CSE1015 – Essentials of Machine Learning

J Component Report

A project report titled

Gesture Recognition Using Machine Learning

By

19BAI1161 MADHUKAR TEMBA

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGENEERING

Submitted to

Dr. R. Rajalakshmi

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

November 2020

DECLARATION BY THE CANDIDATE

I hereby declare that the report titled "***Gesture Recognition Using Machine Learning***" submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Dr. R. Rajalakshmi, Associate Professor, SCOPE, Vellore Institute of Technology, Chennai.**

X

Madhukar Temba
Student

Signature of the Candidate

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. R. Rajalakshmi,** School of Computer Science and Engineering for her consistent encouragement and valuable guidance offered to us throughout the course of the project work.

We are extremely grateful to **Dr. R. Jagadeesh Kannan, Dean,** School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our **Head of the Department** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Gesture Recognition Using Machine Learning"** is a bona-fide work of **Madhukar Temba (19BAI1161)** carried out the "J"-Project work under my supervision and guidance for **CSE1015 – Essentials of Machine Learning**

Dr. R. Rajalakshmi

SCOPE

TABLE OF CONTENTS

ABSTRACT

The aim is to design a handheld device with an inertial measurement unit to record hand gestures made by the person and classify them using machine learning.
The device also has Wi-Fi and an IR transmitter to communicate with computers and televisions. It will send the command to the computer/television according to the gesture made.

For that we use the remake of the Tensor Flow Lite library for the Arduino by Eloquent.

The outcome of the project will be a handheld device which the user moves around to make gestures which will be recognised and sent to the receiving device wirelessly.

INTRODUCTION

There are practical reasons you might want to squeeze ML on microcontrollers, including:

- Function — wanting a smart device to act quickly and locally (independent of the Internet).
- Cost — accomplishing this with simple, lower cost hardware.
- Privacy — not wanting to share all sensor data externally.
- Efficiency — smaller device form-factor, energy-harvesting or longer battery life.

There’s a final goal which we’re building towards that is very important:

- Machine learning can make microcontrollers accessible to developers

who don’t have a background in embedded development.

On the machine learning side, there are techniques you can use to fit neural network models into memory constrained devices like microcontrollers. One of the key steps is the quantization of the weights from floating point to 8-bit integers. This also has the effect of making inference quicker to calculate and more applicable to lower clock-rate devices.

TinyML is an emerging field and there is still work to do — but what’s exciting is there’s a vast unexplored application space out there. Billions of microcontrollers combined with all sorts of sensors in all sorts of places which can lead to some seriously creative and valuable Tiny ML applications in the future.

So, we will use this machine learning technique to make a device which will

recognise the gesture made by moving the device.

Related Works

1. GESTURE BASED MOTION TRACKING KEYBOARD USING MACHINE LEARNING

by 1.TENEPALLI SOWMYA,2. UDARI NARESH,3. VEERABOMMA MANASA,4. RANGAM RAGINI 1,3&4.UG SCHOLAR, 2. ASSISTANT PROFESSOR DEPARTMENT OF ECE, AVN INSTITUTE OF ENGINEERING AND TECHNOLOGY, KOHEDA ROAD, IBRAHIMPATNAM(M), R.R.DIST-501510, HYDERABAD

https://www.researchgate.net/publication/340599168_GESTURE_BASED_MOTION_TRACKING_KEYBOARD_USING_MACHINE_LEARNING

In spite of the emergence of various sensors and input devices, the current methods have not been able to provide an accurate recognition system for gestures. Recognition system such as the one in Kinect using machine learning algorithms shows some of the use cases for these gestures to newer system and domains. The one in our paper provides a far more accurate result along with faster computational speed as it uses the support vector machine (SVM) algorithm and neural networks. This motion recognition system tracks motion made in mid-air by device in a 3D space, log its speed, angular velocity, distance covered and some other variables at real time, and in turn convert the device captured data of motion into characters of English alphabets. The device presented in this paper presents solution based on support vector algorithms and discusses about some concepts raised from the device.

INTRODUCTION The most common techniques in the communication between the human and computer are

performed via simple devices such as mouse and keyboard. Automated gesture tracking devices can serve as a very efficient tool in the case to serve social purpose like the sign language communication for the deaf and verbally impaired person. Although the current devices are intended for easy and user friendly interaction with the user, still they lack when there is a massive data input at some reasonable speed. For these reasons, we need some kind of a new input method working at a greater degree as people communicate that is through gestures. The problem of understanding the sign language, besides being challenging, can be of great social interest as it would be helpful to the deaf and verbally impaired people to get discrete, personal and effective service in the day-to-day life. We envision a system to track gestures made in mid-air, where a computer will automatically translate the motion made by the device by the user to characters of the English alphabet system. The problem of gesture recognition deals with the Alochana Chakra Journal Volume IX, Issue IV, April/2020 ISSN NO:2231-3990 Page No:507detection, analysis and recognition of character from sequence of motion of device. The hardware needed is not complex and inexpensive as it consists of an Arduinopro micro, an mpu-6050 accelerometer and an hc-06 module for Bluetooth communication. Recognition of gestures is a compound problem that has widely attracted attention. Initial methods for this recognition considered of setups fit only for laboratories, i.e. finger movements being converted into voltage signal sent to the computer via a wired glove being worn by a person. Obviously, these kinds of setups are not at all appropriate for general usage as limitations such as the length of the wire. Later, the glove

was replaced by a camera like the Microsoft Kinect 3D which was very costly. We put forth a method for gesture recognition via motion tracking as our step to solve these kinds of problems. Our device is a gesture tracking device which uses machine learning's support vector machine model to convert accelerometer data to a sequence of alphabets. The main aim of the project is to build a device using an Arduino pro micro that translates gestures into words wirelessly.

METHEDOLOGY Flask the sketch to Arduino, open the serial monitor in the Arduino IDE and press the button on Arduino, the reading of gesture would start. Now everything will be working, so we are ready to use the library. We will start by deleting the data folder and its contents and replace it with a new created dataset. This library has been designed to make a keyboard such that each gesture is associated with a character also being case sensitive. This means we can teach the algorithm about 53 different gestures. Start a new recording batch, by recording a sample for a specific gesture. Open the terminal and type: python start.py target=a: 0 port=COM6

- The "target" argument will tell module that we are recording a new sample for a gesture.
- The "a" character characterizes a gesture, and should be unique as it has length of 1 for every gesture .
- The "0" character is the batch number which has to be different each time when we register a new batch so that overriding of sample is avoided.

- The "port" represents the serial port to which Arduino is connected to.

When start.py is running, recording of different gestures by pressing and releasing the button on the Arduino is done. When the button is pressed, recording of data from the accelerometer is done. Every sample gets saved in the data folder as a different file. After this the training of model is done. When dataset is ready, we can use it to train our algorithm. For this open the terminal and write: python learn.py To check out if the model is working, open the terminal and type: python start.py port= predictNo we can make gestures and see it getting predicted correctly. In Python, scikit is a popular library which is used to implement machine learning algorithm, Support Vector Machine is also available in scikit library and follows the structure same as object creation, import library, fitting model and prediction.

APPLICATIONS

- Movements and gestures detection
- Angular rate detection
- User interface which is motion activated
- Platform Stability
- Navigational Boards

CONCLUSION The issue related with this recognition technique is what technology to use for collecting raw data from the device. A large number of recognition techniques are available like the feature extraction, active shape models, template matching. There are also a few segmentation algorithms such as Hand segmentation using HSV Colour, Anticipated static gesture set and sampled storage approach algorithm provide for recognition without much noise. But in

the situation of the "Arduino based movement monitoring keyboard using machine learning," the results will be more precise as it will use the scikit learning (support vector machine algorithm) and thus better predictions with less noise to classify signals into letters. The cost of production of this device will be much less from the current methods of recognition and it will also be working mostly in all the environments such as areas with not proper light conditions. It can be of great social interest too as it would be helpful to the deaf and mute people to get discrete, direct and very effective service in the day to day life.

2. Machine Learning in 3D Space Gesture Recognition Veronica Naosekpam* & Rupam Kumar Sharma Department of CSE and IT, School of Technology, Assam Don Bosco University, Assam, India *Corresponding author: venaosekpam11@gmail.com Received 24 October 2018, Received in revised form 20 April 2019 Accepted 17 July 2019, Available online 31 October 2019

Abstract: The rapid increase in the development of robotic systems in a controlled and uncontrolled environment leads to the development of a more natural interaction system. One such interaction is gesture recognition. The proposed paper is a simple approach towards gesture recognition technology where the hand movement in a 3-dimensional space is utilized to write the English alphabets and get

the corresponding output in the screen or a display device. In order to perform the experiment, an MPU6050 accelerometer, a microcontroller and a Bluetooth for wireless connection are used as the hardware components of the system. For each of the letters of the alphabets, the data instances are recorded in its raw form. 20 instances for each letter are recorded and it is then standardized using interpolation. The standardized data is fed as inputs to an SVM (Support Vector Machine) classifier to create a model. The created model is used for classification of future data instances at real time. Our method achieves a correct classification accuracy of 98.94% for the English alphabets' hand gesture recognition. The primary objective of our approach is the development of a low-cost, low power and easily trained supervised gesture recognition system which identifies hand gesture movement efficiently and accurately. The experimental result obtained is based on use of a single subject.

END OF RELATED WORKS

Methodology

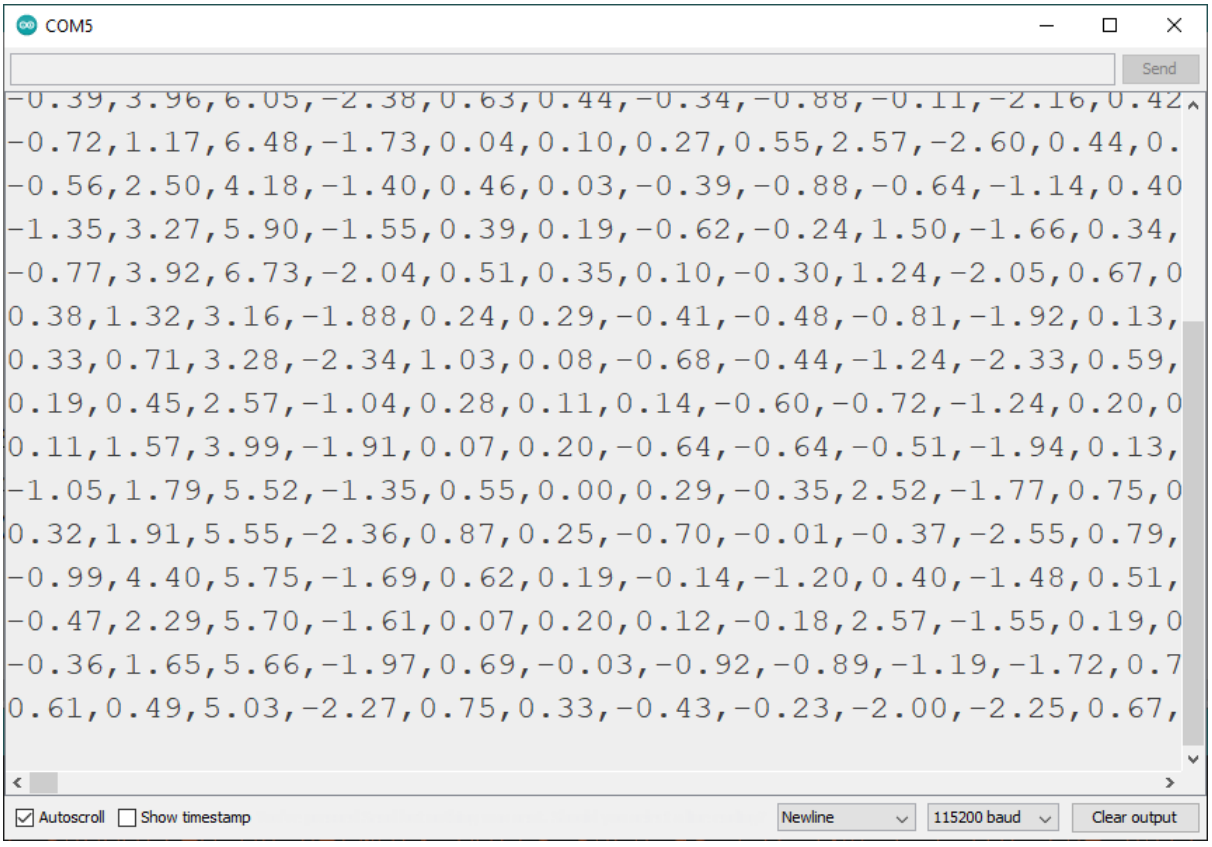
- For training the model:
  We use the Adafruit MPU6050 library to get the output data from the IMU.
- Then we truncate it to make it not exceed specified value.

- After that we repeatedly check if the board is in motion using the IMU data.
- If the board is in motion then we record the data from the IMU and we store it in an array.
- After 3 second delay we print the array on to the serial monitor where each value is separated by a comma.
- After that we calibrate the IMU recording the data and setting a new baseline so that we can detect motion again.
- We continue this process until we collect the required amount of data samples.
- After that we copy the data to excel and arrange it in a dataset.
- Now we create and store separate excel files for each gesture and store it in a folder.

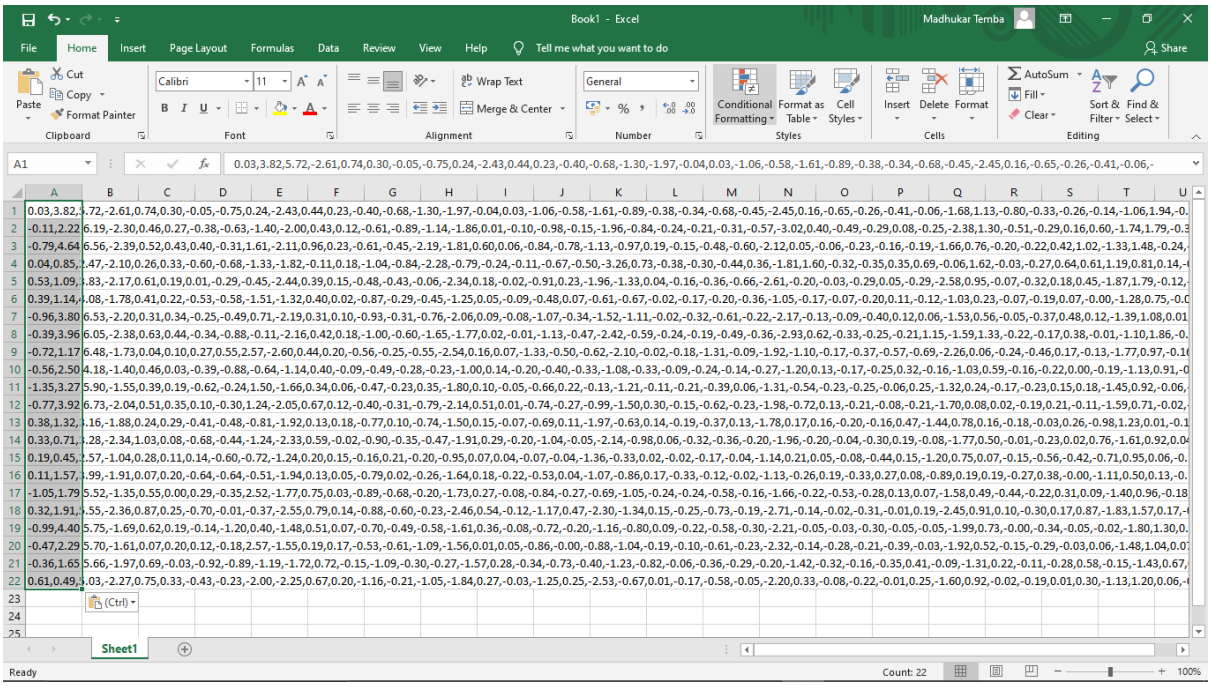
- Then we open python, select the classifier and generate the C code for the machine learning program using micromlgen.

First, we make the required gesture 15-20 times and record it.

