Virtual Hand

23.06.2022

Genero Inc.
Gebze Technical University
Kocaeli/Gebze

Contents

1.Introduction

1.1 Purpose of the document

This document defines how we will realize our project (how many parts we will divide and how we will share it), what the final product will look like, and who will take part in which parts of the project.Also document explains the details of the project. Every module has its own chapter in the report and the implementation of every module is explained in these chapters. In the last chapter, there is the scheduling part and it shows what has been done in the previous weeks.

1.2 About Project

Some people unfortunately do not have certain limbs. Some of them are born with such disabilities and some lose their arms and legs to unfortunate accidents. Such disabled individuals may have difficulties in daily life. There are several solutions for those disabilities such as prosthetic limbs. Those prostheses are used for functional or aesthetic purposes. Brain sends signals to limbs whether or not they exist. We intend to use this in order to move a prosthetic hand. But lack of meaningful data and insufficient hardware, we are only capable of sensing the focus information with our hardware.

1.3 Team Members

Company Name	Genero Inc.
Project Name	Virtual Hand
Team Members	
Özlem Servi	1801042671
Buse Elbirgiç	161044032
Salih Tangel	171044071
Yunus Emre Yumşak	1801042659
Mehmet Hüseyin Yıldız	200104004095
Abdurrahman Bulut	1901042258
Ahmet Tuğkan Ayhan	1901042692
Muhammet Fikret Atar	1801042693
Doğukan Güler	200104004092
Ömer Faruk Erol	171044020
Yusuf Talha Altun	1901042695
Yusuf Fatih Şişman	171044017

1.4 Team Members' Roles

Hardware Team	Data Acquisition Team	Data Processing Team	Communication Team	Visualization Team
Ahmet Tuğkan Ayhan	Özlem Sevri	Özlem Servi	Muhammet Fikret Atar	Salih Tangel
Mehmet Hüseyin Yıldız	Yusuf Talha Altun	Mehmet Hüseyin Yıldız	Doğukan Güler	Buse Elbirgiç
Muhammet Fikret Atar	Yunus Emre Yumşak	Ahmet Tuğkan Ayhan	Ömer Faruk Erol	Yusuf Fatih Şişman
Salih Tangel		Ömer Faruk Erol		
		Abdurrahman Bulut		

2.Hardware Module

2.1 Brief Summary

While doing the research for this project, we realized that receiving signals and creating responses according to these signals in an instant is not possible with fNIRS devices. Because of that we looked for other sensors that can recognize brain signals and produce meaningful data with them.

After eliminating a lot of options(fNIRS, EKG, PET, EMG, etc) we decided to use EEG sensors to receive signals from the brain. The advantages of using EEG are; affordable, easy to build, doesn't harm the body, processing signals doesn't require as much time as fNIRS and so on.

To make the project manageable, we divided the project into 5 modules. These modules are; hardware, data acquisition, data processing, communication and visualization. We also divided our team in such a way so, there are at least 3 people that work on a single module.

While doing the hardware part, we were inspired by someone who is studying Brain and Cognitive Sciences in MIT and followed her instructions. For the rest of the report file, we will be explaining how every module works and how we are planning to create our project in the upcoming weeks.

2.2 Hardware Components

2.2.1 Green LED

To get visual feedback from the circuit, we decided to use a green LED. We could've chosen other colors but we thought strictly following the instructions would decrease our chances of making mistakes during construction of the hardware. To support green LEDs, we will use a 1k ohm resistor.

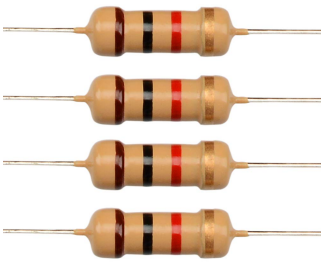
2.2.2 1.5v AAA Battery

To feed the hardware, we will use 4.5v input. To do this we will connect 3 1.5v batteries in series. There will be a total of 6(3x2) batteries which will feed the hardware from 2 different locations.

2.2.3 2x4x 1.5v Battery Holder

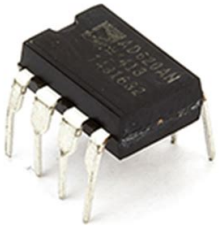
This is the component we will use to get 6V two serial input. There are 2 battery holders in the hardware to increase safety. (also it says in the instruction that this is an efficiency measure)

2.2.4 1k Resistor



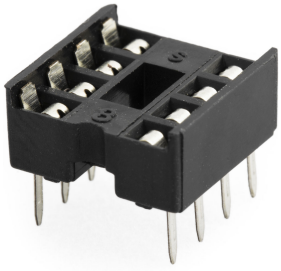
Not only to support LEDs, this 1k resistor is also used to establish sufficient gain on the AD620 chip(which is used to amplify EEG signals). Total of 2 1k resistors will be used in our hardware. One to support LED and one to provide gain to the AD620 chip.

2.2.5 AD620 Chip



AD620 chip is used to amplify EEG signals. The reason behind why we are amplifying EEG signals is, these signals are so small to detect(10-20 millivolts). Thus, using an instrumentation amplifier is a necessity while doing this project.

2.2.6 8 Pin Integrated Socket



To integrate our AD620 chip into our hardware we decided to add an 8 pin integrated socket to our hardware design(not included in the instruction pdf). Using integrated sockets is important because chips can be easily damaged and to prevent any damage can occur on the chip, integrated sockets are used.

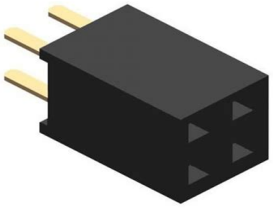
2.2.7 10k uF Capacitor

In the instruction circuit diagram, there is one 4.7k uF capacitor. But instead we will use two 10k uF capacitors placed in series to achieve that. It is necessary to put a capacitor inside a circuit to store electrostatic energy and release this energy to the circuit when it is needed.

2.2.7 2x3 Header

2x3 Headers are used to connect electrodes to circuits. A male-female jumper is sufficient enough to connect electrodes to circuits. Female part of the jumper will be inserted to the header while male part will be attached with alligator clips.

2.2.8 2x2 Header

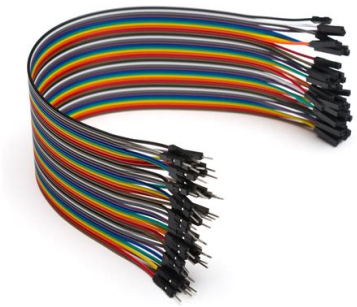


2x2 Headers are used to connect mini stereo plugs into the circuit. After connecting the mini stereo plug, it will be inserted to the mobile phone. (but in our project we will insert it to the external sound card to prevent any damage to the phone or laptop)

2.2.9 Alligator Clips

Alligator clip is a sprung metal clip with long, serrated jaws which will be used for creating a temporary electrical connection.

2.2.10 Jumper Wires

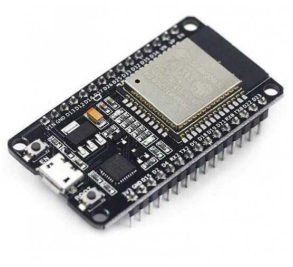


We will use this component to connect two points in our circuit. We will use breadboards and other prototyping tools in order to make it easy to change our circuit as needed.

2.2.11 Electrical Tape

Electrical tape is a safety tape for wires, which will be used to cover and insulate a broad range of cables, wires and other materials that conduct electricity.

2.2.12 ESP 32 Module



ESP32 is the name of the chip that was developed by Espressif Systems. This provides Wi-Fi (and in some models) dual-mode Bluetooth connectivity to embedded devices. We used this device to transfer data from hardware to our machine learning code and interface.

