CSCi 4907
Introduction to IoT and Edge Computing Applications

Prof. Kartik Bulusu, CS Dept.

Week 11 [04/05/2024]	• Introduction to Matrices • Scipy.fftpack • Mosquitto – Open source MQTT broker	• Edge Compute Python codes • In-class Raspberry Pi Lab – Mosqiutto MQTT

```
git clone git@github.com:gwu-csci3907/Spring2024.git

git clone https://github.com/gwu-csci3907/Spring2024.git
```

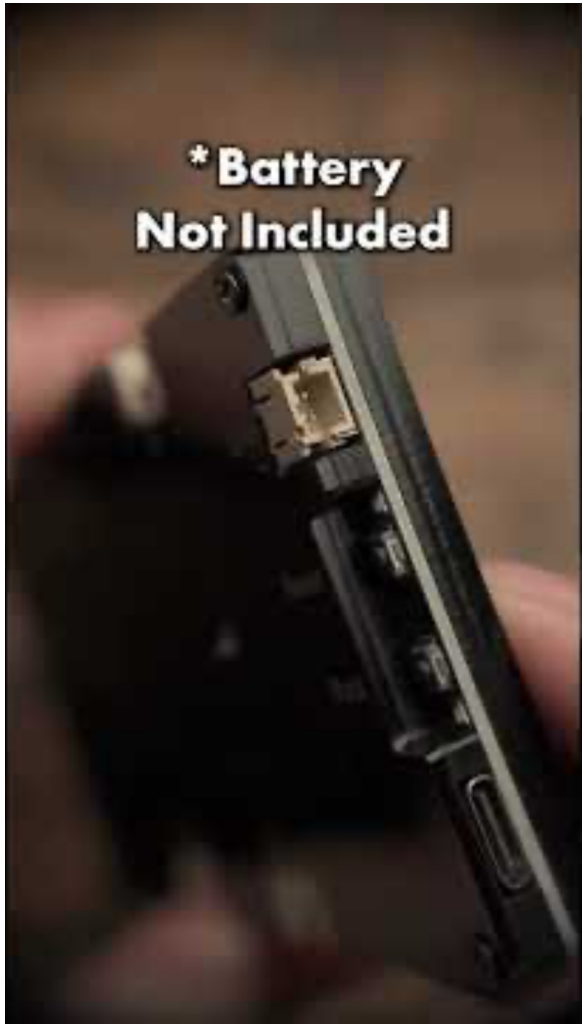
GW School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Spring 2024

Photo: Kartik Bulusu

For full video on RPi see: https://youtu.be/k2C4lbbIH0c?si=imAET8LCsH-kRCjk

YouTube Shorts:
https://youtube.com/shorts/wa6i-k0qRuI?si=FVKpFdCTPO6jJtgl



YouTube Shorts:
https://youtube.com/shorts/njUM5Dib1mk?si=84_vGj-UpS4f0JRP

YouTube Shorts:
https://youtube.com/shorts/qNfzw3c1SII?si=8O9kjxbYIT22d8Np

Final project proposals

Name	Project title	April on March 4, 2024
Georgiana Mois & Peter Wright	Voice Activated RBG Lights	**Approved**
Oliver Kristeya & Warren Nguyen	Laundry Alarm	**Approved**
Selman Eris & William Mai	Posture Correction	**Approved**
Gerald Fattah & Alvin Issac	Real-Time Posture Detection and Correction System	**Approved**
Abdulrahman Alsaleh & Matthew Gouvin	Sleep Monitoring System	**Approved**
Alicia Ha & Jonathan Pang	Smart Pet Feeder	**Approved**
Aleks Haskett & Bridget Orr	Snake Game on RPi	**Approved**
Talia Novack & Liza Mozolyuk	Remember the lecture – the IoT device	**Approved**
Kartik Bulusu	*Nothing to show !!!*	I am flunking this !

Topics to be covered today

Hardware:
SenseHat

Edge computing on the Pi:
FFT + `SciPy.fftpack`

IoT Strategy:
Intro to MQTT

Expectations on student deliverables:

1. Final project presentation
2. Final project demo
3. Final report in a conference-style template

Icon source:
https://thenounproject.com/icon/clean-energy-4339399/

Two questions up for discussion

1. How should we start perceiving an IoT system, physically ?

2. How / Where do we place the “thing” in that system ?

Keywords:
Small, functional, re-envisioning applications, efficient, sensor-driven, smart-sensors, connectivity, autonomy, data-driven, durability, fault tolerance, interoperability

Proximity to data, compute-power, network, distributed-network etc.

All activities today are a part of graded in-class lab
Download codes from github and demonstrate
[10 points]

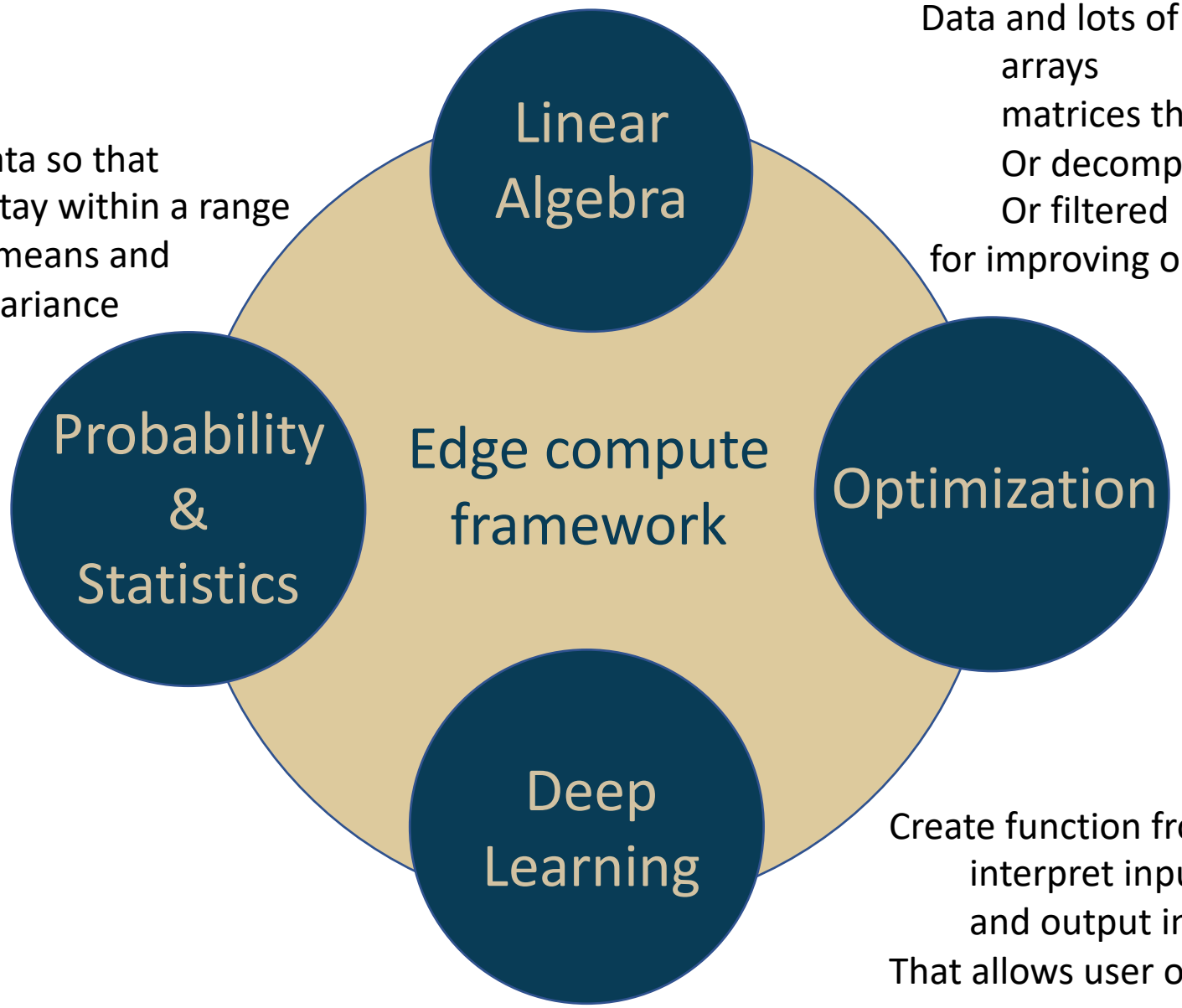
Expanding the Edge framework to Fourier Analysis of data

