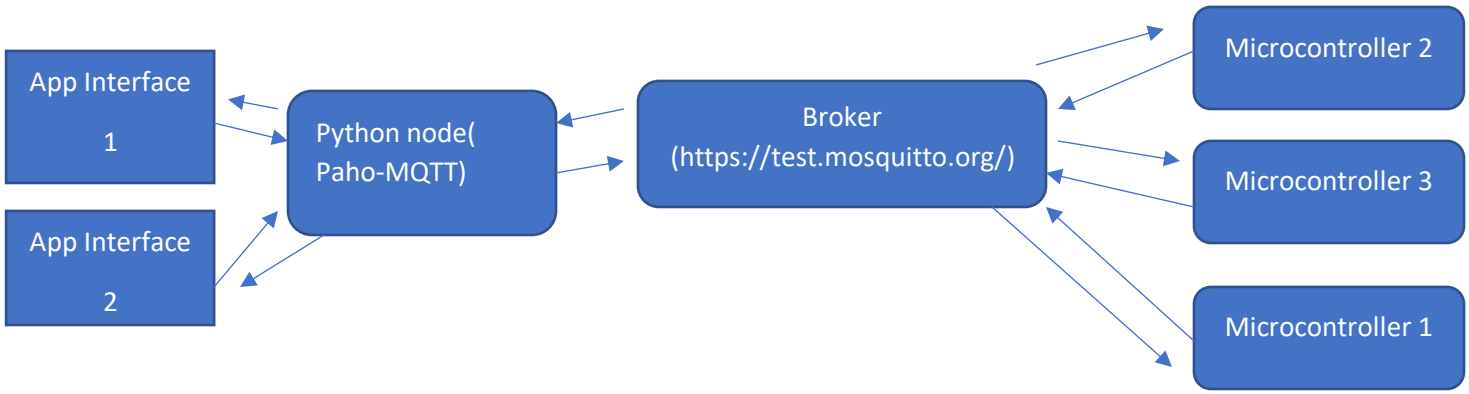
Goal of this description:

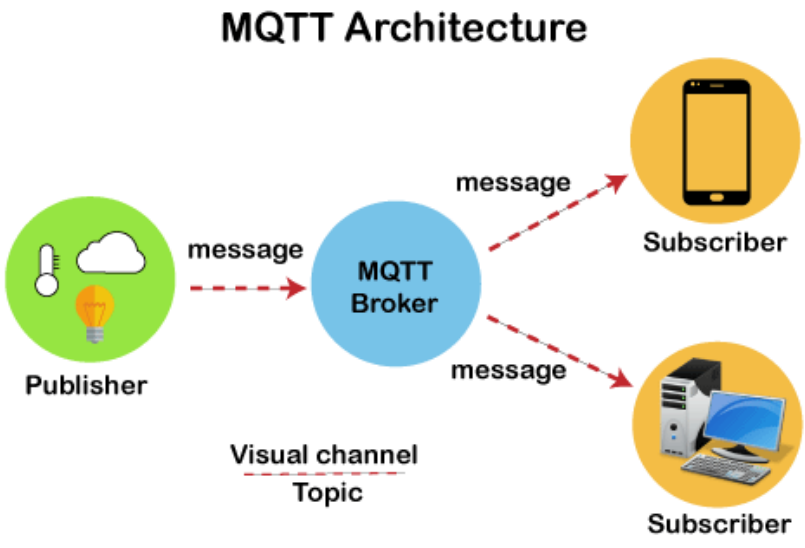
- ➔ Introduce you to the back end architecture that was used in the smart monitor application using python and MQTT, this architect is very powerful due to its ability to be connect with any microcontroller and the software can be a website, phone app, desktop app and more electronic devices.