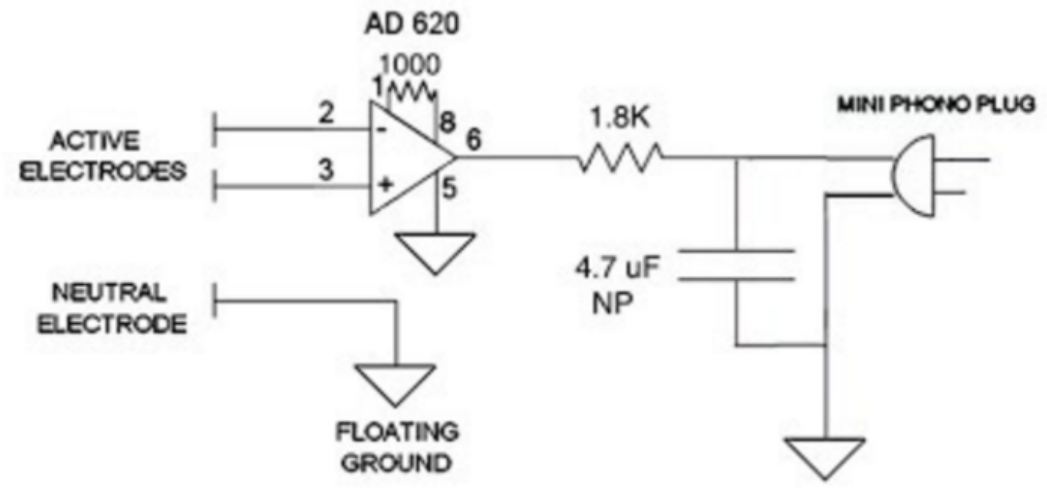
2.2.13 Copper Tape

Copper tape has countless applications in electronics from creating low-profile traces for electrical components to RF-shielding and antenna-making. We used copper tape to receive signals from a person's forehead.

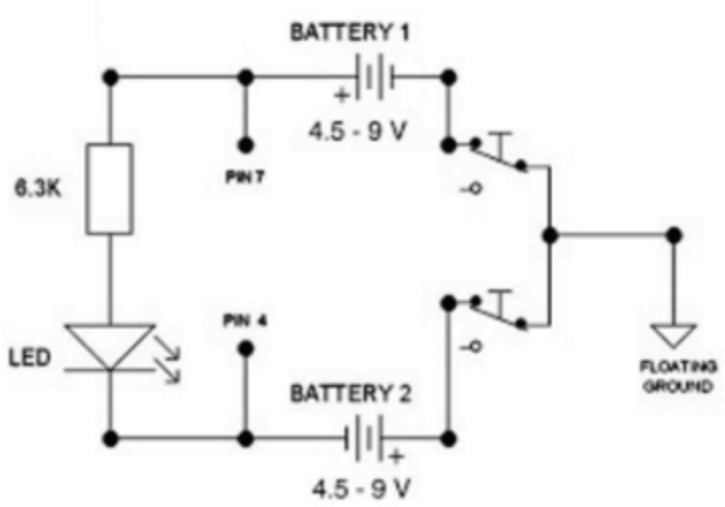
2.2.14 Switch

Switches used in order to control current on the circuit. There are total of 2 switches on the circuit. First one disconnects Battery 1(showed on the diagram) and second one disconnects Battery 2. They both need to pressed at the same time to prevent breaking the circuit.

2.3 Circuit Diagrams



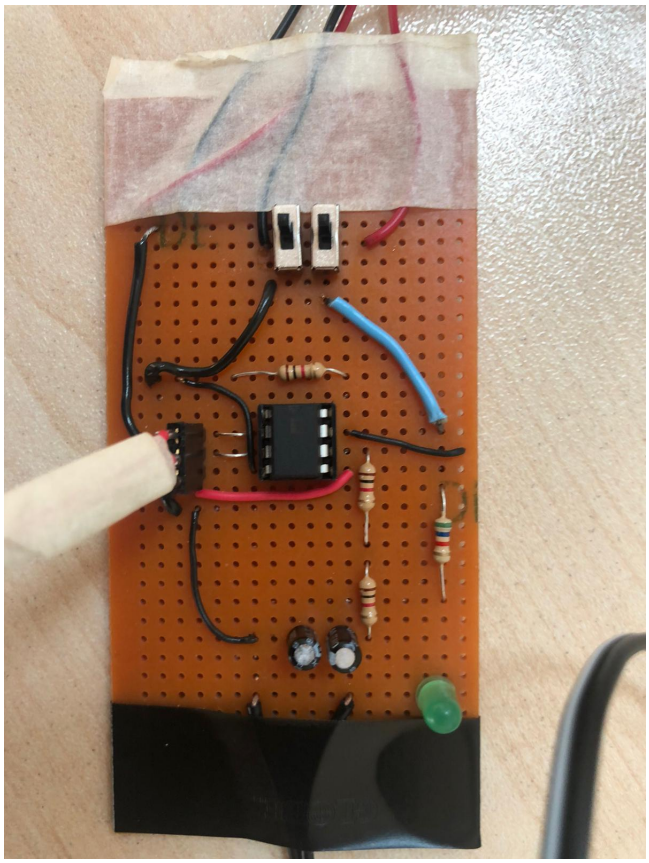
(Circuit Diagram - Analog Part)



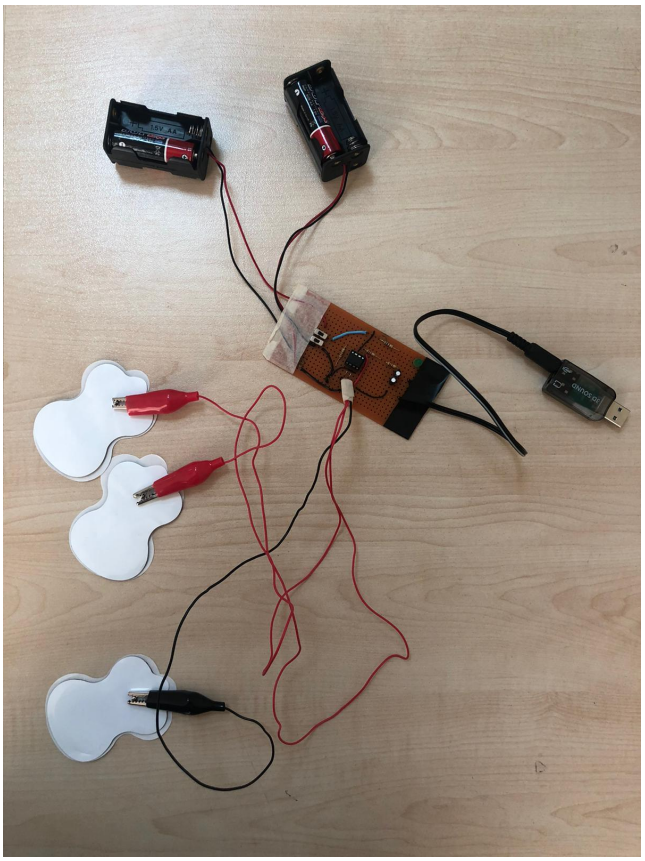
(Circuit Diagram - Battery)

2.4 How Our Hardware Looks Like?

Until the first presentation , we had done this amplificator circuit.

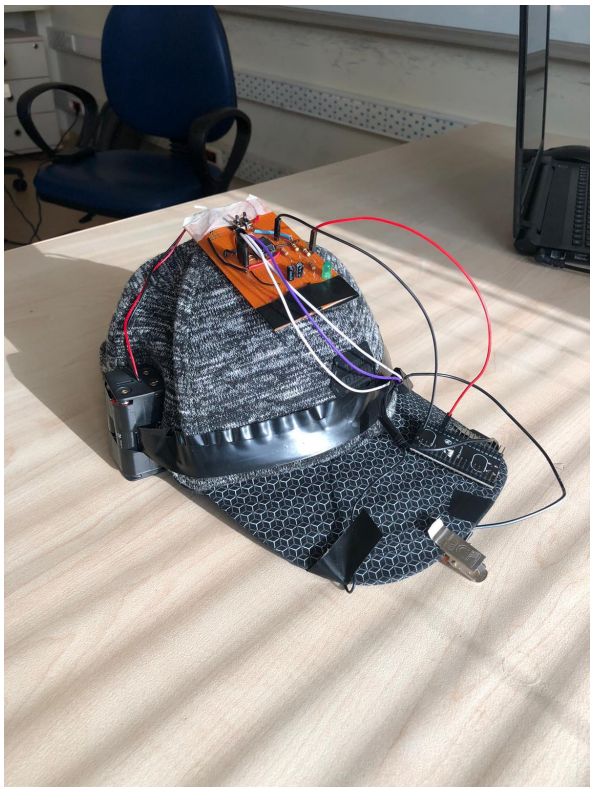


(Closer Look of Circuit)



(General Look of Circuit)

For the final presentation, we placed whole parts of hardware, which are esp32, copper tape, electrical tape, jumper wires, battery holder, battery and amplificator circuit to the head cap. It can be resized for each person. It looks like this below figures. Esp32 module provides to send 25x4 byte data per/sec to our user interface with bluetooth module.



(Front Side)

(Back Side)

On the back side of the module ,we placed the copper wires on the head hat so that they are on the forehead of the person.

