TECHNICAL DOCUMENT

WEATHER STATION WITH MICROPYTHON

TEMPERATURE SIGNAL

Authors

Israt Jahan sumiya

Contents

Contents

Figures

1. INTRODUCTION

Weather describes the condition of the air at a particular time and place. Weather also tells how the air moves (wind), and it might be carrying something such as rain, snows, or clouds. Thunder, lighting, rainbows, snows storm and other special events are also part of weather. Studying about weather is important because it affects our activities. People must be aware of daily weather so that they can plan their daily things well.

The Weather Station is one of the most significant projects for the students of TAMK. We aim to collect most accurate weather information. TAMK has a high-end sensor system on the rooftop to capture all types of weather data like temperature, light, rain, humidity etc. In previous, this project was implemented with C++.But this year our goal is to use python for low consumption with great capabilities. In IOT, python is great choice for the backend side of development as well as software development of devices. There are low system requirements especially for micropython than the C++. This Technical document will guide users how the software process sensor data from the ESP32 board to database.

2. USED COMPONENTS AND TOOLS

Different components and tools are used for this project. Here is the list:

- Hardware Components
    - ESP32(Wemos Lolin32)
    - Breadboard
    - Some wires
    - Resistor 120 ohm and 220 ohms
    - TAMK's rooftop Sensor
    - An USB and ethernet cable
    - Raspberry Pi

- Tools
    - Thonny IDE for using micropython.
    - Esptool for deploying micropython to our esp32 development board.
    - Python3 ruining for Esptool.
    - Advanced IP Scanner for checking raspberry pi IP address.
    - VNC Viewer for Raspberry pi.

3. INSTALLATION AND SETUP

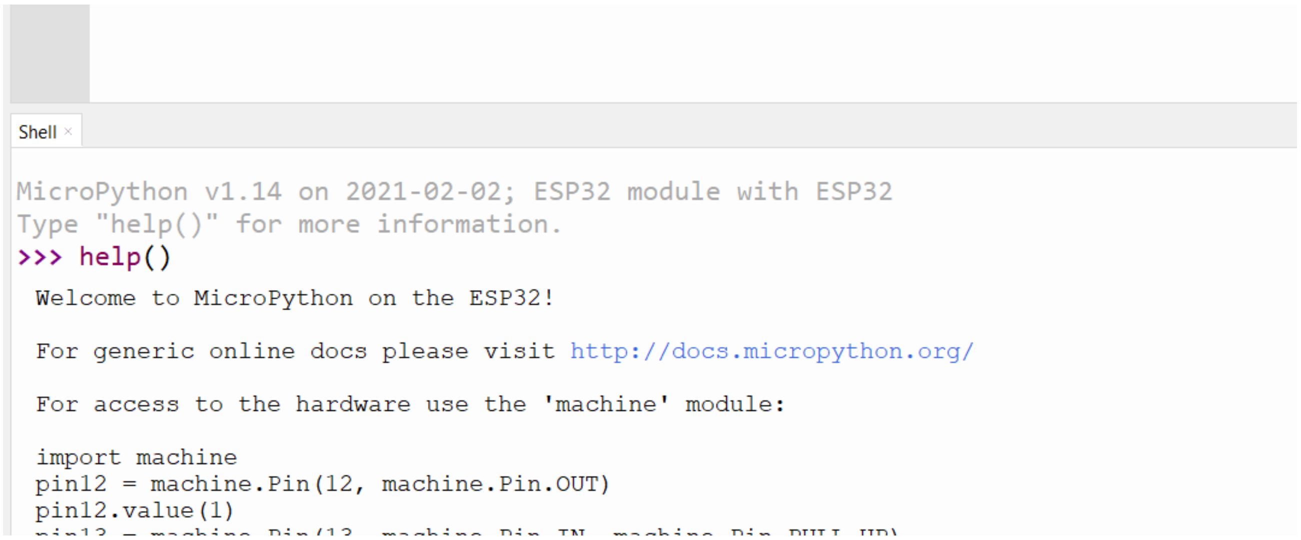
Following all the installation and setup was done in windows operating system.

Python:

1. Download link for python3**:** https://www.python.org/downloads/
2. Install Python.
3. Install a build-in python mqtt-client: Go to the command prompt and type "**pip install paho-mqtt**".

Micropython and Thonny:

1. Download Thonny IDE installation file: https://thonny.org/
2. Run the execution file and install Thonny.
3. Download ESP32 Micropython firmware: https://micropython.org/download/esp32/
4. Download driver for the device port: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers
5. Plugin ESP32 to laptop using micro-USB cable.
6. Lunch Thonny and go to **tools -> Manage plug-ins**. Then type **stool** in the search bar and click "**find package in PyPI**". Then install it.
7. Next navigate to **"Run -> select interpreter"** and select the device to "**MicroPython (ESP32)**". Then in the port option select the driver execution file. Click the install or update firmware. It will open a dialogue. Select the same port. For the firmware browse the downloaded firmware.
8. Make sure to click the checkbox of "**Erase flash before installing**". Then click the install.
9. Close all the dialogue tab. In the bottom of the Thonny it should show an icon with three greater than symbol. Try to write "**help()**" and hit enter. If it responds back, it is successfully installed.



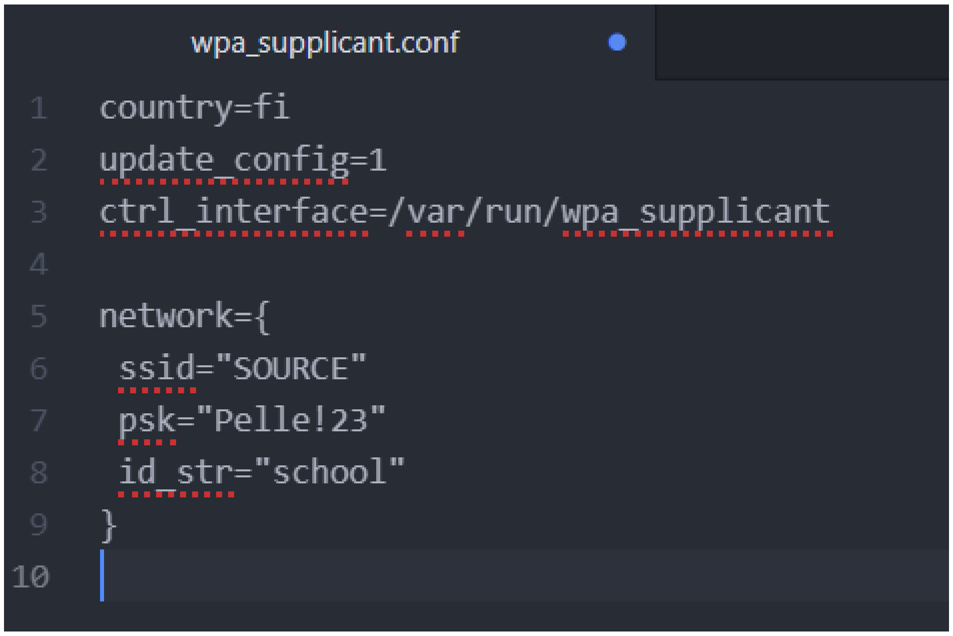
```
MicroPython v1.14 on 2021-02-02; ESP32 module with ESP32
Type "help()" for more information.
>>> help()
 Welcome to MicroPython on the ESP32!

 For generic online docs please visit http://docs.micropython.org/

 For access to the hardware use the 'machine' module:

 import machine
 pin12 = machine.Pin(12, machine.Pin.OUT)
 pin12.value(1)
```

Figure 1: Thonny installation check-up

Raspberry pi 4:

Following this instruction, you will be able to connect raspberry pi to your laptop:

1. Download and install:
    - SD Card formatter: https://www.sdcard.org/downloads/formatter/
    - putty: https://www.putty.org/
    - Advanced IP Scanner: https://www.advanced-ip-scanner.com/
    - Atom: https://atom.io/
2. Download Raspberry pi imager: https://www.raspberrypi.org/software/
3. Insert an 8GB or greater capacity SD card into micro-SD card reader and plug-in to laptop. Open SD Card formatter and format SD card.
4. Open the imager execution file. Install it. It will open a dialogue. Select the operating system. For this project **raspberry pi OS (32-bit)** was used. When you will try to choose the storage, it will automatically detect your SD card. Select it and press "**WRITE**" and then **yes** to continue.
5. Remove the card reader and plug-in again.
6. Open the SD card and create an empty file named "**ssh**" without any extension in the boot partition. When PI boot it looks for this file. Once ssh file is found, it enables ssh and then deletes it automatically. SSH can be also enable from **raspi-config** but for this you need HDMI monitor, a mouse and keyboard.
7. To connect with network, add another file named **wpa_supplicant.conf**
   Copy the following code on that file. You can open such file using **atom**. You can replace ssid and psk (password) for your home network.