- ➔ As it show the architecture the app interface can be a Phone Application or Desktop application, also may be a web application, and the microcontroller can be anything can connect to the MQTT broker
- ➔ The essential part of this architect is the python node which is developed using paho-mqtt library,

Which can publish and subscribe to the broker

An Overview over the MQTT protocol:



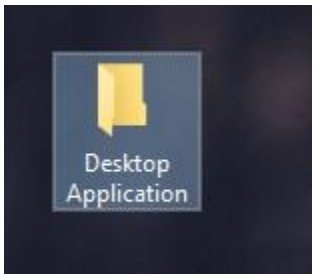
In our case we used developed a Desktop application using PyQt5 and Qt designer with python:

Let us take for example a simple app like this:

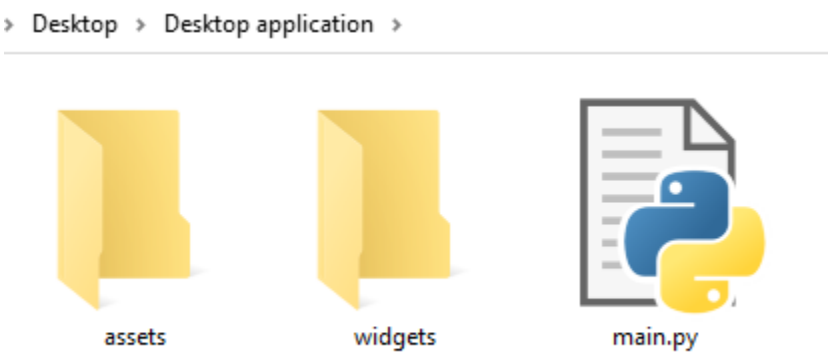
I. Design The Desktop application :

1-> Download Qt Designer from this link: https://build-system.fman.io/qt-designer-download

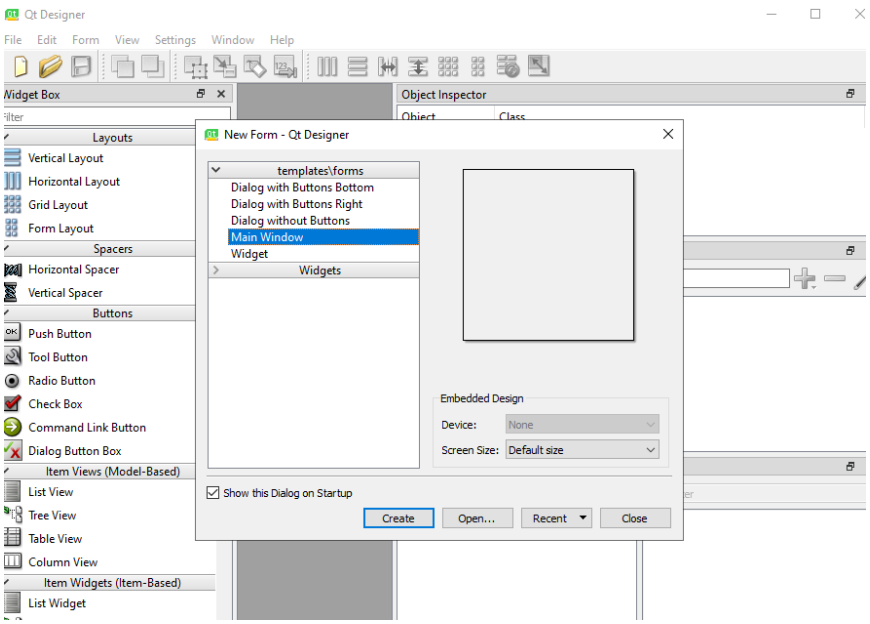
2-> Create a folder for the application:



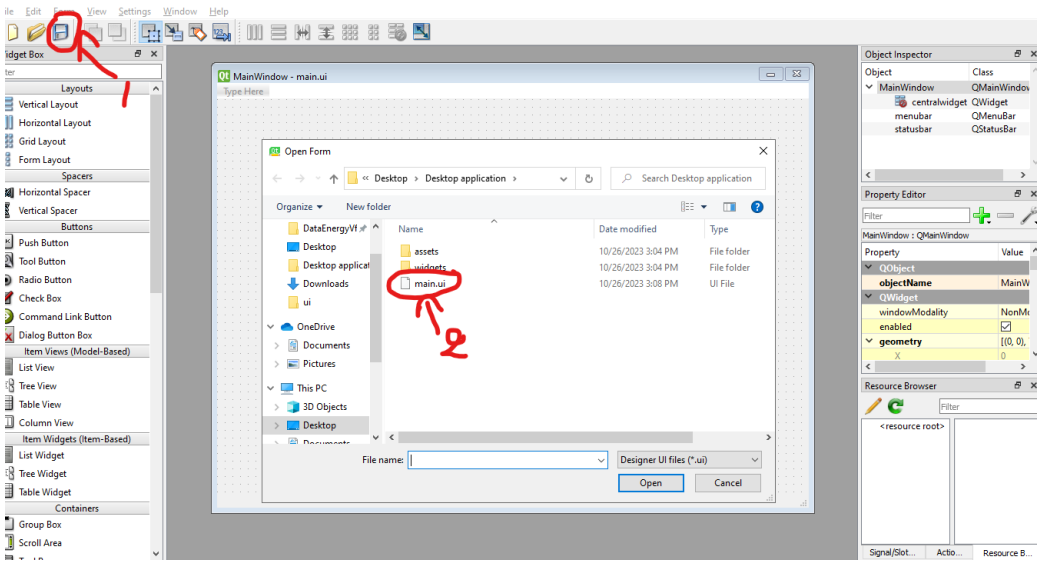
Inside the folder, create subfolders as shown in the figure:



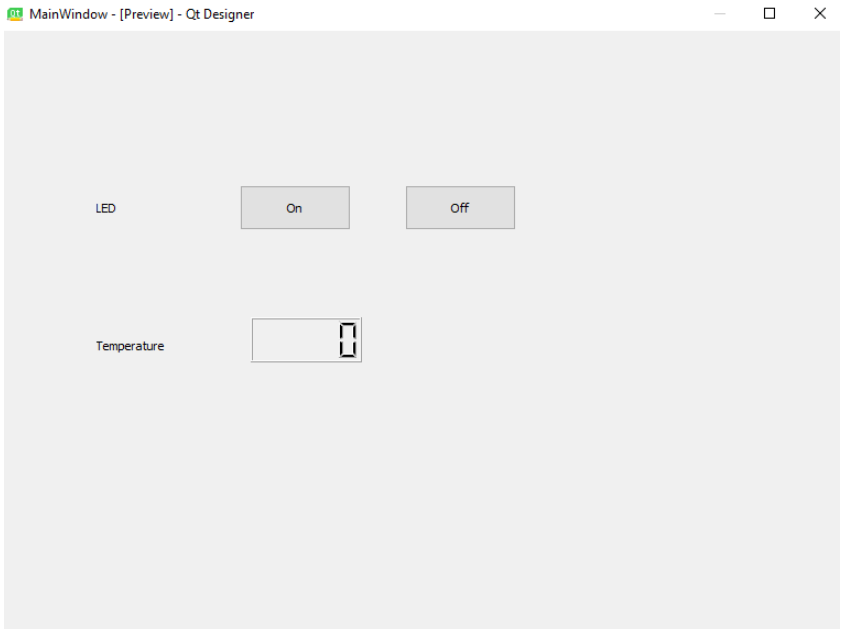
3-> Execute the Qt Designer:



➔ Chose main window and click create, after you create it save it in the same Desktop application folder as main.ui :

