**ECE 510: Intelligent Robotics: III**

**Fritz Robot Head**

**By**

**Abhiraj Abhaykumar Eksambekar**

**Under the guidance of**

**Prof. Marek Perkowski**

**Final Report**

**Table of Contents**

[Introduction 4](#_Toc74319679)

[Hardware purchases 5](#_Toc74319680)

[Hardware build 5](#_Toc74319681)

[Fritz Robot Head 6](#_Toc74319682)

[Hardware flaws 9](#_Toc74319683)

[Hardware upgrades 9](#_Toc74319684)

[Provided software. 10](#_Toc74319685)

[Arduino pin out and connections and descriptions. 11](#_Toc74319686)

[ARDUINO UNO AND MOTOR CONTROL BOARD DESCRIPTION 11](#_Toc74319687)

[Arduino software concept and working with difficulties. 14](#_Toc74319688)

[Windows software’s, library files, audio files, working and usage. 15](#_Toc74319689)

[Gestures Library File: 15](#_Toc74319690)

[Emotion Library File 16](#_Toc74319691)

[GUI.py 16](#_Toc74319692)

[FLOW AND USAGE: 17](#_Toc74319693)

[Add Delete Gesture 18](#_Toc74319694)

[EXAMPLE TO ADD A NEW GESTURE: 21](#_Toc74319695)

[Adding emotions to library manual 27](#_Toc74319696)

[Csv file working and usage 28](#_Toc74319697)

[Additional documentation- Software from other sources description 30](#_Toc74319698)

[Visual Studio Code 30](#_Toc74319699)

[OBS 30](#_Toc74319700)

[Hardware Troubleshooting 31](#_Toc74319701)

[Power Connection Problem 31](#_Toc74319702)

[Software troubleshooting 31](#_Toc74319703)

[Gui Arduino disconnect while running. 31](#_Toc74319704)

[Gui crash due to library improper format. 31](#_Toc74319705)

[Conclusion 32](#_Toc74319706)

**List of Figures**

[Figure 1 Head Front 6](#_Toc74319829)

[Figure 2 Head Back 7](#_Toc74319830)

[Figure 3 Head Right 7](#_Toc74319831)

[Figure 4 Head Left 8](#_Toc74319832)

[Figure 5 Head Top 8](#_Toc74319833)

[Figure 6 Head Top Connection Problem 9](#_Toc74319834)

[Figure 7 Arduino Uno Board 11](#_Toc74319835)

[Figure 8 Arduino Sensor Shield 12](#_Toc74319836)

[Figure 9 Gesture Library file Contents 15](#_Toc74319837)

[Figure 10 Emotion Library Contents 16](#_Toc74319838)

[Figure 11 GUI Flow Chart 17](#_Toc74319839)

[Figure 12 GUI Setup 18](#_Toc74319840)

[Figure 13 Add Gesture Menu 19](#_Toc74319841)

[Figure 14 Add Gesture Details screen 19](#_Toc74319842)

[Figure 15 Add Gestures Motions 19](#_Toc74319843)

[Figure 16 Add Postures Verification 20](#_Toc74319844)

[Figure 17 Add Gesture Verification 20](#_Toc74319845)

[Figure 18 List of gestures present in library 21](#_Toc74319846)

[Figure 19 Add Gesture Step 1 21](#_Toc74319847)

[Figure 20 Add Gesture Step 1.1 21](#_Toc74319848)

[Figure 21 Add Gesture Step 4 22](#_Toc74319849)

[Figure 22 Add Gesture Step 5 22](#_Toc74319850)

[Figure 23 Add Gesture Step 6 22](#_Toc74319851)

[Figure 24 Add Gesture Step 7 22](#_Toc74319852)

[Figure 25 Add Gesture Step 8 23](#_Toc74319853)

[Figure 26 Add Gesture Step 9 24](#_Toc74319854)

[Figure 27 Add Gesture Step 10 24](#_Toc74319855)

[Figure 28 Add Gesture Step 11 25](#_Toc74319856)

[Figure 29 Add Gesture Step 12.1 25](#_Toc74319857)

[Figure 30 Add Gesture Step 12.2 26](#_Toc74319858)

[Figure 31 Add Gesture Step 13 26](#_Toc74319859)

[Figure 32 GUI after Gesture is added 26](#_Toc74319860)

[Figure 33 Add Gesture from CSV 1 28](#_Toc74319861)

[Figure 34 Add Gesture from CSV 2 28](#_Toc74319862)

[Figure 35 Landmark Locations 29](#_Toc74319863)

Introduction

Fritz bit is a primary robot head to show elementary and basic human-like emotions. It has 2 eyebrows, 2 eyes, 2 eyelids, 1 mouth, and 1 neck. Servo motors control these parts. This project aims to control these motors to show emotions and gestures set to correspond to the audio.

Hardware purchases

Purchased hardware includes-

* https://kerkits.com/products/fritz-the-robotic-head - Contains all parts for fritz robot head, Arduino UNO, Servo Shield, and battery attachments.
* Extra components like variable power supply adapter, glue gun, glue sticks, hot glue were purchased from personal funds.

Hardware build

It took around 4-5 days to build this robot head from scratch. https://cdn.shopify.com/s/files/1/1320/7675/files/Fritz\_assembly\_Instructions\_VERS3.pdf?1855087446633294682 link helps to make the head. Glue gun and hot glue came in handy for assembling all the ports.