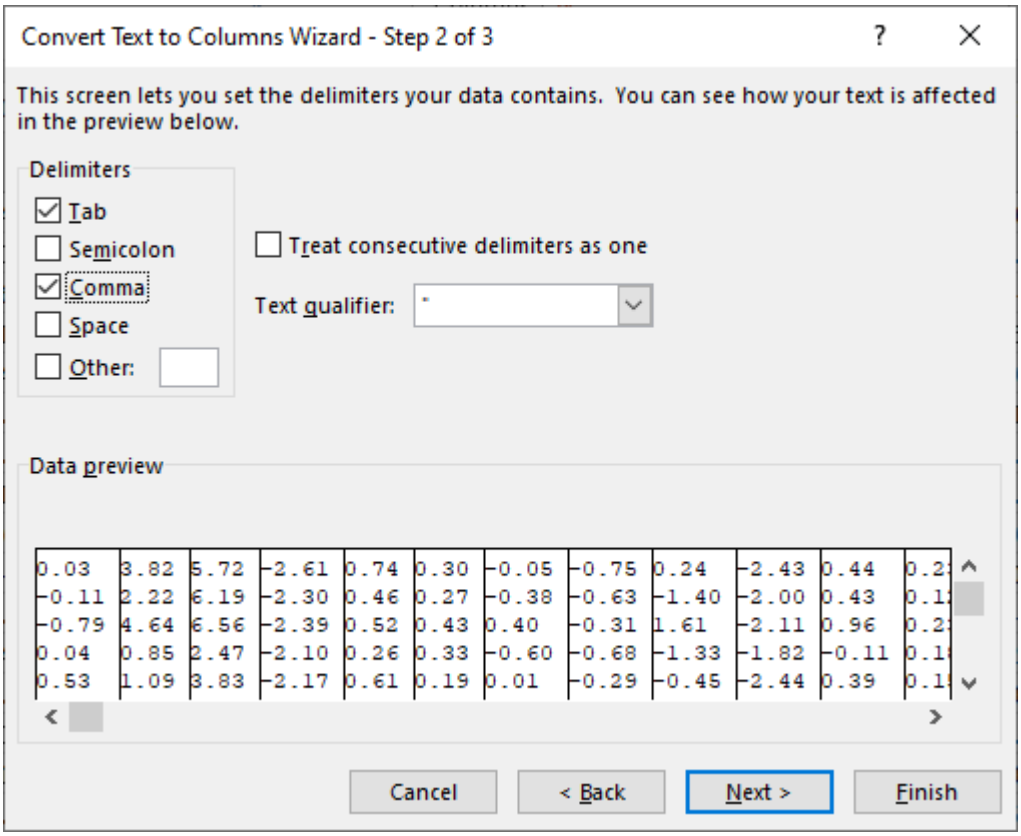
The output on the serial monitor looks like this:



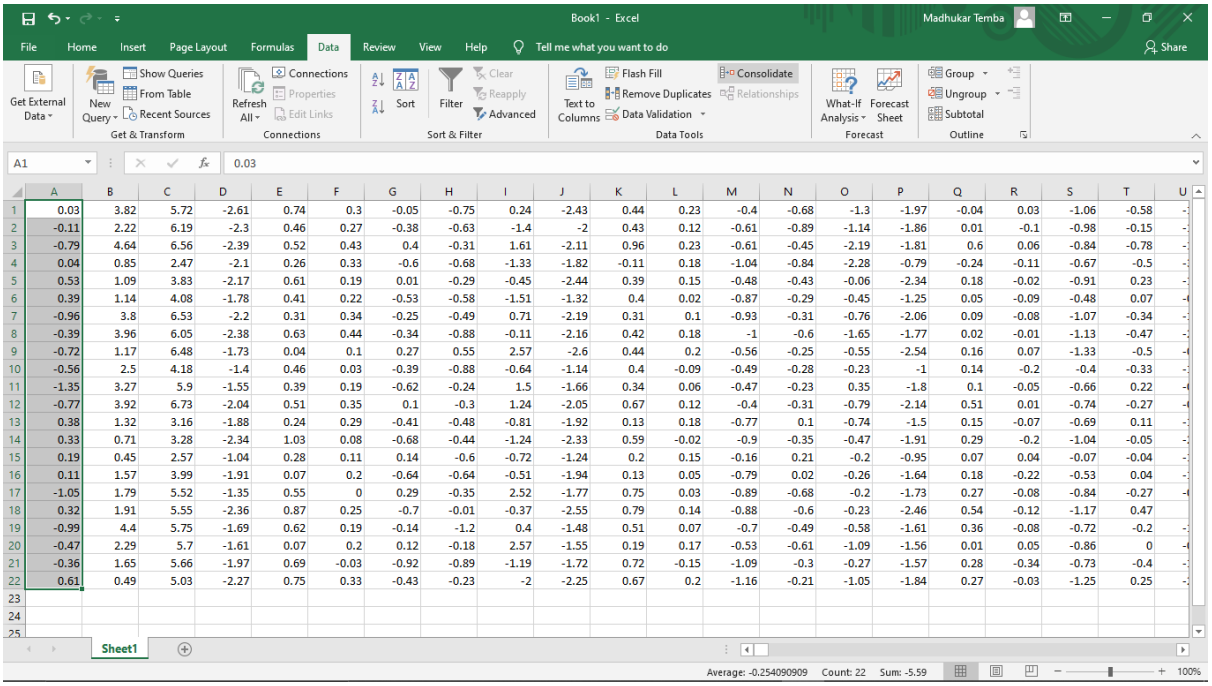
Then copy the data to excel with each file containing one gesture information.



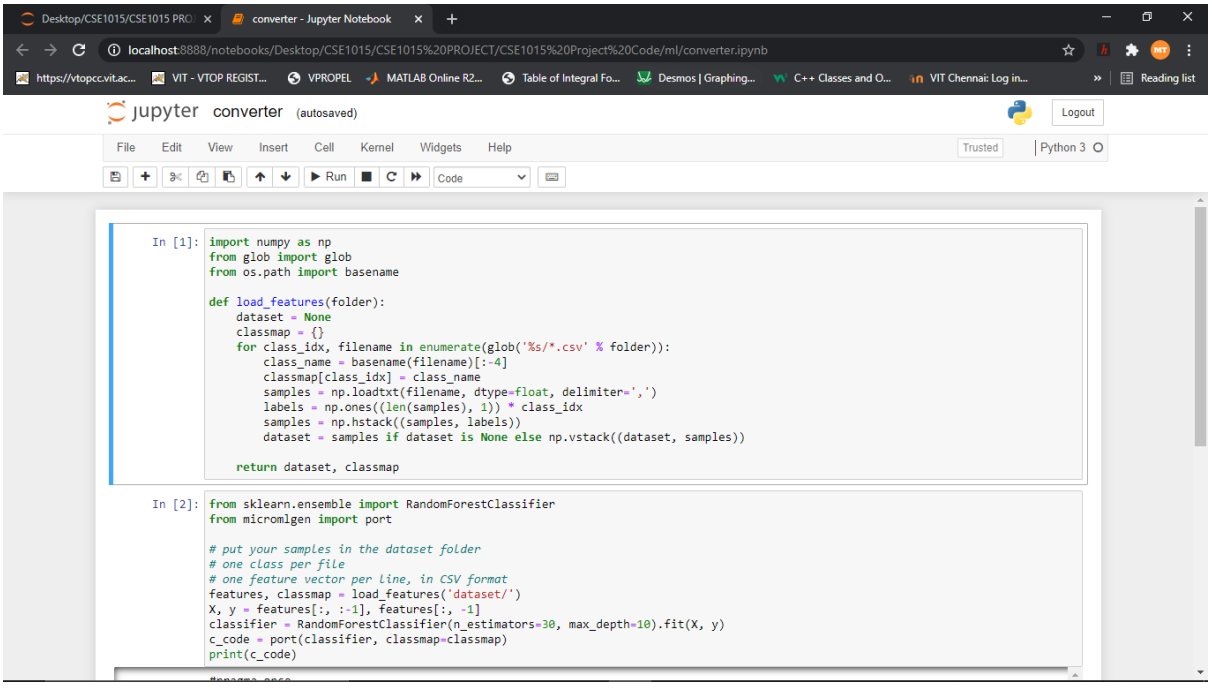
Then arrange the data using text to columns in excel:



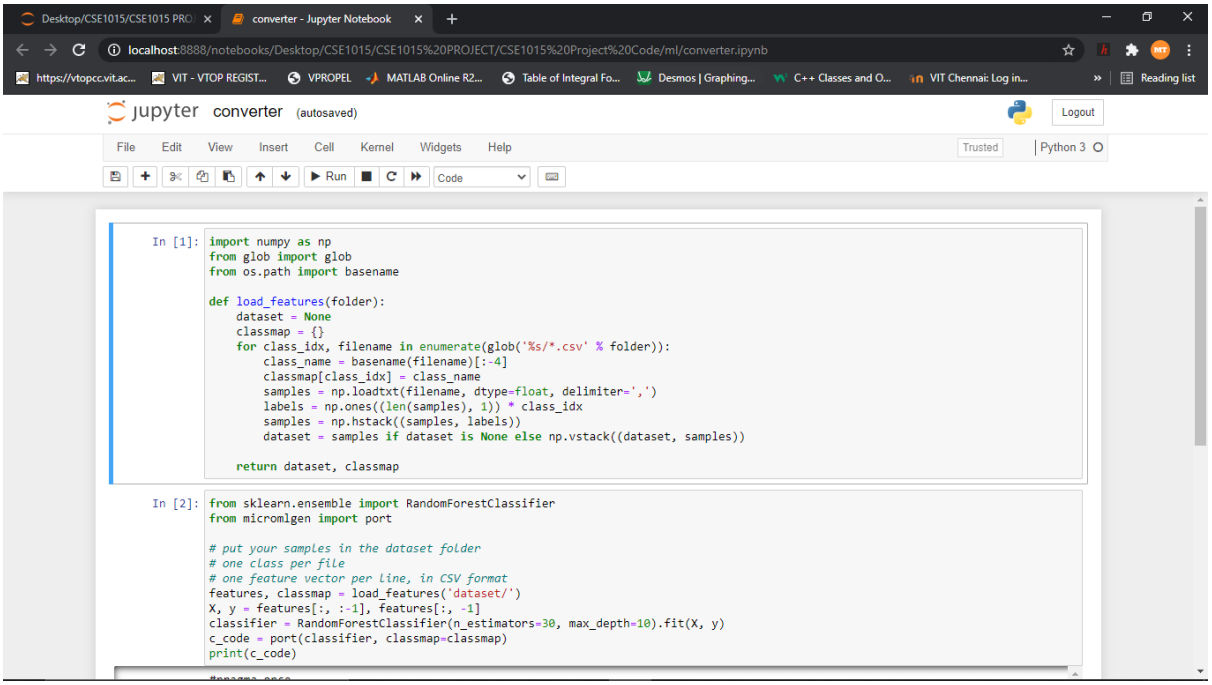
After arranging:



Now move all the gesture files into one folder and run the converter code in python:



Specify the classifier (using Random Forest) here:



The micromlgen converter will output the C code as text:

```
/**
* Convert class idx to readable name
*/
const char* idxToLabel(uint8_t classIdx) {
    switch (classIdx) {
        case 0:
        return "a";
        case 1:
        return "doubletap";
        case 2:
        return "leftswipe";
        case 3:
        return "rightswipe";
        case 4:
        return "tripletap";
        default:
        return "Houston we have a problem";
    }
}
```

Now we copy and paste it in the Arduino IDE in a file named model.h:

After that in the main file we include the model:

```
#include<"model.h">

Eloquent::ML::Port::RandomForest classifier;
```

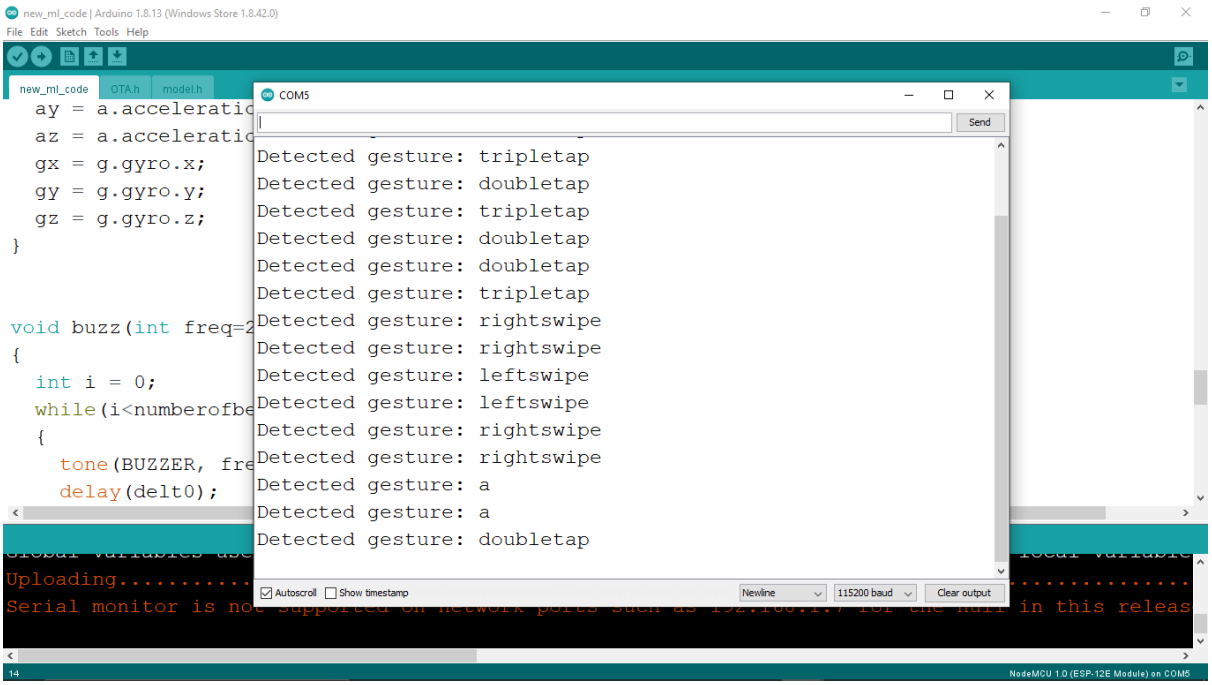
- What the code does:
  The code first calibrates the IMU data.
- Then it checks for motion by the IMU.
- If motion is detected that is if the acceleration is greater than the set

threshold then it will start recording the IMU data to an array.

- It take around 300 samples of the 3 accelerometer axis and 3 gyroscope rotations/sec values.
- After that it uses support vector machine classifier to determine the gesture.
- The accuracy of the classifier is 97% according to the maker.

Now we upload the code to the microcontroller and we will get the output in the serial monitor when we perform the gesture:

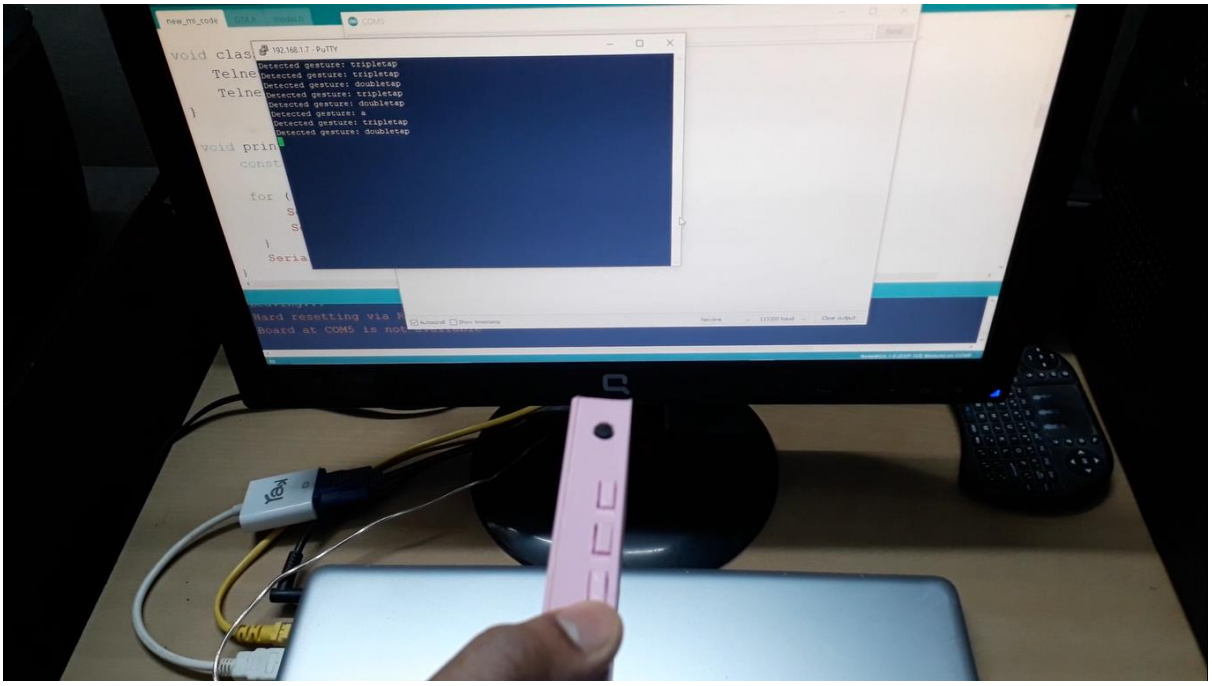
Output on the serial monitor:



The detected gesture is also printed in the screen:

For a wireless serial monitor we can use Telnet Stream:

```
void printfeatures() {
    const uint16_t numFeatures = sizeof(features) / sizeof(float);

    for (int i = 0; i < numFeatures; i++) {
        TelnetStream.print(features[i]);
        TelnetStream.print(',');
    }
    TelnetStream.println();
}
```



THE BUILD PROCESS

1. First we record all the part dimensions:

OLED DIMENSIONS:

length = 28mm

Breadth = 26mm

Height = 3mm

Screen location is 6mm from top and 2.5mm from the sides

Screen length 22mm, breadth 11.5mm

NodeMCU dimensions:

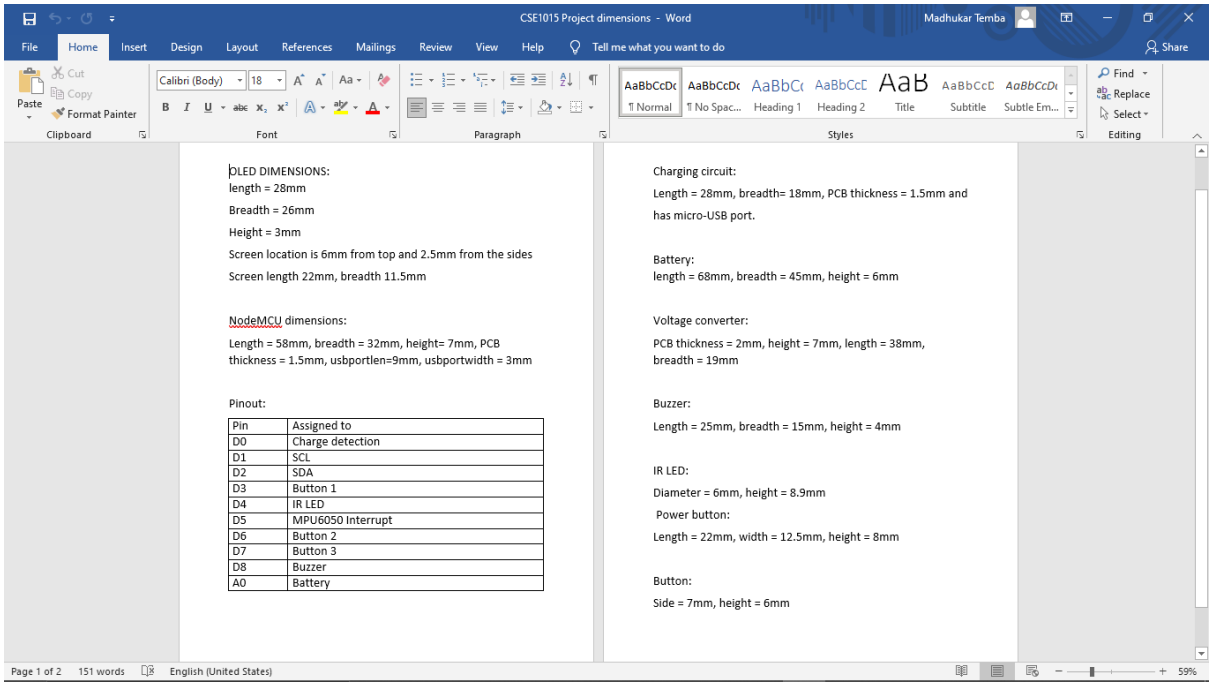
Length = 58mm, breadth = 32mm, height= 7mm, PCB thickness = 1.5mm, usbportlen=9mm, usbportwidth = 3mm

2. Then we make the pinout table:

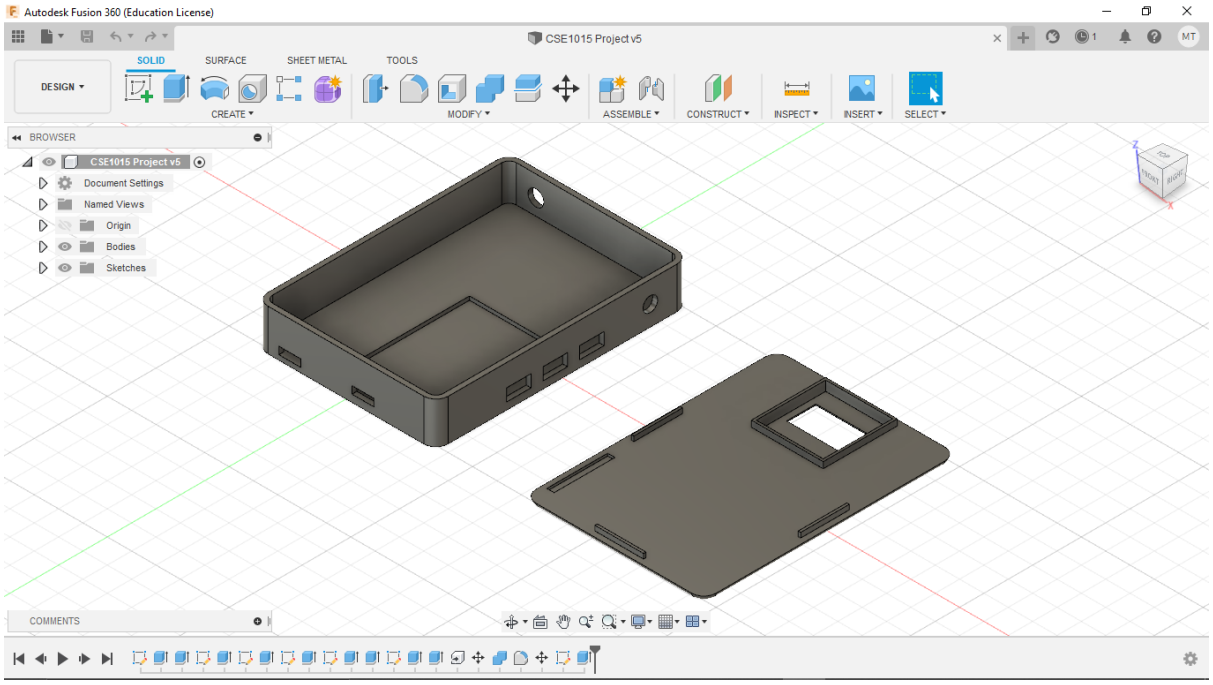
Pinout:

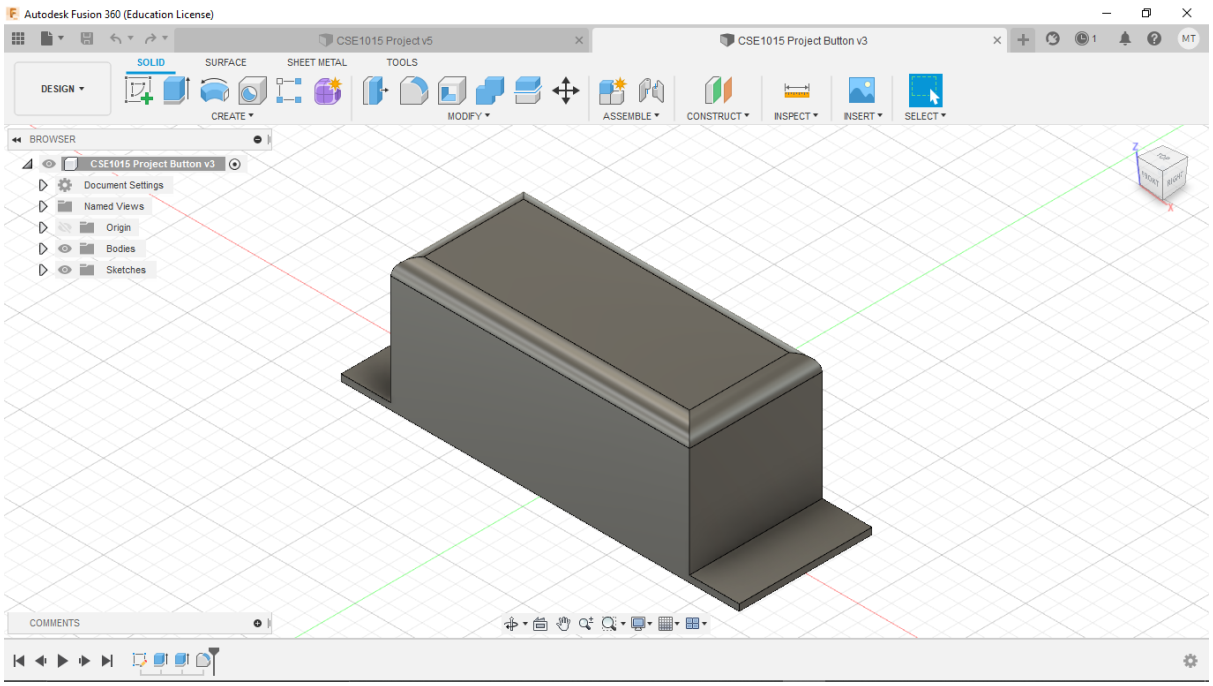
Pin	Assigned to
D0	Charge detection
D1	SCL
D2	SDA
D3	Button 1
D4	IR LED
D5	MPU6050 Interrupt
D6	Button 2
D7	Button 3
D8	Buzzer
A0	Battery

3. After recording all the data:



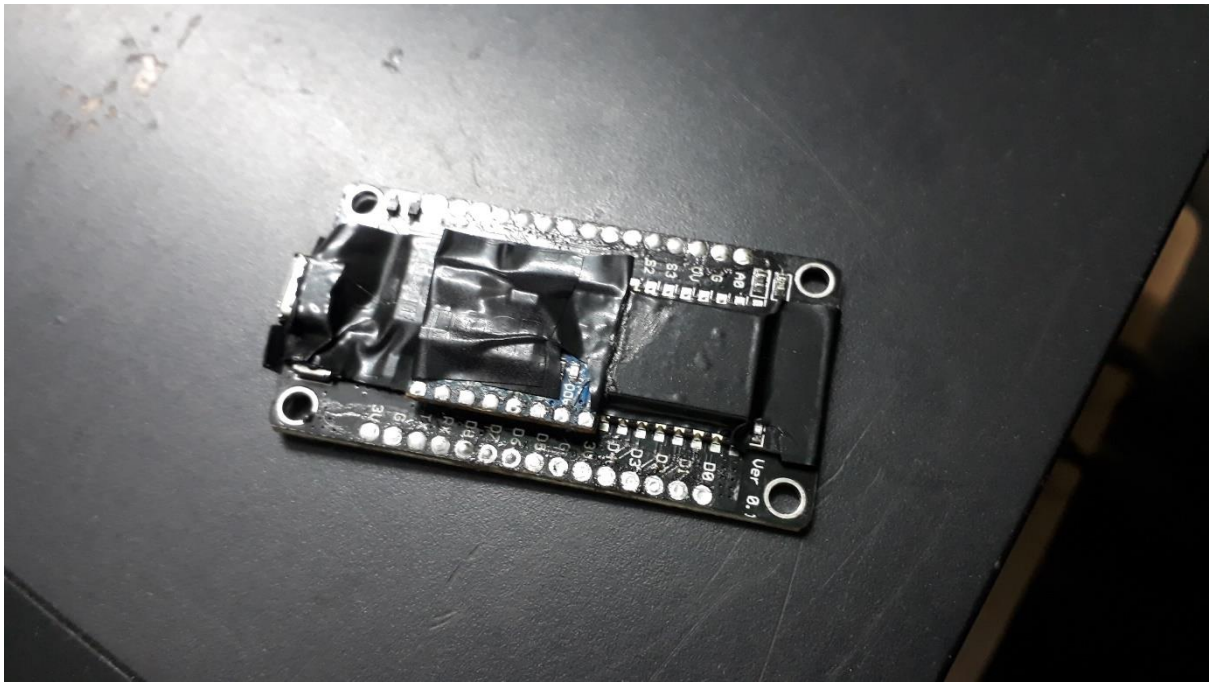
4. Let us make the 3D model in Fusion 360:

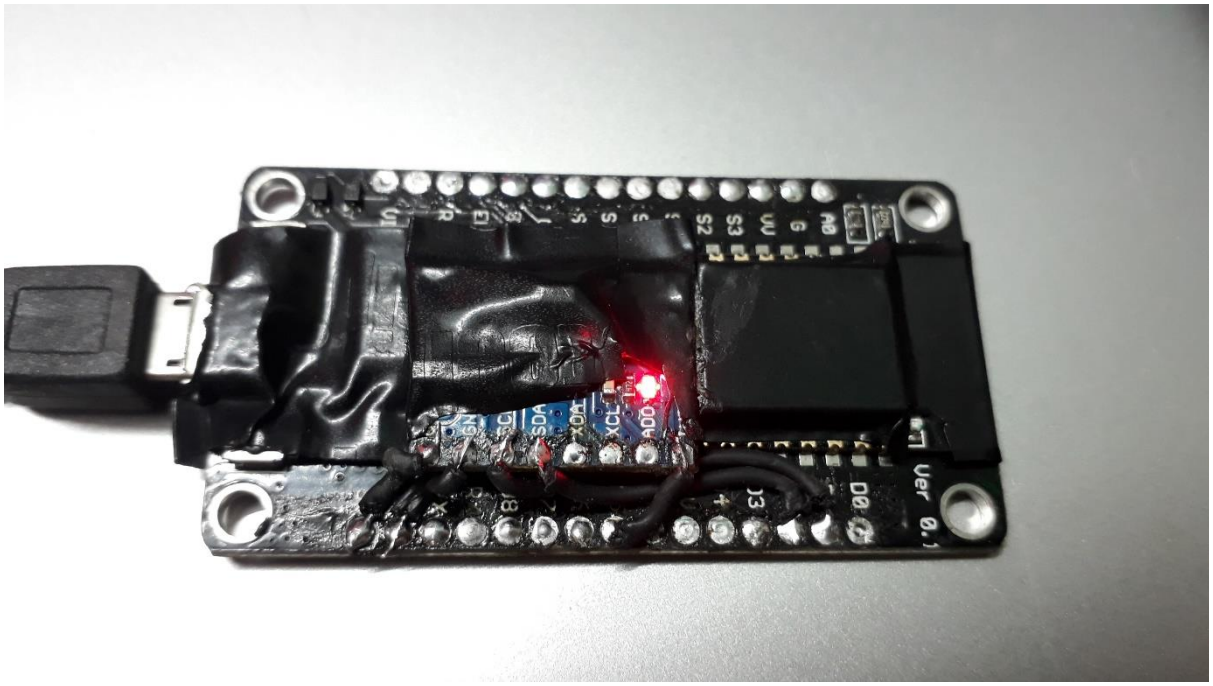




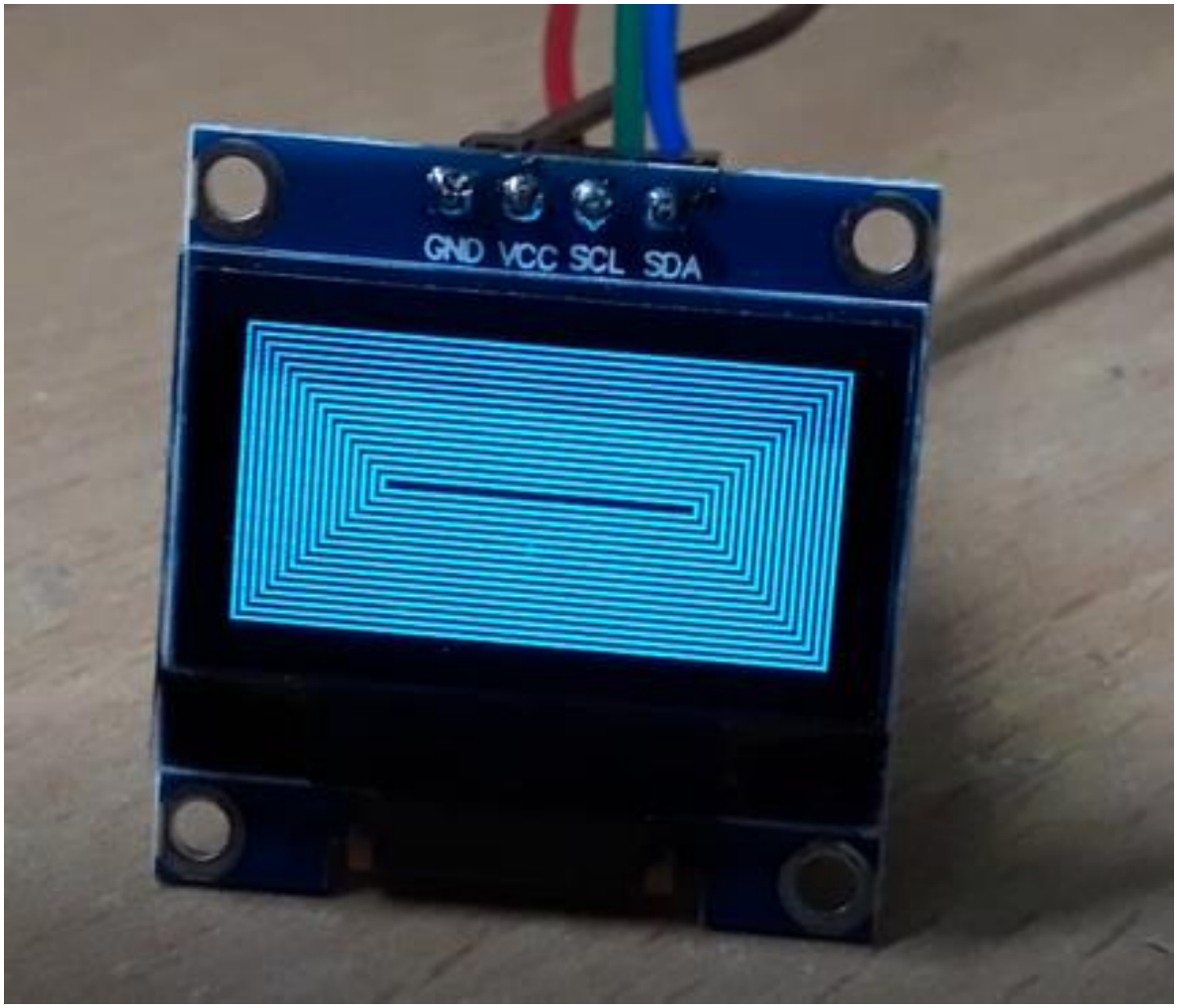
Now let’s start the assembly and testing:

1. Attaching the IMU to the microcontroller to save space:

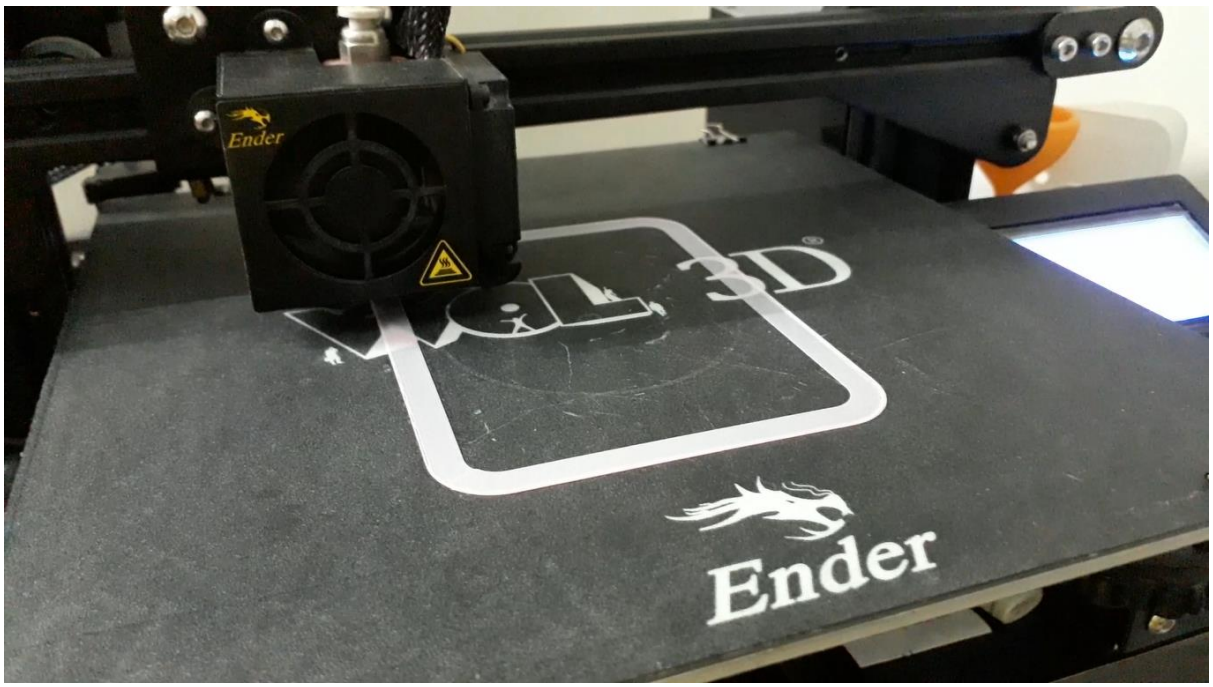




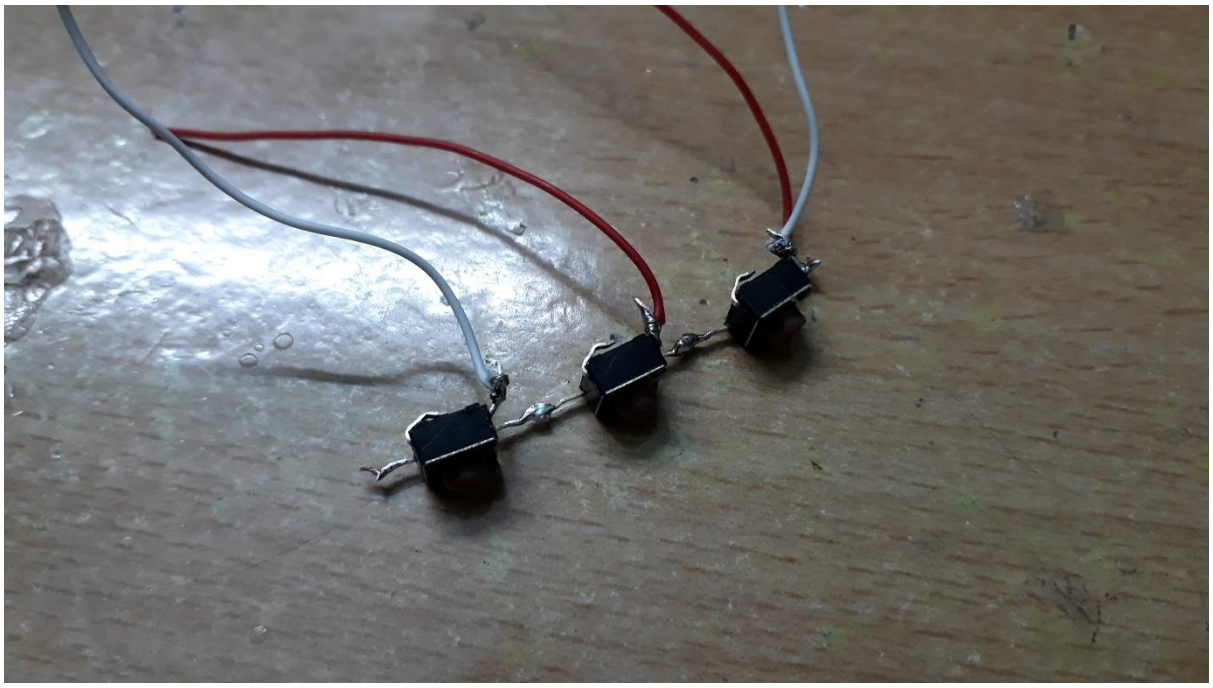
Testing the OLED display:

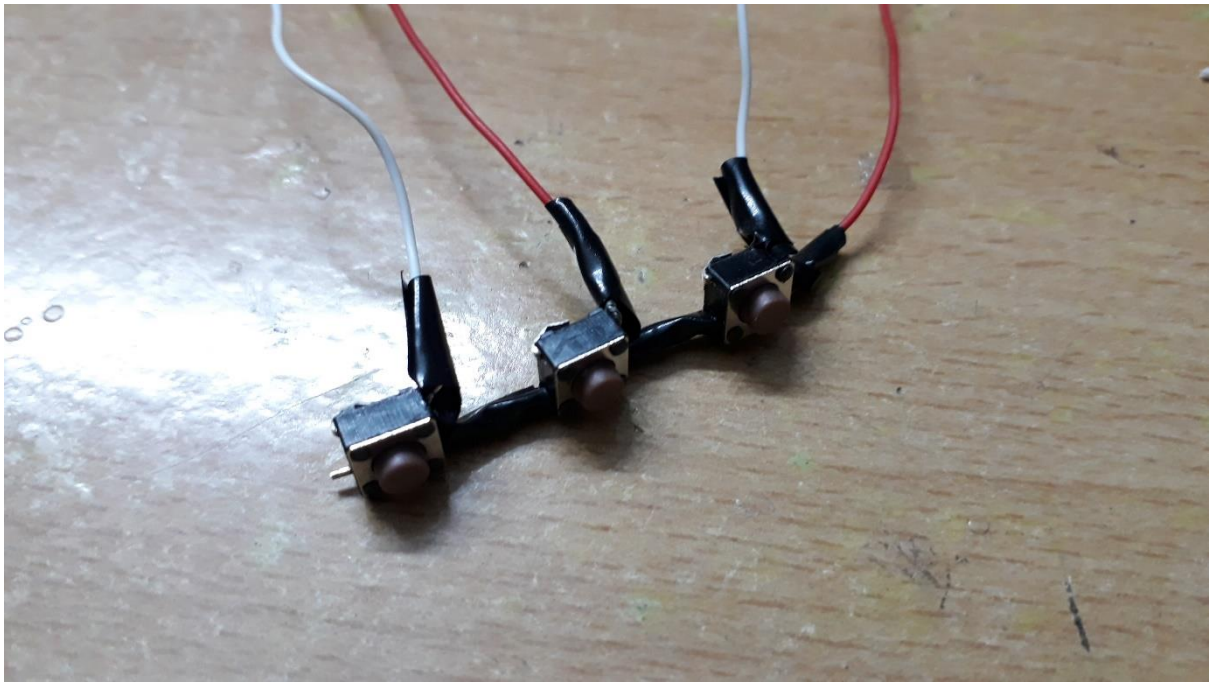


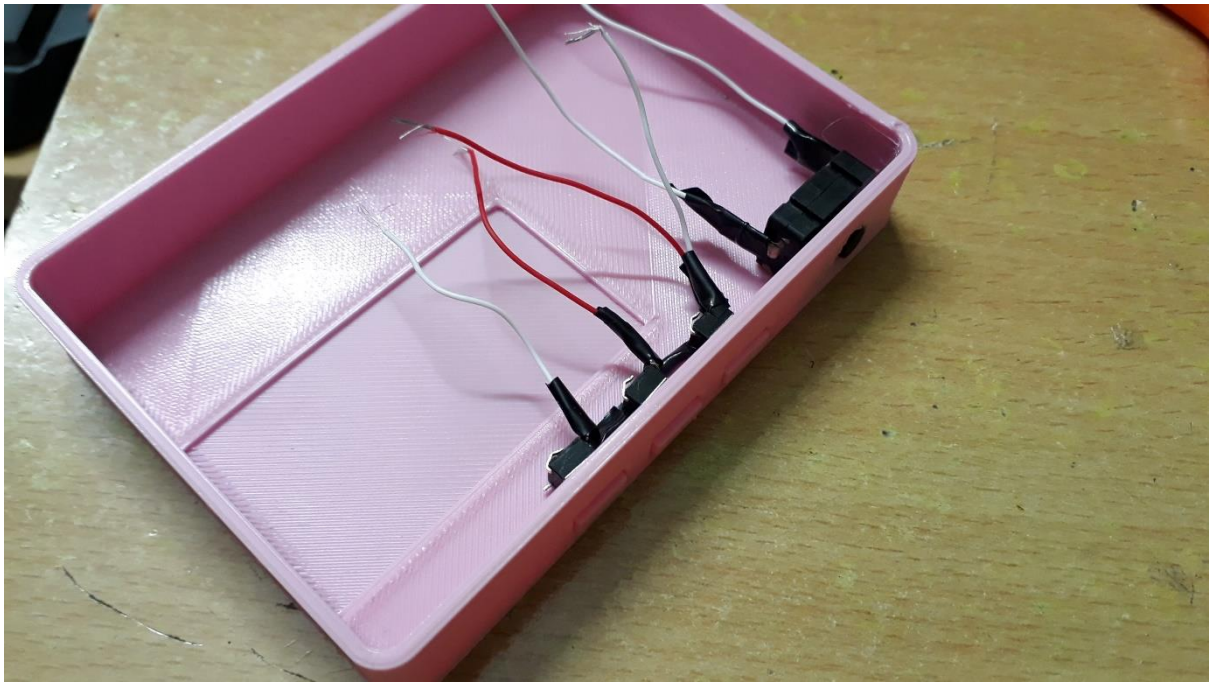
2. 3D printing the parts:

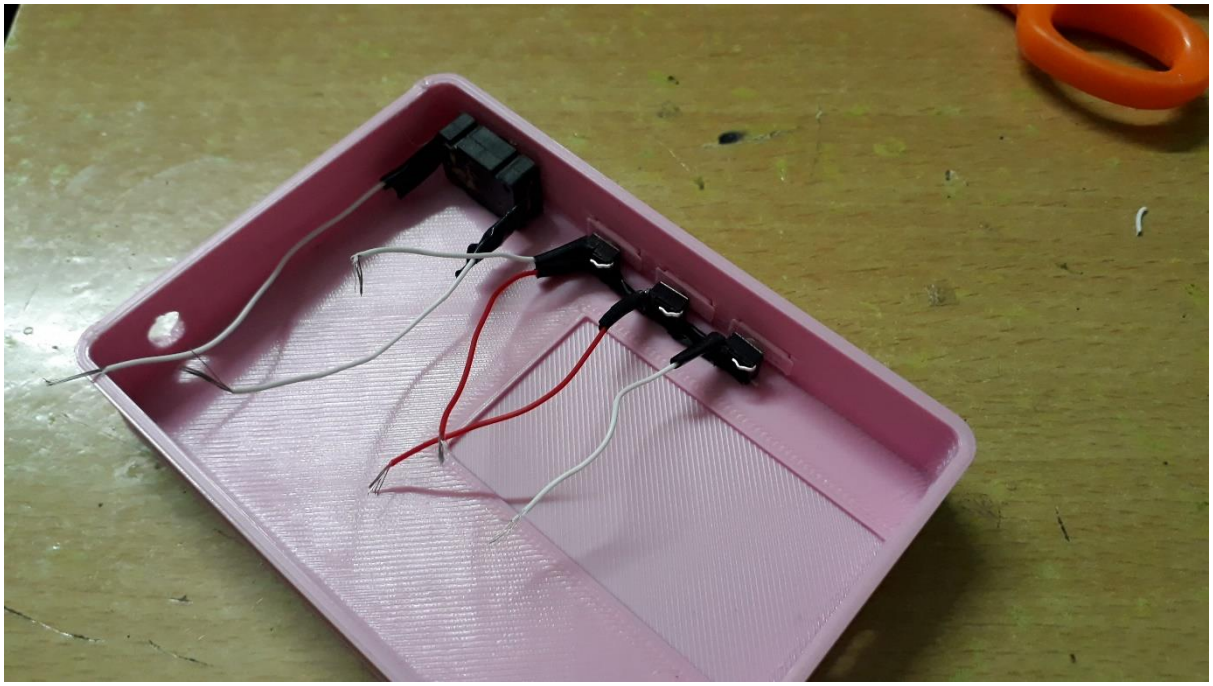


3. Joining the buttons

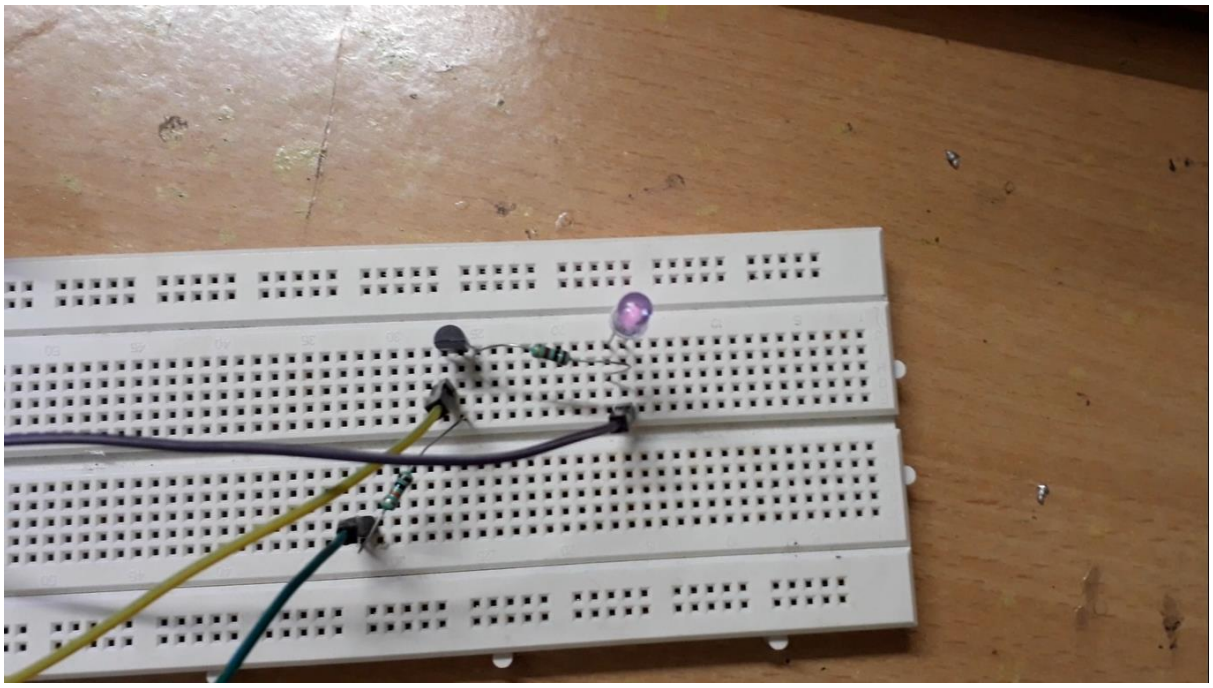




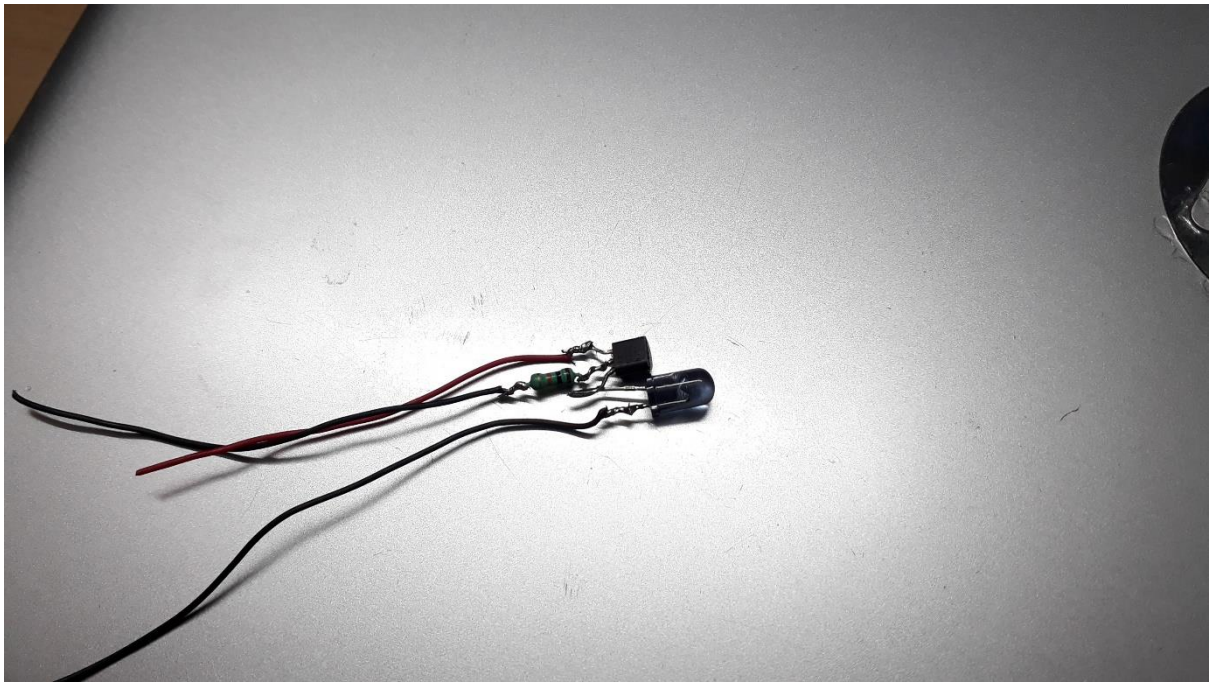


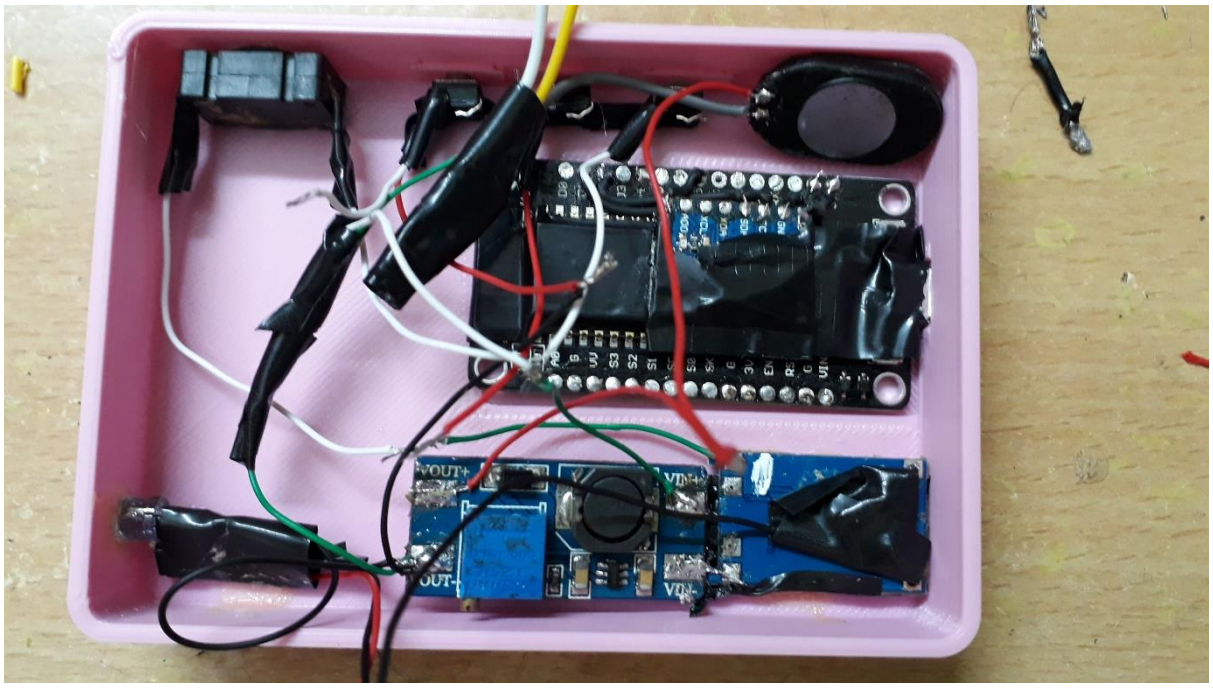


4. Testing the IR led:

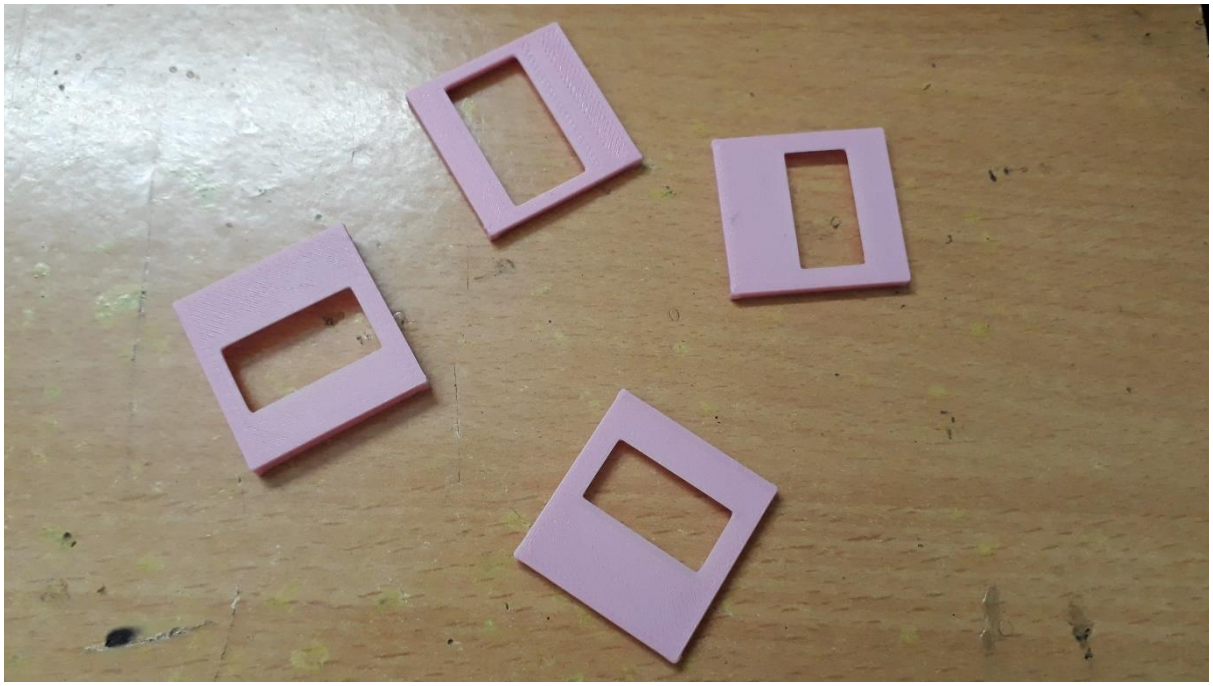


Making the circuit small:

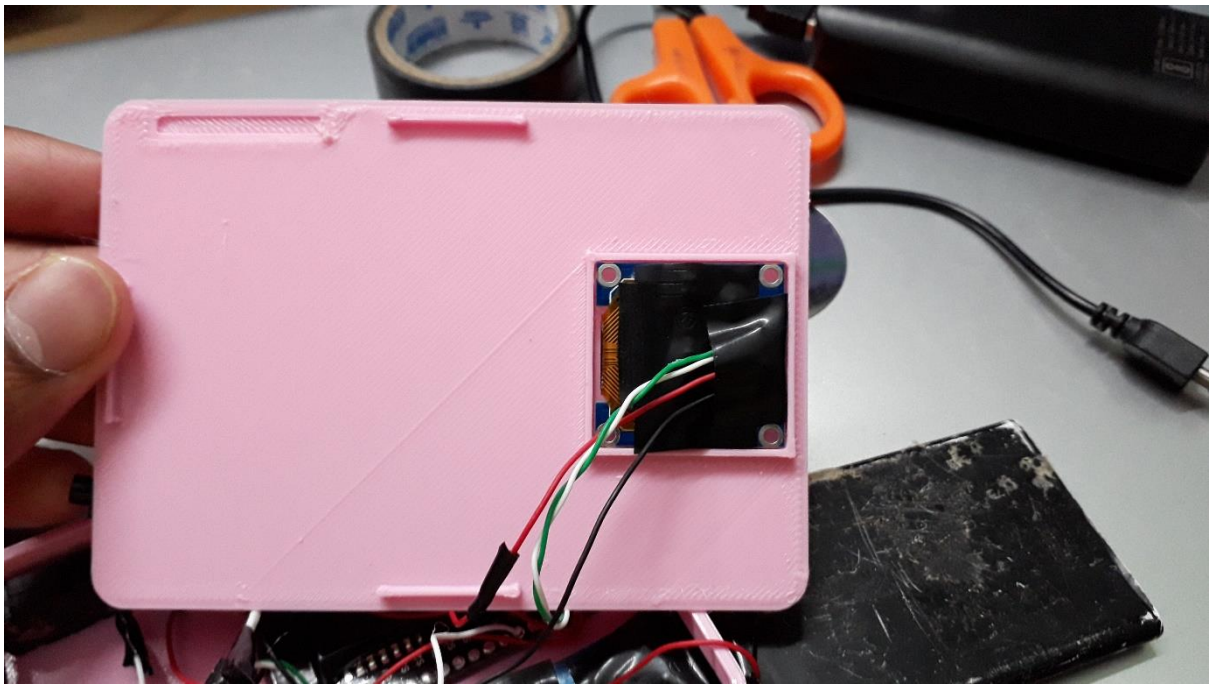




5. Making the test enclosures for the OLED display:



6. 3D printing the front cover:



7. Final Result:


P : -24.98
R : -1.13
V : 311.56

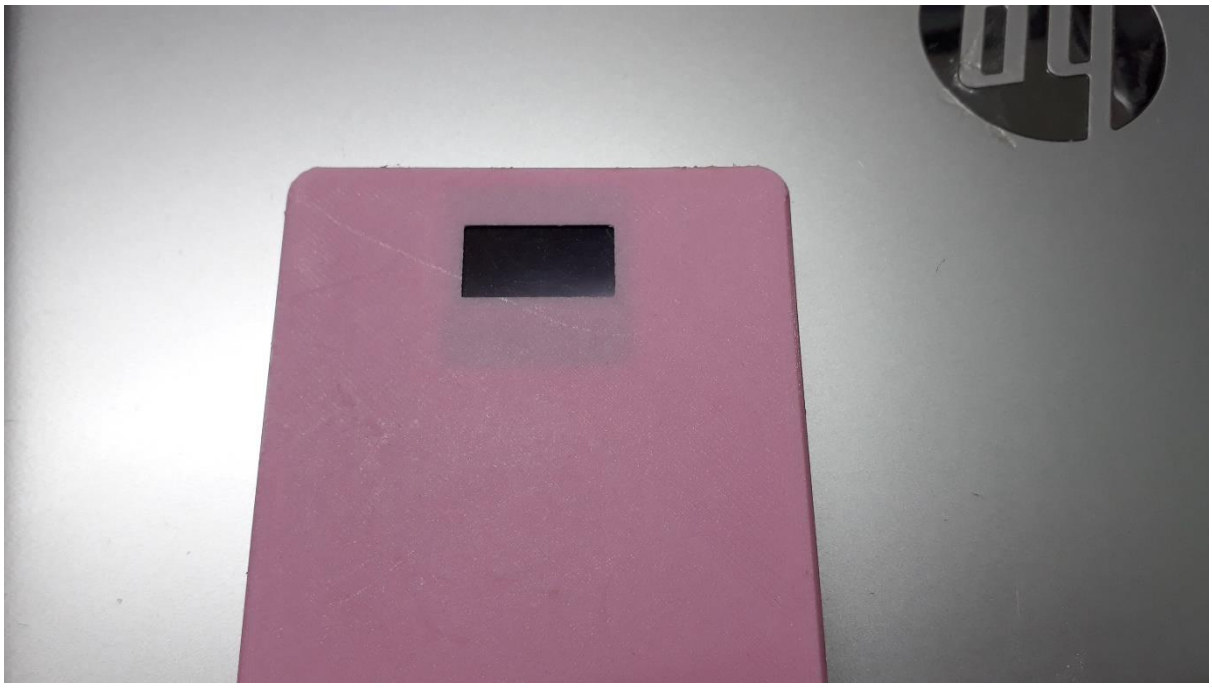
Features:

1. Has 4 buttons, one for power and the rest for controlling the device:

2. Two ports: One for charging and the other one for data transfer.

3. Front mounted IR led to control televisions:

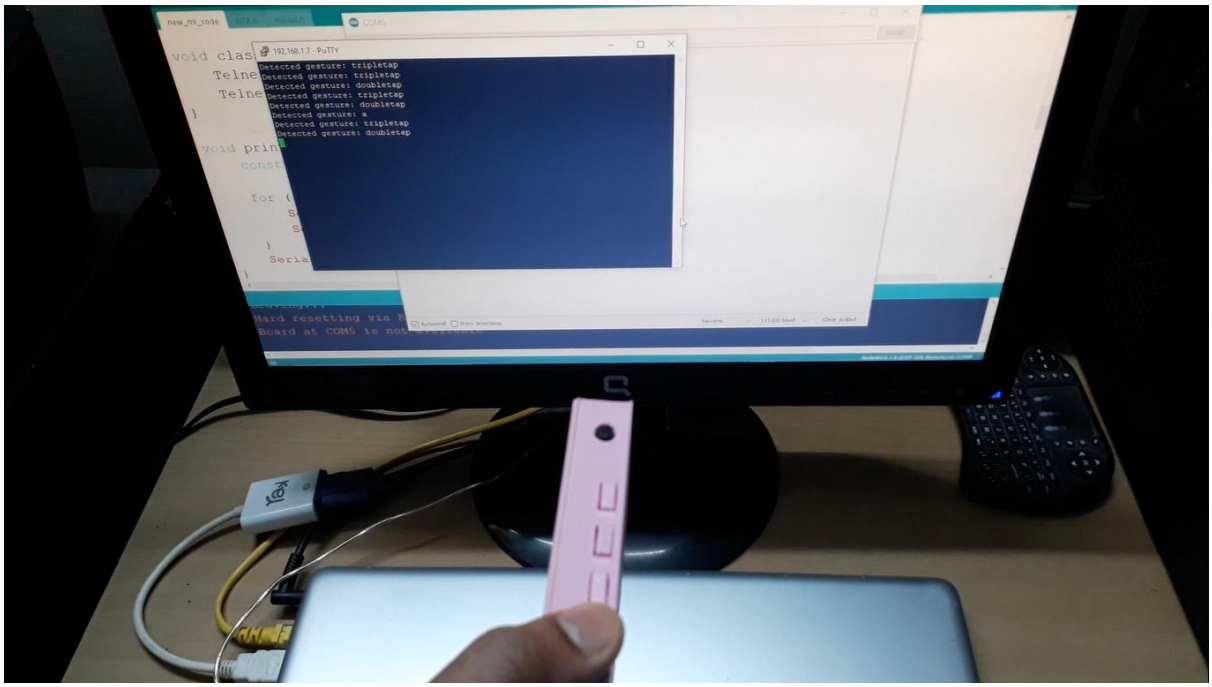
4. Has a 128X64 pixel OLED display:



- Has a 2000mAh battery.
- Up to 24 hours of continuous usage.

- It has Wi-Fi for communicating with computers.
- It has deep sleep capability to save power.
- Buzzer for audio feedback.
- IR led for communicating with televisions.
- Has a 6 axis IMU.
- It has over the air (OTA) update capability.

Running the ML code on the device:



Experiments

I trained many gestures like left swipe, right swipe etc. I also trained the alphabets ‘A’ and ‘B’ to see will it recognise these too or not.

- The device detects the correct gesture about 70 percent of the time.