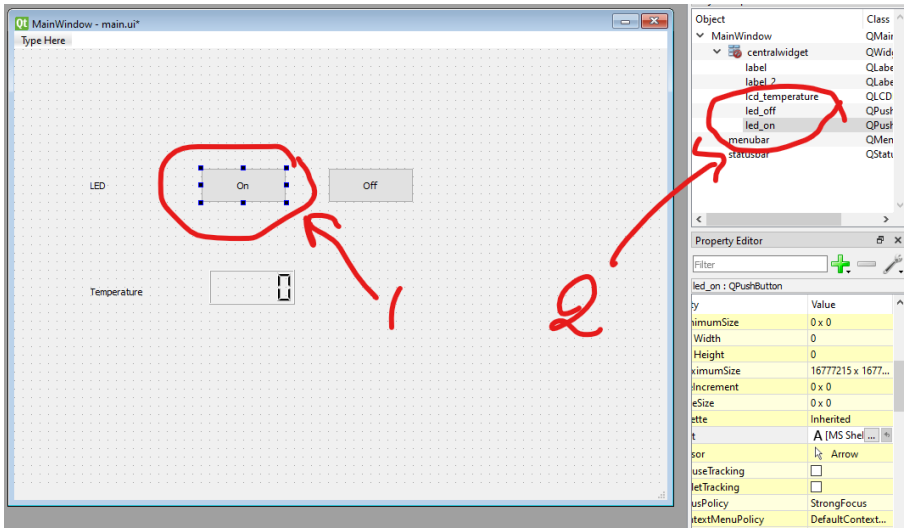


➔ So now we design a simple Interface to control a LED and measure a temperature value :



➔ To design this app just drag and drop from widget box to the main window a Label and push buttons and LCD number.

➔ Rename the push buttons and the LCD Number to



➔ Led_Off – Led_on – lcd_temperature: those names we are going to use them later in the python code so make sure to write them correct.

➔ After you finish make sure to save the design in the Desktop application folder.

II. Code the back end application :

➔ Pyqt5 may seems in the first a little too complex to develop an application with it but its very simple if you have a good knowledge coding with Python and object oriented programming,

➔ It's very hard to show you all the functionality of how to use it, but I'm going to give you the basics to develop our simple interface

➔ If you want to learn more about Pyqt5 here is a link to a book :

https://drive.google.com/drive/u/0/folders/1Wqzr63Ond_AYuXCMfpxKgO5cKTUtmrT9

main.py

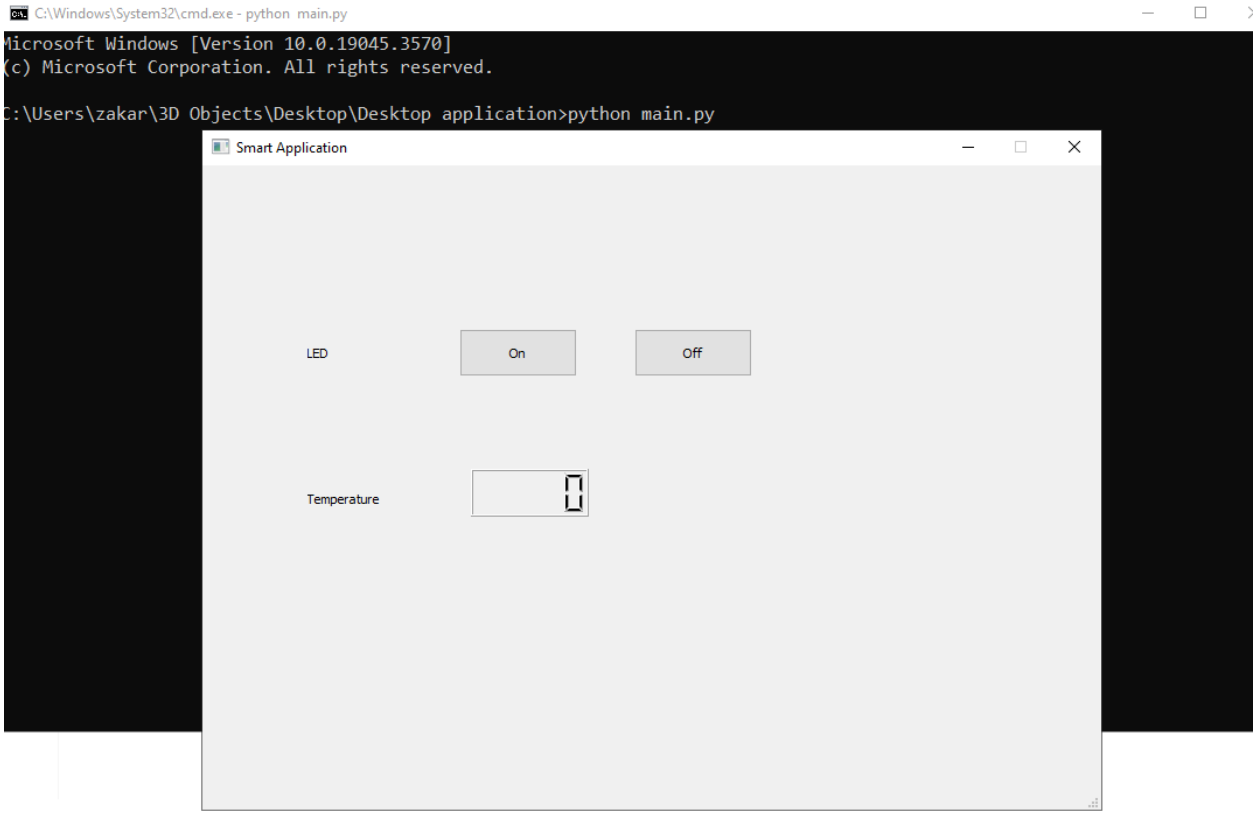
```python
# import library
from PyQt5.QtWidgets import QApplication, QFileDialog, QMainWindow, QMessageBox
from PyQt5 import uic,QtGui
import sys

# initialize the MainWindow of our app
class Main(QMainWindow):
    def __init__(self):
        # call init of QMainWindow
        super().__init__()
        # load our design from Qtdesigner
        uic.loadUi('./main.ui', self)
        # set up our app functionality
        self.set_ui()
        self.buttons()

    def set_ui(self):
        self.setWindowTitle('Smart Application') #This is the title of our App
        self.setFixedSize(770,552)  # This is the size of our Application in the System

    #This is the function where we goanna set the trigged of our push Buttons
    def buttons(self):
        pass

