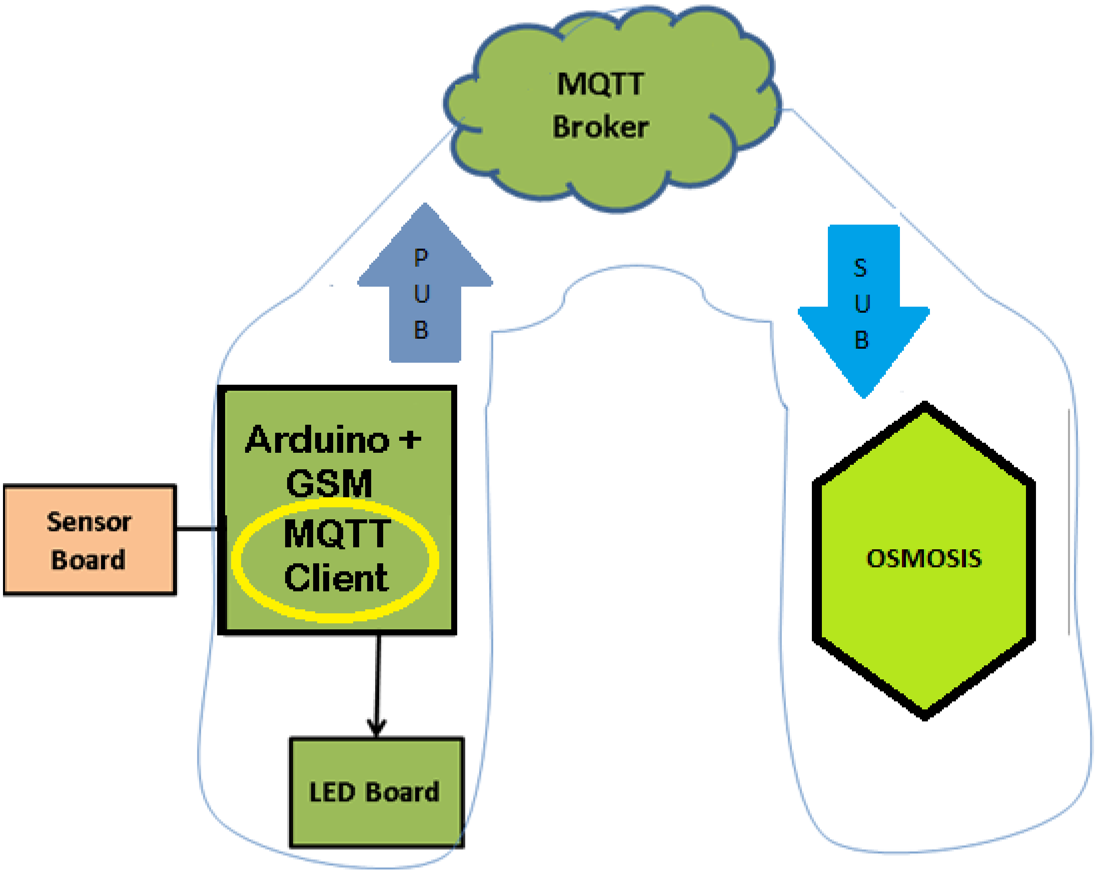
User Manual on

Post data using MQTT

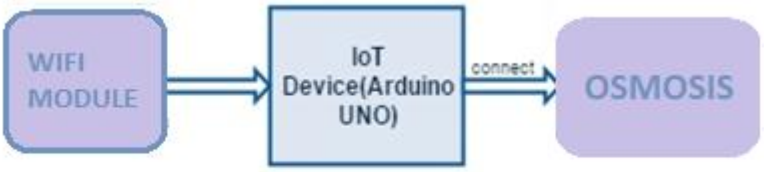
Practical's Objective:

Post data to osmosis platform using MQTT.