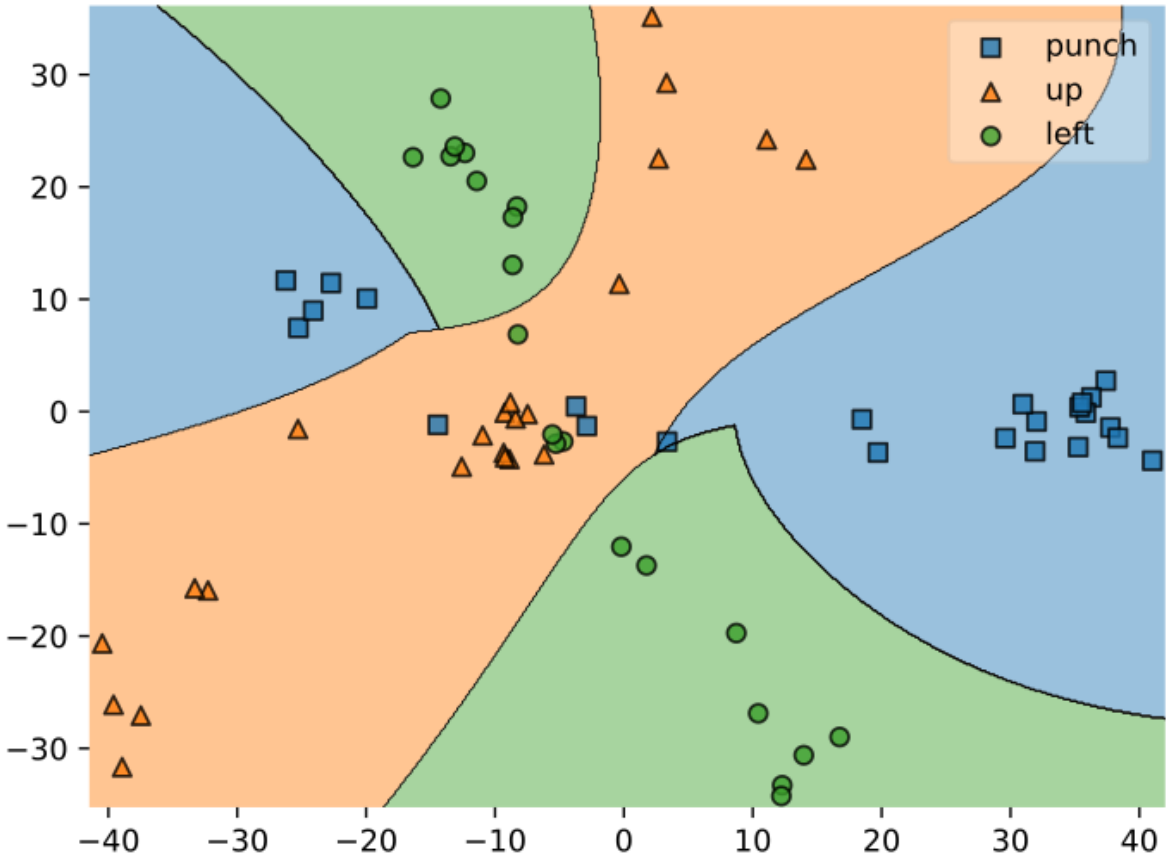
- It has problems with very similar data such as drawing different letters.
- The user has to perform a very similar gesture as made while recording the data to make the device detect the correct one.
- A change in the start time of the recording leads to incorrect classification. So, it is configured to start recording as soon as it detects motion.

Comparative study

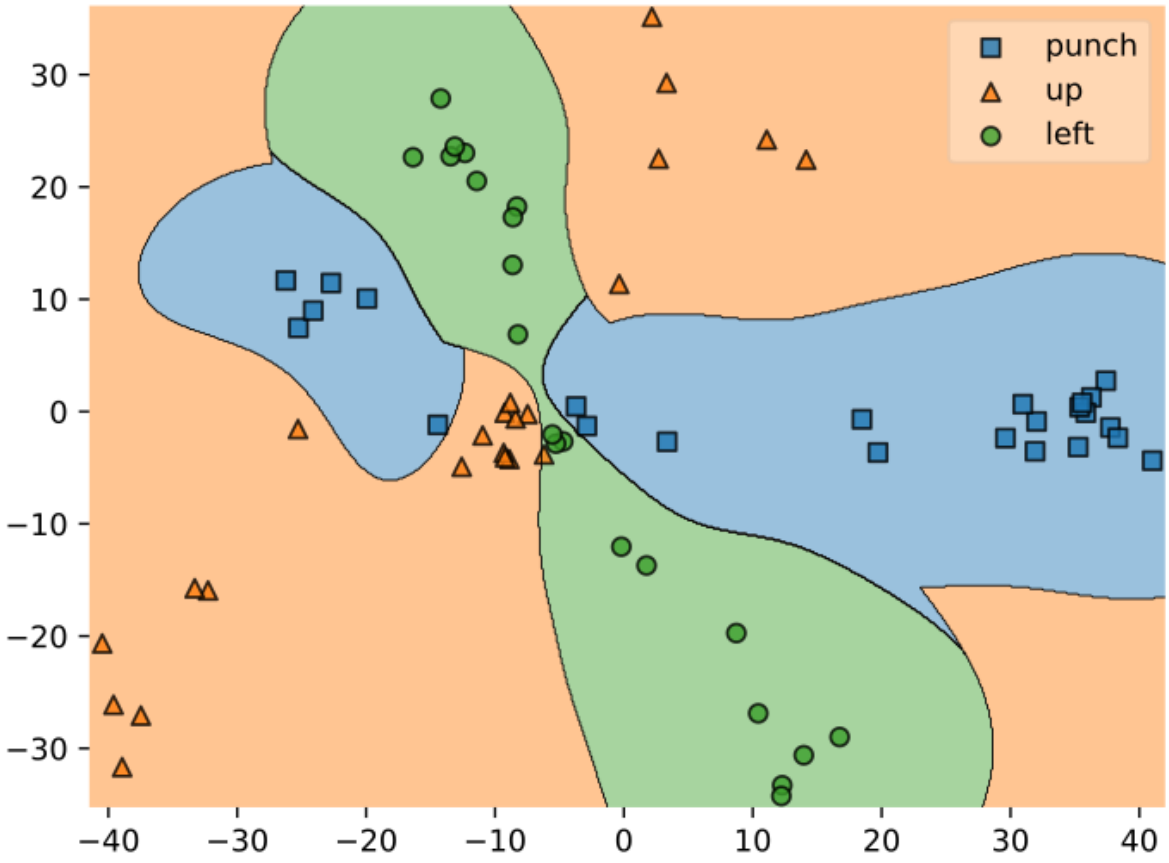
The creator of the ML program Eloquent had more model accuracy than my model.

Kernel	C	Gamma	Degree	Vectors	Flash size	RAM (b)	Avg accuracy
RBF	10	0.001	-	37	53 Kb	1228	99%
Poly	**100**	**0.001**	**2**	**12**	**25 Kb**	**1228**	**99%**

Kernel	C	Gamma	Degree	Vectors	Flash size	RAM (b)	Avg accuracy
Poly	100	0.001	3	25	40 Kb	1228	97%
Linear	50	-	1	40	55 Kb	1228	95%
RBF	100	0.01	-	61	80 Kb	1228	95%



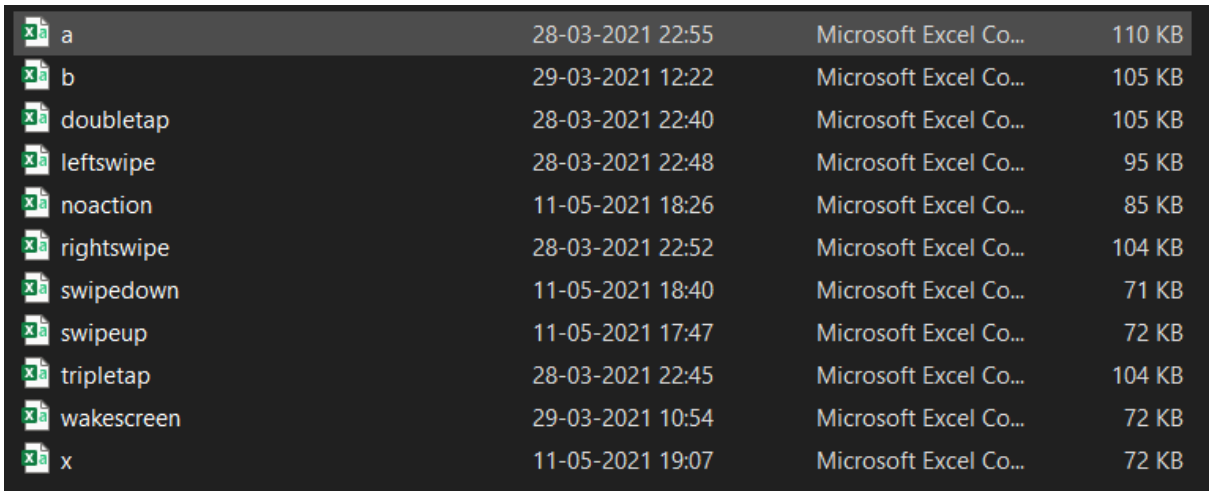
Decision boundaries of 2 PCA components of Gestures features, RBF kernel, 0.001 gamma



Decision boundaries of 2 PCA components of Gestures features, RBF kernel, 0.01 gamma

My Results

11 gestures are in the dataset which are labelled 0 to 10 from top to bottom:



a	28-03-2021 22:55	Microsoft Excel Co...	110 KB
b	29-03-2021 12:22	Microsoft Excel Co...	105 KB
doubletap	28-03-2021 22:40	Microsoft Excel Co...	105 KB
leftswipe	28-03-2021 22:48	Microsoft Excel Co...	95 KB
noaction	11-05-2021 18:26	Microsoft Excel Co...	85 KB
rightswipe	28-03-2021 22:52	Microsoft Excel Co...	104 KB
swipedown	11-05-2021 18:40	Microsoft Excel Co...	71 KB
swipeup	11-05-2021 17:47	Microsoft Excel Co...	72 KB
tripletap	28-03-2021 22:45	Microsoft Excel Co...	104 KB
wakescreen	29-03-2021 10:54	Microsoft Excel Co...	72 KB
x	11-05-2021 19:07	Microsoft Excel Co...	72 KB

The test data was recorded separately and stored in the folder ‘testdata’

The test data was recorded separately and stored in the folder ‘testdata’

atest	11-05-2021 18:57	Microsoft Excel Co...	63 KB
btest	11-05-2021 18:58	Microsoft Excel Co...	81 KB
doubletaptest	11-05-2021 19:01	Microsoft Excel Co...	71 KB
leftswipetest	11-05-2021 17:55	Microsoft Excel Co...	66 KB
noactiontest	11-05-2021 18:10	Microsoft Excel Co...	50 KB
rightswipetest	11-05-2021 17:52	Microsoft Excel Co...	62 KB
swipedowntest	11-05-2021 18:39	Microsoft Excel Co...	61 KB
swipeuptest	11-05-2021 17:46	Microsoft Excel Co...	57 KB
tripletaptest	11-05-2021 17:57	Microsoft Excel Co...	71 KB
wakescreentest	11-05-2021 18:55	Microsoft Excel Co...	38 KB
xtest	11-05-2021 19:07	Microsoft Excel Co...	72 KB

The classifiers tested are:

- Random Forest

- Multi-layer perceptron
- Decision Tree
- Linear Regression
- Logistic Regression
- Gaussian Naïve Bayes
- Support Vector Classifier
- Gaussian Process Classifier

Testing and Visualisation Method

- Each classifier is trained using the dataset and then tested using the test data recorded separately.
- The accuracy, MAE,MSE, RMSE and R2 scores are compared.
- Scatter plot and histogram is also plotted between the actual and predicted values.
- For visualising the classifier the data was converted into 2D data using principal component analysis.

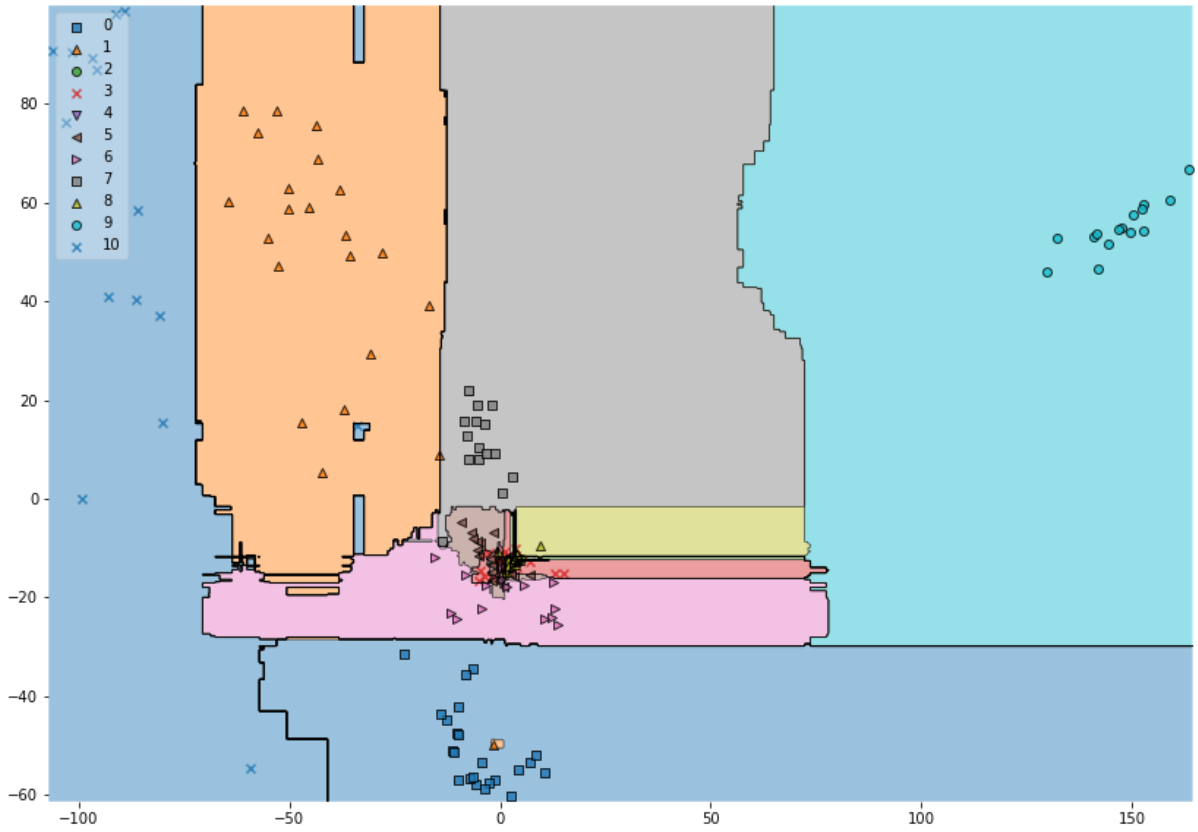
- Please check the .ipynb file named ‘Model Tester and Visualiser’. This file contains the code for testing and visualisation.

1. Random Forest

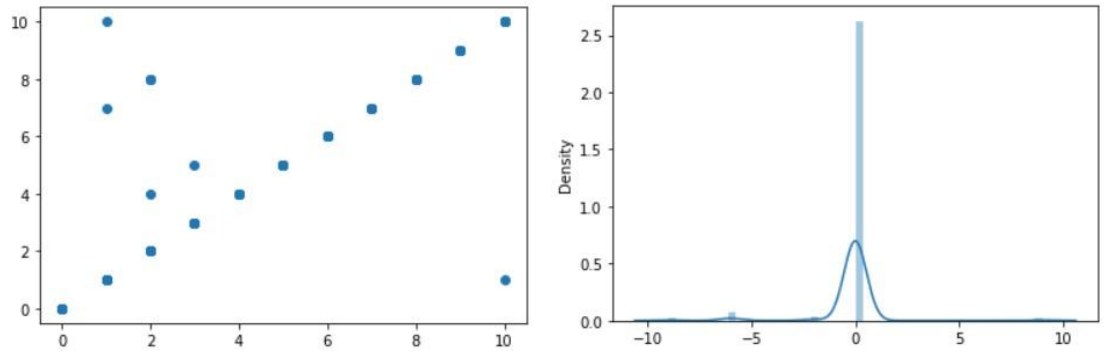
```
[[12  1  0  0  0  0  0  0  0  0  0]
 [ 0 16  0  0  0  0  0  0  0  0  1]
 [ 0  0 14  0  1  0  0  0  0  0  0]
 [ 0  0  0 13  0  1  0  0  0  0  0]
 [ 0  0  0  0 11  0  0  0  0  0  0]
 [ 0  0  0  0  0 13  0  0  0  0  0]
 [ 0  0  0  0  0  0 13  0  0  0  0]
 [ 0  0  0  0  0  0  0 12  0  0  0]
 [ 0  0  0  0  0  0  0  0 15  0  0]
 [ 0  0  0  0  0  0  0  0  0  8  0]
 [ 0  1  0  0  0  0  0  0  0  0 14]]
```

	precision	recall	f1-score	support
0.0	1.00	0.92	0.96	13
1.0	0.89	0.94	0.91	17
2.0	1.00	0.93	0.97	15
3.0	1.00	0.93	0.96	14
4.0	0.92	1.00	0.96	11
5.0	0.93	1.00	0.96	13
6.0	1.00	1.00	1.00	13
7.0	1.00	1.00	1.00	12
8.0	1.00	1.00	1.00	15
9.0	1.00	1.00	1.00	8
10.0	0.93	0.93	0.93	15
accuracy			0.97	146
macro avg	0.97	0.97	0.97	146
weighted avg	0.97	0.97	0.97	146

```
MAE:  0.15753424657534246
MSE:  1.1712328767123288
RMSE:  1.0822351300490707
R Squared:  0.8854371248692204
```



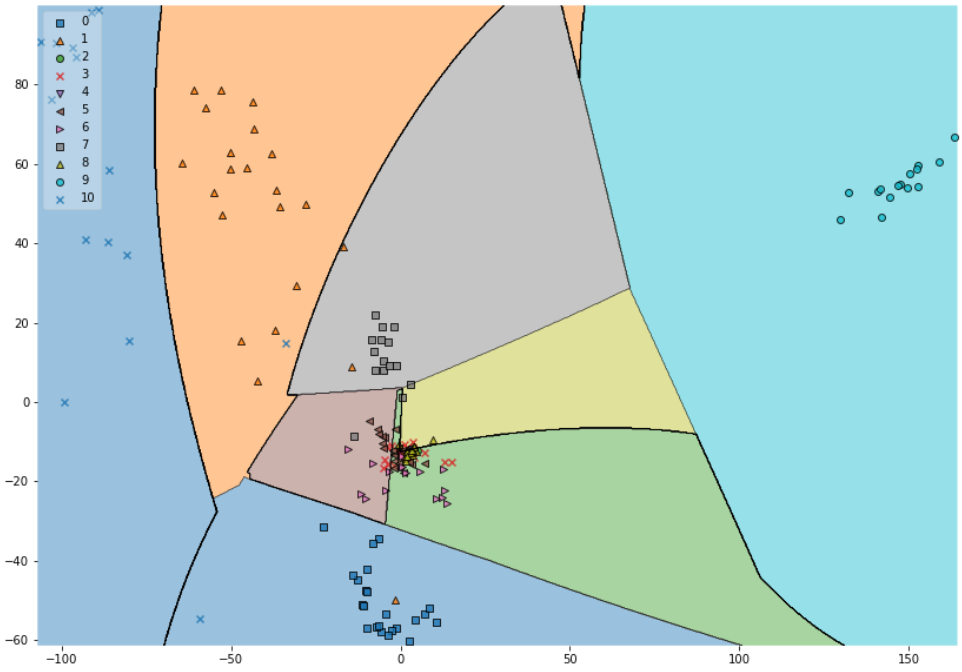
Scatter plot and histogram



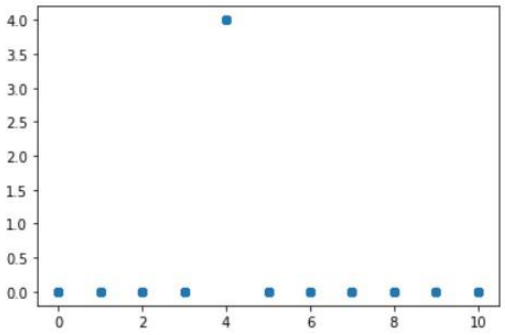
The histogram has plotted the difference between ytest and predictedvalues.