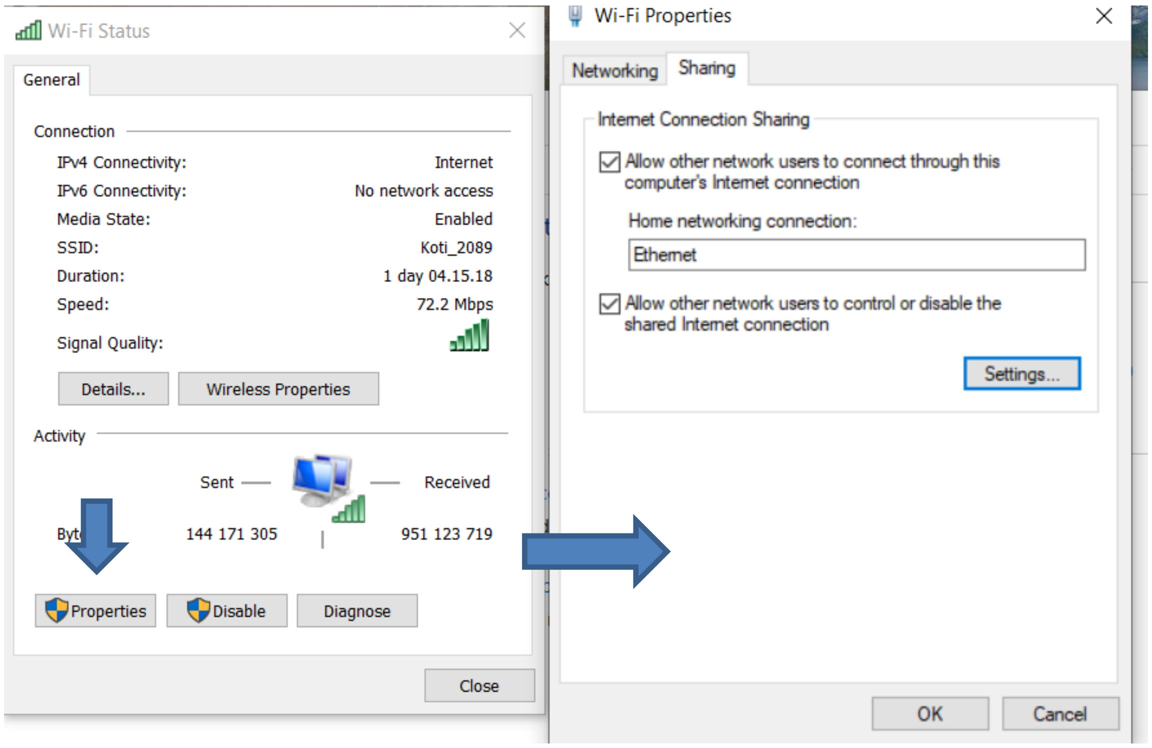


```
country=fi
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
 ssid="SOURCE"
 psk="Pelle!23"
 id_str="school"
}
```

Figure 2: Add wireless network to raspberry-pi.

8. Right click the SD card and eject it. Remove SD card and insert it into your raspberry pi.

9. In your pc go to **Control Panel->Network and Internet->Network and Sharing Center**. Open your wireless network **connections.** Then select **properties**. On the next window click **sharing** and click the **two checkboxes** of allow the other networks and press **ok**. Unless you do this your raspberry pi will not get the IP address.



IPv4 Connectivity:	Internet
IPv6 Connectivity:	No network access
Media State:	Enabled
SSID:	Koti_2089
Duration:	1 day 04.15.18
Speed:	72.2 Mbps
Signal Quality:	

	Sent	Received
Bytes	144 171 305	951 123 719

Internet Connection Sharing

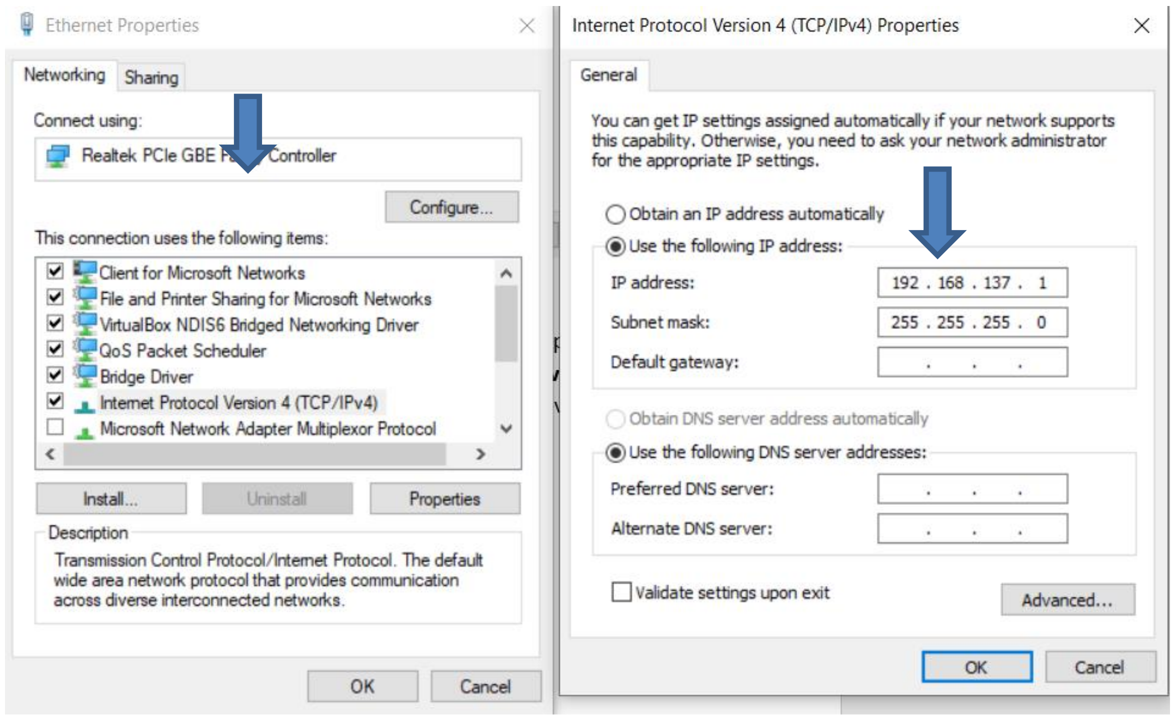
☑ Allow other network users to connect through this computer's Internet connection

Home networking connection: Ethernet

☑ Allow other network users to control or disable the shared Internet connection

Figure 3: Allow other network users to connect.

10. Then connect LAN or ethernet cable from laptop to raspberry pi. Power it up using USB cable or using raspberry pi charger. Now you'll see a new **unidentified network**. Click on the **connections** and go to the **properties**. On networking tab double click on ipv4. You'll see a gateway IP-address is given to the new network. It can be also seen with "**ipconfig /all**" command using command prompt.


Ethernet Properties

Networking | Sharing

Connect using:

Realtek PCIe GBE Fa... Controller

This connection uses the following items:

- [✓] Client for Microsoft Networks
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] VirtualBox NDIS6 Bridged Networking Driver
- [✓] QoS Packet Scheduler
- [✓] Bridge Driver
- [✓] Internet Protocol Version 4 (TCP/IPv4)
- [ ] Microsoft Network Adapter Multiplexor Protocol

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address:]

IP address:	192 . 168 . 137 . 1
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

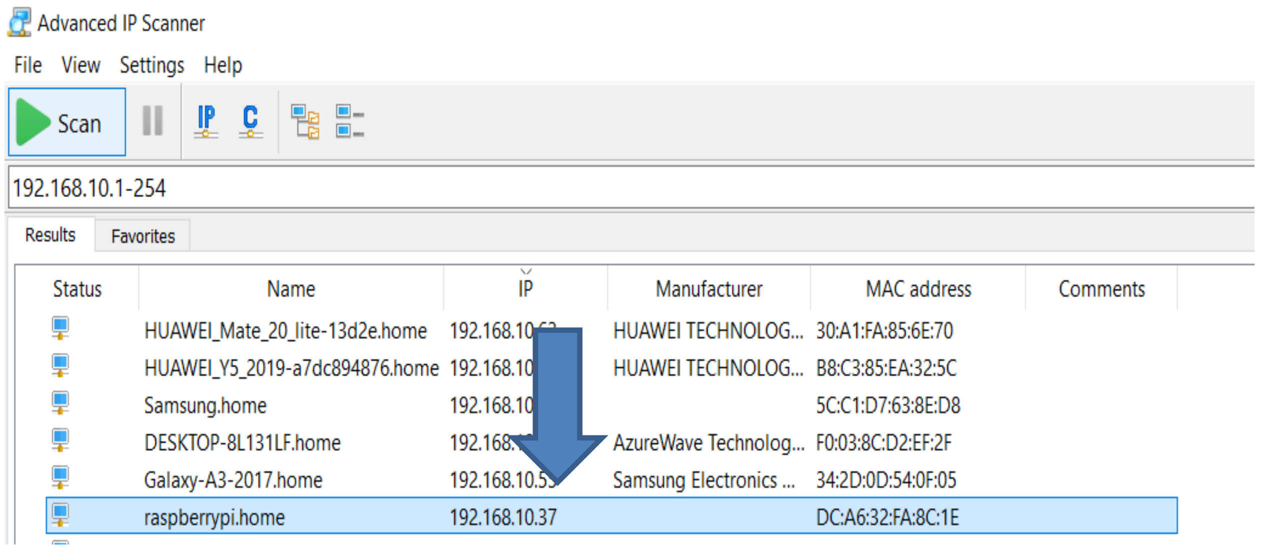
Obtain DNS server address automatically [✓ Use the following DNS server addresses:]

Preferred DNS server:	. . .
Alternate DNS server:	. . .