Goal: To use `scipy` library for signal processing

Source:
https://youtu.be/Cx5Z-OslNWE
Strang, G., Linear Algebra and Learning from Data (2019)

Data and lots of data
arrays
matrices that are special and can be factored
Or decomposed
Or filtered
for improving our understanding of the physical phenomena

Monitor data so that
they stay within a range
means and
variance



Finding matrices
that transform data
and minimize errors
Memory intensive process that usually
happens at cloud-level

Create function from data at cloud-level
interpret input data at edge-level
and output information at edge-level
That allows user or system to take decisions

Source:
https://en.wikipedia.org/wiki/SciPy
https://scipy.org

Explore Signal Processing with Scipy– Python library

SciPy (pronounced /ˈsaɪpaɪ/ "sigh pie"[2]) is a free and open-source Python library used for scientific computing and technical computing.[3]

SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

What is a Matrix ?

DATA

- Arranged in ROWS and COLUMNS
- Typically carries a MEANING

DATA

- Rectangular ARRAY of numbers

ARRAYS

- Two-dimensional arrays
- *m* rows and *n* columns

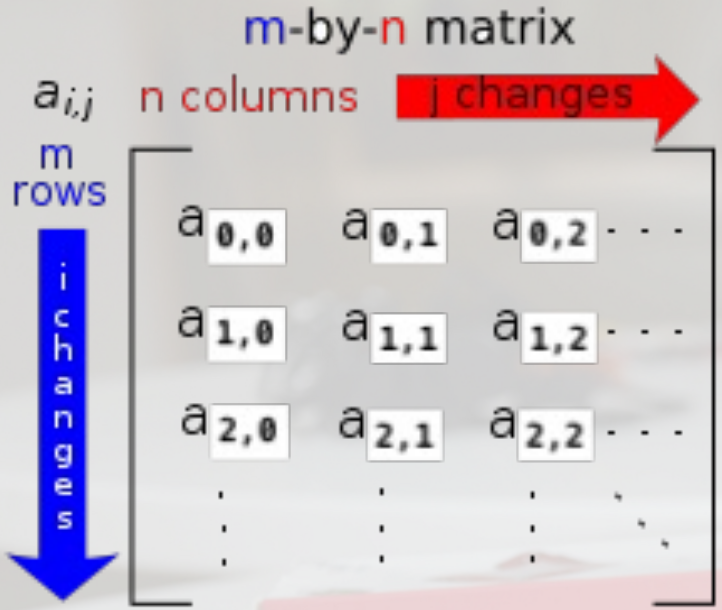
Source: http://giphy.com/search/matrix-gif

$$\begin{bmatrix} 1 & -4 \\ 9 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 15 & 3 & 9 \\ 2 & 5 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 11 & 7 \\ 4 & 2 \\ 6 & 9 \\ 3 & 1 \end{bmatrix}$$

Bookkeeping in a Matrix



Source: http://en.wikipedia.org/wiki/Matrix_(mathematics)

Python:

```
>>> import numpy as np
>>> A = np.matrix([[-1, 2],[3, 4]])
>>> A[0,0]
>>> A[0,:]
>>> A[:,0]
>>> A[1,0]
```

The ORDER of a matrix

- $A_{m \times n}$ *is* $m \times n$
- *Read as "m-by-n"*

a_{ij} *is called an ELEMENT*

- *at the* i^{th} *row and* j^{th} *column of A*

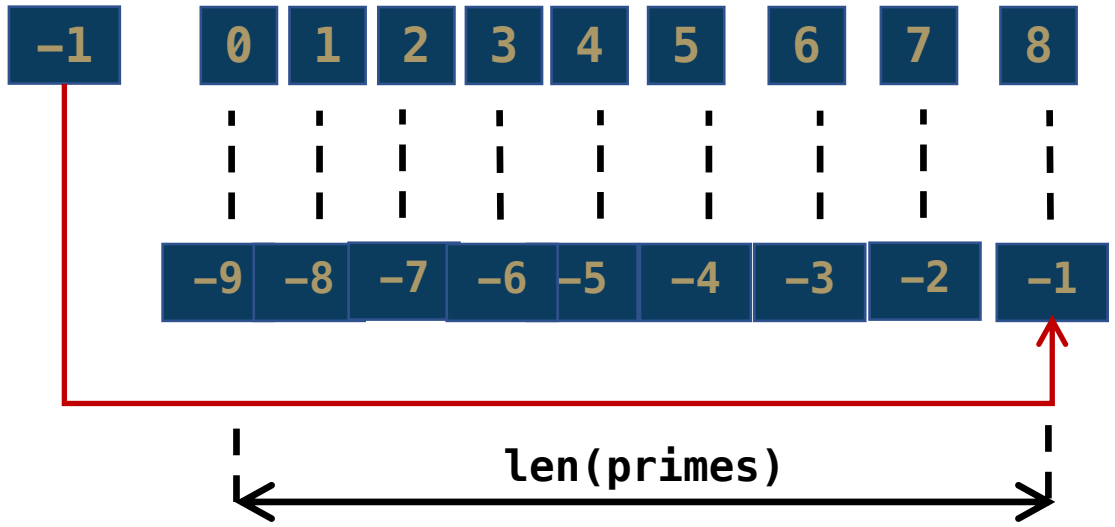
MATLAB:

```
>> A = [-1 2; 3 4]
>> A(1,1)
>> A(1,:)
>> A(:,2)
>> A(2,1)
```