3.Data Acquisition

Data can be any unprocessed fact, value, text, sound, or picture that is not being interpreted and analyzed. Data is the most important part of all Data Analytics, Machine Learning, Artificial Intelligence. Without data, we can't train any model and all modern research and automation will go in vain.

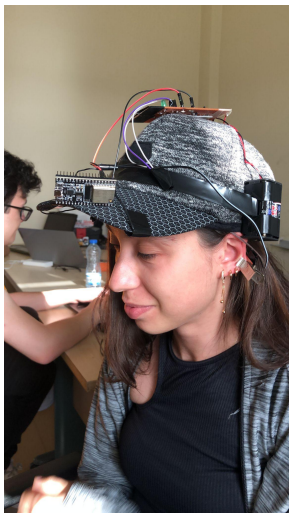
We needed data as we are going to apply machine learning techniques in our project. But we did not have the data we needed for our system. So created a script and collected data from 120 healthy people aged 18-24 years.

We received the signals with the copper bands on the forehead of the hat. We determined the copper band we put on the ear as a reference. In this way, we prevented external factors from affecting our signal.

In order to facilitate our work while collecting data, we asked the subjects to act simultaneously, adhering to the video we prepared beforehand. We set the total data acquisition time for one subject to be 80 seconds and synchronized data acquisition with video . We received 25 data per second, 250 data for each movement type, and 2000 data from one subject in total.

We asked them to focus on thinking they were closing their hands for the first 10 seconds, and relax for the next 10 seconds. Then the simulation part started. In the simulation, we first asked the subject to open and close his hand 5 times, and then to bend all his fingers 5 times, starting from the thumb.

While the subject was making movements, we received the signals generated in the brain by means of our sensor and recorded them in the file. A file was automatically created for each subject. We took advantage of the interface we created to understand whether we could receive the signals properly. When we realized that we received a wrong signal or that there was an error in the signal, we ended the data acquisition process and started the process again after removing the error. Also, while collecting data, we noticed that the subjects were surprised at the transitions between the movements, and we decided not to use the first movements for the movement types in order to avoid the erroneous data set. With these measures, we got a clean data set by purifying it from wrong, incomplete or erroneous data, such as data that occurs as a result of the people we collect data from doing something other than the actions we want from them.



Wearing hat

A photograph from data collection

A photograph from data collection

```
odak_repeat1:2928,2919,2927,2930,2928,2926,2927,2927,2927,2928,2928,2928,2928,2911,2928,2928,2928,2929,2928,2934,292
odak_repeat2:2930,2928,2928,2927,2927,2926,2925,2926,2928,2928,2927,2929,2927,2926,2927,2928,2928,2928,2928,2925,292
odak_repeat3:2928,2926,2938,2928,2925,2923,2929,2922,2924,2923,2927,2923,2923,2923,2927,2913,2925,2927,2923,2925,292
odak_repeat4:2922,2923,2924,2926,2923,2922,2921,2926,2927,2927,2923,2921,2923,2922,2925,2928,2917,2928,2927,2924,292
odak_repeat5:2928,2926,2925,2921,2925,2927,2923,2924,2923,2922,2928,2923,2921,2919,2923,2923,2926,2922,2921,2926,292
rahat_repeat1:2925,2925,2926,2919,2926,2923,2922,2927,2921,2928,2926,2923,2921,2921,2928,2928,2925,2925,2925,2926,29
rahat_repeat2:2927,2923,2922,2919,2914,2922,2927,2923,2925,2919,2921,2921,2918,2921,2922,2917,2921,2918,2923,2921,29
rahat_repeat3:2911,2927,2917,2915,2917,2917,2916,2909,2915,2913,2917,2912,2918,2924,2914,2922,2917,2921,2918,2917,29
rahat_repeat4:2918,2919,2919,2917,2910,2918,2922,2915,2920,2914,2912,2919,2916,2912,2914,2910,2911,2926,2919,2914,29
rahat_repeat5:2928,2910,2912,2911,2919,2911,2913,2908,2913,2911,2925,2918,2913,2913,2913,2915,2914,2914,2915,2912,29
ackapa_repeat1:2903,2909,2905,2911,2911,2908,2907,2906,2909,2910,2906,2907,2910,2909,2909,2905,2905,2903,2904,2910,2
ackapa_repeat2:2906,2901,2902,2901,2907,2908,2903,2907,2903,2905,2904,2901,2909,2902,2906,2908,2906,2906,2906,2908,2
ackapa_repeat3:2896,2903,2903,2898,2897,2901,2902,2907,2901,2902,2907,2901,2903,2903,2905,2902,2903,2907,2901,2901,2
ackapa_repeat4:2905,2903,2903,2899,2907,2907,2903,2907,2909,2909,2902,2905,2912,2902,2907,2910,2905,2913,2905,2899,2
ackapa_repeat5:2903,2911,2901,2903,2911,2910,2901,2911,2903,2901,2911,2909,2911,2907,2902,2902,2911,2903,2905,2907,2
parmak1_repeat1:2903,2907,2909,2911,2904,2906,2911,2905,2903,2903,2907,2906,2900,2908,2909,2901,2905,2903,2903,2918,
parmak1_repeat2:2904,2907,2908,2905,2914,2906,2905,2908,2909,2901,2908,2911,2910,2906,2906,2907,2902,2907,2902,2909,
parmak1_repeat3:2909,2914,2919,2919,2922,2919,2917,2914,2913,2912,2917,2913,2911,2918,2915,2911,2914,2912,2910,2913,
parmak1_repeat4:2914,2914,2913,2917,2917,2918,2918,2914,2916,2917,2918,2921,2916,2918,2919,2917,2914,2925,2917,2920,
parmak1_repeat5:2919,2919,2923,2926,2918,2922,2922,2922,2926,2919,2931,2919,2917,2923,2927,2923,2926,2925,2919,2924,
parmak2_repeat1:2922,2927,2925,2922,2925,2917,2925,2927,2921,2929,2925,2921,2919,2959,2927,2922,2927,2927,2926,2923,
parmak2_repeat2:2900,2923,2917,2922,2927,2919,2918,2935,2928,2927,2927,2907,2918,2927,2919,2926,2917,2921,2901,2919,
parmak2_repeat3:2923,2916,2921,2929,2928,2923,2927,2928,2911,2921,2921,2925,2921,2915,2928,2925,2922,2918,2923,2925,
parmak2_repeat4:2929,2922,2940,2918,2923,2917,2928,2928,2927,2923,2923,2927,2914,2923,2924,2927,2917,2919,2926,2932,
parmak2_repeat5:2919,2919,2917,2926,2933,2919,2928,2920,2923,2923,2930,2921,2927,2918,2922,2924,2923,2919,2928,2934,
parmak3_repeat1:2932,2927,2923,2935,2926,2927,2928,2927,2927,2930,2926,2928,2927,2925,2928,2928,2926,2928,2928,2928,
parmak3_repeat2:2923,2928,2928,2924,2928,2928,2929,2926,2929,2929,2929,2928,2931,2926,2930,2931,2929,2932,2930,2928,
parmak3_repeat3:2924,2926,2925,2928,2929,2927,2928,2929,2933,2927,2929,2931,2928,2929,2928,2928,2928,2928,2931,2928,2941,
parmak3_repeat4:2934,2929,2934,2930,2930,2934,2933,2931,2931,2934,2937,2930,2933,2935,2928,2934,2933,2930,2934,2930,
parmak3_repeat5:2931,2944,2931,2932,2939,2934,2939,2943,2938,2941,2930,2933,2937,2933,2934,2939,2935,2938,2923,2939,
parmak4_repeat1:2943,2934,2933,2937,2945,2943,2941,2942,2947,2942,2942,2941,2942,2944,2943,2938,2945,2942,2945,2947,
parmak4_repeat2:2945,2944,2944,2946,2947,2943,2944,2941,2945,2946,2944,2946,2941,2948,2946,2944,2937,2945,2944,2946,
parmak4_repeat3:2946,2945,2950,2948,2945,2946,2941,2946,2943,2945,2947,2954,2945,2946,2947,2945,2949,2944,2949,2934,
parmak4_repeat4:2944,2943,2944,2947,2946,2944,2944,2944,2944,2943,2953,2943,2941,2947,2944,2944,2946,2944,2944,2945,
parmak4_repeat5:2947,2945,2944,2949,2944,2950,2946,2945,2945,2951,2948,2941,2943,2947,2949,2945,2945,2944,2946,2943,
```

Collected Data File Example

4.Signal Conditioning

Most signals require some form of manipulation that prepares them for the next stage of processing. This is a crucial process for making analogue signals like temperature and vibration intelligible to data acquisition systems or control equipment. If an incoming signal is not optimized for the in-line digitizer via signal conditioning, it can result in measurement inaccuracies and suboptimal levels of performance.