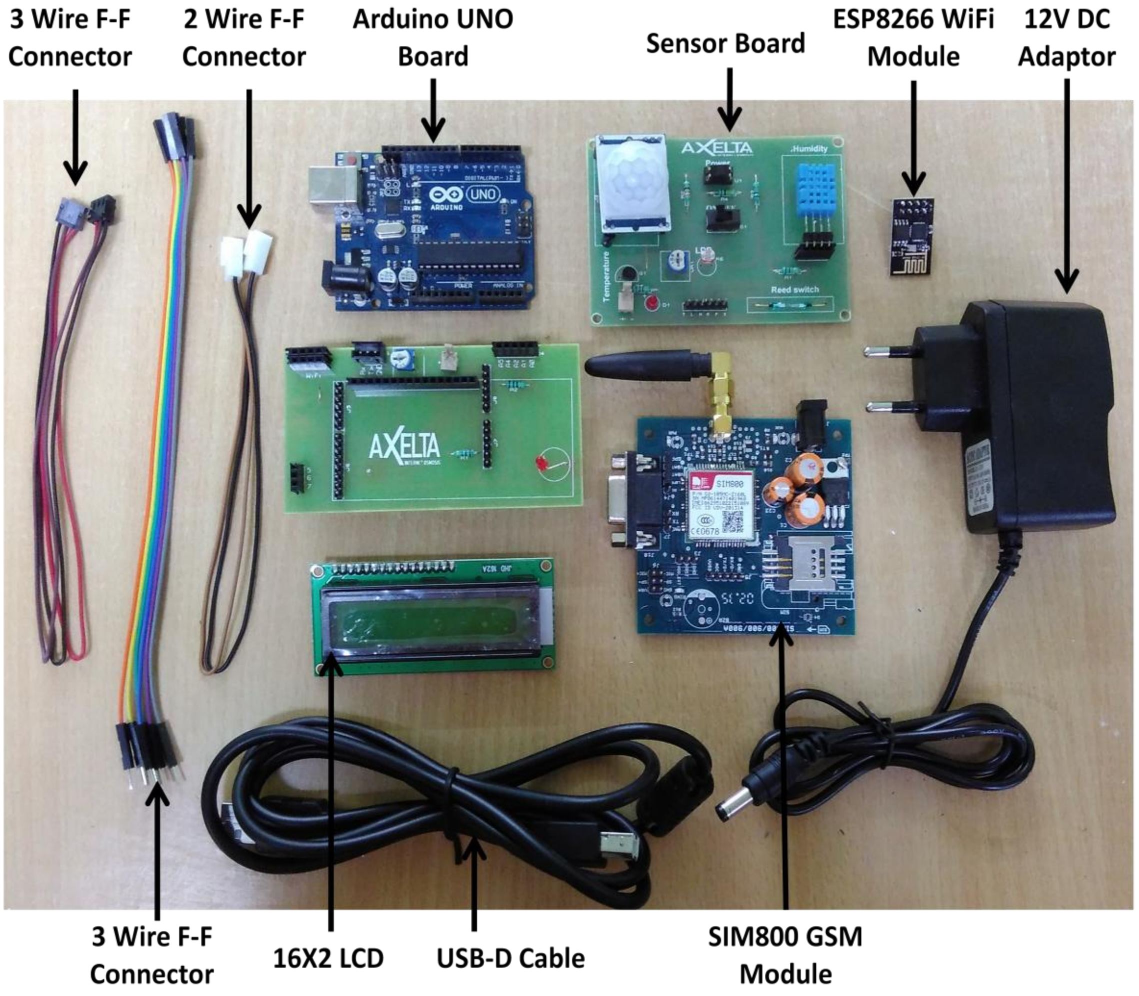
1. **End-End IoT Flow diagram:**



GSM at Node Flow Diagram:



2. **Hardware requirements:**



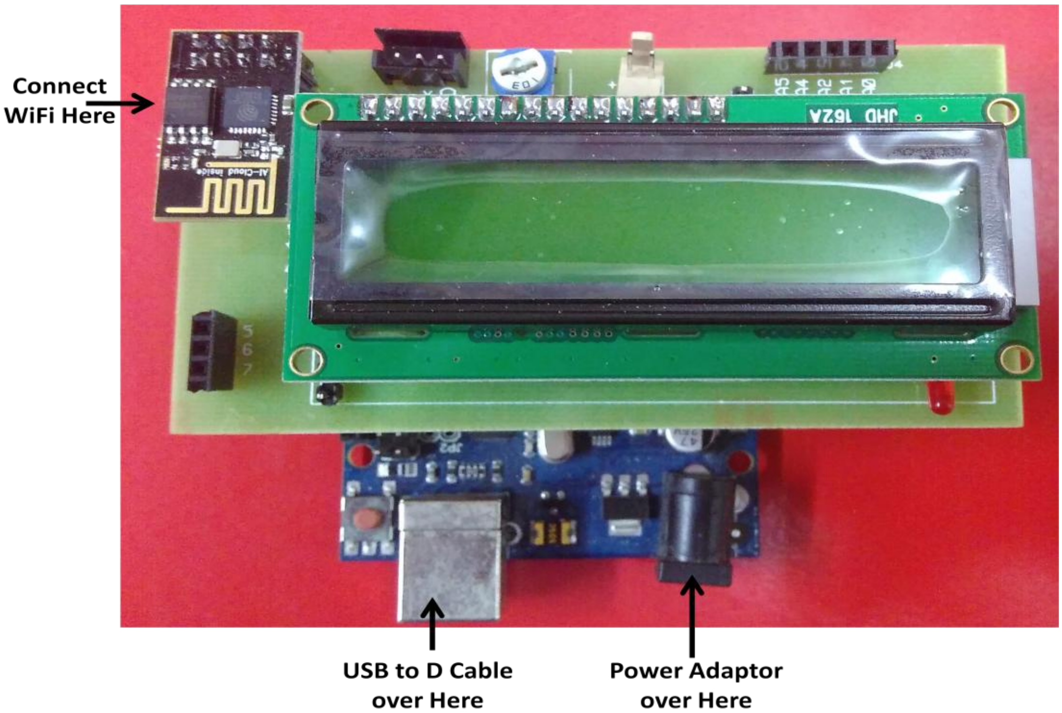
Software Requirement:
Same as that of previous Practical