[ ] Validate settings upon exit

Figure 4: check gateway IP of your new network

11. As the IP is 192.168.137.1 It will in the range of 192.168.137.1-254.Advanced IP Scanner can be used to find the raspberry pi IP.

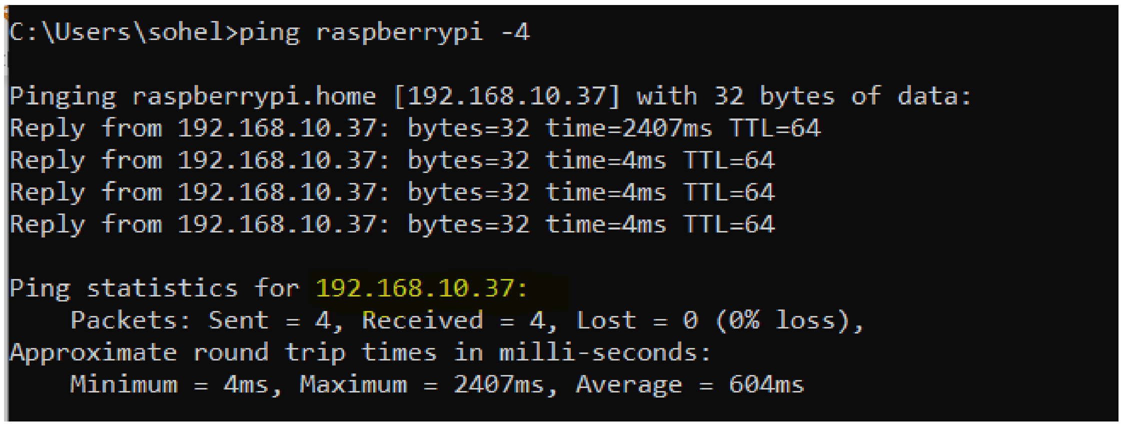

Advanced IP Scanner

192.168.10.1-254

Status	Name	IP	Manufacturer	MAC address	Comments
	HUAWEI_Mate_20_lite-13d2e.home	192.168.10.62	HUAWEI TECHNOLOG...	30:A1:FA:85:6E:70	
	HUAWEI_Y5_2019-a7dc894876.home	192.168.10	HUAWEI TECHNOLOG...	B8:C3:85:EA:32:5C	
	Samsung.home	192.168.10		5C:C1:D7:63:8E:D8	
	DESKTOP-8L131LF.home	192.168.	AzureWave Technolog...	F0:03:8C:D2:EF:2F	
	Galaxy-A3-2017.home	192.168.10.5	Samsung Electronics ...	34:2D:0D:54:0F:05	
	raspberrypi.home	192.168.10.37		DC:A6:32:FA:8C:1E	

Figure 5: check raspberry pi IP of your new network (1)

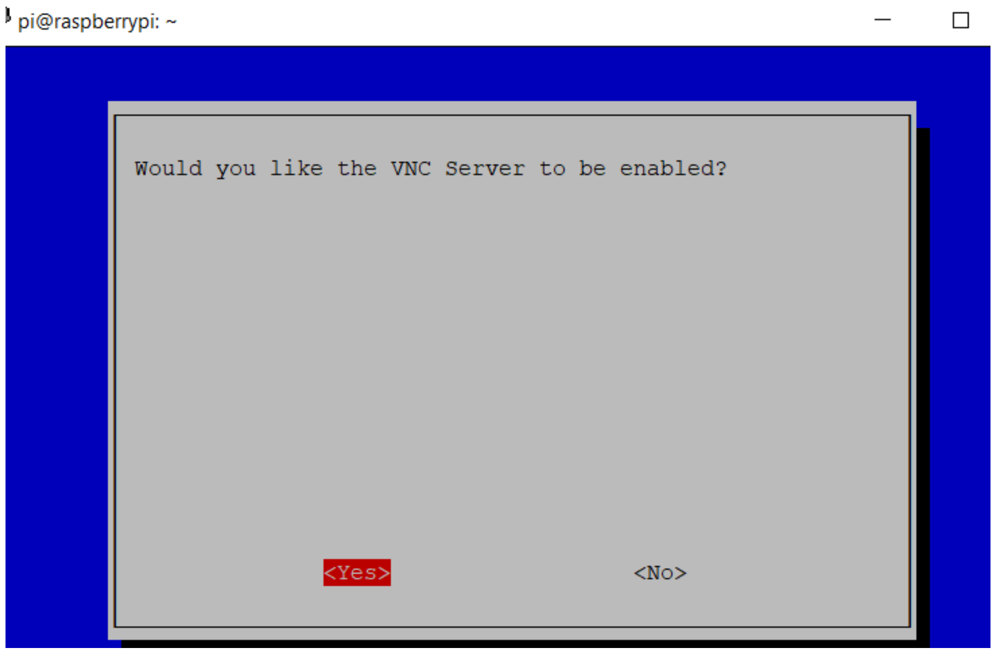
There is another way to find it. Go to command prompt and type "**ping raspberrypi -4"**.



```
C:\Users\sohel>ping raspberrypi -4

Pinging raspberrypi.home [192.168.10.37] with 32 bytes of data:
Reply from 192.168.10.37: bytes=32 time=2407ms TTL=64
Reply from 192.168.10.37: bytes=32 time=4ms TTL=64
Reply from 192.168.10.37: bytes=32 time=4ms TTL=64
Reply from 192.168.10.37: bytes=32 time=4ms TTL=64

Ping statistics for 192.168.10.37:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 2407ms, Average = 604ms
```

Figure 5: check raspberry pi IP of your new network (2)

12. Open **Putty** and login with raspberry pi IP. The default user is **pi** and password is **raspberry**. Use “**sudo raspi-config”** command. It will open a window. It can be controlled by the upper or lower arrow and tab of your keyboard. Select **2. Display option** and hit enter. Set the resolution. We have used **DMT Mode 85 1280x720 60Hz 16:9** and then **ok**. Again, go to **3. Interface option** and enable **VNC.** Now you can use VNC viewer. Basically, we will be using this graphical desktop to write some code. Also, Putty can be used then you will need editor like VIM.



```
pi@raspberrypi: ~

Would you like the VNC Server to be enabled?

<Yes>                    <No>
```

Figure 6: Enable VNC

13. Mosquitto in raspberry:

    Run the following commands to install mosquitto:
    >>**sudo apt install -y mosquitto mosquitto-clients**

    Make sure moasquitto service is enabled:

>> **sudo systemctl enable mosquitto.service**

Check mosquitto is running or not:

>> **ps -ef | grep mosquitto**

Run mosquitto as a deamon
>> **mosquitto -d**

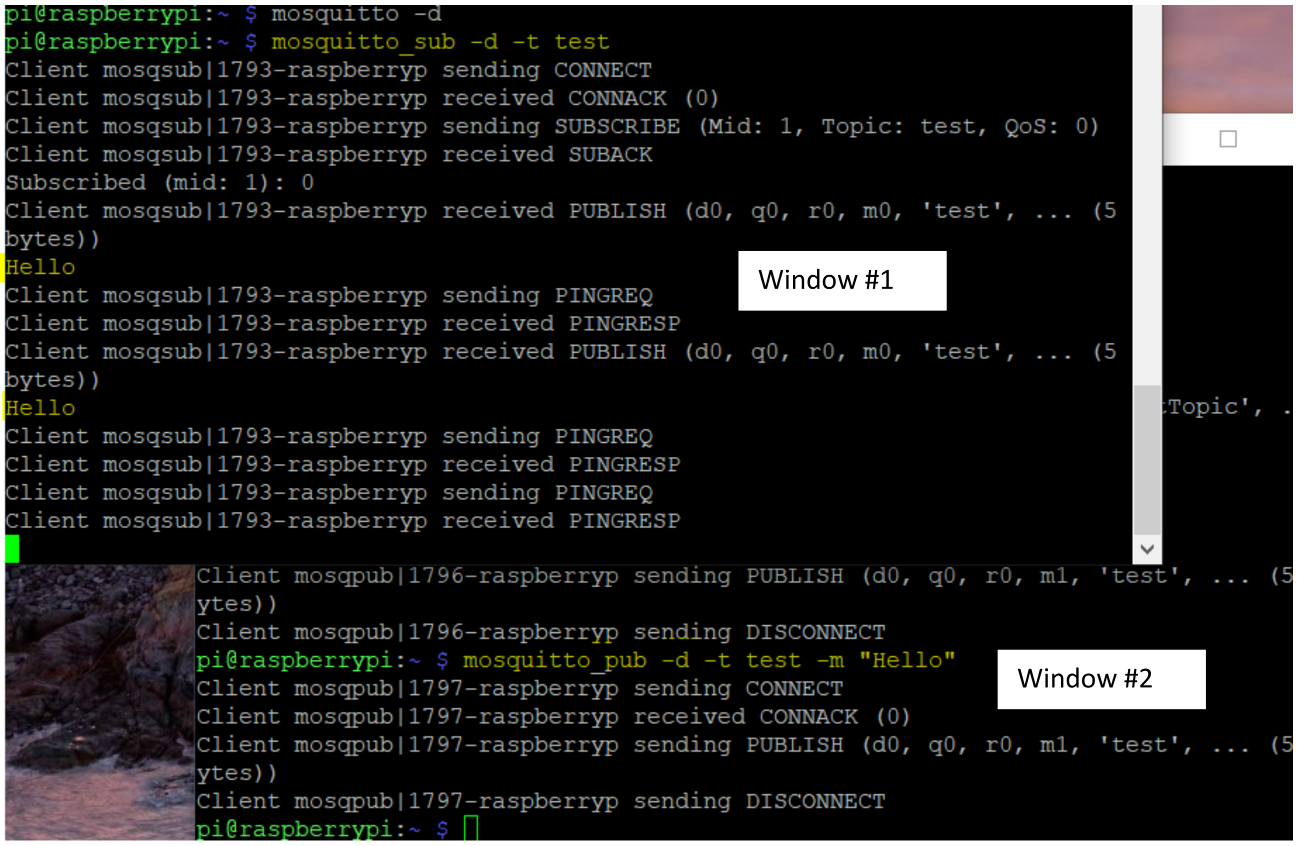
To subscribe to an MQTT topic with mosquitto client open terminal window #1 and give the following command:

pi@raspberry: ~ $ **mosquitto_sub -d -t test**

Now you've subscribed to a topic called test. To publish a massage on this topic, open another window #2. Run this command:

pi@raspberry: ~ $ **mosquitto_pub -d -t testTopic -m "Hello"**

It should show the "hello" message to the other window.



```
pi@raspberrypi:~ $ mosquitto -d
pi@raspberrypi:~ $ mosquitto_sub -d -t test
Client mosqsub|1793-raspberryp sending CONNECT
Client mosqsub|1793-raspberryp received CONNACK (0)
Client mosqsub|1793-raspberryp sending SUBSCRIBE (Mid: 1, Topic: test, QoS: 0)
Client mosqsub|1793-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|1793-raspberryp received PUBLISH (d0, q0, r0, m0, 'test', ... (5
bytes))
Hello
Client mosqsub|1793-raspberryp sending PINGREQ
Client mosqsub|1793-raspberryp received PINGRESP
Client mosqsub|1793-raspberryp received PUBLISH (d0, q0, r0, m0, 'test', ... (5
bytes))
Hello
Client mosqsub|1793-raspberryp sending PINGREQ
Client mosqsub|1793-raspberryp received PINGRESP
Client mosqsub|1793-raspberryp sending PINGREQ
Client mosqsub|1793-raspberryp received PINGRESP
```

```
Client mosqpub|1796-raspberryp sending PUBLISH (d0, q0, r0, m1, 'test', ... (5
ytes))
Client mosqpub|1796-raspberryp sending DISCONNECT
pi@raspberrypi:~ $ mosquitto_pub -d -t test -m "Hello"
Client mosqpub|1797-raspberryp sending CONNECT
Client mosqpub|1797-raspberryp received CONNACK (0)
Client mosqpub|1797-raspberryp sending PUBLISH (d0, q0, r0, m1, 'test', ... (5
ytes))
Client mosqpub|1797-raspberryp sending DISCONNECT
pi@raspberrypi:~ $
```

Figure 7: Sending and receiving massages with moasquitto client.

14. Install python Mqtt client library. Run the following command:
    >> **sudo pip3 install paho-mqtt**

4. SOFTWARE FLOW DIAGRAM

4.1. General diagram

Initially, there are sensors in the rooftop of TAMK. To capture the signals received from sensors, we have used micropython on ESP32. After doing some transformation and conversion of signals. Raspberry Pi acts as an IoT gateway which has MQTT broker plays as an intermediate communicator receiving and sending data forward to database. The idea of how the weather station work is described in the picture below.

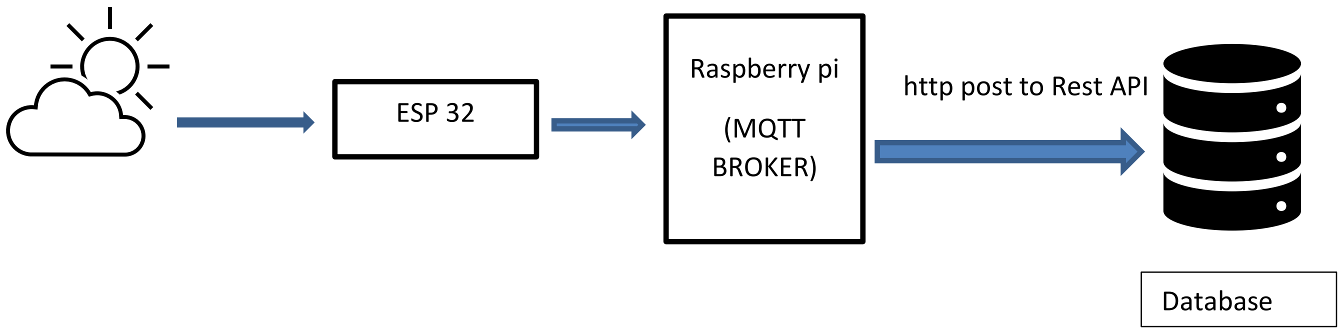


Figure 8: Weather station project

4.2. Circuitry

The board is the main part where can receive and convert the signal before sending to MQTT broker. The board contains: ESP32(Wemos Lolin32), breadboard, some wires and two resistors.

Microcontroller

The Wemos Lolin32 OLED is a development board with ESP32 microcontroller and build-in I2C oled display. To control this display in micropython, we have used ssd1306 library by Adafruit. The code for the library can be found here .Save it to your esp with the name **ssd1306.py**. It features a **BOOT**, **EN** (reset) button and **micro-USB port** at the back. Boot is secure, and flash are encrypted. Micro-USB port used for loading the programs from computer. In micropython, there will be a special purpose file named "**boot.py"**. While installing a fresh instance of micropython firmware on esp32, boot.py will be written in the filesystem automatically. This default file has some code, but it is commented out. So, initially it is inactive. If any code has this file on esp device, it will run automatically next time on each power-up. Basically, we had used this file with the code of connecting to Wi-Fi. Some models are bit different, but all board should work as similar way. This board doesn't have much accessible GPIOS as regular ESP32, but it is still suitable for this small project. Some technical specs of esp32 are given below:

Operating Voltage	3.3volt
Digital IO pins	22
Analogue input pins	6(SVP, SVN,32,33,34,35)
Analogue output pins	2(25,26)
LED build-in	GPIO5
I2C communication	SCL=GPIO4, SDA=GPIO5
Frequency (max)	240MHz
Flash	4MB

The image below shows the pinout of our used device:

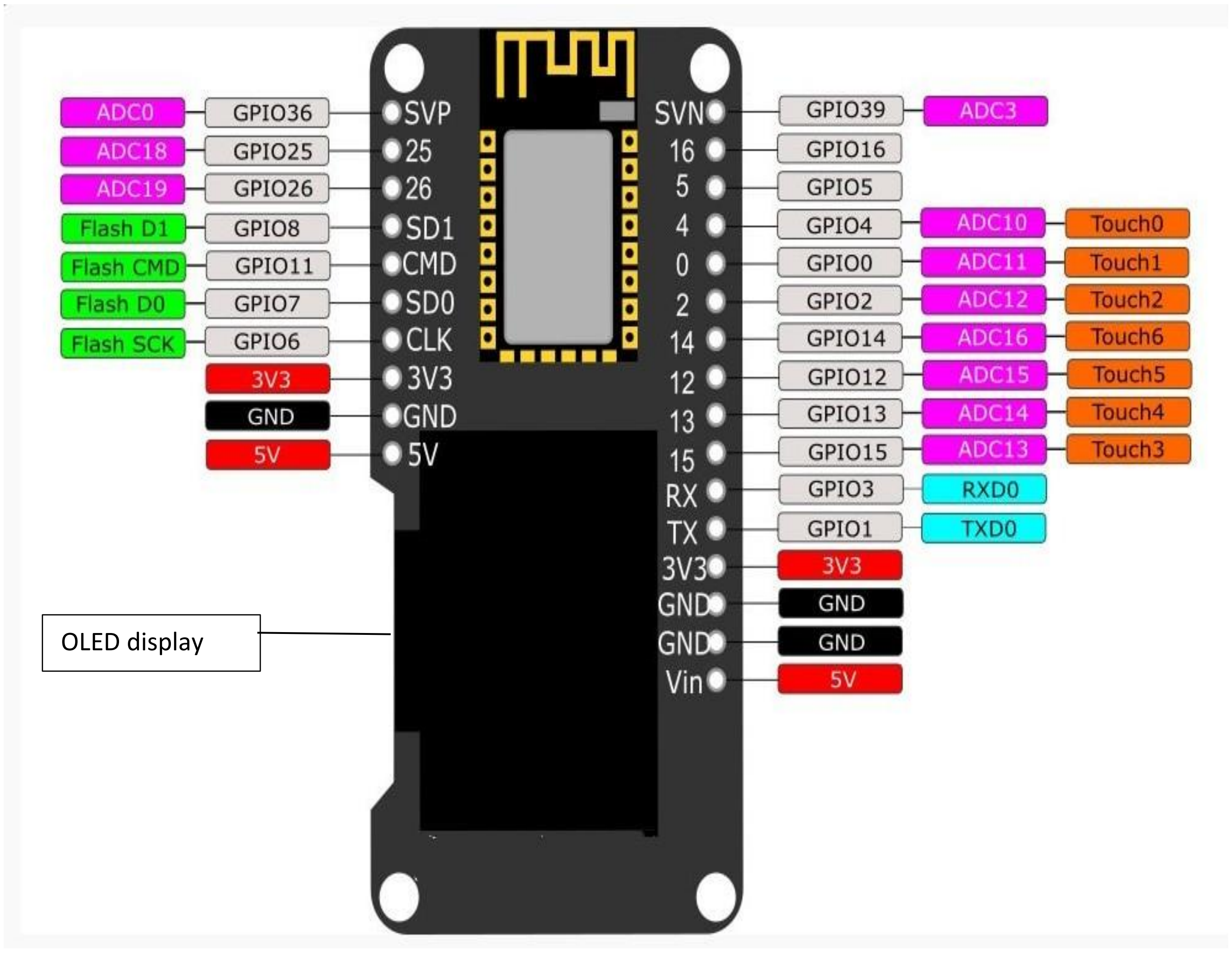
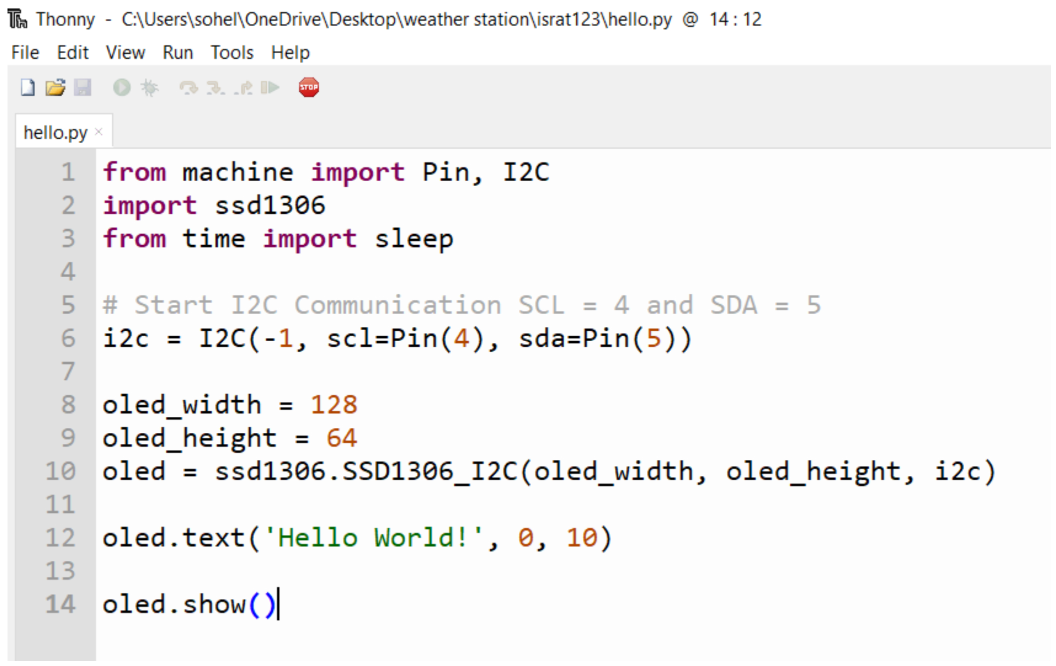


Figure 9: ESP32 WROOM-32 PINOUT (SOURCE: internet)

Example code:

Here is a code that prints **Hello World!** on OLED:


Thonny - C:\Users\sohel\OneDrive\Desktop\weather station\israt123\hello.py @ 14 : 12
File Edit View Run Tools Help
hello.py

```
from machine import Pin, I2C
import ssd1306
from time import sleep

# Start I2C Communication SCL = 4 and SDA = 5
i2c = I2C(-1, scl=Pin(4), sda=Pin(5))

oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

oled.text('Hello World!', 0, 10)

oled.show()
```

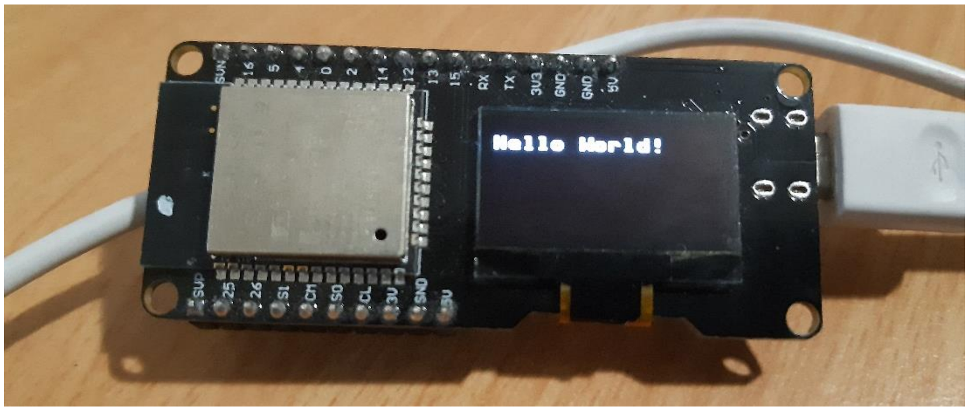


Figure 10: Print message to OLED

Voltage

All the IO pins of esp32 run at 3.3v but our sensor is 5v. So, voltage divider ratio rule can be used for the voltage drop. The formula for calculating output voltage based on ohm's law and is shown below:

$$V_{OUT} = V_S \times R_2 \div R_1 + R_2$$

where, $V_{OUT} = output\ voltage(v)$

$V_s = \ voltage\ source(v)$

$$R_1 = resistance\ of\ first\ resisitor(\Omega)$$

$$R_2 = resistance\ of\ second\ resisitor(\Omega)$$

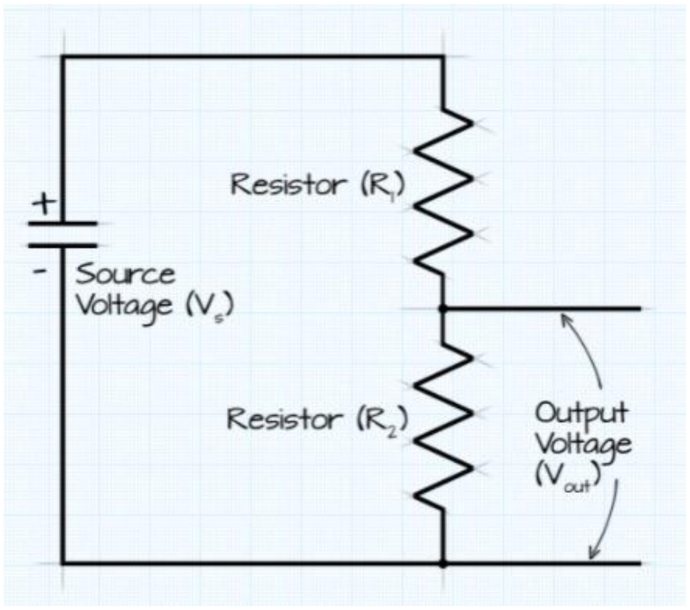


Figure 11: Voltage divider circuit

Example: Our source voltage was 5v, expected output voltage 3.3v and one of resister we choose 120 ohm. Calculating with this formula for other resistor should use around in 232 ohms. So, we choose second one 220v.