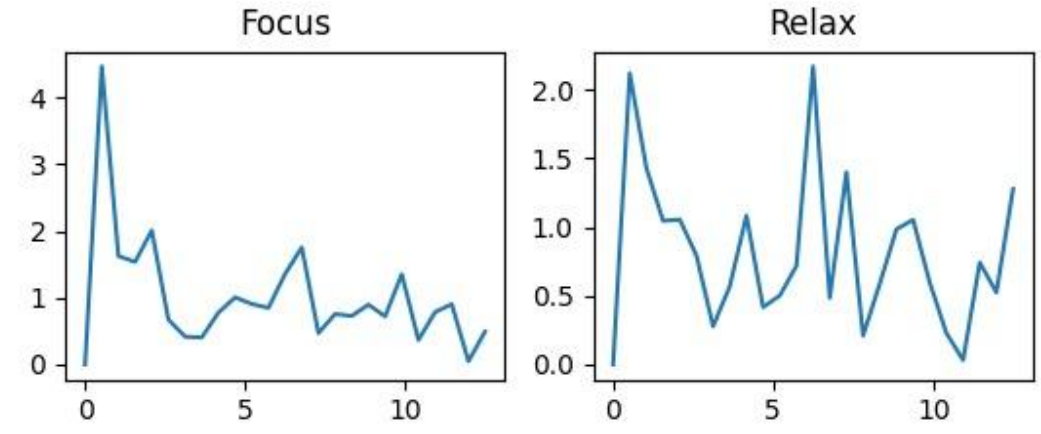
Signal conditioning needs vary in capabilities depending on your sensor. Different measurement types and sensor architectures have different signal conditioning requirements. For example, low-voltage analogue signals will typically need to be amplified and subsequently filtered to reduce background noise before digitization. In our project, we also use an amplifier for amplification and filtering.

4.1 Fast Fourier Transform

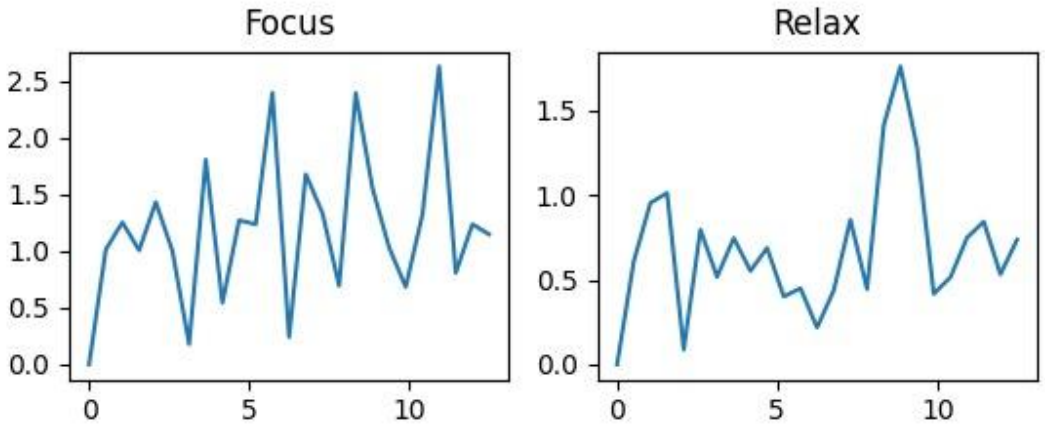
A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

Since the signals we receive from the time domain in our system cannot produce healthy outputs in machine learning, we need to convert them to the frequency domain. We used FFT for this conversion. When we insert the data we received from the subjects into FFT, we can make a healthier data analysis by comparing the data generated for each subject.

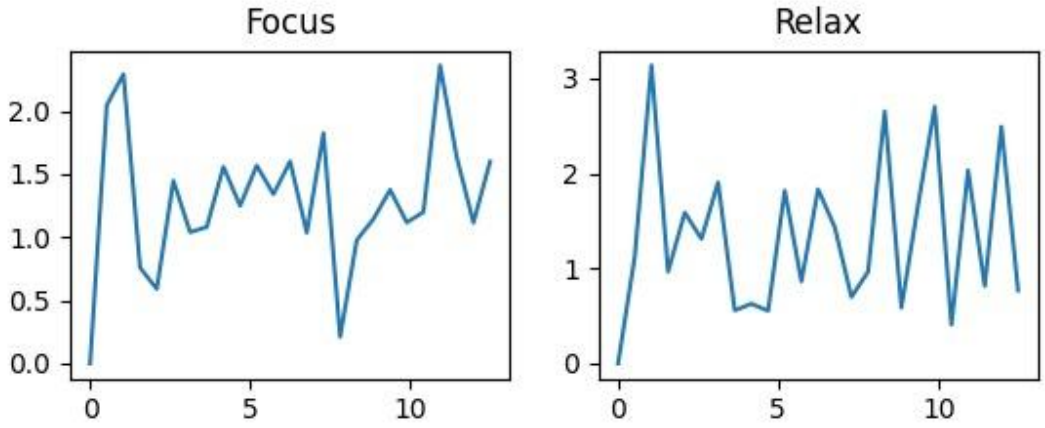
4.1.1 FFT Graphic Examples



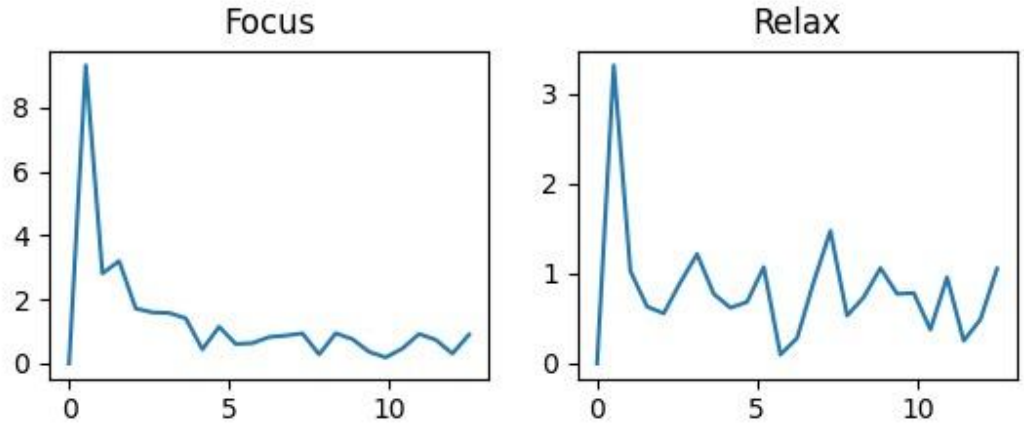
(Subject 1)



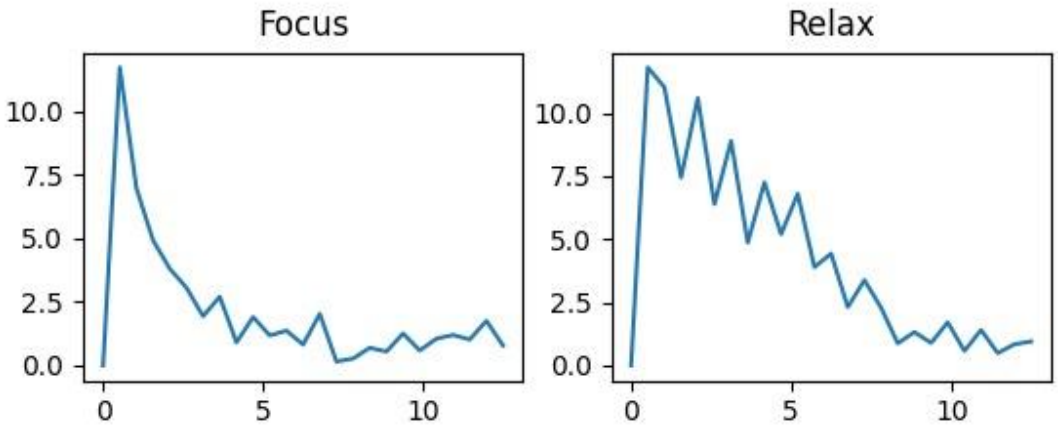
(Subject 2)



