**Design Project Report**

**Internet of Things**

**EEE F411**

****

**SUBMITTED BY**

|  |  |
| --- | --- |
| **NAME** | **ID NO.** |
| Aditya Agrawal | 2017A3PS0201P |
| Nihar Zanwar | 2017A8PS0270P |
| D. Krishna Swaroop | 2017A3PS0315P |
| Aman Shukla | 2017A8PS0436P |

**Submitted to Dr. Vinay Chamola for partial fulfilment of the course Internet of Things (EEE F411).**

**Introduction**

The project is to design an IoT system to monitor the temperature and pH levels of water flowing out of factories. The entire setup can be thought of as having two stages-

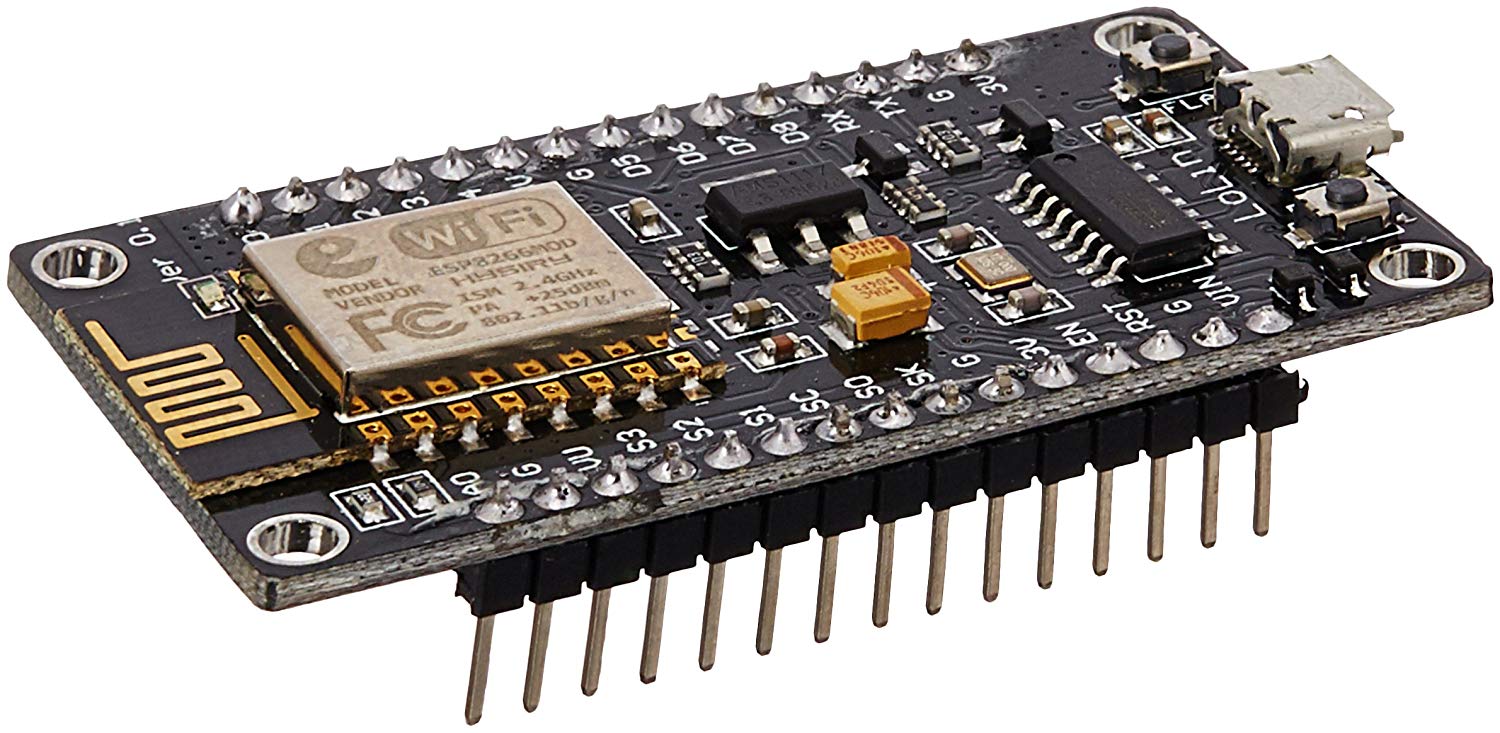
Data Transmission and Storage

Data Acquisition

The first stage involves NodeMCU, DS18B20 temperature sensor and a pH sensor.