Arduino UNO Board Connections:

Don’t change the sensor connections. They will also remainsame.

Connecting WIFI module to Arduino UNO Board

1) **Don’t connect WiFi Module ESP8266 to the J9 on LCD Shieldunless you give12V Supply to the Arduino Board.**
2) After plugging ESP8266 as shown below connect your PC to Arduino Board via USB Cable.



Make sure that all the Hardware connections are proper as explained above.

Programming:

Click the link: https://drive.google.com/open?id=1CFgC2BHNjvn1KGnTcSbrZHb2MEh997F1

Step by Step Explanation for the Program

```
#include <PubSubClient.h>

#include <dht11.h>
#include<LiquidCrystal.h> //include LCD library to work with LCD display
#include <SoftwareSerial.h>
#include <WiFiEspClient.h>
#include <WiFiEsp.h>
#include <WiFiEspUdp.h>
#include <PubSubClient.h>

SoftwareSerial WiFi_Serial(3, 4); /*WiFi module connected to pins 3-Rx, 4-Tx
LiquidCrystal lcd(8, 9, 10, 11, 12, 13); //LCD connected to pins RS-En, E-Rs, 10-D4, 11-D5, 12-D6,
13-D7*/

#define WIFI_AP "Axelta"
#define WIFI_PASSWORD "Axelta140218"
String CN = "client";
String clientName1; /*Define string variable for client name */
```

```
/*==============================================================================
=============*/
/*CHANGE THE PARAMETERS BELOW ACCORDING TO YOUR DEVICE INFORMATION
CREATED ON AXELTA OSMOSIS*/
/*==============================================================================
=============*/

#define Device_No       "WKSP-01"            // Enter Device No
#define CN          "Workshop"           // Enter Client Name
#define Device_Type     "SAM"                // Enter Your Device Type
#define Device_Key      "LUJPIOJV7L5NHCWQ1F4J" // Enter Your Device Key
#define Node_No        "001"                // Enter Your Node number
#define Name           "MQTTTEST"            // Enter Your Name

/*==============================================================================
=============*/

//#define TOKEN "ARDUINO_DEMO_TOKEN"

// DHT
int R = 5;  // REED SWITCH
int P = 6;  // POWER
int X = 7;  // PIR
int T = A2; // TEMPERATURE - A2 has already been defined as Analog Pin 2 in
int H = A5; // HUMIDITY
int L = A4; // LIGHT
int CNT = 7;
float CELSIUS, HUM, LIGHT;
dht11 DHT11;
int i = 1;
int Start_chck = 0;

char osmosisServer[] = "osmosis.axelta.com";

// Initialize the Ethernet client object
WiFiEspClient espClient;

PubSubClient client(espClient);

int status = WL_IDLE_STATUS;
unsigned long lastSend;

void setup() {
  // initialize serial for debugging
  Serial.begin(9600);
  InitWiFi();
  client.setServer( osmosisServer, 1883 );
  lastSend = 0;
}
```

```
void loop() {
  status = WiFi.status();
  if ( status != WL_CONNECTED) {
    while ( status != WL_CONNECTED) {
      Serial.print("Attempting to connect to WPA SSID: ");
      Serial.println(WIFI_AP);
      // Connect to WPA/WPA2 network
      status = WiFi.begin(WIFI_AP, WIFI_PASSWORD);
      delay(500);
    }
    Serial.println("Connected to AP");
  }

  if ( !client.connected() ) {
    reconnect();
  }

  if ( millis() - lastSend > 10000 ) { // Update and send only after 1 seconds
    getAndSendTemperatureAndHumidityData();
    lastSend = millis();
  }

  client.loop();
}

/*=============================================================================
=============*/
/*                    FUNCTION TO GET TEMPERATURE SENSOR OUTPUT                    */
/*=============================================================================
=============*/

void TEMPERATURE()
{
  Serial.print("Geeting Temp\t");
  lcd.clear();
  int value_temp = analogRead(T); //Read analog value of temperature sensor output from pin A2
  delay(10);
  value_temp = analogRead(T);
  delay(10);
  float millivolts_temp = (value_temp / 1023.0) * 5000; //convert it to milli volts output ([actual
temperature output from sensor] * [Input voltage (5V = 5000mV)] / [Resolution of ADC 2^10 =
1024])
  CELSIUS = millivolts_temp / 10;
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(CELSIUS);
  Serial.print("T:\t");
  Serial.print(CELSIUS); Serial.print("\t");
}
```

```
/*==============================================================================
=============*/
/*                    FUNCTION TO GET HUMIDITY SENSOR OUTPUT                              */