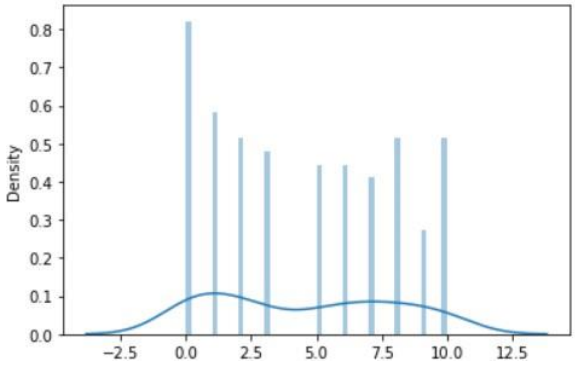
SVC

```
[[13  0  0  0  0  0  0  0  0  0  0]
 [ 0 16  0  0  0  0  0  0  0  0  1]
 [ 0  0  6  0  1  0  0  0  8  0  0]
 [ 0  0  0 13  0  1  0  0  0  0  0]
 [ 0  0  0  0 11  0  0  0  0  0  0]
 [ 0  0  0  0  0 13  0  0  0  0  0]
 [ 0  0  0  0  0  0 13  0  0  0  0]
 [ 0  0  0  0  0  0  0 12  0  0  0]
 [ 0  0  0  0  0  0  0  0 15  0  0]
 [ 0  0  0  0  0  0  0  0  0  8  0]
 [ 0  3  0  0  0  0  0  0  0  0 12]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        13
         1.0       0.84      0.94      0.89        17
         2.0       1.00      0.40      0.57        15
         3.0       1.00      0.93      0.96        14
         4.0       0.92      1.00      0.96        11
         5.0       0.93      1.00      0.96        13
         6.0       1.00      1.00      1.00        13
         7.0       1.00      1.00      1.00        12
         8.0       0.65      1.00      0.79        15
         9.0       1.00      1.00      1.00         8
        10.0       0.92      0.80      0.86        15

    accuracy                           0.90       146
   macro avg       0.93      0.92      0.91       146
weighted avg       0.93      0.90      0.90       146

MAE:  0.6027397260273972
MSE:  4.2465753424657535
RMSE:  2.060722043960746
R Squared:  0.5846258328591618
```



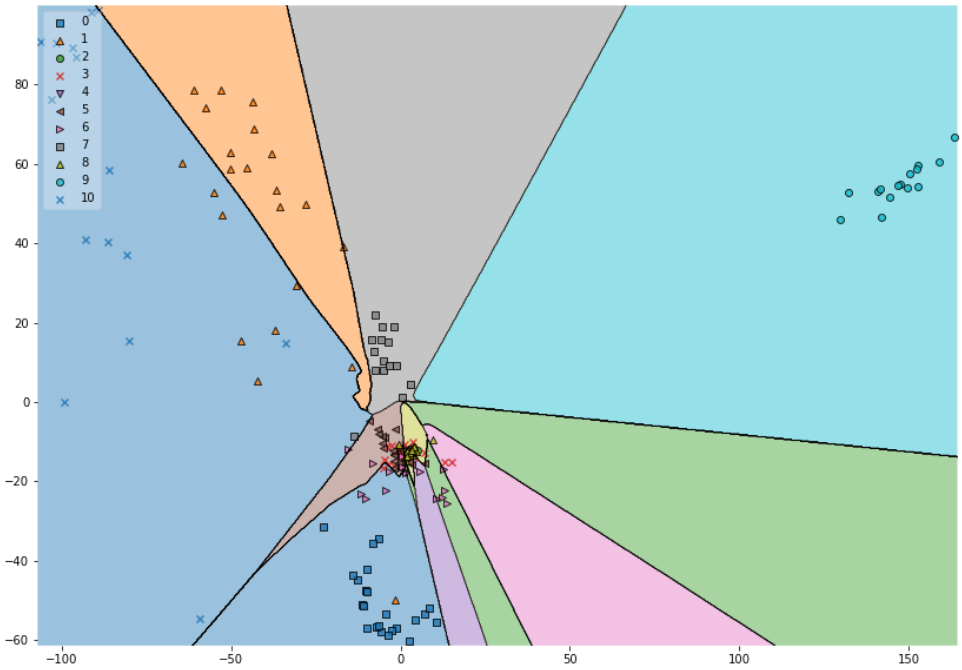
Scatter plot and histogram



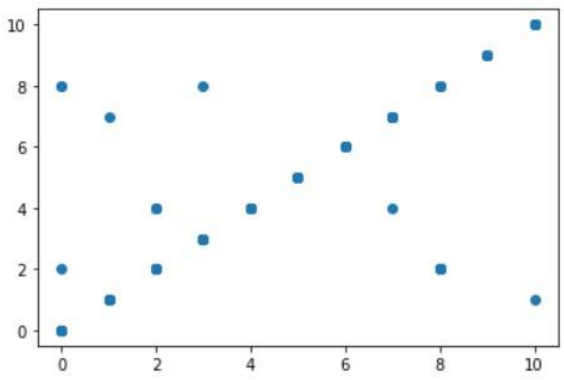


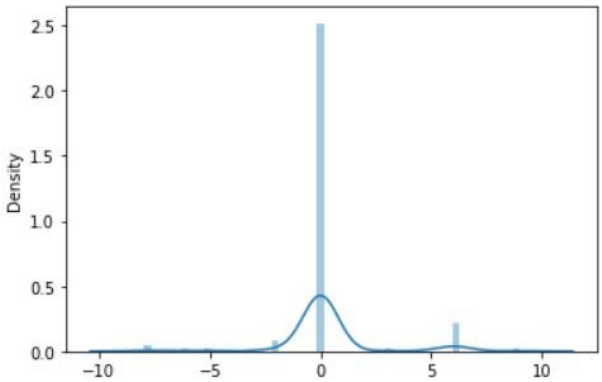
Multi Layer Perceptron

```
[[10  0  1  0  0  0  0  0  2  0  0]
 [ 0 16  0  0  0  0  0  1  0  0  0]
 [ 0  0 12  0  3  0  0  0  0  0  0]
 [ 0  0  0 13  0  0  0  0  1  0  0]
 [ 0  0  0  0 11  0  0  0  0  0  0]
 [ 0  0  0  0  0 13  0  0  0  0  0]
 [ 0  0  0  0  0  0 13  0  0  0  0]
 [ 0  0  0  0  1  0  0 11  0  0  0]
 [ 0  0 11  0  0  0  0  0  4  0  0]
 [ 0  0  0  0  0  0  0  0  0  8  0]
 [ 0  1  0  0  0  0  0  0  0  0 14]]
              precision    recall  f1-score   support

         0.0       1.00      0.77      0.87        13
         1.0       0.94      0.94      0.94        17
         2.0       0.50      0.80      0.62        15
         3.0       1.00      0.93      0.96        14
         4.0       0.73      1.00      0.85        11
         5.0       1.00      1.00      1.00        13
         6.0       1.00      1.00      1.00        13
         7.0       0.92      0.92      0.92        12
         8.0       0.57      0.27      0.36        15
         9.0       1.00      1.00      1.00         8
        10.0       1.00      0.93      0.97        15

    accuracy                           0.86       146
   macro avg       0.88      0.87      0.86       146
weighted avg       0.87      0.86      0.85       146

MAE:  0.773972602739726
MSE:  4.732876712328767
RMSE:  2.1755175734359784
R Squared:  0.537058791138195
```



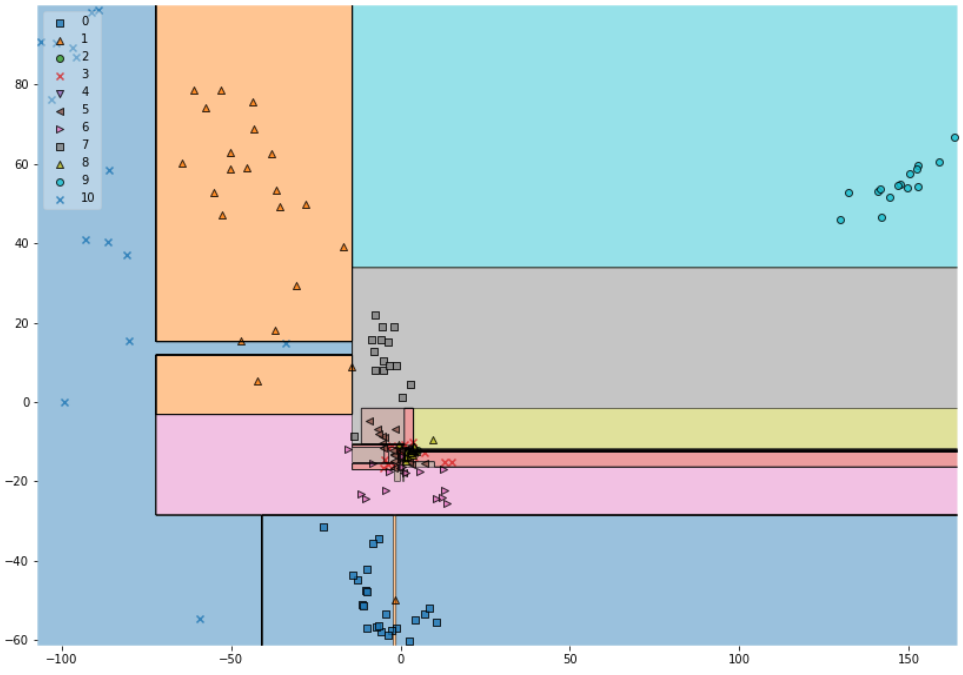
Scatter plot and histogram



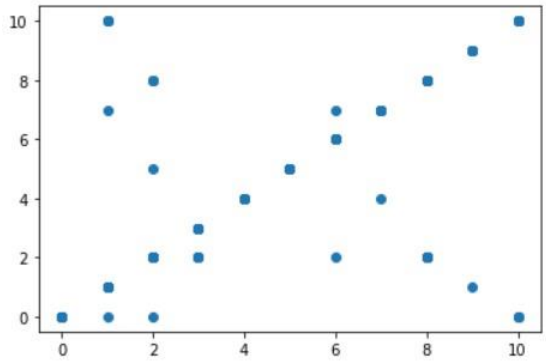


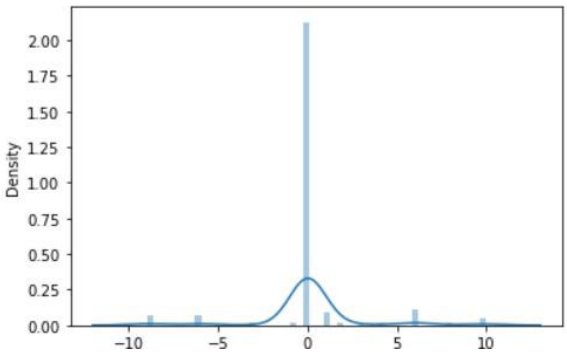
Decision Tree

```
[[13  0  0  0  0  0  0  0  0  0  0]
 [ 1 11  0  0  0  0  0  1  0  0  4]
 [ 1  0 10  0  0  1  0  0  3  0  0]
 [ 0  0  4 10  0  0  0  0  0  0  0]
 [ 0  0  0  0 11  0  0  0  0  0  0]
 [ 0  0  0  0  0 13  0  0  0  0  0]
 [ 0  0  1  0  0  0 11  1  0  0  0]
 [ 0  0  0  0  1  0  0 11  0  0  0]
 [ 0  0  6  0  0  0  0  0  9  0  0]
 [ 0  1  0  0  0  0  0  0  0  7  0]
 [ 3  0  0  0  0  0  0  0  0  0 12]]
              precision    recall  f1-score   support

         0.0       0.72      1.00      0.84        13
         1.0       0.92      0.65      0.76        17
         2.0       0.48      0.67      0.56        15
         3.0       1.00      0.71      0.83        14
         4.0       0.92      1.00      0.96        11
         5.0       0.93      1.00      0.96        13
         6.0       1.00      0.85      0.92        13
         7.0       0.85      0.92      0.88        12
         8.0       0.75      0.60      0.67        15
         9.0       1.00      0.88      0.93         8
        10.0       0.75      0.80      0.77        15

    accuracy                           0.81       146
   macro avg       0.85      0.82      0.83       146
weighted avg       0.84      0.81      0.81       146

MAE:  1.0410958904109588
MSE:  7.47945205479452
RMSE:  2.7348586900961664
R Squared:  0.26840549916484646
```