The parts were difficult to connect as they become more compact. It took much time to make proper connections between moving parts and motors in very tight spaces. There are 11 moving parts, each connected to a servo motor. The motors connected to facial expressions are mini servo motors, whereas the motors responsible for moving the neck horizontal and vertical are powerful with high torque capacity. Eyebrows are rotated using a single motor. Eyelids are also controlled using a single motor. The eyeball is controlled using 2 motors. One of the motors is used for horizontal motion and other one for vertical motion of the eyeballs. The mouth has 3 moving parts. The right and left parts of lip are controlled directly with the motor. They move up and down. The third motor is connected to lower jaw. It also moves vertically making an impression of speaking.

Fritz Robot Head



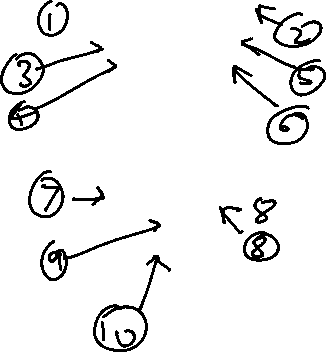


Figure 1 Head Front

|  |  |
| --- | --- |
| 1 | Left Eyebrow |
| 2 | Right Eyebrow |
| 3 | Left Eyelid |
| 4 | Left Eyeball |
| 5 | Right Eyelid |
| 6 | Right Eyeball |
| 7 | Left Lip |
| 8 | Right Lip |
| 9 | Bottom Lip |
| 10 | Neck |



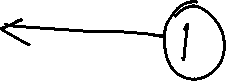


Figure 2 Head Back

1**-** All servo motors connected to sensor board which is mounted on Arduino.

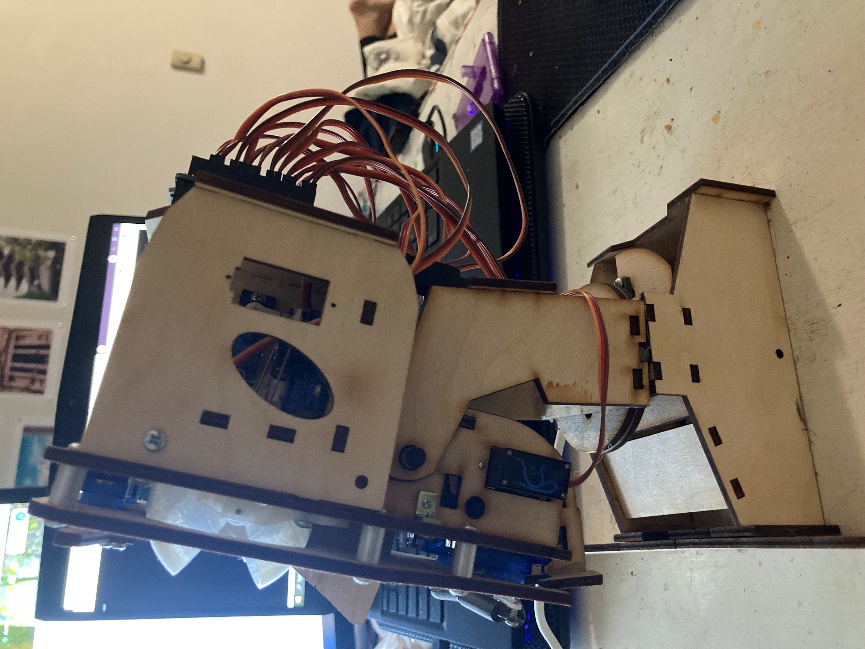
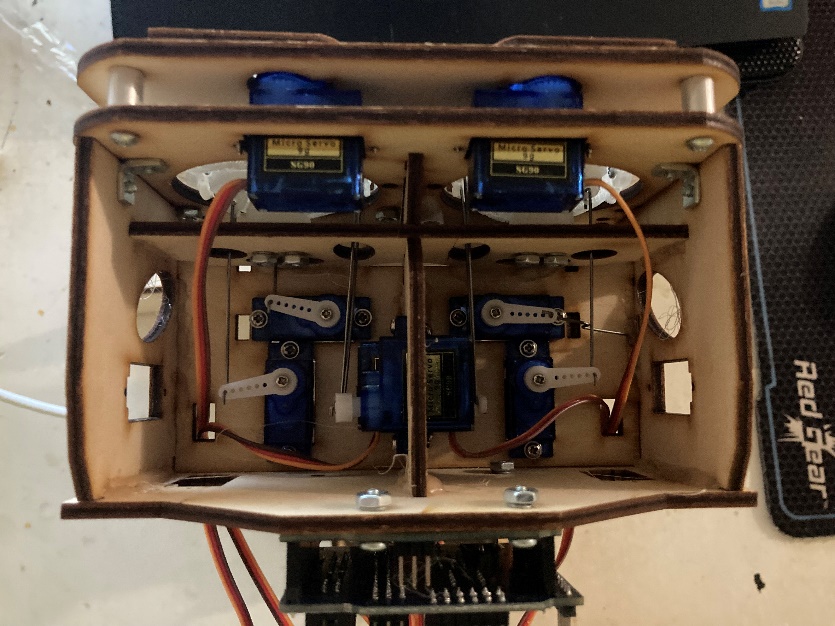


Figure 3 Head Right



Figure 4 Head Left



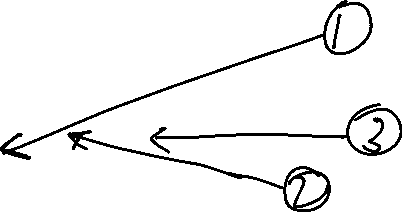


Figure 5 Head Top



Figure 6 Head Top Connection Problem

1. The metal connection between motor and eyeball that is loose.
2. Hole from where this contact needs to pass to find attachment to eye.
3. Eye hinge where the metal contact is to be attached.