```

```python
# if code run from main
if __name__ == '__main__':
    # create app in sys
    app = QApplication(sys.argv)
    # load our MainWindow
    main = Main()
    # show the window
    main.show()

     # except end process
    try:
        sys.exit(app.exec_())
    except SystemExit:
        print('Clossing window...')

```

You need to download the necessary Libraries in your operating system to run the code successfully:

- ➔ pip install PyQt5
- ➔ pip install pyqt5-tools
- ➔ pip install pyqt5-plugins

- ➔ if you did everything exactly till, run the main.py code from the CMD, you should see the application executed :



➔ After that insert the code bellow in the main.py, you're goanna find the whole application the my GitHub if you want to copy and paste it, all the lines are with comments description :

main.py

```python
#Import the paho Library :
import paho.mqtt.client as paho
#this is a thread responsible of the connection with the broker
class MqttApp(QThread):
    temperature = pyqtSignal(str) # this is the signal that goanna recieve the value of the temperature

    # Function of handling the Mqtt parameters necessary for the broker
    def run(self):
        self.client = paho.Client() # create a client for the broker
        self.client.on_connect = self.on_connect # function when connection
        self.client.on_message = self.on_message
        self.client.on_publish = self.on_publish
        self.client.connect("test.mosquitto.org", 1883, 60)  # we connect to the mosquitto broker
        self.client.loop_forever()

    # when connect success to the broker, subsciber to the topic
    def on_connect(self, client, userdata, flags, rc):
        # this is the topic where the microntorller's goanna send the data
        self.client.subscribe("SmartApplication/temperature")

    # when the value's goanna received we goanna stock it on the temperature signal
    # the from the signal we goanna show it on the main window LCD Number
    def on_message(self, client, userdata, msg):
        topic, message = msg.topic, msg.payload.decode("utf-8")
        print(topic + " -> " + str(message))
        if topic == "SmartApplication/temperature":
            self.temperature.emit(str(message))

    # This is the function when we goanna send the commands of the LED to turn on or Off
    def publish_msg(self, topic, message):
        ret = self.client.publish(topic, message)
    def on_publish(self, client, userdata, result):
            print("data published")
```

Moreover, this code also which is responsible of the execution of the MQTT thread:

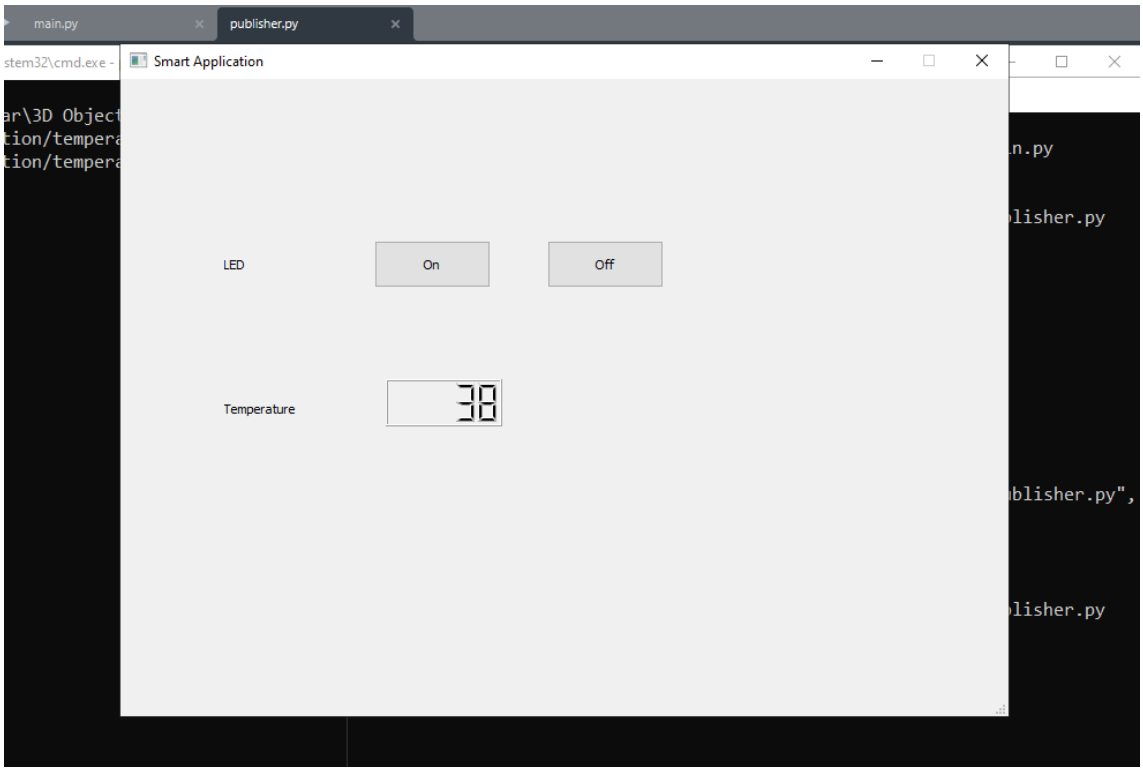
```python
# initialize the MainWindow of our app
class Main(QMainWindow):
    def __init__(self):
        # call init of QMainWindow
        super().__init__()
        # load our design from Qtdesigner
        uic.loadUi('./main.ui', self)
        # set up our app functionality
        self.set_ui()
        self.buttons()
        self.start_subscribing() # calling the mqtt thread function

    # those two functions are responsible of handling the temperature signal
    # to show in the main window, it takes the value from the MQTT thread then it rewrite it
    # in the main window and then to the LCD Number widget box
    def start_subscribing(self):
        self.thread = MqttApp()
        self.thread.temp.connect(self.set_temp) # receiving the temperature value
        self.thread.start()
    def set_temp(self, temp):
        self.lcd_temperature.display(temp) # this line responsible of LCD Number change value
```

➔ Now we completed successfully the Desktop application to test it we're going to run a publisher.py script for sending a fake temperature value in the same topic to test the application :

main.py | publisher.py

```python
import paho.mqtt.client as paho
from time import sleep

broker = "test.mosquitto.org"
port = 1883


def on_publish(client, userdata, result):  # create function for callback
    print("data published \n")


client1 = paho.Client("control1")  # create client object
client1.on_publish = on_publish  # assign function to callback
client1.connect(broker, port)

while True:
    ret = client1.publish("SmartApplication/temperature", 37)
    sleep(4)
    ret = client1.publish("SmartApplication/temperature", 38)
    sleep(4)
    ret = client1.publish("SmartApplication/temperature", 35)
    sleep(4)

```

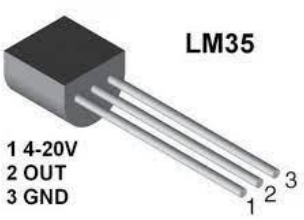
➔ Run the application main.py and the publisher.py in the same time, If everything was right we except to see the results :



