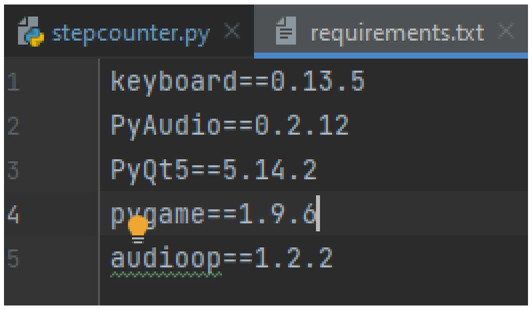
Multidimensional Sensing Techniques
Assignment 2 – Audio Sensors
Md Mahbub Islam – ID: 2202978

https://github.com/it-teaching-abo-akademi/assignment-2---audio-sensors-Md-Mahbub-Islam

Introduction

In this project, we aim to get familiarized with audio sensors. We will be using python for this lab with the provided script. The script has several dependencies which need to be installed before we can run the script. We will install them through the 'pip install NAME' command on our computer. Before we run the script, we will connect as many microphones (at least two) to our computer as possible to function as our audio sources. We can also connect our android devices to our computer using third-party software such as droidcam or WO Mic. Here are few links related to the use of a mobile phone as a microphone on a PC.

A requirements.txt file was created for easy installation of the modules.



```
keyboard==0.13.5
PyAudio==0.2.12
PyQt5==5.14.2
pygame==1.9.6
audioop==1.2.2
```

Figure 1: requirements file

Modules could be easily installed using command

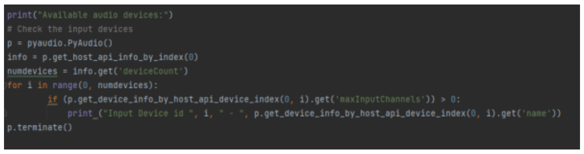
Pip -r requirements.txt

The goal is to get familiarized with audio volume data and do some basic calculations such as mean and variances. Since a method wasn't mentioned numpy library functions were used to calculate these values.

Primary Setup

The provided program uses PyQt5 library to display the GUI program.

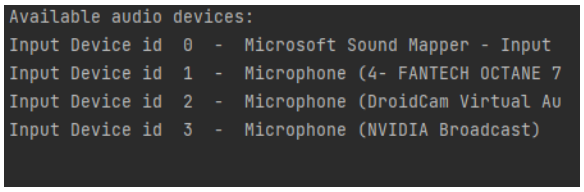
Pyaudio library gets a list of all available audio devices.



```
print("Available audio devices:")
# Check the input devices
p = pyaudio.PyAudio()
info = p.get_host_api_info_by_index(0)
numdevices = info.get('deviceCount')
for i in range(0, numdevices):
        if (p.get_device_info_by_host_api_device_index(0, i).get('maxInputChannels')) > 0:
            print ("Input Device id ", i, " - ", p.get_device_info_by_host_api_device_index(0, i).get('name'))
p.terminate()
```

Figure 2: Getting devices list

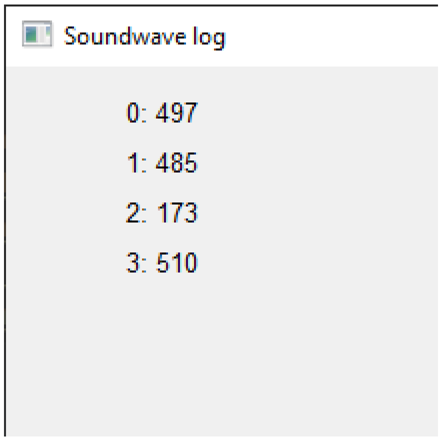
They output is displayed in the console.



```
Available audio devices:
Input Device id  0  -  Microsoft Sound Mapper - Input
Input Device id  1  -  Microphone (4- FANTECH OCTANE 7
Input Device id  2  -  Microphone (DroidCam Virtual Au
Input Device id  3  -  Microphone (NVIDIA Broadcast)
```

Figure 3 : Devices list

For me it was 3 devices. The NVIDIA Broadcast device is just a filter audio source for main devices.

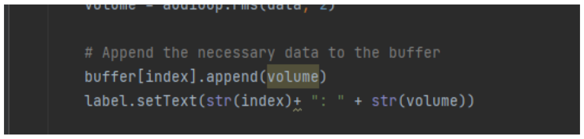


```
Soundwave log

0: 497
1: 485
2: 173
3: 510
```

Figure 4: Raw sensor log for volumes

The GUI shows sound volume at a certain time. It updates very rapidly.