The above image shows why it is difficult to reattach any contact that has been separated after the head is completely assembled.

Hardware flaws

The connection to these moving parts is made in too compact spaces. Connections for eyeballs and eyelids are to be made properly. If the connection is lost during rotation of the motor, it is very difficult to connect them again. It might be even necessary to open the joint connections just to make the connections. But most of the parts are joined to each other by hot glue gun and therefore connections might not be able to rebuild again.

Another hardware flaw is the Arduino to PC serial communication wire obstructs the vertical motion of the neck.

Hardware upgrades

Cutting the robot head back and attaching a latch/hinge to place it back can be helpful. This upgrade will enable access to the moving part attachments a little easier from behind. Another possible upgrade is placing the Arduino board at the base instead of back. Placing Arduino differently will free the vertical motion of the neck.

Provided software.

The designers provide a software for this project. Link to this software and instructions to using it: https://kerkits.com/pages/fritz-support-page-assembly-software-installation-etc. This software provides gui with variety of options. There are set of predefined gestures like smiling, angry, sad, disappointed, and many more. After adding more gestures, they are shown under this same label. To add new gestures according to a audio, there is a very easy and convenient way. User can add a audio file to the gui. The software shows an audio map for that audio, and an image of the fritz robot head. The user can click on the audio visualization to a specific point in time to set gestures at that time. Once the user clicks on audio map at required time, the user can move parts of robot image which corresponds to each specific motor on the robot head. Similar way, user can put multiple gestures for the audio file at different point of the audio.

Arduino pin out and connections and descriptions.

ARDUINO UNO AND MOTOR CONTROL BOARD DESCRIPTION

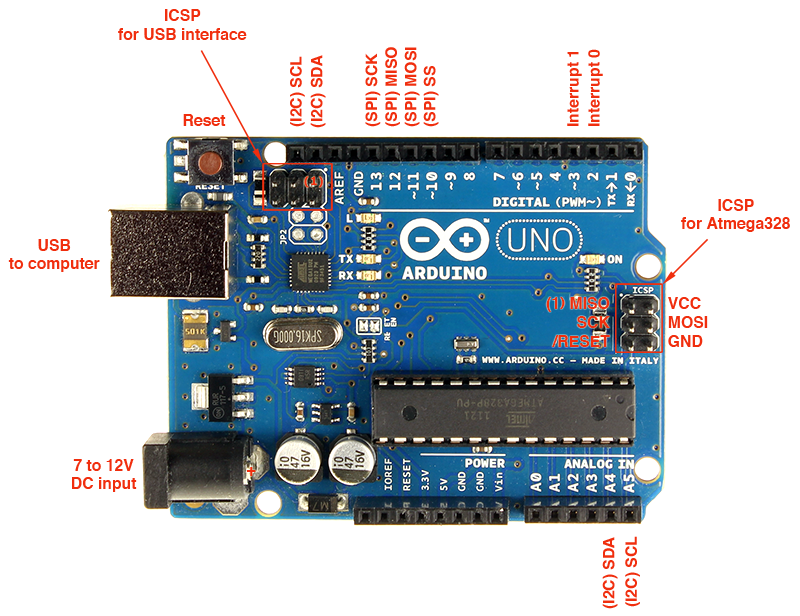


Figure 7 Arduino Uno Board

Arduino board is a embedded microcontroller board for simple project developments. This board uses atmega-328p microcontroller. A proper firmware is already written onto this microcontroller. A logic can be written in C++ to control GPIO, and communication parts on this board. Arduino has 13 digital pints and 6 analog pins. It can be programmed with IDEs provided on the Arduino website. The IDE provides editor, compiler and uploader to Arduino. It also provides serial communication window to send data serially to Arduino and also display data received serially from Arduino. Arduino board can be supplied with power source in many ways. Connecting USB port to computer powers up the Arduino. It can also be powered up by giving 5v supply to Vin pin and ground to ground pin. It is not recommended to use Vin pin as if the voltage goes above some threshold, it can damage the entire board. Best practice to power the Arduino board is to use the DC input port. This port is connected to a voltage regulator and therefore handles excess voltage being supplied to microcontroller. And therefore, protectes the microcontroller from damage.