Indexing and Slicing Lists

Retrieve list-elements with a range of values

```
>>> primes = [2, 3, 5, 7, 9, 11, 13, 17, 19]
```



start stop

```
>>> primes[2:5]
[5, 7, 9]
```

start stop step

```
>>> primes[0:7:2]
[2, 5, 9, 13]
```

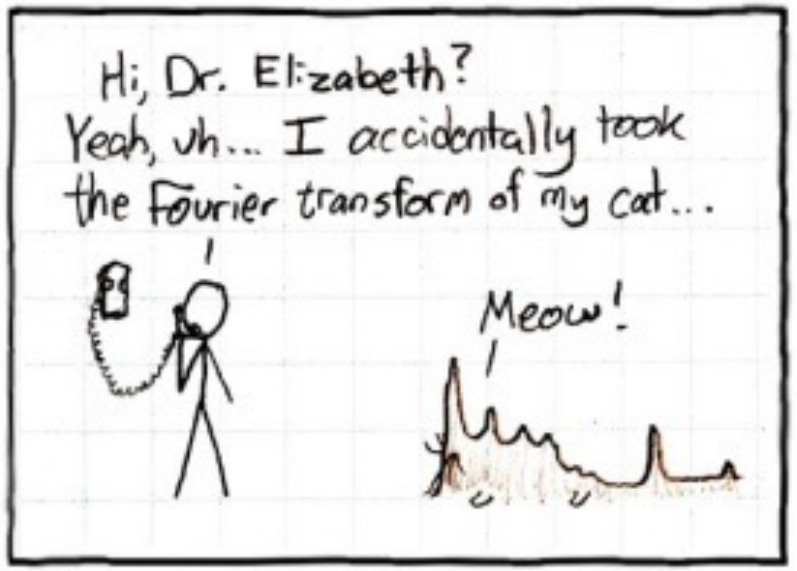
start stop step

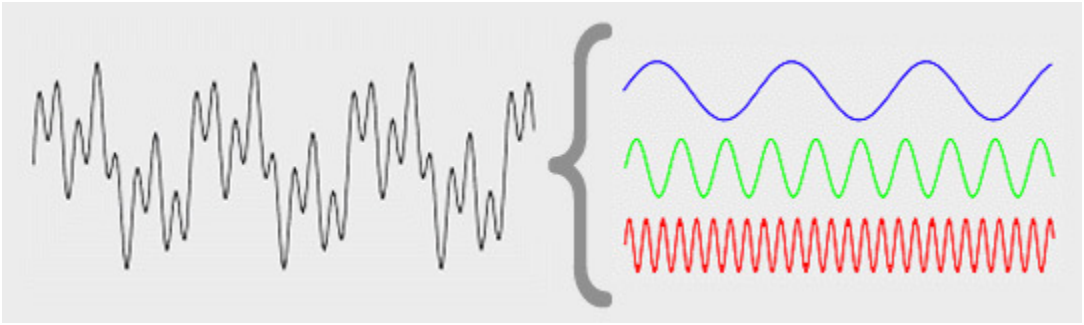
```
>>> primes[8:2:-2]
[19, 13, 9]
```

start: at the index value
step: up or down at the increment value (default = 1)
stop: at the index value but not including it

Fourier transform: non-mathematical introduction

Seeing information in harmonics

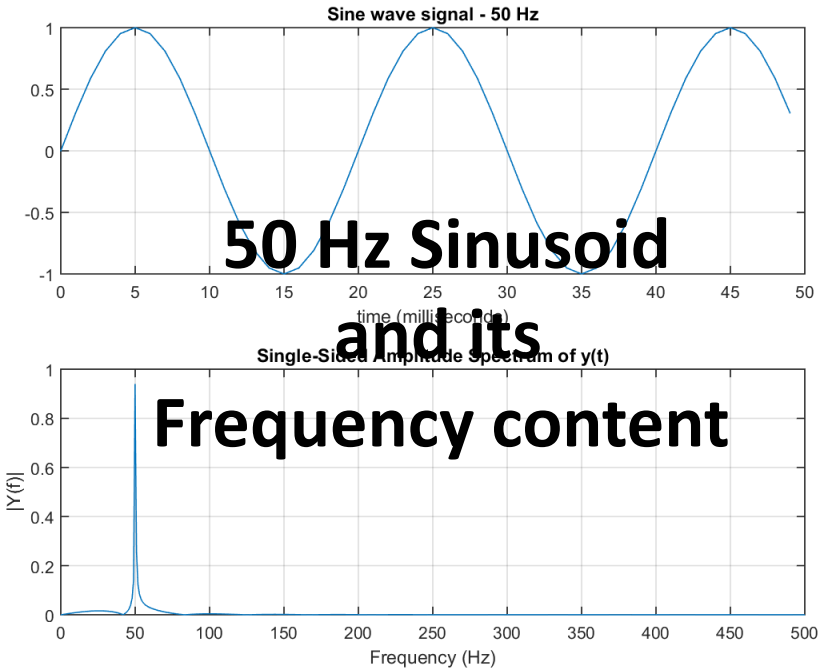


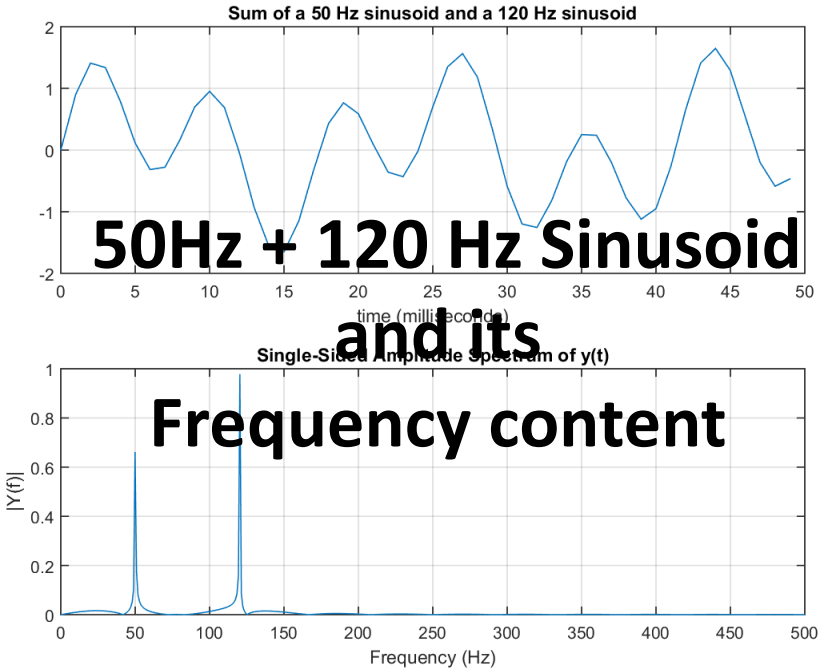


Source:

1. http://www.blog.radiator.debacle.us/2013/02/narrative-systems-workflow-using.html
2. http://xkcd.com/26/
3. http://en.wikipedia.org/wiki/File:Joseph_Fourier_%28circa_1820%29.jpg