**Microcontroller**

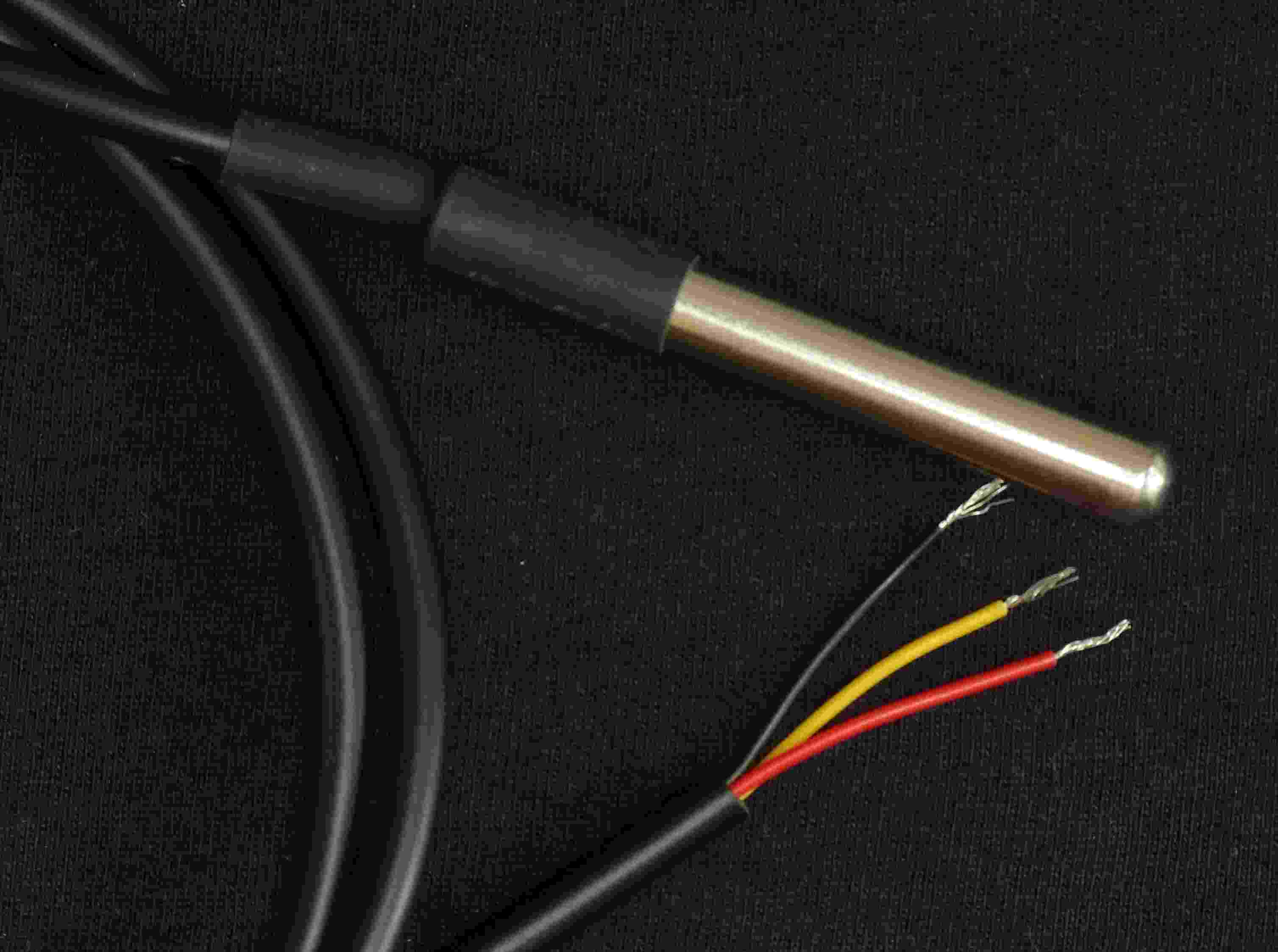
**NodeMCU** is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.



**Temperature Sensor**

We have used the **DS18B20** temperature sensor manufactured by Dallas to measure temperature of the water stream. This is a digital thermometer that can provide 9-bit to 12-bit Celsius temperature measurements. It communicates over a 1-Wire bus i.e; requires only one data line for communication with a microcontroller. Although the setup we have designed has only one such sensor attached to the NodeMCU, each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. The sensor requires a pull-up resistor between the Data and Vcc lines.

The operating temperature is -55 - 125 which makes it suitable for our application. The datasheet specifies that the sensor is waterproof but we couldn’t verify a certified IP rating. In order to ensure that the system stays functional for longer period of time, it is advisable to put them in a silicon enclosure.