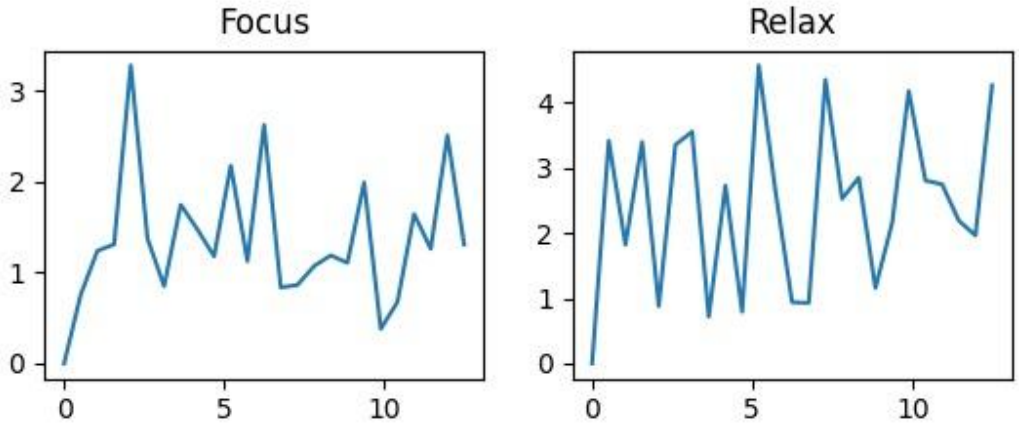
(Subject 3)



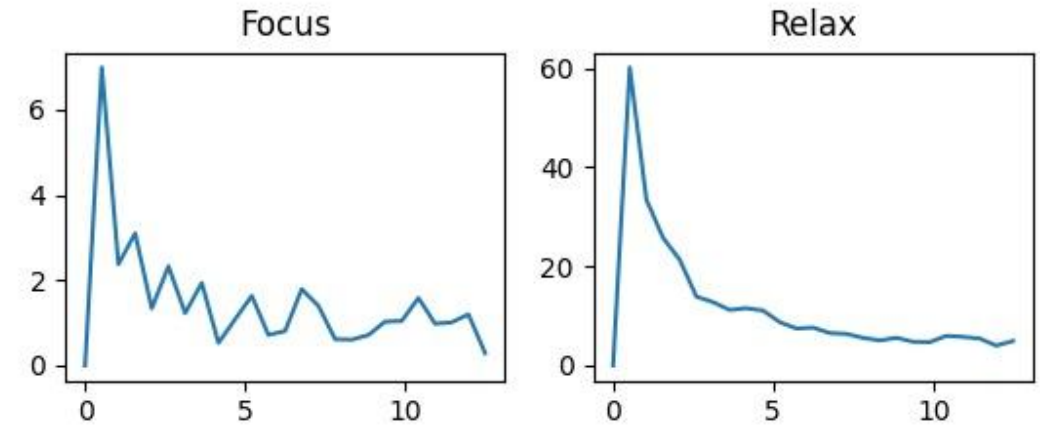
(Subject 4)



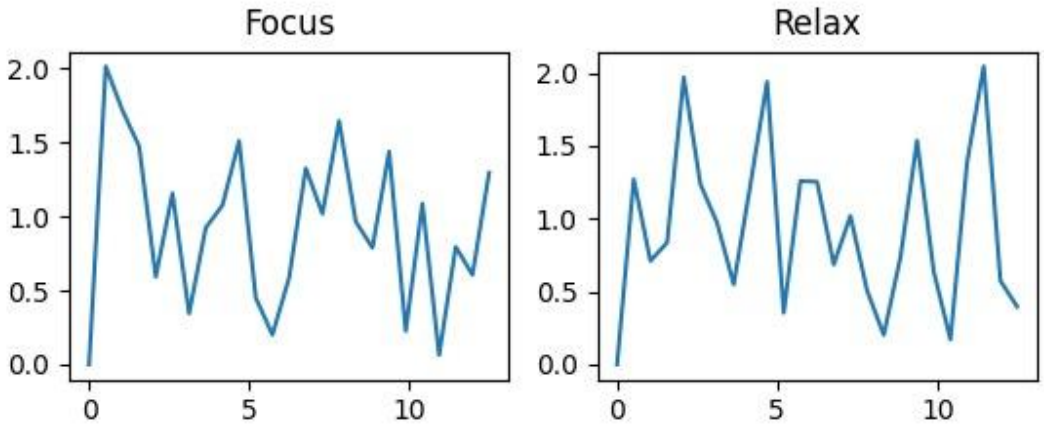
(Subject 5)



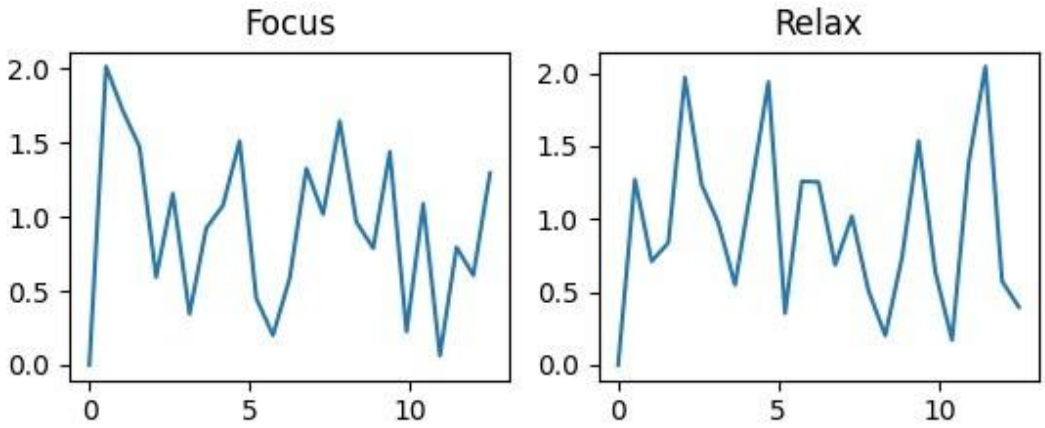
(Subject 6)



(Subject 7)



(Subject 8)



(Subject 9)

5.Data Processing

5.1 Data Pre-Processing

After getting enough data we need to process those to make them meaningful. We preferred using machine learning solutions. We used keras python library for this purpose. All data from every person is in a file separately and results are labeled as “odak” or “rahat”. We did a pre-processing to make all the data ready for the learning model.

All data files are revised with a small script. The small changes have been made to make all things ready. After all changes, all data is merged in a dataframe which is about 4500 samples. The results are encoded. Moreover the signal values were very close to each other, so we did z score normalization.

5.2 Data Processing Models (For Fingers, Relax, Focus)

We experienced several number of models for trainings to see results :

- **Decision Tree (14.3 %)**

```
Decission Tree

In 11
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (precision_score, recall_score,f1_score, accuracy_score,mean_squared_error,mean_absolute_error, roc_curve,
 classification_report,auc)


model = DecisionTreeClassifier()
model.fit(x_train, y_train)


y_pred = model.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred , average="weighted")
precision = precision_score(y_test, y_pred , average="weighted")
f1 = f1_score(y_test, y_pred, average="weighted")

print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("recall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)

accuracy
0.143
precision
0.145
recall
0.143
f1score
0.143
```

- **KNN (15.9 %)**

KNN

```
In 12
1 from sklearn.neighbors import KNeighborsClassifier
2 knn = KNeighborsClassifier(n_neighbors = 3, p = 2)
3 knn.fit(x_train, y_train)
4 print("Accuracy: {}".format(knn.score(x_test,y_test)))
```

```
Accuracy: 0.1591304347826087
```