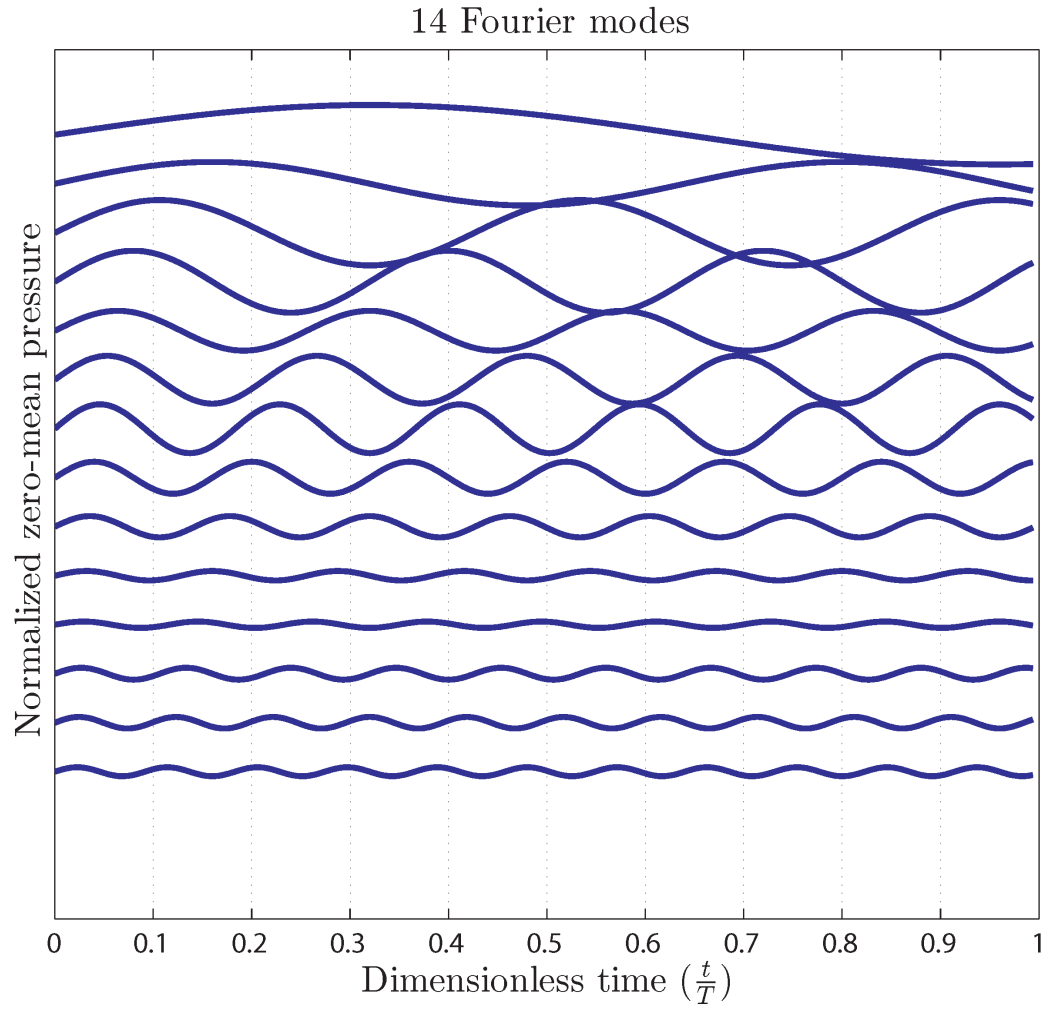
How do data/signals look in Fourier space ?



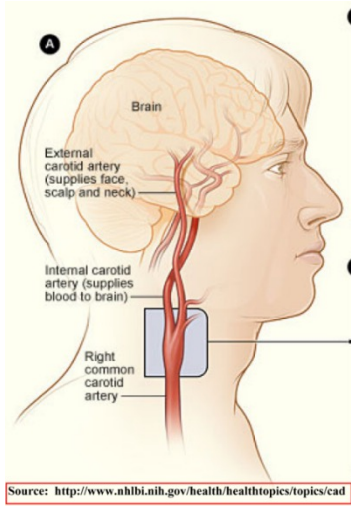


How do data/ signals look in Fourier space ?

Interpreting Fourier transform through an example



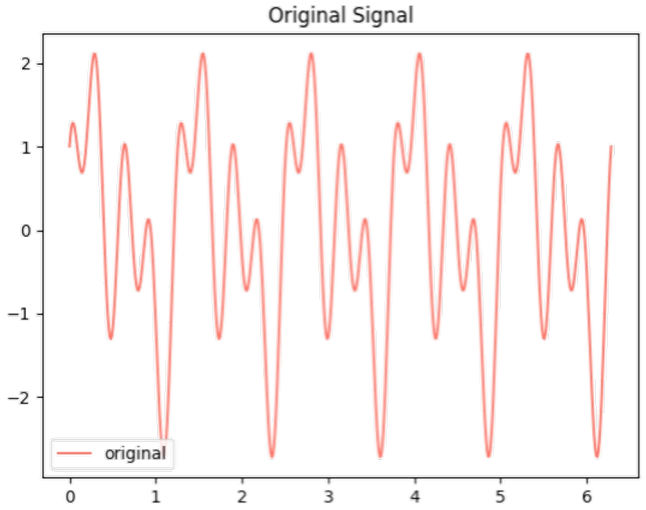
- **Decomposition of the pressure-time signal related to the carotid artery waveform**



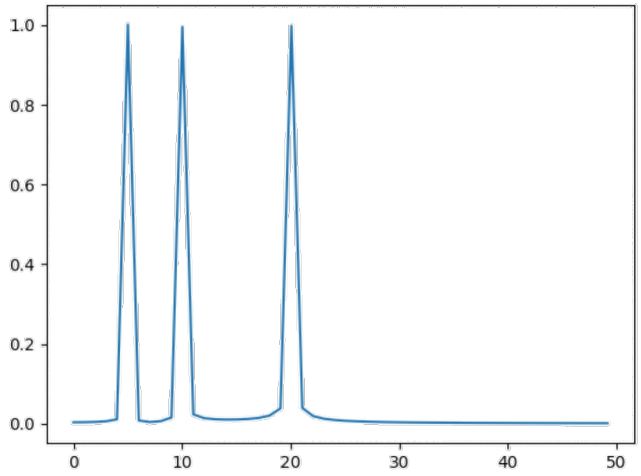
- **Summation of 14 wavetrains (Fourier modes) give the original waveforms**

Practical look at a code using `scipy.fftpack` library

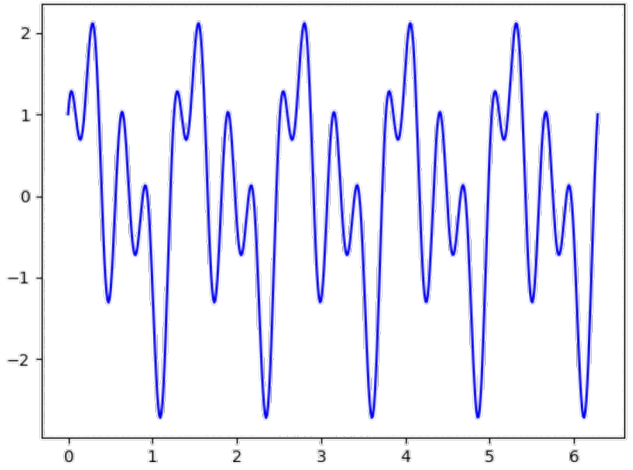
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, ifft, fftfreq