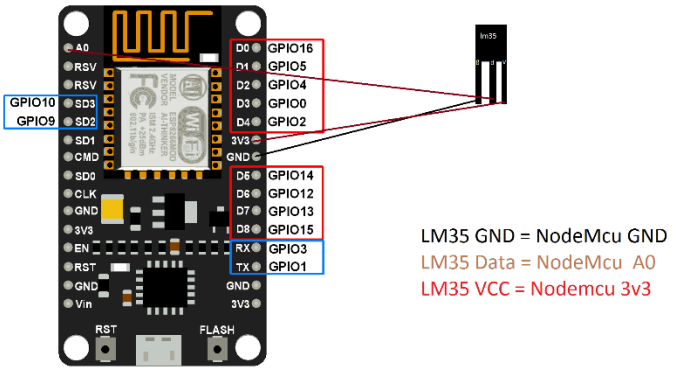
IV. Code the Microcontroller :

In our prototype, we are going to use esp8266 as a microcontroller, which will receive the commands for the LED and send the temperature value:

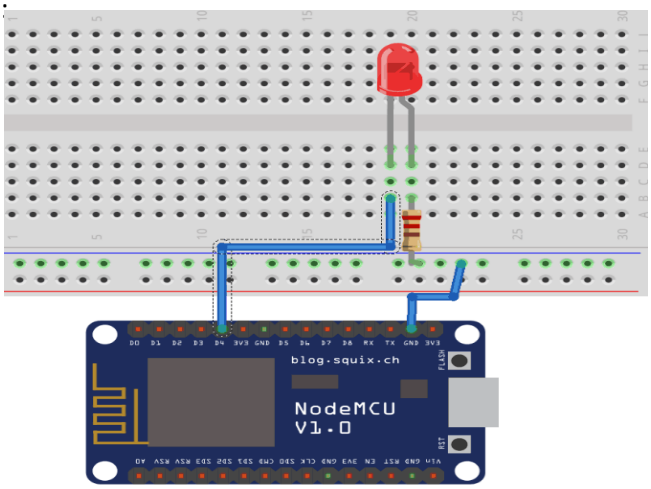
➔ About the Sensors we're going to use LM35 sensor and a simple LED with 220 ohm resistance



➔ The LM35 with the ESP8266 montage wiil be as shown in the figure:



➔ And The LED on series with the resistance 220 ohms and connected to D4 pin as shown in the figure below :



➔ First Let's write a function responsible of Reading the temperature Value and return its Value on Celsius :

esp1 §

```
7
8  #define LM35_pin A0
9
10 // This Function take the LM35 pin which is A0
11 // And Calculate the Temperature in Celisus
12 int get_temperature(int sensor_pin){
13     int analogValue  = analogRead(sensor_pin);
14     millivolts = (analogValue / 1024.0) * 3300; // 3300 is the voltage provided by NodeMCU ESP8266
15     celsius = millivolts / 10;
16     return celsius;
17 }
```

➔ About the LED we're going to Turn it On whenever when we receive the 1 message on the LED MQTT Topic with the instruction :

```
// Command the LED
switch (msg_led){
  case '0':
    digitalWrite(LED, LOW);
    break;
  case '1':
    digitalWrite(LED, HIGH);
    break;
  }
```

➔ Now to connect the ESP With the MQTT Broker and be able to send and receive data we need a few library's we're going to use :

esp1 §

```
1  // Neccessary Librarys
2  #include <ESP8266WiFi.h>
3  #include <PubSubClient.h>
```

➔ To download the PubSubClient Library :
https://www.arduino.cc/reference/en/libraries/pubsubclient/