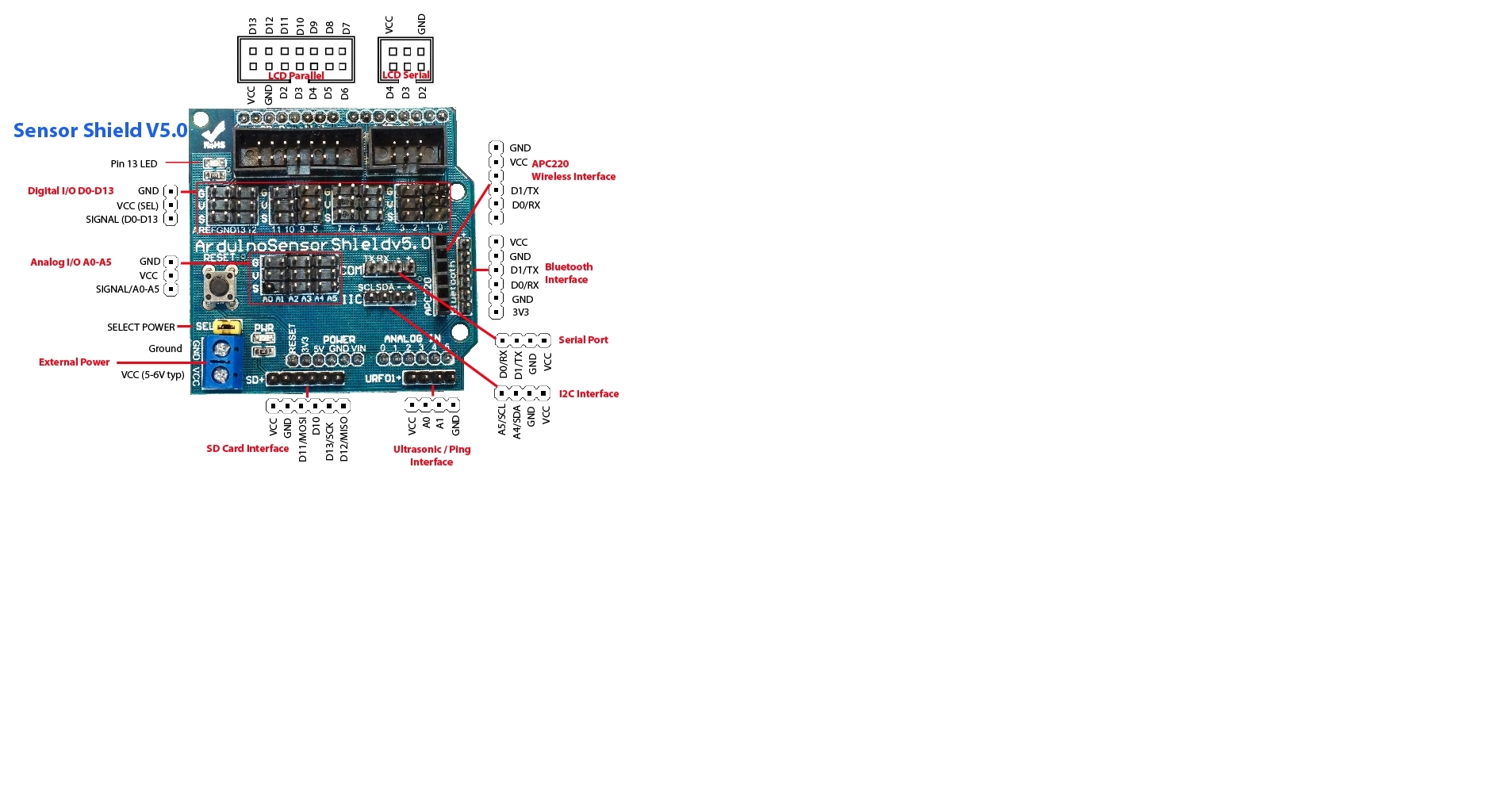
****

Figure 8 Arduino Sensor Shield

This is a simple Arduino sensor shield that goes on top of Arduino uno. It provides ground and VCC signal for each digital and analog pin on Arduino separately on its side. It also provides set of different connectors as required for power, ground, and TX/RX pins. It also has a external power input port. It can be supplied up to 9V of power. This power will be regulated to 5 volts and be provided on all the Vcc pins that are on the side of analog and digital pins.

Arduino is connected to computer via serial communication cable. The motor shield is placed on the Arduino uno. This shild provides each analog and digital pin a +5 and Gnd pin on its sides. This makes it easier to connect servo motors directly to this shield without needing a breadboard. This board also has power in to supply required 5V external power supply to the motors and external components. The following table contains face part connected to respective motor and its corresponding control pin on Arduino board.

|  |  |
| --- | --- |
| Face part | Arduino Pin |
| Left Eyebrow | D2 |
| Right Eyebrow | D3 |
| Left Eyelid | D4 |
| Right Eyelid | D5 |
| Horizontal Eye Left | D6 |
| Horizontal Eye Right | D7 |
| Vertical Eye Left | A0 |
| Vertical Eye Right | D9 |
| Left Lip | D10 |
| Right Lip | D11 |
| Bottom Lip | D12 |
| Neck Horizontal | D13 |
| Neck Vertical | D8 |

Current setup uses variable power adapter and not provided battery connector.

Arduino software concept and working with difficulties.

Arduino is responsible for controlling the servo motors. Arduino should be always careful with motor rotation that it does not exceed certain limits, otherwise, it will damage the face parts beyond repair. Therefore, minimum and maximum limits are defined for each motor separately through testing.

The script initializes servo class for each motor. Once the Arduino boots up, it setups servo motor to corresponding Arduino pin. It also initializes serial communication with a baud rate of 57600. During initialization, it also finds center position of each motor and sets it to 50% rotation of its minimum and maximum set limits.

After initialization is completed successfully, the Arduino runs into loop to check any serial communication available between PC and Arduino and then process the data as per.