To convert ADC value (analogue reading) to DC voltage following formula can be used:

$$V = adc\ value \times V_{ref} \div Resolution$$

Example: If we use 10-bit resolution will be 2^{10}or 1024.In our case Voltage reference is 3.3v

For more details: https://youtu.be/S34R4zg03uE

Circuit (wiring diagram)

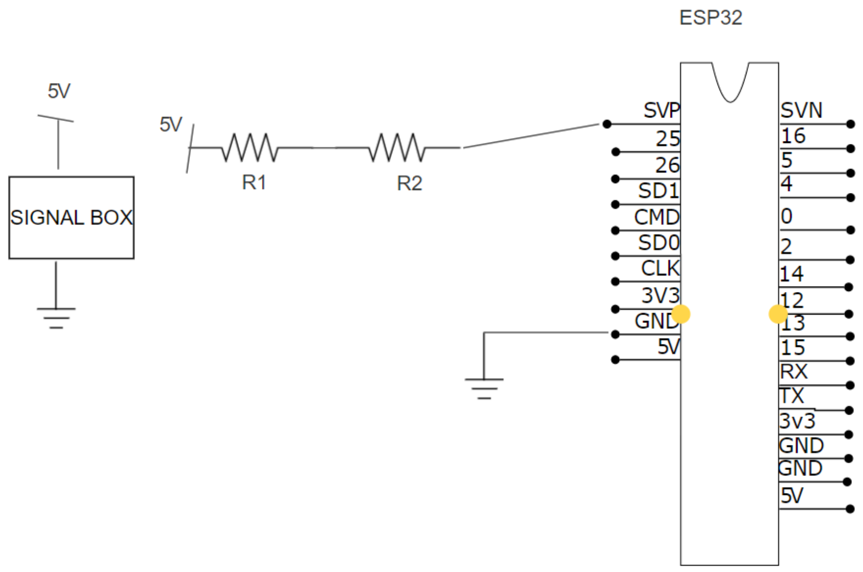
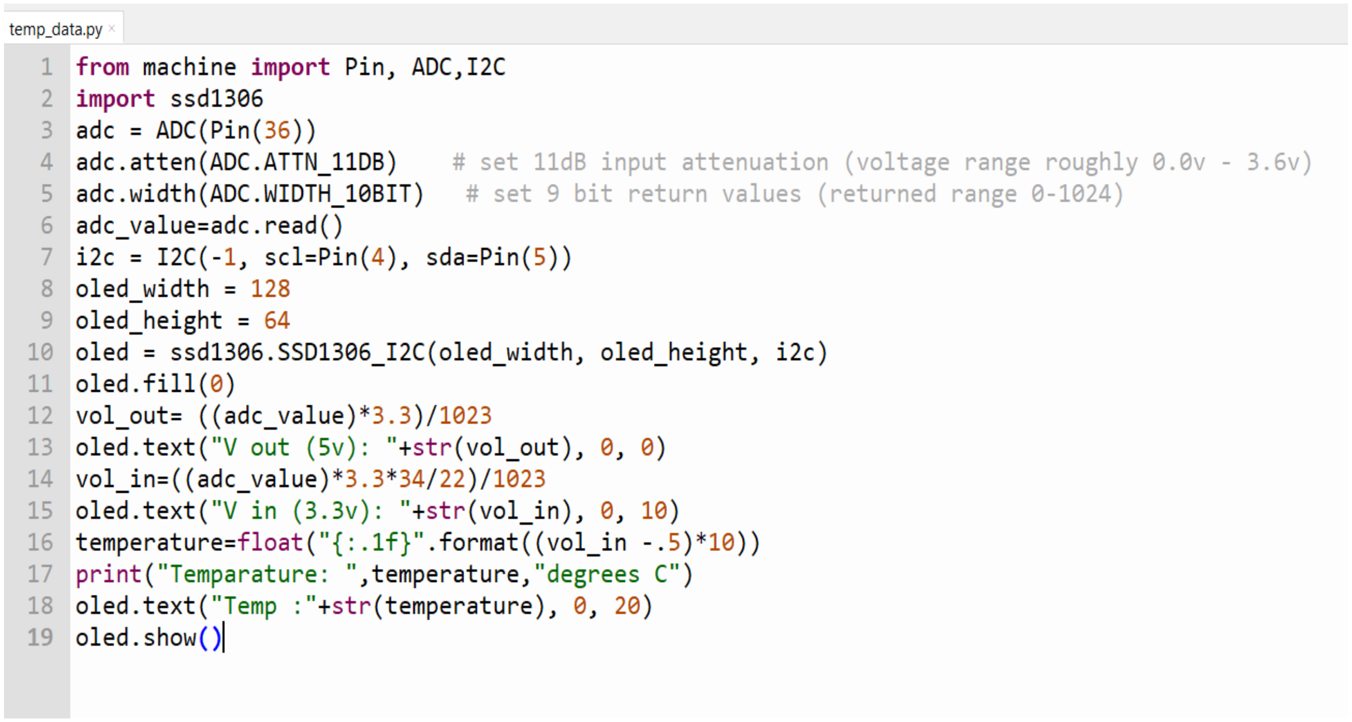


Figure 12: Circuit for temperature sensor

Example Code



```python
from machine import Pin, ADC,I2C
import ssd1306
adc = ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)    # set 11dB input attenuation (voltage range roughly 0.0v - 3.6v)
adc.width(ADC.WIDTH_10BIT)   # set 9 bit return values (returned range 0-1024)
adc_value=adc.read()
i2c = I2C(-1, scl=Pin(4), sda=Pin(5))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)
oled.fill(0)
vol_out= ((adc_value)*3.3)/1023
oled.text("V out (5v): "+str(vol_out), 0, 0)
vol_in=((adc_value)*3.3*34/22)/1023
oled.text("V in (3.3v): "+str(vol_in), 0, 10)
temperature=float("{:.1f}".format((vol_in -.5)*10))
print("Temparature: ",temperature,"degrees C")
oled.text("Temp :"+str(temperature), 0, 20)
oled.show()
```

Output:

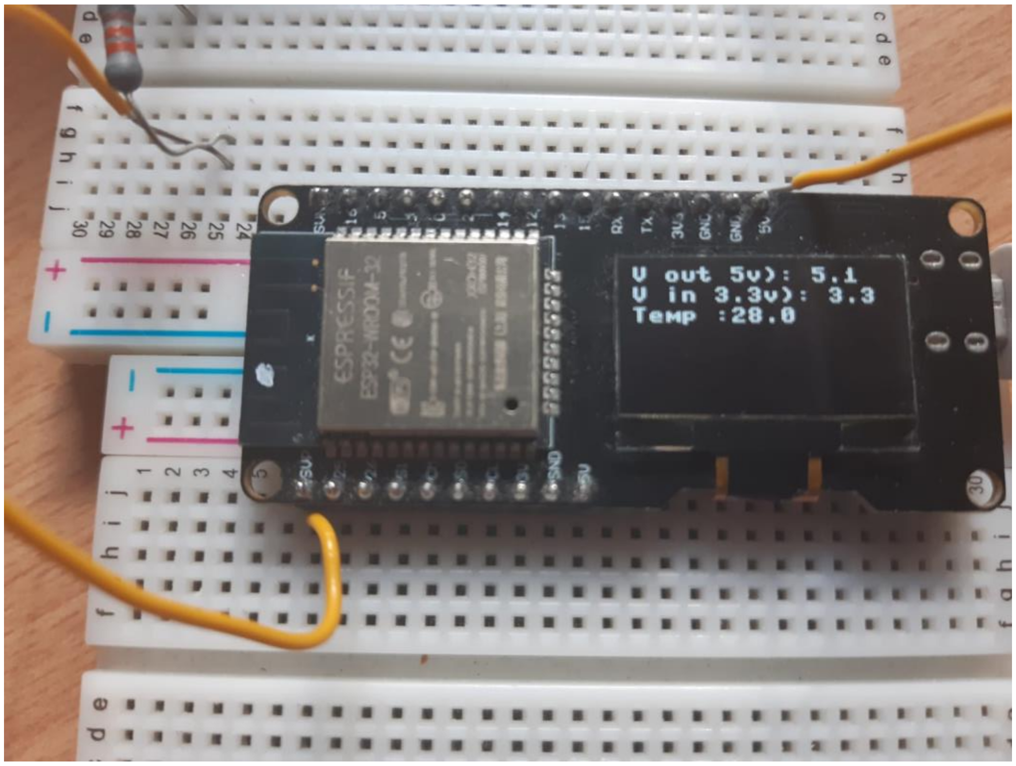


Figure 13: Temperature data

MQTT CLIENT TO MQTT BROKER

Once the temperature signal has been tested with the esp board, it is time to publish the signal data. Here esp board is working as MQTT client and raspberry pi as MQTT broker. At first, we tried to publish data on raspberry pi. Mosquitto is running on it. So, it will be able to send back the data to all subscriber who subscribed on the same topic. Then, we had subscribed the same topic from the broker and decode the data. Finally, after some transformation we send the data to rest API. All the files are available in gitlab: https://gitlab.tamk.cloud/tamk-projects/summer-projects/2021/021-weather-station-improvements

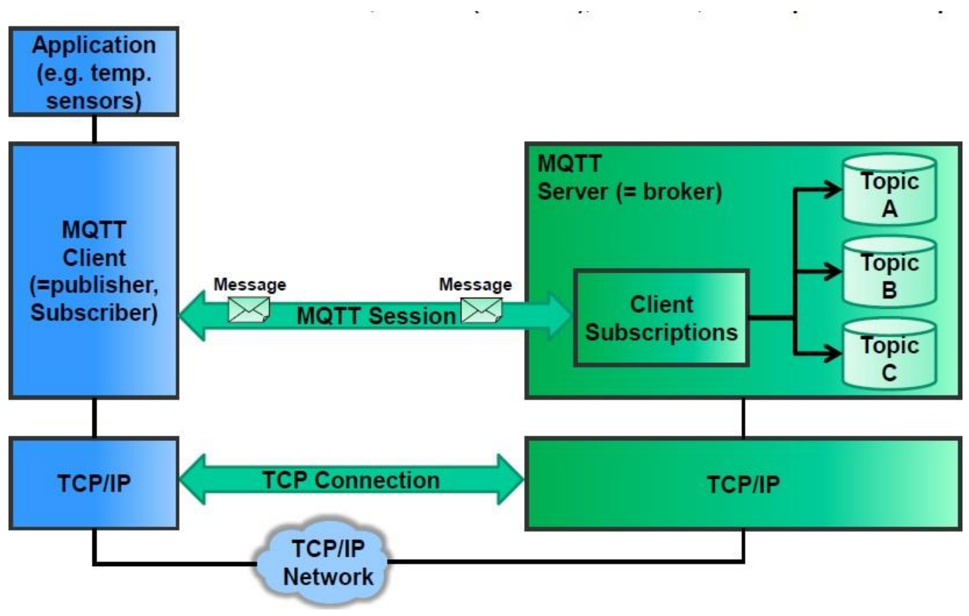
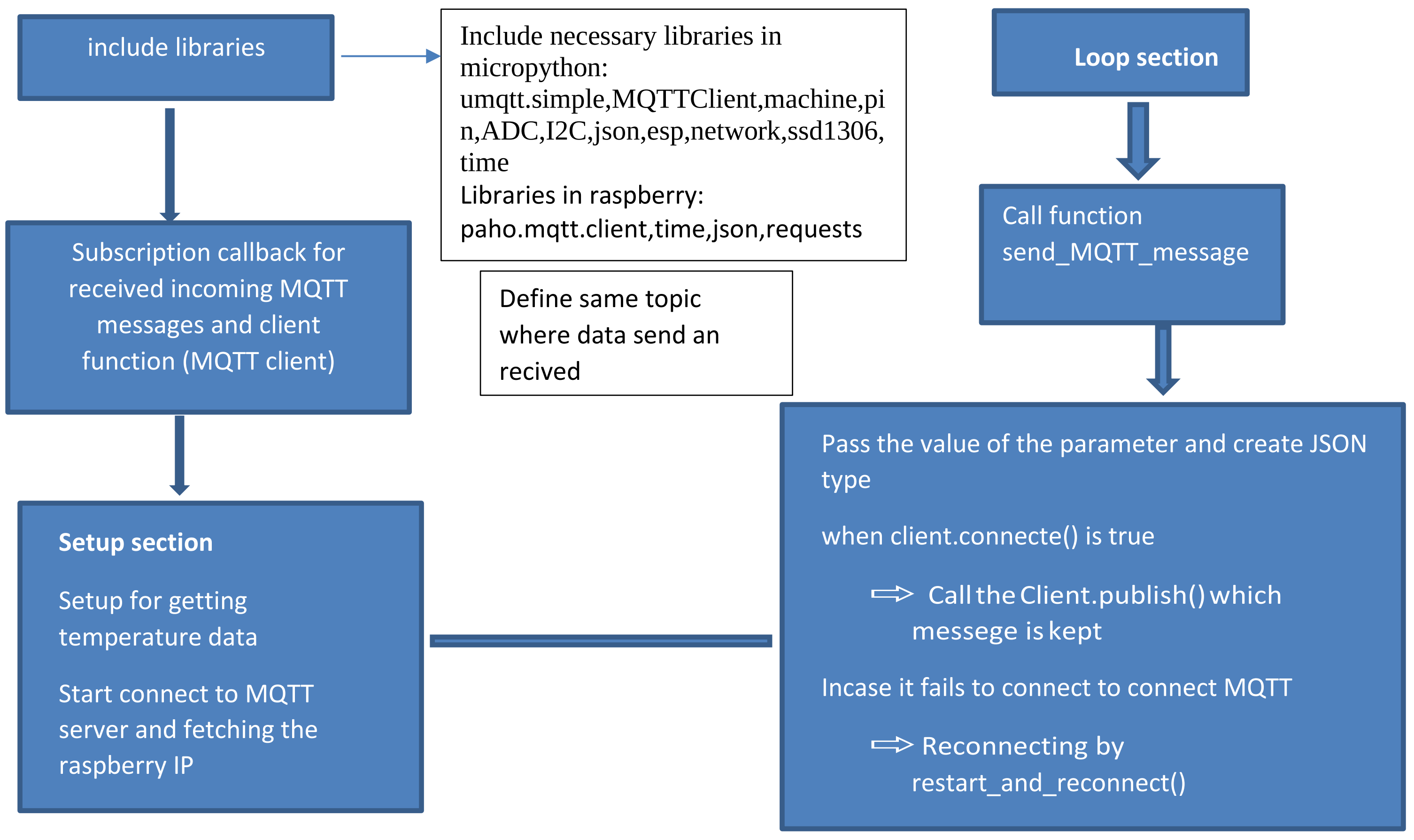


Figure *14: Flowchart from MQTT client to server – source: Embedded System Course*

Detail software flowchart:



include libraries

Include necessary libraries in micropython: umqtt.simple,MQTTClient,machine,pin,ADC,I2C,json,esp,network,ssd1306, time
Libraries in raspberry: paho.mqtt.client,time,json,requests

Subscription callback for received incoming MQTT messages and client function (MQTT client)