The following part of the code was used to show each device volumes



```
volume = audioop.rms(data, 2)

# Append the necessary data to the buffer
buffer[index].append(volume)
label.setText(str(index)+ ": " + str(volume))
```

Figure 5: Showing current volumes of each devices

Mean and Variance (Current Time)

If the number of devices is N then the mean of the volumes is denoted by

$$\overline{\boldsymbol{X}} = \frac{\sum \boldsymbol{V_i}}{\boldsymbol{N}}$$

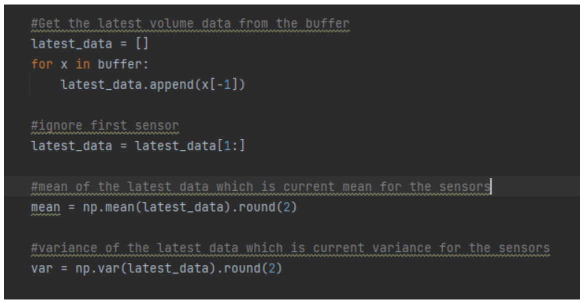
Vi is the volumes of each device in current frame

$\overline{\mathbf{X}}$ is the mean.

Similarly, variance was calculated Using the following formula.

$$S^2 = \frac{\sum(x_i - \overline{x})^2}{n-1}$$

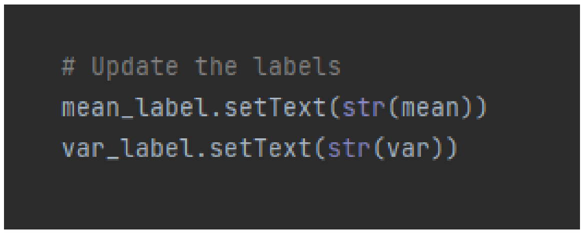
In the following code I used numpy library to calculate mean and varience



```
#Get the latest volume data from the buffer
latest_data = []
for x in buffer:
    latest_data.append(x[-1])

#ignore first sensor
latest_data = latest_data[1:]

#mean of the latest data which is current mean for the sensors
mean = np.mean(latest_data).round(2)

#variance of the latest data which is current variance for the sensors
var = np.var(latest_data).round(2)
```

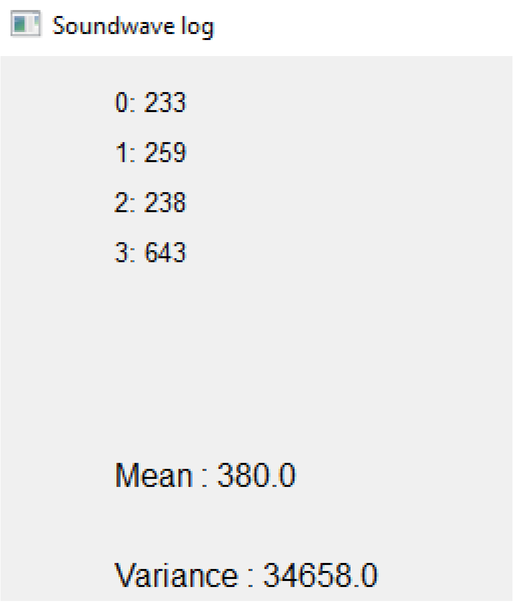
Figure 6: Finding mean and variance

Then set the labels to show them in GUI



```
# Update the labels
mean_label.setText(str(mean))
var_label.setText(str(var))
```

Figure 7: Displaying mean and variance

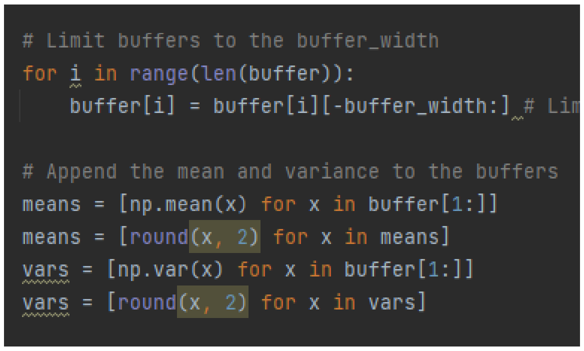


```
Soundwave log

0: 233
1: 259
2: 238
3: 643



Mean : 380.0

Variance : 34658.0
```