The loop start checks for any data available serially. If no data present, continue the loop. If serial data present, it reads the data byte by byte. It checks if the first byre is for character ‘g’. If it is, the flag for command receiving is set and continues the loop. In the next loop, if another byte is available and if it corresponds to an alphabet, it stores it for later use. The next 2 bytes in serial communication should correspond to digits which will be percent the motor needs to rotate in accordance with the minimum and maximum limits set. The previous alphabet saved is used after the digits are received. The following command correspond the parts on face connecting the motor.

|  |  |
| --- | --- |
| ga | Left Eyebrow |
| gb | Right Eyebrow |
| gc | Left Eyelid |
| gd | Right Eyelid |
| ge | Horizontal Eye Left |
| gf | Horrizontal Eye Right |
| gh | Vertical Eye Left |
| gi | Vertical Eye Right |
| gj | Left Lip |
| gk | Right Lip |
| gl | Bottom Lip |
| gm | Neck Horizontal |
| gn | Neck Vertical |

As previously stated, pc sends percent value to Arduino, therefore, Arduino needs to process this value to rotate the motor as per. Arduino script maps the percent value to the min and max limits of the motor and then writes the angle to the motor.

This is the fastest way of data transfer since another method, where Arduino reads entire string instead of a single character. Reading entire string takes noticeable time and causes delay in execution and even data loss.

Windows software’s, library files, audio files, working and usage.

I have created multiple software’s in python for this project which cover different tasks for this project. Even though you may think the program is simple, there is strong logic behind it that took too much time to write the script for the ease of user and towards correctness of required working functionality.

Gestures Library File:

This file contains information of audio file, button name and gestures mapping relative to the audio. It is in JSON format. It is read as dictionary in python script.

The following image shows format of this file.

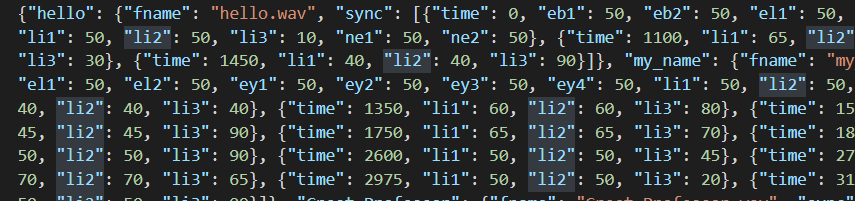


Figure 9 Gesture Library file Contents

The library consists of nested libraries. The keys of dictionary indicates name/title which is later used as button name as well. The 2nd layer dictionary has 2 keys, one is “fname” indicating audio file name, and other being sync. The sync key has as a list of dictionaries as its value. These dictionaries have one time key, indication at which point in time of the audio, the corresponding gesture is to be made. The time is in microseconds. The next key parameters correspond to motor attached to part and its corresponding movement change is 0-99 percent value.

The following table are useful to understand more about these names and values for motors.

|  |  |  |  |
| --- | --- | --- | --- |
| Face Part | Library Name | Percent Range | Corresponding Motion |
| Left Eyebrow | eb1 | 00-99 | Up – Down |
| Right Eyebrow | eb2 | 00-99 | Up – Down |
| Left Eyelid | el1 | 00-99 | Close – Open |
| Right Eyelid | el2 | 00-99 | Close – Open |
| Horizontal Eye Left | ey1 | 00-99 | Left – Right |
| Horizontal Eye Right | ey2 | 00-99 | Left – Right |
| Vertical Eye Left | ey3 | 00-99 | NOT WORKING |
| Vertical Eye Right | ey4 | 00-99 | Up – Down |
| Left Lip | li1 | 00-99 | Up – Down |
| Right Lip | li2 | 00-99 | Up – Down |
| Bottom Lip | li3 | 00-99 | Open – close |
| Neck Horizontal | ne1 | 00-99 | Left – Right |
| Neck Vertical | ne2 | 00-99 | Up – Down |

Emotion Library File

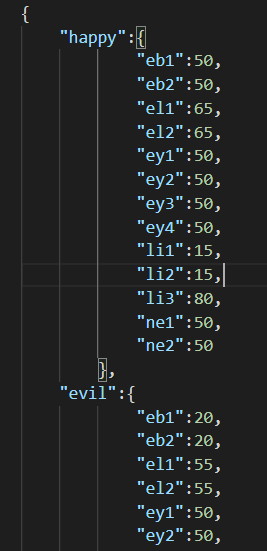


Figure 10 Emotion Library Contents

This library file is same as the gesture file but it has only 2 levels of dictionary. The first key indicates the name of emotion to be used as button name. The nested dictionary has values of motors to be set for respective expression.

GUI.py

This script makes it easy for user to show gestures on the fritz robot head and play the corresponding video simultaneously taking the information from gesture library. It also features showing emotions regardless of any audio present taken from the emotion library.