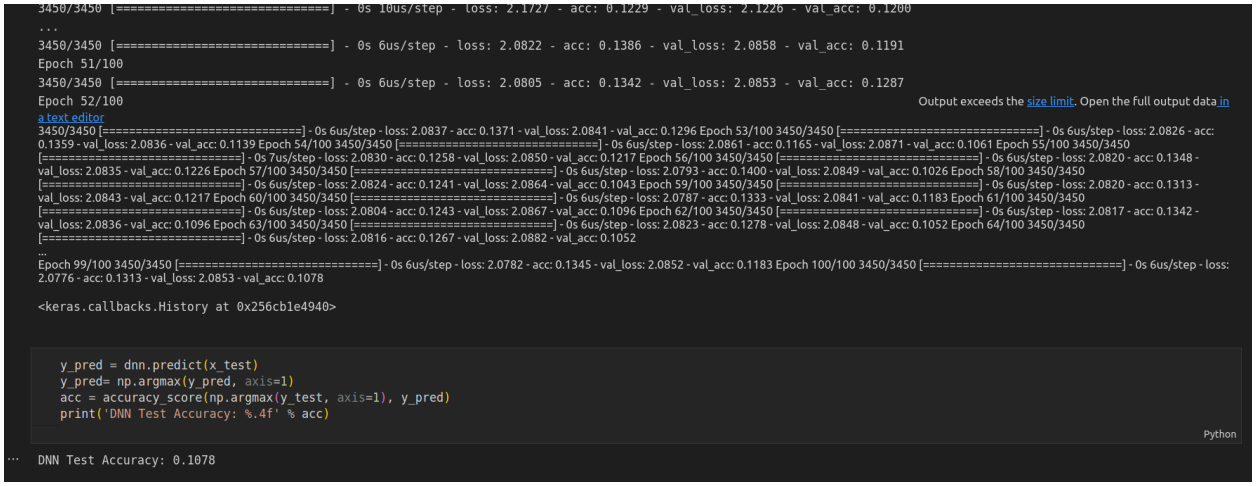
- **Best Random Forest Model (13.6 %)**

Best Random Forest Model

```
In 13
1 import numpy as np
2 from sklearn.ensemble import RandomForestClassifier
3 clf = RandomForestClassifier(max_depth=5, n_estimators=300)
4
5 clf.fit(x_train, y_train)
6
7 clf.score(x_test, y_test)
```

```
Out 13
0.13652173913043478
```

- **DNN (10.78 %)**



```
3450/3450 [==============================] - 0s 100s/step - loss: 2.1727 - acc: 0.1229 - val_loss: 2.1226 - val_acc: 0.1200
...
3450/3450 [==============================] - 0s 6us/step - loss: 2.0822 - acc: 0.1386 - val_loss: 2.0858 - val_acc: 0.1191
Epoch 51/100
3450/3450 [==============================] - 0s 6us/step - loss: 2.0805 - acc: 0.1342 - val_loss: 2.0853 - val_acc: 0.1287
Epoch 52/100
```

Output exceeds the size limit. Open the full output data in a text editor

```
3450/3450 [==============================] - 0s 6us/step - loss: 2.0837 - acc: 0.1371 - val_loss: 2.0841 - val_acc: 0.1296 Epoch 53/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0826 - acc: 0.1359 - val_loss: 2.0836 - val_acc: 0.1139 Epoch 54/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0861 - acc: 0.1165 - val_loss: 2.0871 - val_acc: 0.1061 Epoch 55/100 3450/3450 [==============================] - 0s 7us/step - loss: 2.0830 - acc: 0.1258 - val_loss: 2.0850 - val_acc: 0.1217 Epoch 56/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0820 - acc: 0.1348 - val_loss: 2.0835 - val_acc: 0.1226 Epoch 57/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0793 - acc: 0.1400 - val_loss: 2.0849 - val_acc: 0.1026 Epoch 58/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0824 - acc: 0.1241 - val_loss: 2.0864 - val_acc: 0.1043 Epoch 59/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0820 - acc: 0.1313 - val_loss: 2.0843 - val_acc: 0.1217 Epoch 60/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0787 - acc: 0.1333 - val_loss: 2.0841 - val_acc: 0.1183 Epoch 61/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0804 - acc: 0.1243 - val_loss: 2.0867 - val_acc: 0.1096 Epoch 62/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0817 - acc: 0.1342 - val_loss: 2.0836 - val_acc: 0.1096 Epoch 63/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0823 - acc: 0.1278 - val_loss: 2.0848 - val_acc: 0.1052 Epoch 64/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0816 - acc: 0.1267 - val_loss: 2.0882 - val_acc: 0.1052
...
Epoch 99/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0782 - acc: 0.1345 - val_loss: 2.0852 - val_acc: 0.1183 Epoch 100/100 3450/3450 [==============================] - 0s 6us/step - loss: 2.0776 - acc: 0.1313 - val_loss: 2.0853 - val_acc: 0.1078

<keras.callbacks.History at 0x256cb1e4940>
```

```
y_pred = dnn.predict(x_test)
y_pred= np.argmax(y_pred, axis=1)
acc = accuracy_score(np.argmax(y_test, axis=1), y_pred)
print('DNN Test Accuracy: %.4f' % acc)
```

Python

```
DNN Test Accuracy: 0.1078
```

As can be seen from these results. The highest train success is about 16 %. This shows us that the data is not correct or errored. Because the success values are very low. To improve this we may prefer a better professional device to get signals.

After compiling the model we made a real-time working script that communicates with esp-32 to get signals and puts all 50 inputs to the model and redirects the output to the unity simulation and simulation shows the finger movements.

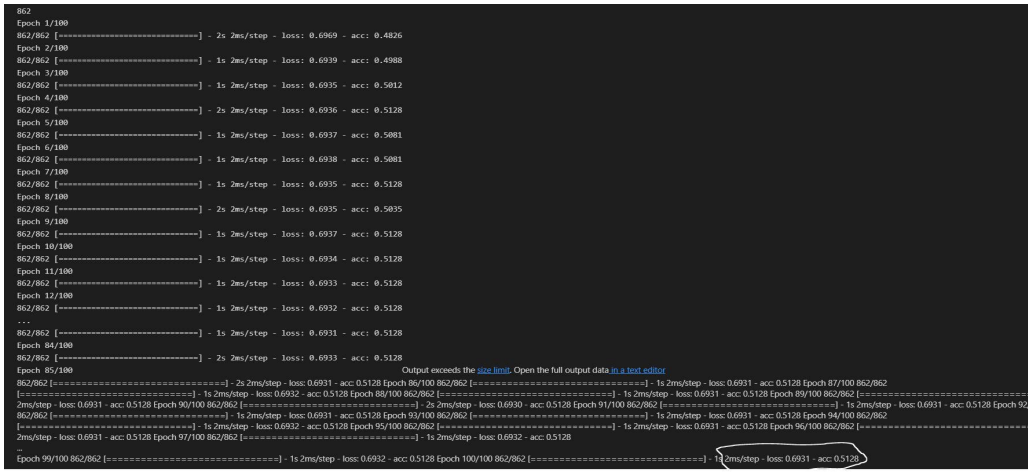
5.3 Data Processing Model (For only Relax and Focus Situation)

We used the same algorithms only for focus and relaxation states.

- **DNN (Accuracy : 0.5128)**

```
from keras.models import Sequential
from keras.layers import Dense

x_train = np.array(x_train)
y_train = np.array(y_train)

print(len(x_train))

model = Sequential()
model.add(Dense(500, activation='relu', input_dim=50))
model.add(Dense(100, activation='relu'))
model.add(Dense(50, activation='relu'))
#model.add(Dense(1, activation='softmax'))
model.add(Dense(2, activation='softmax'))

# opt = keras.optimizers.SGD(learning_rate=0.001)
model.compile(optimizer='Adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=100, batch_size=2)
```

Python



```
862
Epoch 1/100
862/862 [==============================] - 2s 2ms/step - loss: 0.6969 - acc: 0.4826
Epoch 2/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6939 - acc: 0.4988
Epoch 3/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6935 - acc: 0.5012
Epoch 4/100
862/862 [==============================] - 2s 2ms/step - loss: 0.6936 - acc: 0.5128
Epoch 5/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6937 - acc: 0.5081
Epoch 6/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6938 - acc: 0.5081
Epoch 7/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6935 - acc: 0.5128
Epoch 8/100
862/862 [==============================] - 2s 2ms/step - loss: 0.6935 - acc: 0.5035
Epoch 9/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6937 - acc: 0.5128
Epoch 10/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6934 - acc: 0.5128
Epoch 11/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6933 - acc: 0.5128
Epoch 12/100
862/862 [==============================] - 1s 2ms/step - loss: 0.6932 - acc: 0.5128
...
862/862 [==============================] - 1s 2ms/step - loss: 0.6931 - acc: 0.5128
Epoch 84/100
862/862 [==============================] - 2s 2ms/step - loss: 0.6933 - acc: 0.5128
Epoch 85/100                                    Output exceeds the size limit. Open the full output data in a text editor
862/862 [==============================] - 2s 2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 86/100 862/862 [==============================] - 1s 2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 87/100 862/862
[==============================] - 1s 2ms/step - loss: 0.6932 - acc: 0.5128 Epoch 88/100 862/862 [==============================] - 1s 2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 89/100 862/862 [==============================
2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 90/100 862/862 [==============================] - 2s 2ms/step - loss: 0.6930 - acc: 0.5128 Epoch 91/100 862/862 [==============================] - 1s 2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 92
862/862 [==============================] - 1s 2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 93/100 862/862 [==============================] - 1s 2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 94/100 862/862
[==============================] - 1s 2ms/step - loss: 0.6932 - acc: 0.5128 Epoch 95/100 862/862 [==============================] - 1s 2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 96/100 862/862 [==============================
2ms/step - loss: 0.6931 - acc: 0.5128 Epoch 97/100 862/862 [==============================] - 1s 2ms/step - loss: 0.6932 - acc: 0.5128
...
Epoch 99/100 862/862 [==============================] - 1s 2ms/step - loss: 0.6932 - acc: 0.5128 Epoch 100/100 862/862 [==============================] - 1s 2ms/step - loss: 0.6931 - acc: 0.5128
```

- Additionally, we applied FFT to our data and trained it in our models.