Figure 8: Displayed data

Mean and Variance (100 Data point)

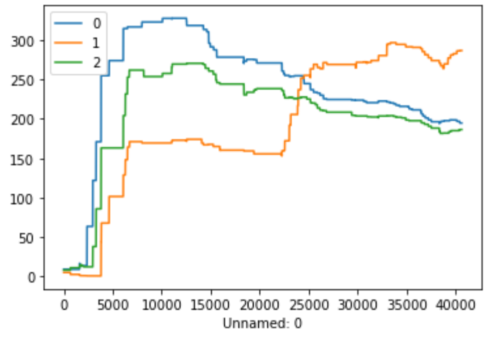
I saved the volume points for each device in a buffer. In this case 100 was used as buffer length.



```
# Limit buffers to the buffer_width
for i in range(len(buffer)):
    buffer[i] = buffer[i][-buffer_width:] # Lim

# Append the mean and variance to the buffers
means = [np.mean(x) for x in buffer[1:]]
means = [round(x, 2) for x in means]
vars = [np.var(x) for x in buffer[1:]]
vars = [round(x, 2) for x in vars]
```

Figure 9: 100 data points saved

Then saved the data points for later analysis in Matplotlib.
After that a few thousand data point is saved for the mean of the last data points



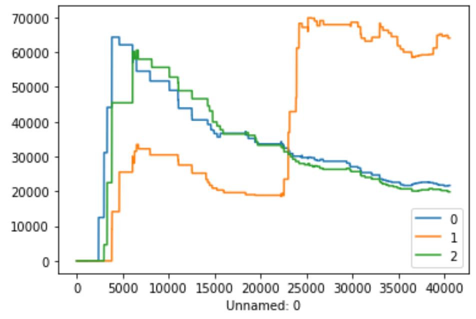


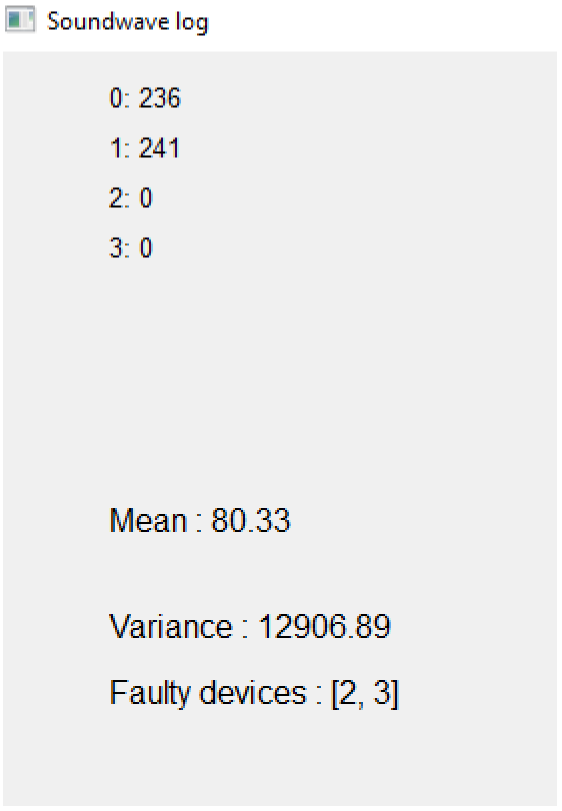
Figure 10: Plots of 100's data point mean and variance, above and below respectively

Each data point correspoinds to 100 last overvation. So it starts with 100, then 101, 102, Etc each point only taking last 100 in calculation.

In my case the main speaker and NVIDIA broadcast filter data seems to follow almost similar path. As expected. Droid cam shows similar path at the start. But picks up as I put it closer to sound source.

Faulty Device Finding

It is not exactly easy to find faulty device. But assuming the faulty device won't pickup sound over certain threshold. Simplest system would be to find if the device volume passes certain threshold.



Soundwave log

```
0: 236
1: 241
2: 0
3: 0

Mean : 80.33

Variance : 12906.89

Faulty devices : [2, 3]
```

Figure 11: Faulty devices

In this scenarios device 2,3 is shown as faulty as they have lower volume compared to mean. (device configuration is different as this test was done in another time compared to previous tests)

Conclusion

We can get many interesting data from the audio signals. Mean smooth outs the data. And variance shows how divergent our data is. We can used that to our advantage for specific tasks.