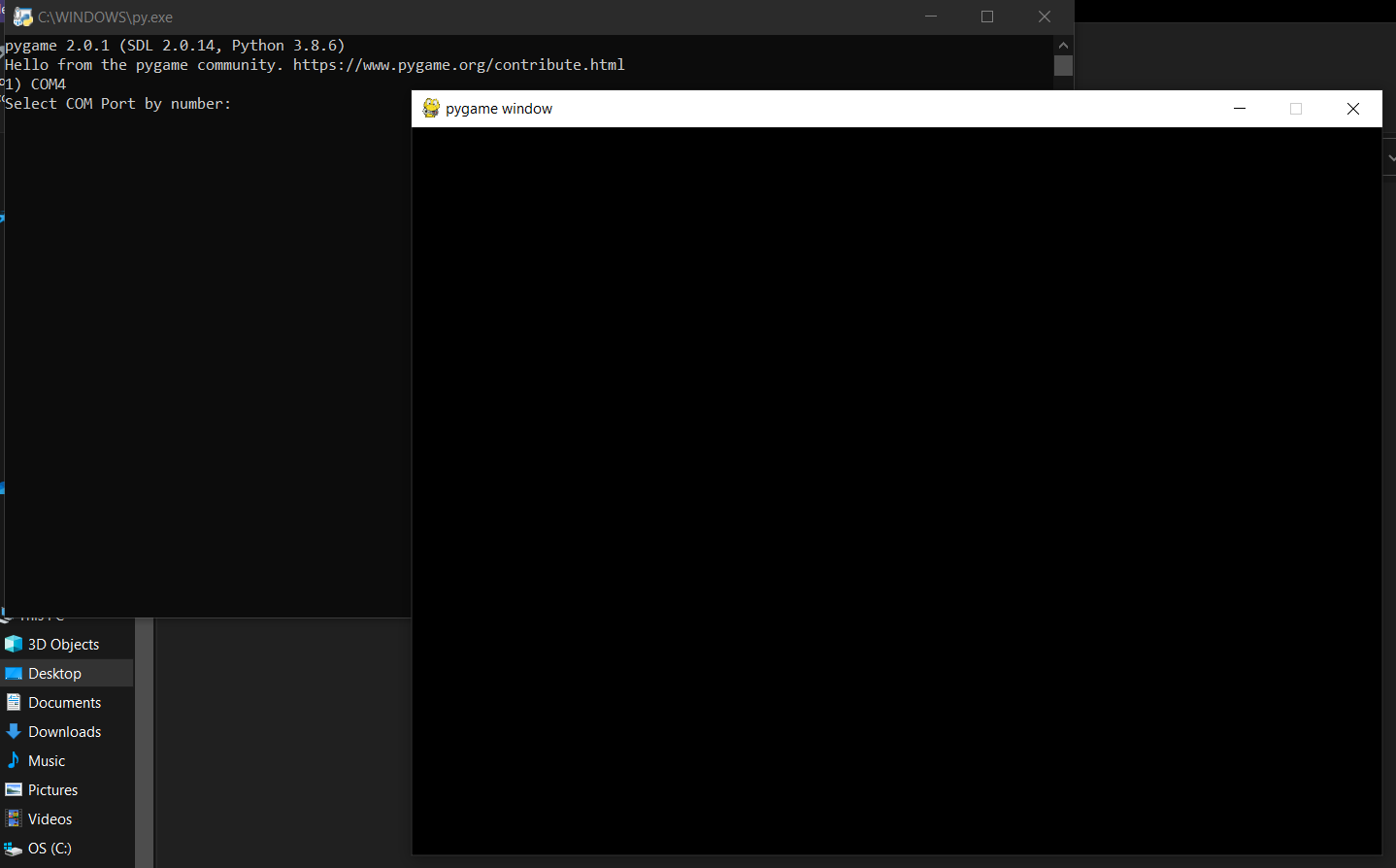
Diagram

Description automatically generated

Figure 11 GUI Flow Chart

FLOW AND USAGE:

The programming and logic in this program is strong that updating gestures and expressions will create new buttons automatically without needing to do changes in the program. Start the program execution, it will open command line and black screen GUI.





Enter Port No.

Figure 12 GUI Setup

The command line will show list of serial COM Ports available, enter number corresponding to COM Port connected to Arduino board. If correct port is used, the GUI will show buttons for all the gestures and expressions in the library.

User can click on any of the buttons to show the gestures / expression. If gesture button is pressed, the script will start an audio corresponding to that button and will send motor movement commands serially to Arduino depending on run time of audio. After finishing the execution, the user can again select between any of the buttons. If user press the expression button, the script will send relative motor movement commands to Arduino to show that facial expression. To exit the program, just hit the usual ‘X’ to close and exit the application. There are four more buttons to move forward and backward with the list of buttons on display. As per the scrip, maximum of 10 gestures and 8 emotions are displayed on the screen. If there are more gestures and emotions in the library file than these value, these 4 buttons will help navigating the list to show next or previous set of gestures or emotions

Add Delete Gesture

The file addDelGestures.py helps to easily add or remove and even check list of gestures present in the library file.

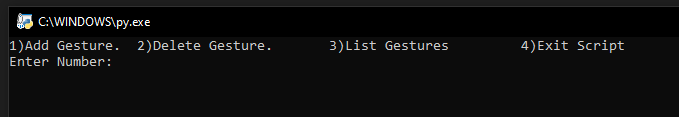
****

Figure 13 Add Gesture Menu

1st is menu screen. It gives 4 options to modify library, add gesture, delete gesture, list all gestures present in the library, and exit the script. Enter corresponding number to execute that function.

While adding gestures, first thing the script will ask is for gesture name and audio file name. Complete audio file name, with extension should be entered.

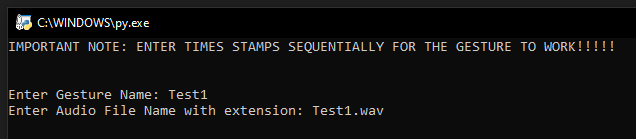


Figure 14 Add Gesture Details screen

Next screen, it will ask for at which timestamp, what should be the angle for each motor to show a posture. The values should be in range of 10-99. Refer the table mentioned in previous sections for how the percent react for each motor for facial expression. Important note is to enter time stamps sequentially.