For interfacing the sensor to the NodeMCU, we have used the OneWire library. The **OneWire** library provides routines for communication via the Dallas OneWire Protocol. OneWire is a Master/Slave protocol, and all communication cabling required is a single wire. 

**pH Sensor**

Write some bakchodi here. I don’t know what sensor we are using.

At this point, we have described the setup of the microcontroller and sensors. This concludes the Data Acquisition stage of the entire setup.

Moving onto the **Data Transmission and Storage**, the data collected from the sensor is constantly beamed over a **Wi-Fi** connection to a laptop running a MQTT broker called ‘**Mosquitto**’. The **MQTT broker** is run through a Python script on the laptop.

Eclipse Mosquitto is an open source message broker that implements the MQTT protocol using a **publish/subscribe** model. This model makes it suitable for low power IoT applications.



The MQTT broker publishes the incoming data on a database which is the subscriber in this case. We are using InfluxDB as the database.

**InfluxDB** is an open-source data store developed by Influxdata for any use case involving large amounts of time-stamped data, log data, application metrics, IoT sensor data, and real-time analytics. It is a time series database designed to handle high write and query loads.



**ChronoGraf** is the UI and administrative component of the InfluxDB platform. It allows the user to quickly see real-time visualizations of the data.

The following is the code to be uploaded on the NodeMCU.

*#include <OneWire.h>*

*#include <ESP8266WiFi.h>*

*#include <PubSubClient.h>*

*#include <DallasTemperature.h>*

*#define SensorPin A0 //pH meter Analog output to Arduino Analog Input 0*

*#define Offset 0.00 //deviation compensate*

*#define LED 13*

*#define samplingInterval 2000*

*#define printInterval 8000*

*#define ArrayLenth 40 //times of collection*

*const char\* ssid = "aditya";*

*const char\* password = "adityaaditya123";*

*const char\* mqttServer = "192.168.43.248";*

*const int mqttPort = 1883;*

*int pHArray[ArrayLenth]; //Store the average value of the sensor feedback*

*int pHArrayIndex=0;*

*// GPIO where the DS18B20 is connected to*

*const int oneWireBus = 4;*

*// Setup a oneWire instance to communicate with any OneWire devices*

*OneWire oneWire(oneWireBus);*

*// Pass our oneWire reference to Dallas Temperature sensor*

*DallasTemperature sensors(&oneWire);*

*WiFiClient espClient;*

*PubSubClient client(espClient);*

*float temperatureF;*

*float temperatureC;*

*void setup() {*

*// Start the Serial Monitor*

*Serial.begin(9600);*

*// Start the DS18B20 sensor*

*sensors.begin();*

*pinMode(LED,OUTPUT);*

*WiFi.begin(ssid, password);*

*while (WiFi.status() != WL\_CONNECTED) {*

*delay(500);*

*Serial.println("Connecting to WiFi..");*

*}*

*Serial.println("Connected to the WiFi network");*

*client.setServer(mqttServer, mqttPort);*

*client.setCallback(callback);*

*while (!client.connected()) {*

*Serial.println("Connecting to MQTT...");*

*if (client.connect("ESP8266Client")) {*

*Serial.println("connected");*

*} else {*

*Serial.print("failed with state ");*

*Serial.print(client.state());*

*delay(2000);*

*}*

*}*

*}*

*void callback(char\* topic, byte\* payload, unsigned int length) {*

*Serial.print("Message arrived in topic: ");*

*Serial.println(topic);*

*Serial.print("Message:");*

*for (int i = 0; i < length; i++) {*

*Serial.print((char)payload[i]);*

*}*

*Serial.println();*

*Serial.println("-----------------------");*

*}*

*void loop() {*

*static unsigned long samplingTime = millis();*

*static unsigned long printTime = millis();*

*static float pHValue,voltage;*

*if(millis()-samplingTime > samplingInterval)*

*{*

*pHArray[pHArrayIndex++]=analogRead(SensorPin);*

*if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;*

*voltage = avergearray(pHArray, ArrayLenth)\*5.0/1024; //ph\_call*

*sensors.requestTemperatures();*

*temperatureC = sensors.getTempCByIndex(0);*

*temperatureF = sensors.getTempFByIndex(0);*

*pHValue = 3.5\*voltage+Offset;*

*pHValue = pHValue/2;*

*samplingTime=millis();*

*}*

*if(millis() - printTime > printInterval) //Every 800 milliseconds, print a numerical, convert the state of the LED indicator*