/*==============================================================================
=============*/

void HUMIDITY()
{
  Serial.print("Getting Hum ");
  int chk = 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");
}

/*==============================================================================
=============*/
/*                    FUNCTION TO GET LDR SENSOR OUTPUT                              */
/*==============================================================================
=============*/

void LIG()
{
  Serial.print("Getting Lig ");
  int value_lig = analogRead(L);
  delay(10);
  value_lig = analogRead(L);
  float volts_lig = (value_lig / 1023.0) * 5;
  LIGHT = 500 / (4 * ((5 - volts_lig) / volts_lig)); // calculate the Lux = 500/[R1 * ((Vin -
Vsense)/Vsense)]
  lcd.setCursor(0, 1);
  lcd.print("L:");
  lcd.print(LIGHT);
  Serial.print("L: ");
  Serial.print(LIGHT); Serial.println("\t");
  delay(2000);
  lcd.clear();
}

/*==============================================================================
=============*/
/*                    FUNCTION TO GET POWER SENSOR OUTPUT                              */
/*==============================================================================
=============*/

void POWER()
{
  Serial.print("PWR STATUS\t");
  if (digitalRead(P) == LOW) // if output form sensor is '0' then print NO power
  {
    lcd.setCursor(0, 0);
    lcd.print("P:OFF");
    Serial.print("P:OFF"); Serial.print("\t");
  }
  else
  {
    lcd.setCursor(0, 0);
    lcd.print("P:ON");
    Serial.print("P:ON"); Serial.print("\t");
  }
}

/*================================================================================
=============*/
/*                  FUNCTION TO GET REED SWITCH SENSOR OUTPUT                    */
/*================================================================================
=============*/

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");
  }
}

/*================================================================================
=============*/
/*                        FUNCTION TO GET PIR SENSOR OUTPUT                      */
/*================================================================================
=============*/

void PIR()
{
  Serial.print("PIR STATUS ");
  if (digitalRead(X) == LOW)
  {
    lcd.setCursor(0, 1);
    lcd.print("X:YES");
     }
  else
  {
    lcd.setCursor(0, 1);
    lcd.print("X:NO ");
    Serial.print("X:NO"); Serial.println("\t");
  }
  delay(2000);
}

/*=============================================================================
=============*/


void getAndSendTemperatureAndHumidityData()
{
  TEMPERATURE();
  HUMIDITY();
  LIG();
  POWER();
  REED();
  PIR();
  String data; //form the JSON string with the available sensor data
  data += "{\"device_no\":\"" + String(Device_No) + "\",\"client\":\"" + String(CN) +
"\",\"device_type\":\"" + String(Device_Type) + "\",\"device_key\":\"" +  String(Device_Key) +
"\",\"node_no\":\"" + String(Node_No) + "\",\"Name\":\"" + String(Name) + "\",\"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 += "\"}";

  // Send payload
  client.publish( "data/LUJPIOJV7L5NHCWQ1F4J", (char*) data.c_str());
  Serial.println( data );
  Serial.print("sizeof data: ");
  Serial.println(data.length());
}
```

```
void InitWiFi()
{
  // initialize serial for ESP module
  WiFi_Serial.begin(9600);
  // initialize ESP module
  WiFi.init(&WiFi_Serial);
  // check for the presence of the shield
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue
    while (true);
  }

  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(WIFI_AP);
    // Connect to WPA/WPA2 network
    status = WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    delay(500);
  }
  Serial.println("Connected to AP");
}

void reconnect()
{
  if (!client.connected())
  {
    //Serial.print("Attempting MQTT connection...");

    if (client.connect((char*) clientName1.c_str()))
    {
      Serial.println("connected");
      if (client.publish("data/LUJPIOJV7L5NHCWQ1F4J", "HELLO"))
        Serial.println("Publish ok");
      else
      {
        Serial.println("Publish failed");
      }
      //Serial.println("Publish failed");
      delay(1000);

    }
    else
    {
      delay(500);
    }
  }
}
```