N = 1000
T = 1
x = np.linspace(0, 2*np.pi*N*T, N)
y1 = np.cos(20*x)
y2 = np.sin(10*x)
y3 = np.sin(5*x)

y = y1 + y2 + y3
# Produces an original signal
```




```
fy = fft(y)
# finds the fft
```

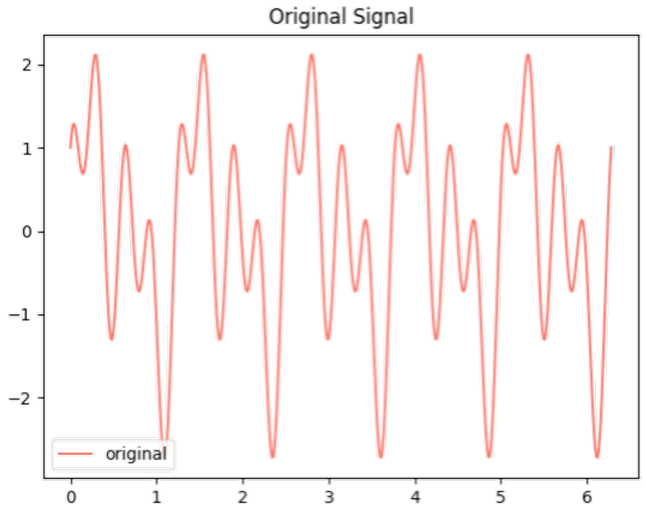


```
y4 = ifft(fy)
# finds the inverse fft
```

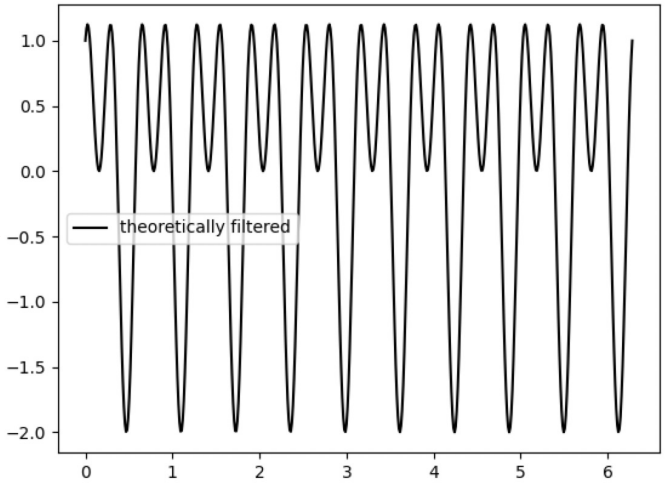


Practical look at Fourier filtering using `scipy.fftpack`

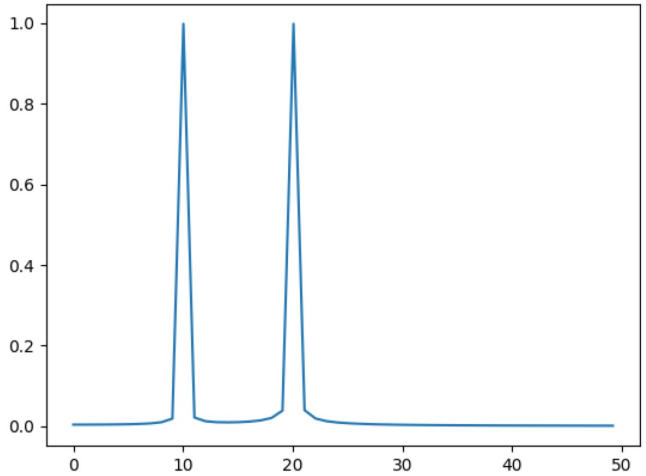
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, ifft, fftfreq

N = 1000
T = 1
x = np.linspace(0, 2*np.pi*N*T, N)
y1 = np.cos(20*x)
y2 = np.sin(10*x)
y3 = np.sin(5*x)

y = y1 + y2 + y3
# Produces an original signal
```



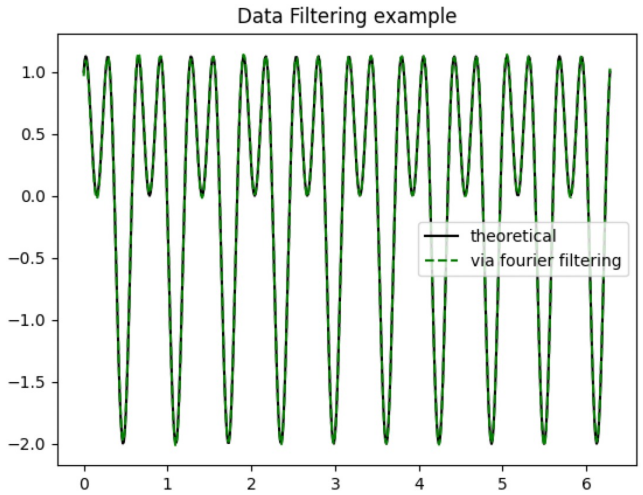

```
act = y1 + y2
# Produces a theoretically
# filtered signal
```




```
fy_act = fft(act)
# finds the fft
```



```
freqs = fftfreq(N)
nwaves = freqs*N # wave numbers

fft_vals = fft(y)

# Fourier filtering of 5 Hz signal
fft_new = np.copy(fft_vals)
fft_new[np.abs(nwaves)==5] = 0.0

# inverse fourier transform to
# reconstruct the filtered data
filt_data = np.real(ifft(fft_new))
```



Explore MQTT Basics
Message Queuing Telemetry Transport

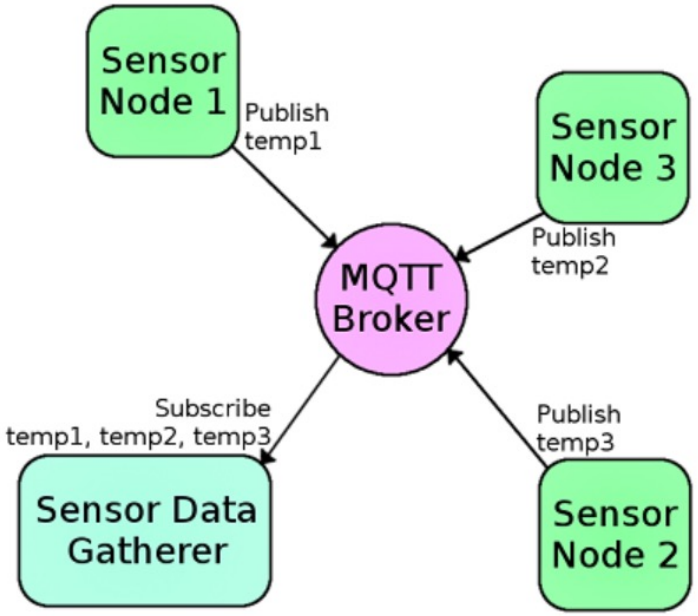
Goal: To understand how publishing and subscribing works practically

School of Engineering
& Applied Science
THE GEORGE WASHINGTON UNIVERSITY

GW

Prof. Kartik Bulusu, CS Dept.

MQTT paradigm



Hardware

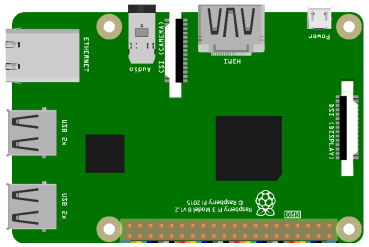
Broker

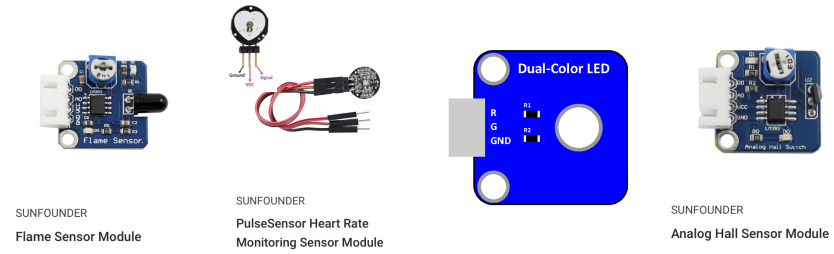
- The broker is the server
- It distributes the information to the interested devices connected to the server.