*{*

*Serial.print(temperatureC);*

*Serial.println("ºC");*

*Serial.print(temperatureF);*

*Serial.println("ºF");*

*Serial.print("Voltage:");*

*Serial.print(voltage,2);*

*Serial.print(" pH value: ");*

*Serial.println(pHValue,2);*

*digitalWrite(LED,digitalRead(LED)^1);*

*printTime=millis();*

*client.publish("topic/ph", String(pHValue).c\_str());*

*client.subscribe("topic/ph");*

*client.publish("topic/temp", String(temperatureC).c\_str());*

*client.subscribe("topic/temp");*

*}*

*}*

*double avergearray(int\* arr, int number){*

*int i;*

*int max,min;*

*double avg;*

*long amount=0;*

*if(number<=0){*

*Serial.println("Error number for the array to avraging!/n");*

*return 0;*

*}*

*if(number<5){ //less than 5, calculated directly statistics*

*for(i=0;i<number;i++){*

*amount+=arr[i];*

*}*

*avg = amount/number;*

*return avg;*

*}else{*

*if(arr[0]<arr[1]){*

*min = arr[0];max=arr[1];*

*}*

*else{*

*min=arr[1];max=arr[0];*

*}*

*for(i=2;i<number;i++){*

*if(arr[i]<min){*

*amount+=min; //arr<min*

*min=arr[i];*

*}else {*

*if(arr[i]>max){*

*amount+=max; //arr>max*

*max=arr[i];*

*}else{*

*amount+=arr[i]; //min<=arr<=max*

*}*

*}//if*

*}//for*

*avg = (double)amount/(number-2);*

*}//if*

*return avg;*

*}*

The above code will connect the NodeMCU to the specified IP address over a Wi-Fi connection while also computing the temperature and pH values from the sensor data. The NodeMCU transfers these values over Wi-Fi to **Python scripts** running on the laptop.

The Python scripts are as follows:

*#!/usr/bin/env python3*

*import paho.mqtt.client as mqtt*

*import time*

*import requests*

*# This is the Subscriber*

*url = "http://localhost:9999/api/v2/write?org=iot\_project&bucket=sensors&precision=s"*

*headers = {"Authorization":"Token ZfENj35c11xxbeNTd0fDh6F9PoQ-pO3Ji1sClFFzXRv7tsgSWBFLkVMZ10RxQWfCigTBelTG20yicO1\_1hp10g=="}*

*##data = "ph,sensor=sensor1 ph\_value=22 " + str(int(time.time()))*

*##r = requests.post(url, data=data, headers=headers)*

*def on\_connect(client, userdata, flags, rc):*

*print("Connected with result code "+str(rc))*

*client.subscribe("topic/val")*

*def on\_message(client, userdata, msg):*

*print("received ph"+msg.payload.decode())*

*data = "ph,sensor=sensor1 ph\_value="+msg.payload.decode()+" " + str(int(time.time()))*

*r = requests.post(url, data=data, headers=headers)*

*client = mqtt.Client()*

*client.connect("localhost",1883,60)*

*client.on\_connect = on\_connect*

*client.on\_message = on\_message*

*client.loop\_forever()*

The second script is as follows:

*#!/usr/bin/env python3*

*import paho.mqtt.client as mqtt*

*import time*

*import requests*

*# This is the Subscriber*

*url = "http://localhost:9999/api/v2/write?org=iot\_project&bucket=sensors&precision=s"*

*headers = {"Authorization":"Token ZfENj35c11xxbeNTd0fDh6F9PoQ-pO3Ji1sClFFzXRv7tsgSWBFLkVMZ10RxQWfCigTBelTG20yicO1\_1hp10g=="}*

*##data = "temp,sensor=sensor1 temp\_value=22 " + str(int(time.time()))*

*##r = requests.post(url, data=data, headers=headers)*

*def on\_connect(client, userdata, flags, rc):*

*print("Connected with result code "+str(rc))*

*##print("name of client is " + client)*

*client.subscribe("topic/temp")*

*##client.subscribe("topic/ph")*

*def on\_message(client, userdata, msg):*

*print("received temp" + msg.payload.decode())*

*##print(userdata.decode())*

*##print(msg.payload.decode())*

*data = "temp,sensor=sensor1 temp\_value=" + msg.payload.decode() + " "+ str(int(time.time()))*

*r = requests.post(url, data=data, headers=headers)*

*client = mqtt.Client()*

*client.connect("localhost",1883,60)*

*client.on\_connect = on\_connect*

*client.on\_message = on\_message*

*client.loop\_forever()*