-DNN (Accuracy : 0.5139)

```
from scipy import pi
from scipy.fftpack import fft
def timeTofreq(data, i):

    arrayData = np.array(data.iloc[i])
    meanData = np.mean(arrayData)
    normalizedData = arrayData - meanData

    freq_data = fft(normalizedData)
    return 2/N * np.abs (freq_data)
```

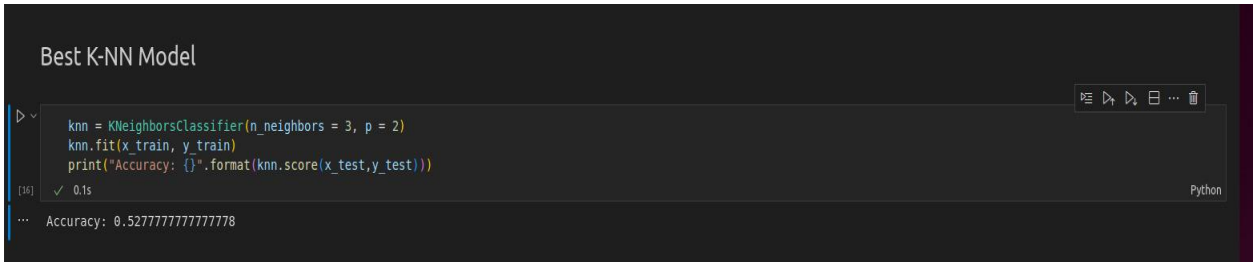
```
from sklearn.model_selection import train_test_split
df = pd.read_csv('all_data.csv', sep = ',', header=None)

odak = df[df[50] == 'odak'].iloc[:,0:50]
rahat = df[df[50] == 'rahat'].iloc[:,0:50]


fodak = []
frahat = []

for i in range(0, len(odak)):
  fodak.append(timeTofreq(odak, i))

for i in range(0, len(rahat)):
  frahat.append(timeTofreq(rahat, i))

arr = [0]*len(fodak)

for i in range(len(frahat)):
  arr.append(1)


#print(arr)

arr = np.array(arr)
print(arr)
train = fodak + frahat
train  = np.array(train)

x_train, x_test, y_train, y_test = train_test_split(train, arr, test_size= 0.25, random_state=42)
```

```
    #loaded_model = pickle.load(open(filename, 'rb'))
    loaded_model = model
    #result = loaded_model.score(x_test, y_test)
    print(loaded_model.evaluate(x_test, y_test))

    predict_prob=model.predict([x_test])
    predict_classes=np.argmax(predict_prob,axis=1)

    #print(len(x_test))
    for i in range(len(predict_classes)):
        print("Actual: ", y_test[i], ",    Predicted: ", predict_classes[i])
    print(predict_classes)
    #print(len(predict_classes))

    for result in predict_prob:
        print(str(result))
```

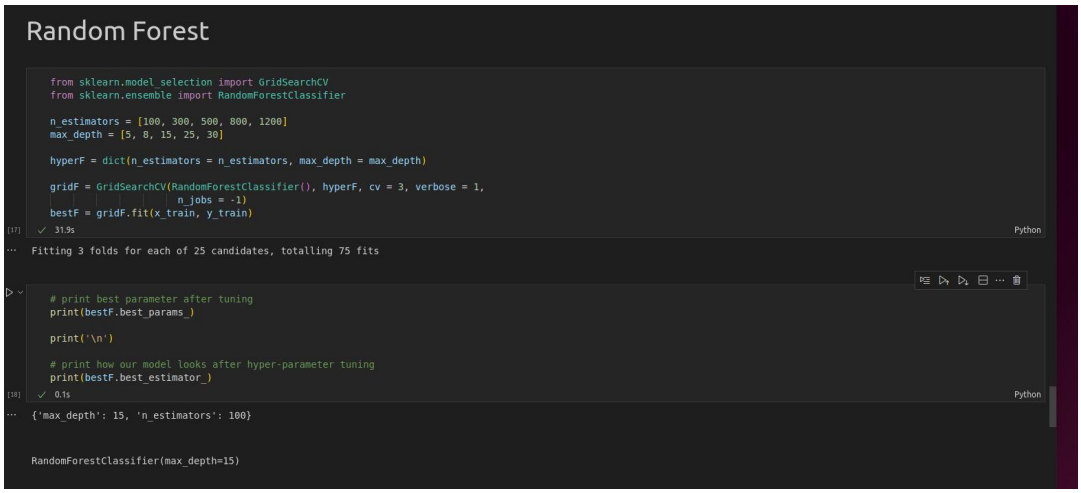
```
[18]
...  Output exceeds the size limit. Open the full output data in a text editor
     9/9 [==============================] - 0s 3ms/step - loss: 3.6899 - accuracy: 0.5139
     [3.6898531913757324, 0.5138888955116272]
     Actual:  0 ,    Predicted:  0
     Actual:  1 ,    Predicted:  1
     Actual:  0 ,    Predicted:  0
     Actual:  0 ,    Predicted:  0
     Actual:  1 ,    Predicted:  1
     Actual:  0 ,    Predicted:  1
     Actual:  1 ,    Predicted:  0
     Actual:  0 ,    Predicted:  1
     Actual:  0 ,    Predicted:  1
     Actual:  0 ,    Predicted:  0
     Actual:  1 ,    Predicted:  1
     Actual:  1 ,    Predicted:  1
     Actual:  1 ,    Predicted:  1
     Actual:  0 ,    Predicted:  1
     Actual:  1 ,    Predicted:  0
     Actual:  0 ,    Predicted:  0
```

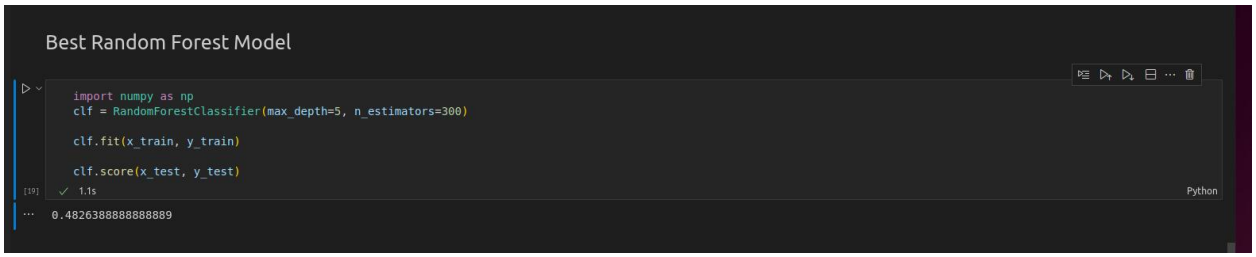
-KNN (Accuracy : 0.59375)

-BEST KNN (Accuracy : 0.52777)



-RANDOM FOREST (Accuracy : 0.48263)





As a result, we got the best result in the KNN model(0.59375).

6.Communication (Not used in final presentation)

6.1 Brief Summary

The EEG signals from the brain should be processed on the computer and the output should be sent to unity side by python.This section summarizes the simultaneous transfer of data processed in python to the unity. This connection can be made with Python Pyautogui library.

6.2 PyAutoGUI

PyAutoGUI lets your Python scripts control the mouse and keyboard to automate interactions with other applications. The API is designed to be simple.

PyAutoGUI has several features:

- Moving the mouse and clicking in the windows of other applications. Sending keystrokes to applications (for example, to fill out forms).
- Take screenshots, and given an image (for example, of a button or checkbox), and find it on the screen.
- Locate an application's window, and move, resize, maximize, minimize, or close it (Windows-only, currently).
- Display alert and message boxes.