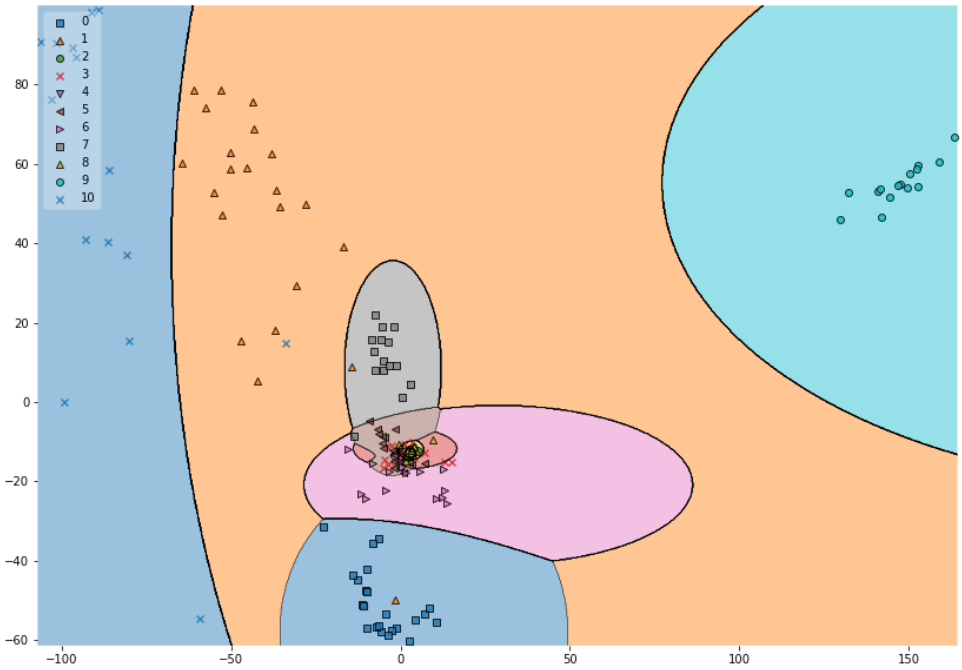
Scatter plot and histogram



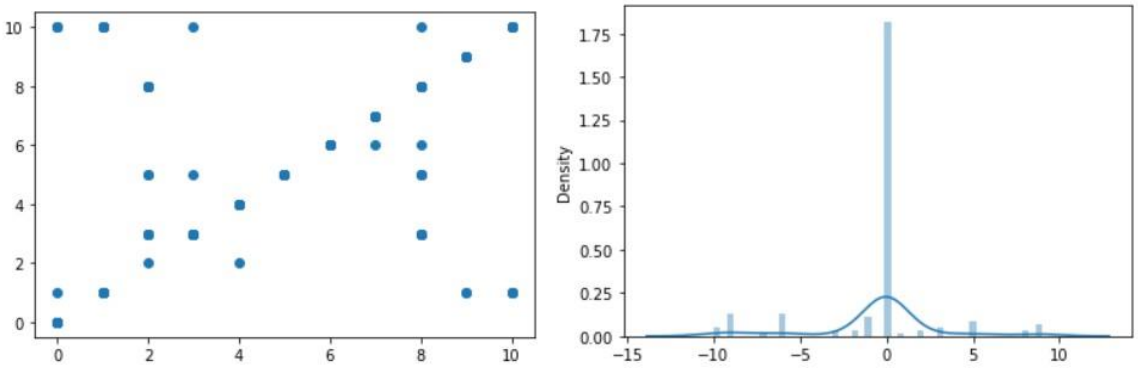


Gaussian Naïve Bayes

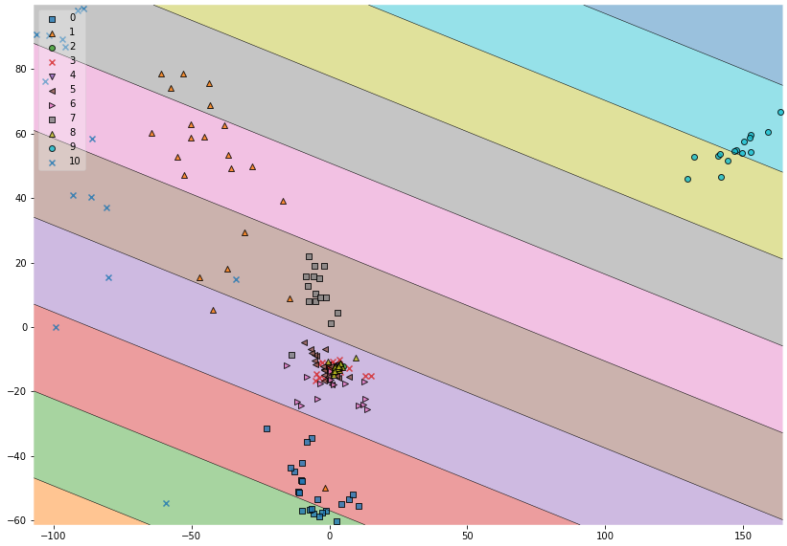
```
[[ 9  1  0  0  0  0  0  0  0  3]
 [ 0 10  0  0  0  0  0  0  0  7]
 [ 0  0  1  5  0  2  0  0  7  0  0]
 [ 0  0  0 12  0  1  0  0  0  0  1]
 [ 0  0  1  0 10  0  0  0  0  0  0]
 [ 0  0  0  0  0 13  0  0  0  0  0]
 [ 0  0  0  0  0  0 13  0  0  0  0]
 [ 0  0  0  0  0  0  1 11  0  0  0]
 [ 0  0  0  5  0  3  1  0  5  0  1]
 [ 0  2  0  0  0  0  0  0  0  6  0]
 [ 0  4  0  0  0  0  0  0  0  0 11]]
              precision    recall  f1-score   support

         0.0       1.00      0.69      0.82        13
         1.0       0.59      0.59      0.59        17
         2.0       0.50      0.07      0.12        15
         3.0       0.55      0.86      0.67        14
         4.0       1.00      0.91      0.95        11
         5.0       0.68      1.00      0.81        13
         6.0       0.87      1.00      0.93        13
         7.0       1.00      0.92      0.96        12
         8.0       0.42      0.33      0.37        15
         9.0       1.00      0.75      0.86         8
        10.0       0.48      0.73      0.58        15

    accuracy                           0.69       146
   macro avg       0.73      0.71      0.70       146
weighted avg       0.70      0.69      0.67       146

MAE:  1.7054794520547945
MSE:  12.417808219178083
RMSE:  3.5238910623312525
R Squared:  -0.21463445971990236
```



Scatter plot and histogram

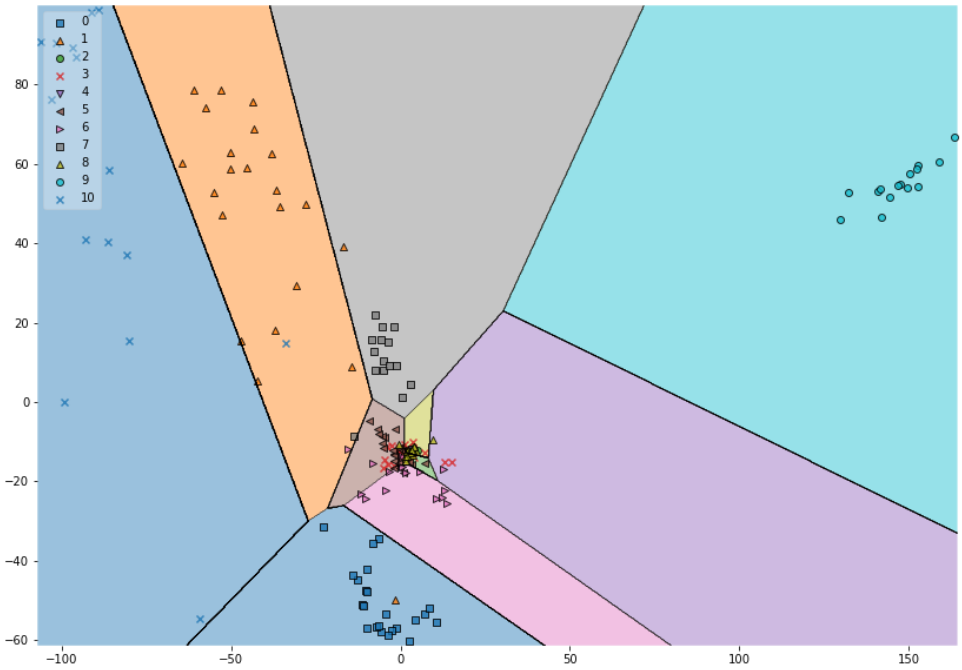


Linear Regression

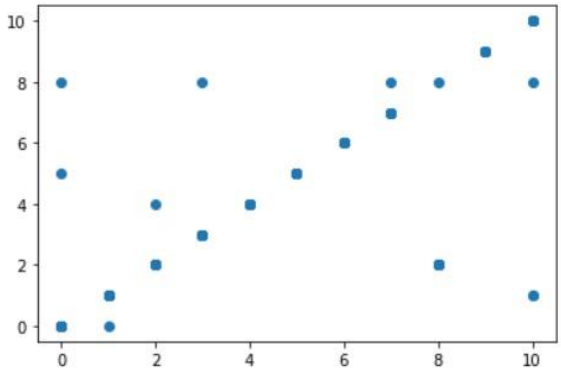


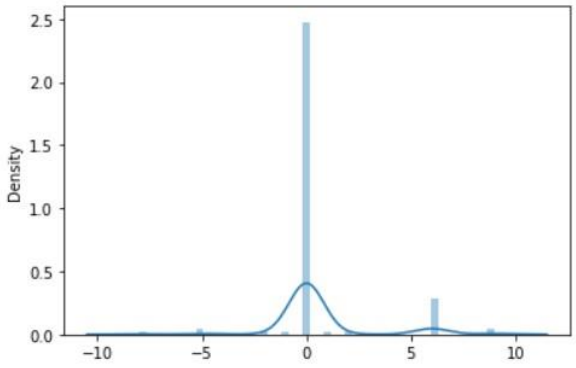
Logistic Regression

```
[[11  0  0  0  0  1  0  0  1  0  0]
 [ 1 16  0  0  0  0  0  0  0  0  0]
 [ 0  0 14  0  1  0  0  0  0  0  0]
 [ 0  0  0 13  0  0  0  0  1  0  0]
 [ 0  0  0  0 11  0  0  0  0  0  0]
 [ 0  0  0  0  0 13  0  0  0  0  0]
 [ 0  0  0  0  0  0 13  0  0  0  0]
 [ 0  0  0  0  0  0  0 11  1  0  0]
 [ 0  0 14  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  8  0]
 [ 0  2  0  0  0  0  0  0  1  0 12]]
              precision    recall  f1-score   support

         0.0       0.92      0.85      0.88        13
         1.0       0.89      0.94      0.91        17
         2.0       0.50      0.93      0.65        15
         3.0       1.00      0.93      0.96        14
         4.0       0.92      1.00      0.96        11
         5.0       0.93      1.00      0.96        13
         6.0       1.00      1.00      1.00        13
         7.0       1.00      0.92      0.96        12
         8.0       0.20      0.07      0.10        15
         9.0       1.00      1.00      1.00         8
        10.0       1.00      0.80      0.89        15

    accuracy                           0.84       146
   macro avg       0.85      0.86      0.84       146
weighted avg       0.83      0.84      0.83       146

MAE:  0.863013698630137
MSE:  5.410958904109589
RMSE:  2.326146793327882
R Squared:  0.47073291606248047
```



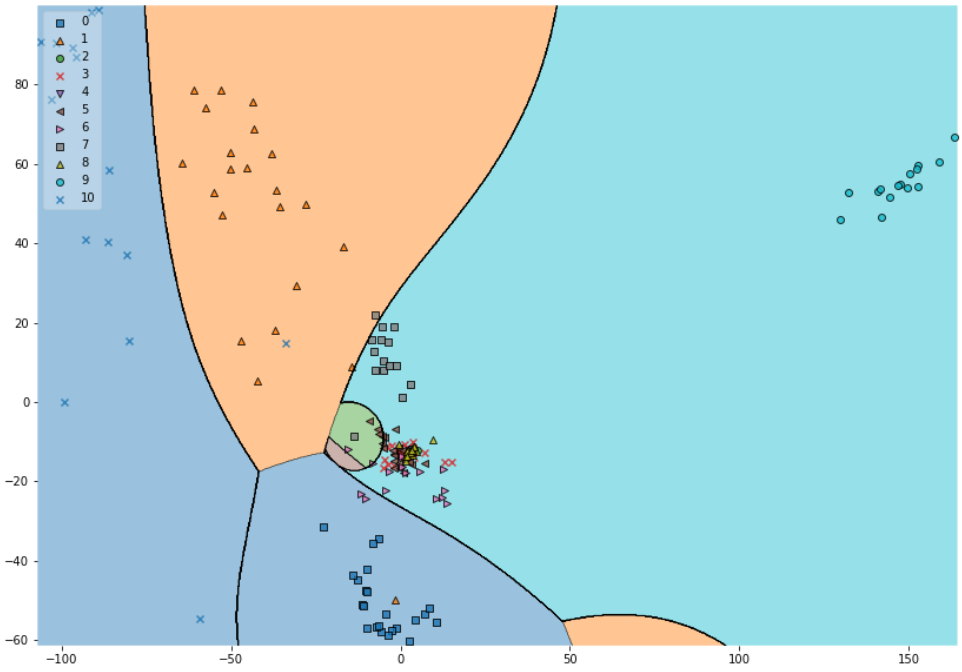
Scatter plot and histogram



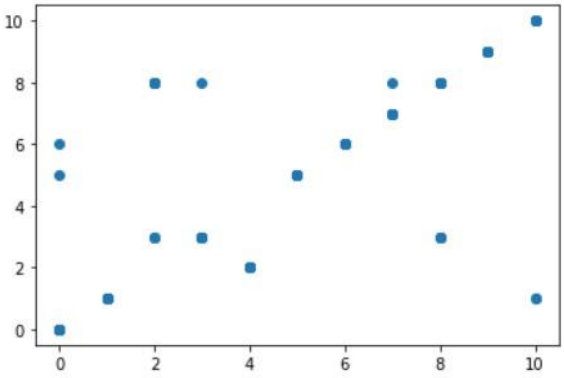


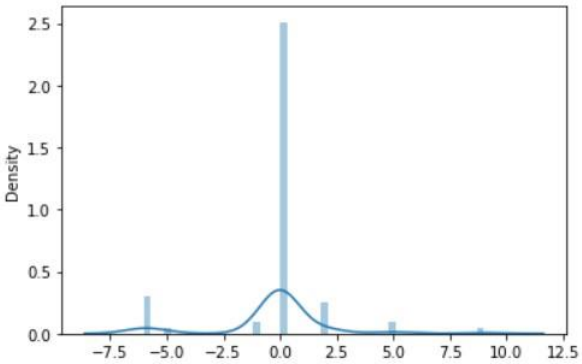
Gaussian Process

```
[[11  0  0  0  0  1  1  0  0  0  0]
 [ 0 17  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  3  0  0  0  0 12  0  0]
 [ 0  0  0 13  0  0  0  0  1  0  0]
 [ 0  0 11  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 13  0  0  0  0  0]
 [ 0  0  0  0  0  0 13  0  0  0  0]
 [ 0  0  0  0  0  0  0 11  1  0  0]
 [ 0  0  0  4  0  0  0  0 11  0  0]
 [ 0  0  0  0  0  0  0  0  0  8  0]
 [ 0  2  0  0  0  0  0  0  0  0 13]]
              precision    recall  f1-score   support

         0.0       1.00      0.85      0.92        13
         1.0       0.89      1.00      0.94        17
         2.0       0.00      0.00      0.00        15
         3.0       0.65      0.93      0.76        14
         4.0       0.00      0.00      0.00        11
         5.0       0.93      1.00      0.96        13
         6.0       0.93      1.00      0.96        13
         7.0       1.00      0.92      0.96        12
         8.0       0.44      0.73      0.55        15
         9.0       1.00      1.00      1.00         8
        10.0       1.00      0.87      0.93        15

    accuracy                           0.75       146
   macro avg       0.71      0.75      0.73       146
weighted avg       0.71      0.75      0.72       146

MAE:  1.0410958904109588
MSE:  5.671232876712328
RMSE:  2.3814350456630824
R Squared:  0.4452744993667517
```



Scatter plot and histogram





Comparison

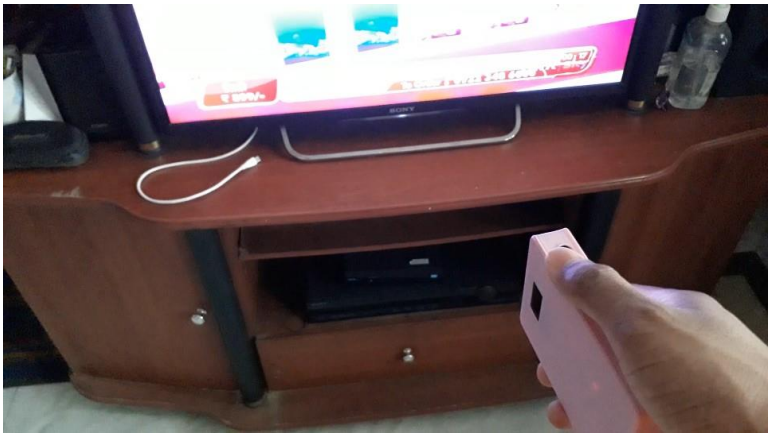
Classifier	Accuracy(%)	MAE	MSE	RMSE	R2 Score
Random Forest	97	0.15	1.17	1.08	0.88
MLP	86	0.77	4.73	2.17	0.537
SVC	90	0.60	4.24	2.06	0.584
Naïve Bayes	69	1.70	12.41	3.52	-0.214
Logistic	84	0.86	5.41	2.32	0.407
Linear	Error	3.22	24.15	4.91	-1.36
Decision Tree	81	1.04	7.47	2.73	0.26
Gaussian Process	75	1.04	5.67	2.38	0.44

Inference

- The random forest classifier has the highest accuracy among all these classifiers.
- The Naïve Bayes classifier has the lowest accuracy among all these classifiers.
- SVM takes up a lot of memory when converted into C code:

```
Error compiling for board NodeMCU 1.0 (ESP-12E Module).

cc1plus.exe: out of memory allocating 150435692 bytes
exit status 1
Error compiling for board NodeMCU 1.0 (ESP-12E Module).
```

Using the device to control a TV (check the given link for all videos)



Conclusion

We have successfully implemented machine learning on a microcontroller and used it to recognise the gestures made by the user. We have also seen that Random Forest classifier gave the highest accuracy among all classifiers of 97%.

We have made a handheld device which can recognise the gestures and send it to a computer as an input device. We can also control televisions using the IR sender built in to the device.

For the project files and videos, please go to this link:

https://drive.google.com/drive/folders/1BVnznAhVSvPpDGUZEza924hyiwaR41iG?usp=sharing

References

1. Machine learning on Arduino by Eloquent: https://eloquentarduino.github.io/2019/12/how-to-do-gesture-identification-on-arduino/#tocrun-the-inference
2. Video tutorial by Neutrino: https://www.youtube.com/watch?v=NuA1YfmMS0w&list=PLnE78kV3yoOCpjkf3mPX4Q4f3bYR4GM-v&index=4
3. Tensorflow for microcontrollers:

https://blog.tensorflow.org/2019/11/how-to-get-started-with-machine.html

4. ESP8266 and computer communication by Instructables: https://www.instructables.com/ESP8266-and-Python-Communication-ForNoobs/