Client

- The device that connects to broker to send or receive information.

Source: https://learn.sparkfun.com/tutorials/introduction-to-mqtt#the-basics





SUNFOUNDER
Flame Sensor Module

SUNFOUNDER
PulseSensor Heart Rate Monitoring Sensor Module

SUNFOUNDER
Analog Hall Sensor Module

Messaging

Topic

- The name that the message is about.
- Clients publish, subscribe, or do both to a topic.

Subscribe

- Clients tell the broker which topic(s) they're interested in.

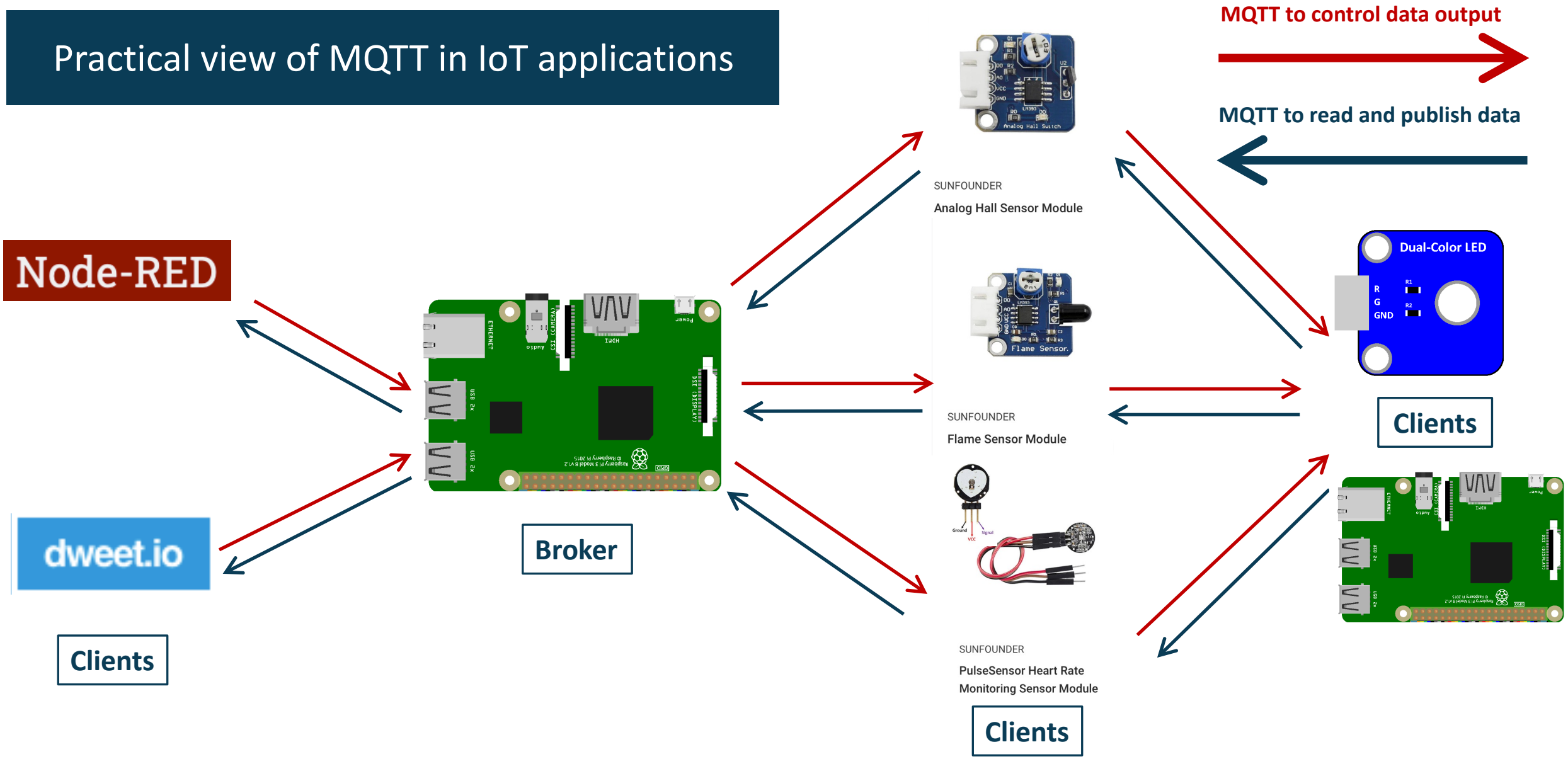
Publish

- Clients that send information to the broker to distribute to interested clients based on the topic name.

QoS

- Quality of Service to the broker
- Integer value ranging from. 0-2.

Practical view of MQTT in IoT applications



Source
https://mqtt.org/getting-started/ :
https://mosquitto.org
https://github.com/eclipse/mosquitto
https://projects.eclipse.org/projects/iot.mosquitto

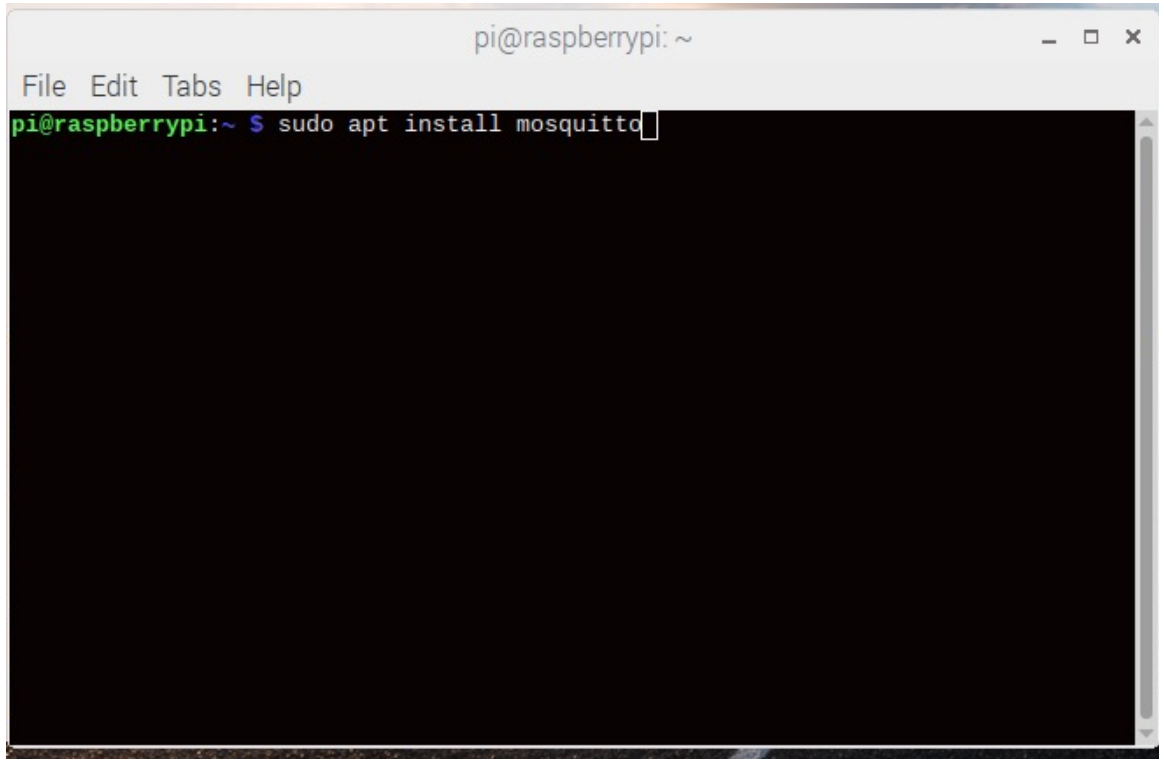
Eclipse Mosquitto - An open source MQTT broker