6.3 Required Connections In Unity and Python

Connections To Be Made In Unity: (Not used in final presentation)

Assigning the necessary keys to the necessary hand movements:

- **Z :** little finger
- **X :** ring finger
- **C :** middle finger
- **V :** index finger
- **Space :** thumb
- **B :** the opposite of the case
- **G :** close all
- **H :** open all

Connections To Be Made In Python: (Not used in final presentation)

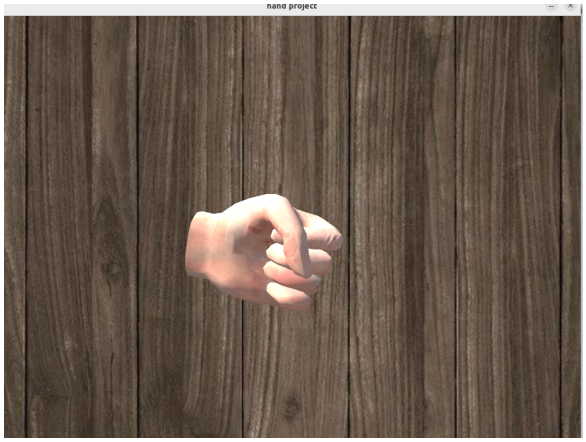
- Importing the necessary socket library.
- Assigning the keys for the necessary hand movements.

7.Simulation Design Module (Not used in final presentation)

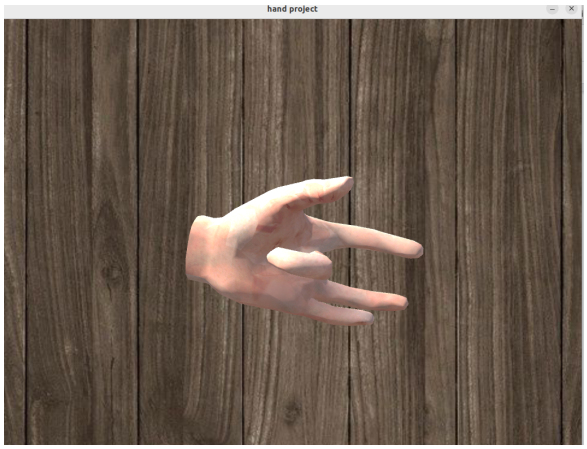
7.1 Unity

In the hardware part, hand modeling was done with unity. Each finger can be opened and closed separately. Our simulation depends on five inputs. These inputs are taken from the machine learning model used in the data processing part of the project and each input is used to control different fingers. When the input is received, the relevant finger is closed and then opened again. These entries are assigned to some keys on the keyboard and it is assumed that the keys are pressed according to the input in our C# code.

Typical hand movement:




hand project

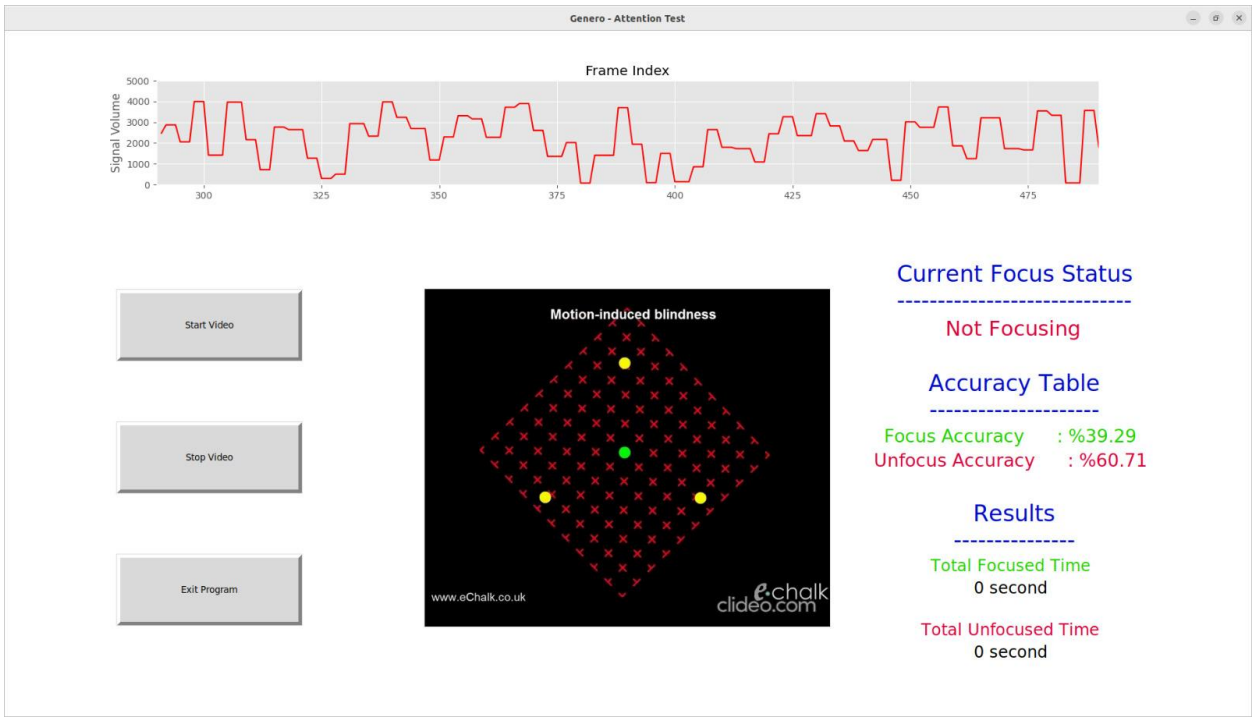

hand project


nana project


nana project

8.User Interface

We have an interface that communicates with the prepared hardware. Our customer enters the focus test with the interface, and we show our customers the results of the analysis of the focusing and relaxing situations.



9.Timeline of Project

Between 5-6th week, the hardware team put together all necessary hardware and tried to receive the first data from a brain and succeed.

Meanwhile, the software team started to develop the virtual hand. We made 6 different inputs coming from hardware and these inputs processed throughout our code and finally shown as a result on the virtual hand(a unity program).

After finishing the hardware, the communication group tried to make a connection between the hardware and software. In early stages of the semester, inputs don't have to produce meaningful output on virtual hands but the important part is, sending signals from the hardware to software.

After passing this stage successfully, we started to testing part. We selected hundreds of different people and asked them to move their fingers one by one. Then the testing group categorized these signals for each finger.

After finishing the tests, the machine learning part came to play. This test stage probably lasted for 1 week, in that time we improved our software(virtual hand) and EEG device.

In the last week, we changed our project from hand to only focus. So in this stage, we trained our model with focus-relax data. Then we implemented a new user interface program to use model which we created.

10.Conclusion

As technology improves and we understand how our brains work, the impact of EEG devices on our life will improve. Even with today's technology, EEG devices are used in many different fields. Even though our hardware lacks of meaningful data to use, with more sufficient(expensive) hardwares it can have valuable data pool and can be a life-changing opportunity for people who suffer such disabilities. With further development and access to better equipment, it is possible to make a movement on a real prosthetic hand. Possible use cases for EEG device used in this project can be listed as; prosthetic limbs for hand, leg, arm, individual fingers, VR environments, gaming, etc.

References

- Ricker, E. (2015). *Eeg: cheap and small.*

  printing and building toward the portable, printable, personal brain lab (incorporate with harry and also 3d print my own mri of brain)

- Noronha, C. (2022, April 08). *Differences between eeg, nirs, fmri and meg.*

  https://www.brainlatam.com/blog/differences-between-eeg-nirs-fmri-and-meg-924

- Front. Neurosci. (2013, December 26). *Meg and eeg data analysis with mne-python*

  https://www.frontiersin.org/articles/10.3389/fnins.2013.00267/full

- Jas, M. , Gramfort, A. , Engemann, D. (2021, October 01). *Process meg/eeg data with plotly in python/v3*

  https://plotly.com/python/v3/ipython-notebooks/mne-tutorial/

- Newman, A. (2020-2021). *Artifacts in eeg data.*

  https://neuraldatascience.io/7-eeg/erp_artifacts.html

- Kumar, V.(2020, June 7). *Processing eeg data with python.*

  https://medium.com/@vishal.mi19/processing-eeg-data-with-python-8036fd7336ca

- Goektepe, Pinar. (2019, September 19). *Visualize eeg data.*

  https://neuro.inf.unibe.ch/AlgorithmsNeuroscience/Tutorial_files/DataVisualization.html

- Anderson, C. (2013). *Getting started with eeg data.*

  https://www.cs.colostate.edu/eeg/data/json/doc/tutorial/_build/html/artifacts.html