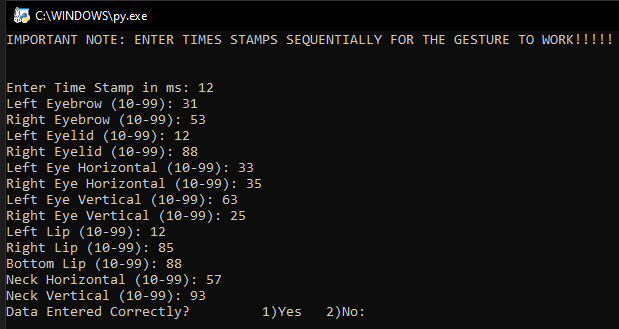


Figure 15 Add Gestures Motions

After the data for posture is entered, it asks for confirmation if entered data is correct, enter 1 to acknowledge correct entry otherwise no. After that, It will again ask if you wish to enter more timestamps and enter the value as per.

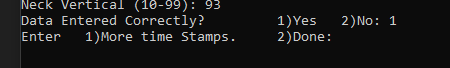


Figure 16 Add Postures Verification

If you want to enter more timestamps press 1 and it will again ask you timestamp and all corresponding posture values and confirmation. Do this for all timestamps you want to set postures for. Once done, enter 2 to confirm all entries. It will take you to new screen where it will show you all the timestamp values and corresponding posture values you have entered. It asks for the last time if all the values are correct. If you wish to write the gesture to the library enter 1 otherwise if you find anything wrong enter 2 and you can start over again.

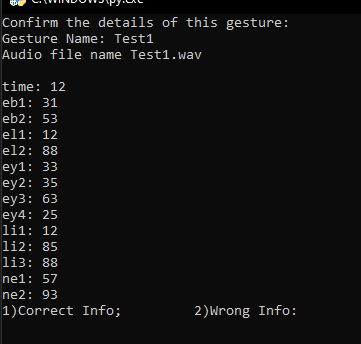


Figure 17 Add Gesture Verification

To delete any gestures existing in the library enter 2 from menu screen. It will show all the list of gestures present in the library file. Enter the number of the gesture you wish to delete. If you do not wish to delete anything, just enter 0.

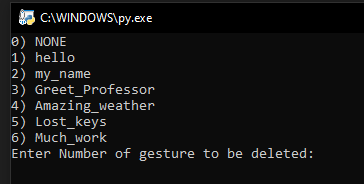


Figure 18 List of gestures present in library

After you are done modifying the library file, you can either enter 4 from menu screen or hit X button on top right corner of the window.

To be able to run the gesture, it is important to put the corresponding audio file with specified name into the AudioFiles folder. Otherwise, the gesture button wont work and it will just keep crashing.

EXAMPLE TO ADD A NEW GESTURE:

Step 1) Record an audio in preferably wave, mp3 or m4a format. You can use any of the recording softwares available or provided by the operating system.

Step 2) Place this audio file in AudioFiles folder.

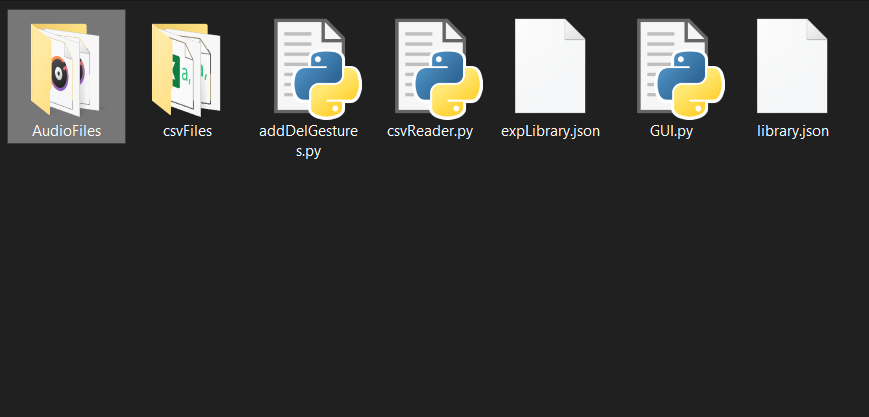


Figure 19 Add Gesture Step 1

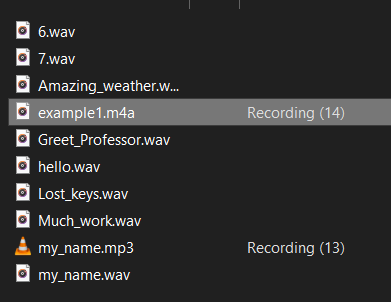


Figure 20 Add Gesture Step 1.1

Step 3) Run “addDelGestures.py” script.

Step 4) Enter 1 to add new gesture.

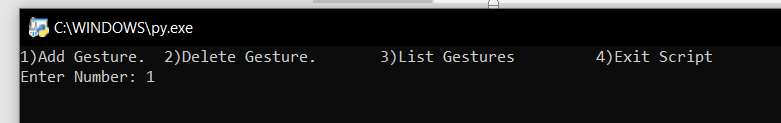


Figure 21 Add Gesture Step 4

Step 5) It will ask for gesture name, name anything you want. This name will be later used in gui as button name.

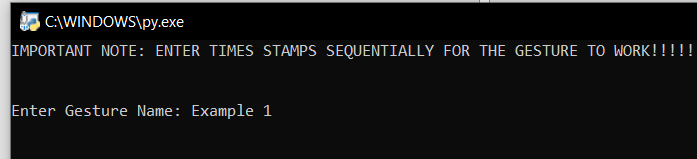


Figure 22 Add Gesture Step 5

Step 6) Enter the name of audio file copied in the audioFiles folder with proper extension.