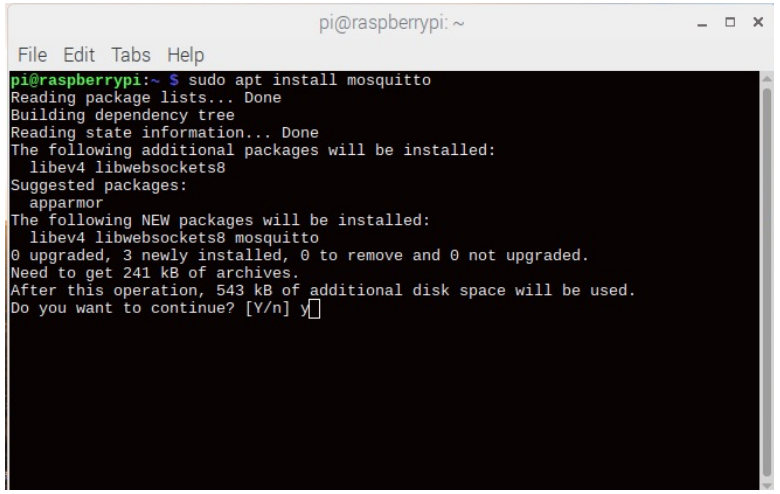
Eclipse Mosquitto provides a lightweight server implementation of the MQTT protocol that is suitable for all situations from full power machines to embedded and low power machines.

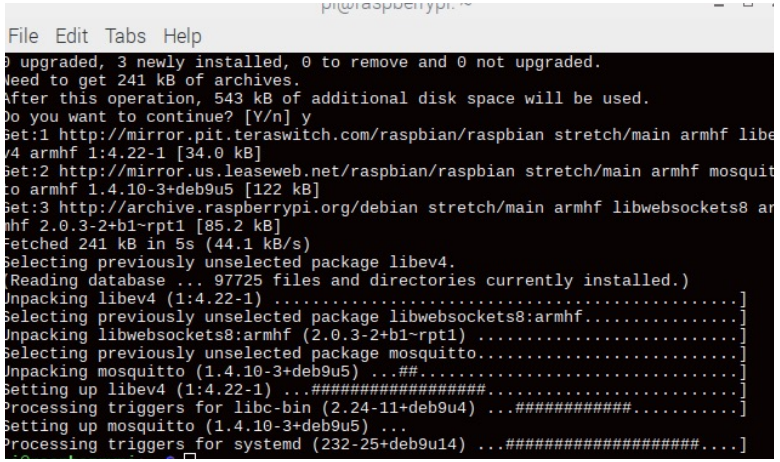
Sensors and actuators, which are often the sources and destinations of MQTT messages, can be very small and lacking in power. This also applies to the embedded machines to which they are connected, which is where Mosquitto could be run.

Step-1: Eclipse Mosquitto - An open source MQTT broker

```
sudo apt-get update
sudo apt-get upgrade
sudo apt install mosquitto
```