➔ Before the Setup function we're going to initialize some variables that we're going to need in our code as the Network Wife parameters and the MQTT Brokers :

esp1 §

```
1  // Neccessary Librarys
2  #include <ESP8266WiFi.h>
3  #include <PubSubClient.h>
4
5  // WiFi
6  const char *ssid = "wifi_name";     // Enter your WiFi name
7  const char *password = "wifi_password"; // Enter WiFi password
8
9  // MQTT Broker
10 const char *mqtt_broker = "test.mosquitto.org";
11 const char *topic = "SmartApplication/LED";
12 const char *mqtt_username = "";
13 const char *mqtt_password = "";
14 const int mqtt_port = 1883;
15
16
17 // Create esp and Wifi Objects
18 WiFiClient espClient;
19 PubSubClient client(espClient);
```

➔ After That we're going to initialize the variables we're goanna use in for transferring data :

```
// Initializing the Variables
#define LED D3
#define LM35_pin A0

// This is the global Variable responsible of Reciving led Commande
char msg_led;

// Initializing the variables neccessary for reading the Temperature Value
int analogValue = analogRead(LM35_pin);
float millivolts = (analogValue / 1024.0) * 3300; // 3300 is the voltage provided by NodeMCU
float celsius = millivolts / 10;
char msg_out[20];  // this is the string that we're going to publish

```

➔ Now for the Setup Function :

➔ Normal Initializing :

```
void setup() {

  #Initializing the Pins
  pinMode(LED, OUTPUT);
  pinMode(LM35_pin, INPUT);
  // Initializing the LED on
  digitalWrite(LED, HIGH);

  // Set software serial baud to 9600;
  Serial.begin(9600);

```

➔ wife Connection :

```

  // connecting to a WiFi network
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to the WiFi network");
  delay(500);
  // End Wifi Connection
```

➔ Now The Connection to the MQTT Broker :

```
58  // connecting to a mqtt broker
59  client.setServer(mqtt_broker, mqtt_port);
60  client.setCallback(callback);
61  while (!client.connected()){
62    String client_id = "esp8266-client-";
63    client_id += String(WiFi.macAddress());
64    Serial.printf("The client %s connects to the public mqtt broker\n", client_id.c_str());
65    if (client.connect(client_id.c_str(), mqtt_username, mqtt_password)){
66      // print message
67      Serial.println("Public mosquitoo mqtt broker connected");
68      }
69    else{
70      Serial.print("failed with state ");
71      Serial.print(client.state());
72      delay(2000);
73      }
74  } // End MQTT Connection
75
```

➔ Now Finally in our Setup Function we're going to subscriber to the only topic we need :

```
76    // subscribe to the LED topic
77    client.subscribe(topic);
78    delay(2000);
79  }
```

➔ Now the Turn for the loop Function which is goanna be very short, first we need an instruction which is goanna execute the mqtt functionality in an infinite loop to keep receiving the messages published by the application :

```
void loop() {
  client.loop();  // client mqtt loop forever
```

➔ After that we're going to read the temperature value and publish it :

```
void loop() {
  client.loop();  // client mqtt loop forever
  analogValue = analogRead(LM35_pin);
  millivolts = (analogValue / 1024.0) * 3300; // 3300 is the voltage provided by NodeMCU
  celsius = millivolts / 10;
  sprintf(msg_out, "%2.f", celsius); // Formatting the temperature value to char
  Serial.print("Temp : ");
  Serial.println(msg_out);
  // Publishe the Temperature data to the broker
  client.publish("SmartApplication/temperature", msg_out);
}
```

➔ Now the Final part is the callback function which is run whenever a message received in the topic we subscribe to it :

```
// Function Called when a message arrived
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]);
     msg_led = (char)payload[i]; // Saving the Reciving Command of the LED
  }
```

➔ Due to that the message we're going to receive is just a one character we save it in the msg_led variable and we write a switch block in the same callback function to command the LED :

```
// Function Called when a message arrived
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]);
     msg_led = (char)payload[i]; // Saving the Reciving Command of the LED }

  // Command the LED
  switch (msg_led){
    case '0':
      digitalWrite(LED, LOW);
      break;
    case '1':
      digitalWrite(LED, HIGH);
      break;
   }
  Serial.println();
  Serial.println("-----------------------");
}
```

➔ Finally you're going to find all the code in The GitHub repository in the Link bellow :

https://github.com/zakariarhiba/MQTT-Python-Architecture

➔ If you did everything's right, you should Run everything in the same time and enjoy the real time data transferring.