Define same topic where data send an recived

Setup section

Setup for getting temperature data

Start connect to MQTT server and fetching the raspbery IP

Loop section

Call function send_MQTT_message

Pass the value of the parameter and create JSON type

when client.connecte() is true

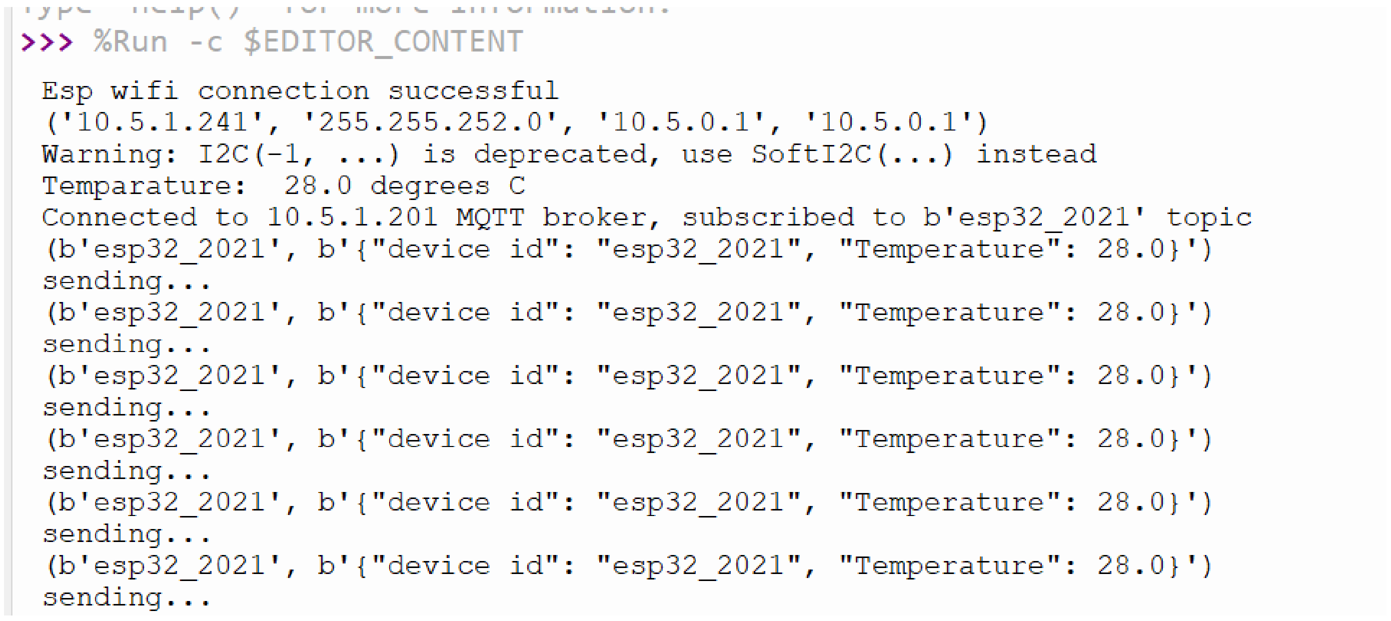
⇨ Call the Client.publish() which messege is kept

Incase it fails to connect to connect MQTT

⇨ Reconnecting by restart_and_reconnect()

Output:

Run **sendToRpi.py** file on esp.



```
>>> %Run -c $EDITOR_CONTENT

 Esp wifi connection successful
 ('10.5.1.241', '255.255.252.0', '10.5.0.1', '10.5.0.1')
 Warning: I2C(-1, ...) is deprecated, use SoftI2C(...) instead
 Temparature:  28.0 degrees C
 Connected to 10.5.1.201 MQTT broker, subscribed to b'esp32_2021' topic
 (b'esp32_2021', b'{"device id": "esp32_2021", "Temperature": 28.0}')
 sending...
 (b'esp32_2021', b'{"device id": "esp32_2021", "Temperature": 28.0}')
 sending...
 (b'esp32_2021', b'{"device id": "esp32_2021", "Temperature": 28.0}')
 sending...
 (b'esp32_2021', b'{"device id": "esp32_2021", "Temperature": 28.0}')
 sending...
 (b'esp32_2021', b'{"device id": "esp32_2021", "Temperature": 28.0}')
 sending...
 (b'esp32_2021', b'{"device id": "esp32_2021", "Temperature": 28.0}')
 sending...
```

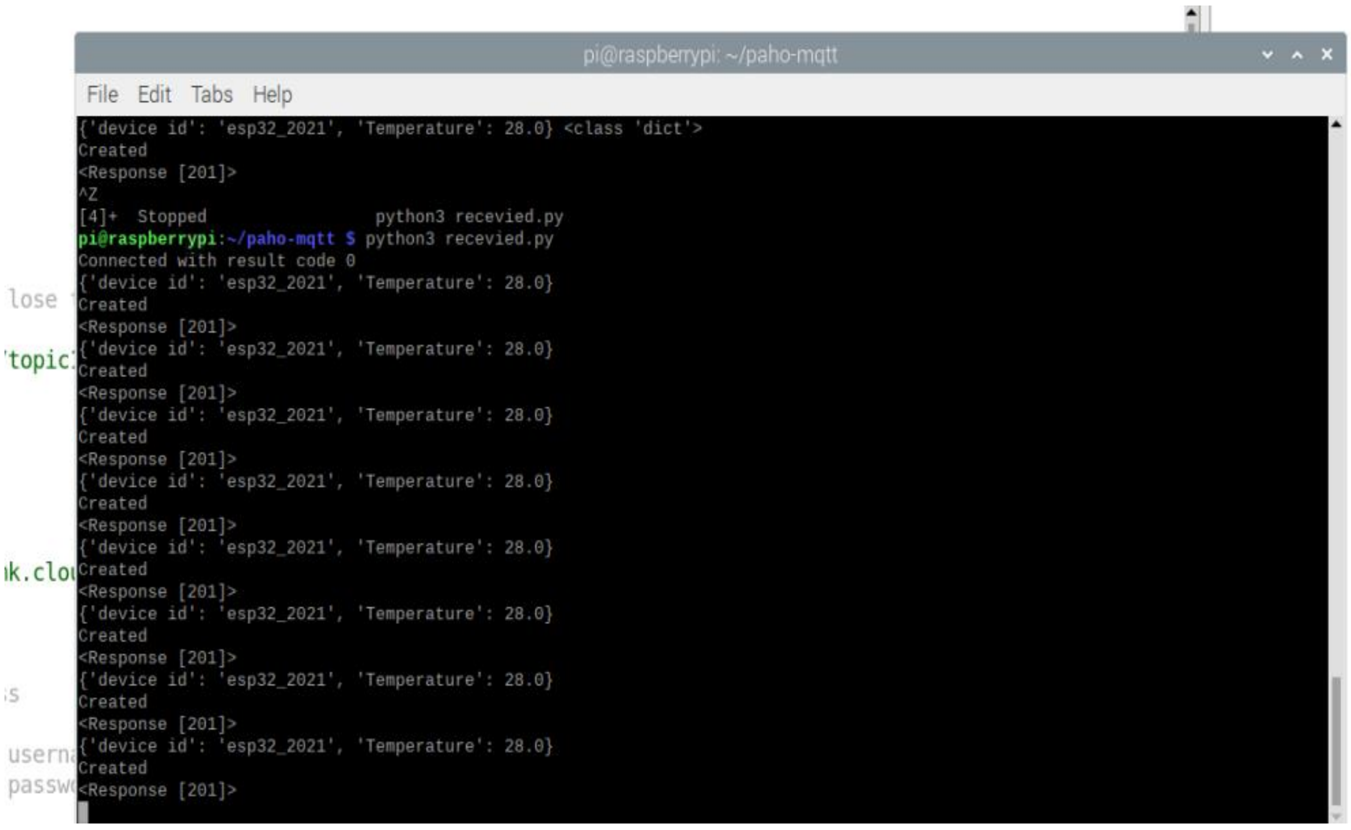
MQTT SERVER TO REST API

Once the data is transforming and sending to the MQTT broker. It is time push the data to the database (http://webapi19sa-1.course.tamk.cloud/v1/weather)

Open terminal on raspberry pi and run received_inRpi.py file.

- Raspberry PI/Virtual Machines
  - Run as IoT gateway, when sending messages Raspberry will forward the messages.
  - MQTT broker named Mosquitto and python code to construct and forward the http POST messages to API.

Output:



```
{'device id': 'esp32_2021', 'Temperature': 28.0} <class 'dict'>
Created
<Response [201]>
^Z
[4]+  Stopped                 python3 recevied.py
pi@raspberrypi:~/paho-mqtt $ python3 recevied.py
Connected with result code 0
{'device id': 'esp32_2021', 'Temperature': 28.0}
Created
<Response [201]>
{'device id': 'esp32_2021', 'Temperature': 28.0}
Created
<Response [201]>
{'device id': 'esp32_2021', 'Temperature': 28.0}
Created
<Response [201]>
{'device id': 'esp32_2021', 'Temperature': 28.0}
Created
<Response [201]>
{'device id': 'esp32_2021', 'Temperature': 28.0}
Created
<Response [201]>
{'device id': 'esp32_2021', 'Temperature': 28.0}
Created
<Response [201]>
{'device id': 'esp32_2021', 'Temperature': 28.0}
Created
<Response [201]>
{'device id': 'esp32_2021', 'Temperature': 28.0}
Created
<Response [201]>
```

5. REFERENCES

https://randomnerdtutorials.com/getting-started-micropython-esp32-esp8266/

https://docs.micropython.org/en/latest/esp32/quickref.html?highlight=dht

https://youtu.be/5srvxIm1mcQ

https://youtu.be/S34R4zg03uE

https://youtu.be/Oo-B98WWre8

https://youtu.be/BzAo48cFhS8

https://youtu.be/lvmNLuHj25o

https://youtu.be/ZkmPRzZPqzc

https://youtu.be/gGB8iw7R-rM

https://youtu.be/S34R4zg03uE