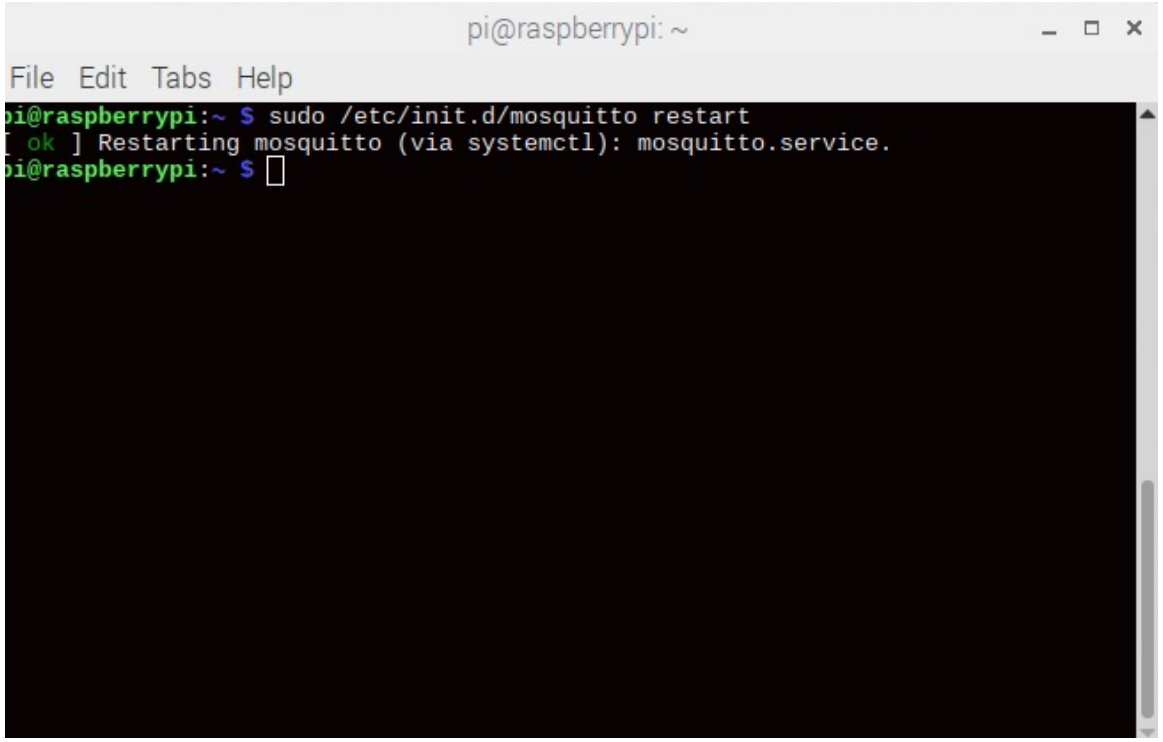






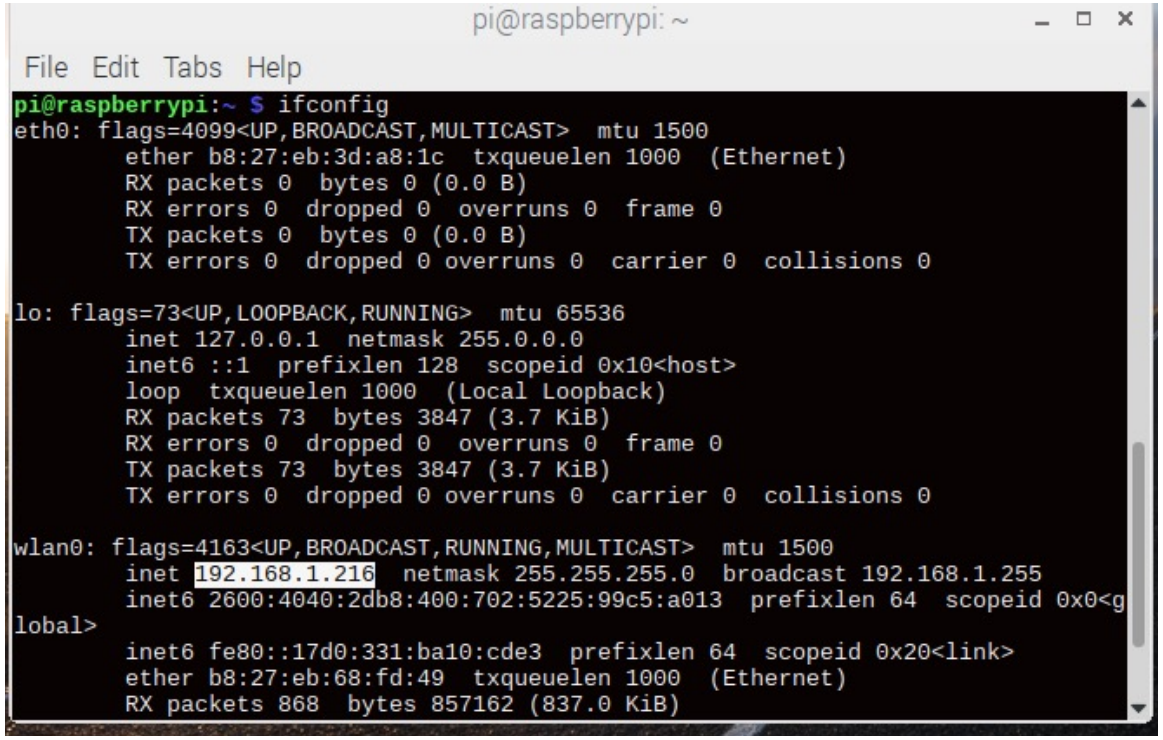
Step-2: `restart` Mosquitto

```
sudo /etc/init.d/mosquitto restart
```



Step-3: Get your IP address

```
ifconfig
```

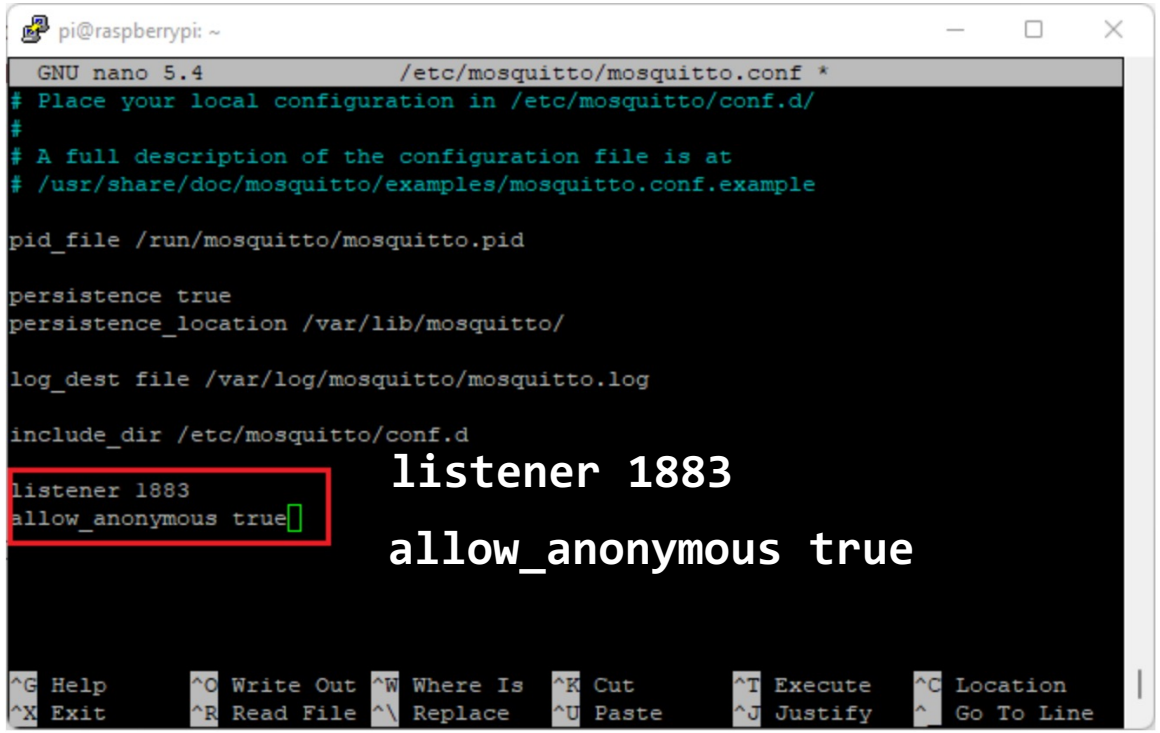



```
hostname –I   # this is also OK to use
```

Source:
https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/
https://forums.raspberrypi.com/viewtopic.php?t=331096

Step-4: Enable Remote Access to Mosquitto Broker (No Authentication)

```
sudo nano /etc/mosquitto/mosquitto.conf
```




```
listener 1883
allow_anonymous true
```

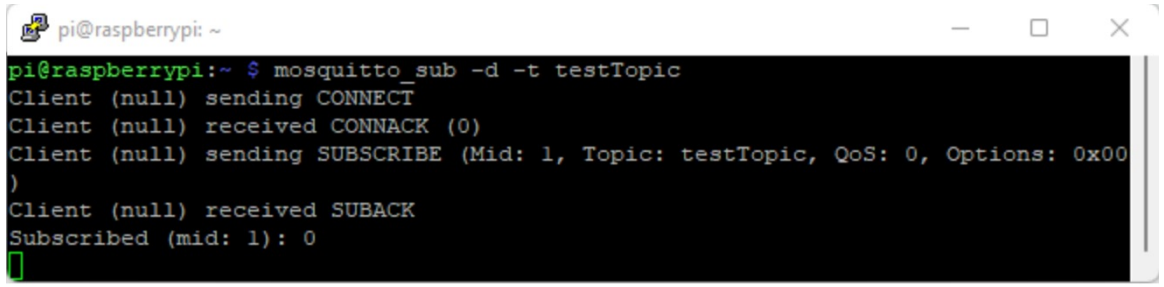
```
sudo systemctl restart mosquitto
```

```
sudo systemctl status mosquitto
```

Source:
https://randomnerdtutorials.com/testing-mosquitto-broker-and-client-on-raspbbery-pi/

Step-4: Publishing "Hello World!" Message to *testTopic* Topic

```
# Install mosquitto-clients
sudo apt install -y mosquitto mosquitto-clients

# Open a terminal window and type the following:
mosquitto_sub -d -t testTopic
```




```
mosquitto_sub -v -t '#' -h 192.168.1.248
```

```
mosquitto_pub -d -t testTopic -m "Hello world!"
```