serial.print("Getting Hum ");
intchk = DHT11.read(H);
    HUM=DHT11.humidity;
lcd.setCursor(9,0);lcd.print("H:");
lcd.print(HUM);
Serial.print("H: ");
Serial.print("HUM"); Serial.print("\t");
   }
void LIG()
   {
Serial.print("Getting Lig ");
intvalue_lig=analogRead(L);
floatmillivolts_lig =(value_lig /1023.0)*5000;
LIGHT=millivolts_lig /10;
lcd.setCursor(0,1);
lcd.print("L:");
lcd.print(LIGHT);
Serial.print("L: ");
Serial.print(LIGHT);Serial.println("\t");
delay(2000);
lcd.clear();
   }
```

**Step24**:

24.1 Start a void “POWER” function and check the concerned digital pin/variable is LOW.

24.2 If it is LOW, display the status on the LCD with the help of “lcd.print” function and “lcd.setCursor” functions respectively.

24.3 Else, display its status.

```
void POWER()
   {
Serial.print("PWR STATUS\t");
if(digitalRead(P)==LOW)
    {
lcd.setCursor(0,0);
lcd.print("P:ON");
Serial.print("P:ON");Serial.print("\t");
```

```
      }
else
      {
lcd.setCursor(0,0);
lcd.print("P:OFF");
Serial.print("P:OFF");Serial.print("\t");
      }
    }
```

**Step25**: write similar functions to read digital Reed switch status.

```
void REED()
    {
Serial.print("        REED STATUS ");
if(digitalRead(R)==LOW)
      {
lcd.setCursor(6,0);
lcd.print("R:OPEN");
Serial.print("R:OPEN");Serial.print("\t");
      }
else
      {
lcd.setCursor(6,0);
lcd.print("R:CLOSE");
Serial.print("R:CLOSE");Serial.print("\t");
      }
   }
void PIR()
    {
Serial.print("PIR STATUS ");
if(digitalRead(X)==LOW)
      {
lcd.setCursor(0,1);
lcd.print("X:YES");
Serial.print("X:YES");Serial.print("\t");
      }
else
      {
```

```
lcd.setCursor(0,1);
lcd.print("X:NO ");
Serial.print("X:NO");Serial.println("\t");
     }
delay(2000);
   }
```

**Step26**:

26.1 Start a function with the name “loop”. This is the Main loop/function of the whole code/program.

26.2 Call the “lcd.clear” function to erase any old/garbage data on the lcd.

26.3 With the help of “lcd.setCursor” function, set the lcd cursor position to 0th column and 0th row.

26.4 With the help of “lcd.print” function, display some string data – “Axetla Systems”.

26.5 Call the delay function and after some time delay clear the lcd using “lcd.clear” function.

26.6 Start an infinite “while” loop.

26.7 Check if the variable “cnt” value is less than 1.

26.8 Call the function “GSM_Check” if returns with “OK” then Call “httpinit”, “httppost” and “gsmterm” sequentially with 2000 milli seconds delay, using “delay” function in milli seconds.

26.9 Call the “lcd.clear” function and display the string “DATA POSTED” using “lcd.print” function.

26.10 If “GSM_Check” returns with no connection then call the functions **“TEMPERATURE”,“HUMIDITY”,“LIG(Light)”,“POWER”** and **“REED”** sequentially

26.11 Make the “cnt” variable value as 4 again.

26.12 Else if the compared “**CNT**” value is not less than 1, then call the functions **“TEMPERATURE”,“HUMIDITY”,“LIG(Light)”,“POWER”** and **“REED”** sequentially with 2000 milli seconds delay.
26.13 Decrease the “**CNT**” value by 1 for every loop.

```
 /********MAIN LOOP*********/
void loop()
  {
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Axetla Systems");
delay(1000);
lcd.clear();
while(1)
     {
if(CNT<1)   //CNT for Time Delay
     {
if(Start_chck==0)
     {
GSM_Check();
i=0;
delay(2000);
     }
else
     {
gsminit();
delay(2000);
httpinit();
delay(2000);
httppost();
delay(2000);
gsmterm();
delay(2000);
     CNT=4;
i=1;
Start_chck=0;
     }
     }
TEMPERATURE();
HUMIDITY();
LIG();
POWER();
REED();
```

```
PIR();
Serial.println();
```

**Step20:**

**Send Sensor Data to your own Web Service. You need to come back to this step after completing the Web Services Tutorial**

Replace new URL created by Web services as shown below

**Finally Sensor Data is posted to Server; output can be seen by logging to server.**

```
GSM_Serial.println("AT+HTTPPARA=\"URL\",\"http://aws.axelta.com/services/data\"");
```

**Replace with your webserviceURL here**

http://IPAddress:8080/WebServicesTraining/rest/service/storeData

IPAddress: Provide your IP address

Again upload the code to Arduino UNO Board, then Data will be posted to above configured URL.

**Step21:**

To check whether data has stored in mongo Db, run the below URL

***http://IPAddress:8080/WebServicesTraining/rest/service/deviceNo/DeviceNo***

IPAddress: Provide your IP address

DeviceNo: Provide your device number