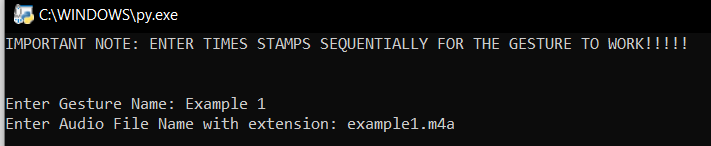


Figure 23 Add Gesture Step 6

Step 7) It will ask for time stamp you want to set the posture for. IMPORTANT: ENTER ALL THE TIMESTAMPS SEQUENTIALLY. The timestamp values should be in ms values.

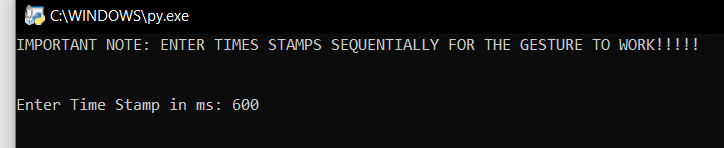


Figure 24 Add Gesture Step 7

Step 8) After you enter the timestamp, it will ask you for values of each moving part one by one. The values should be written between 10-99. Refer to the tables above know how these percent do value move the motors on the robot head.

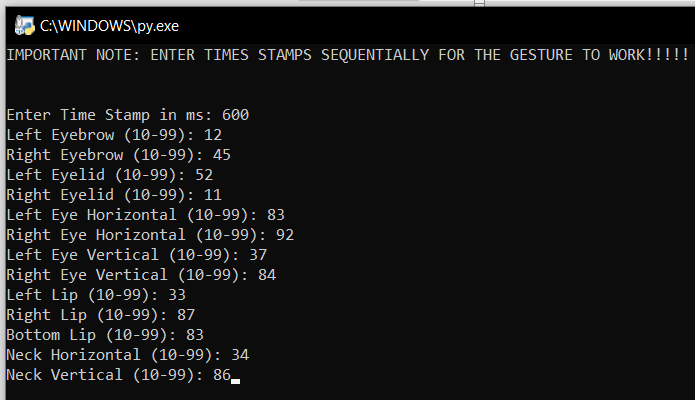


Figure 25 Add Gesture Step 8

Step 9) After you have entered these values, the script will ask for confirmation whether the entered data is correct or not. If correctly entered, press 1 otherwise, 2. If you enter 1, the values will be stored in temporary variable and if you enter 2, the data will be erased.

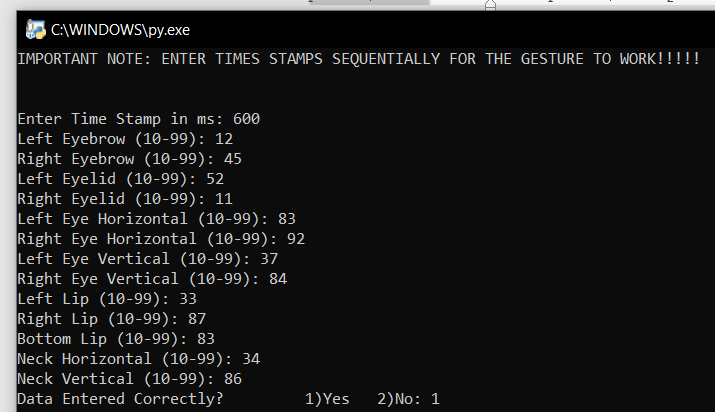


Figure 26 Add Gesture Step 9

Step 10) It will ask if you want to enter more postures for different timestamps. To do so enter 1 otherwise enter 2. If you enter 1, it will repeat steps 7, 8 and 9 until you are done.



Figure 27 Add Gesture Step 10

Step 11) After entering all the required posture values for different required timestamps, the script will show all the entered timestamps and its corresponding values. You need to confirm these values to save it to the library file. Enter 1 to confirm and write the gesture to the library file. If you wish not to store the date you can enter 2 and the data will be erased and wont be saved to the library.

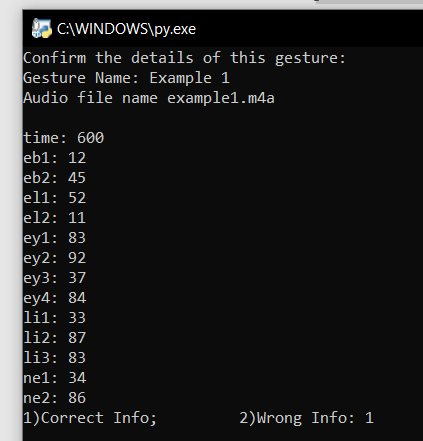


Figure 28 Add Gesture Step 11

Step 12) After successfully storing the data to the library, the user will be notified, and menu screen will be showed again. Repeat these steps to add more new gestures.

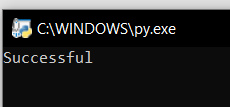


Figure 29 Add Gesture Step 12.1

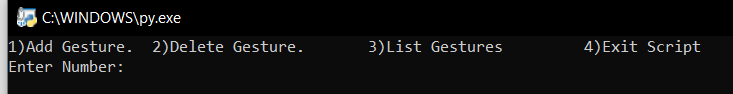


Figure 30 Add Gesture Step 12.2

Step 13) To check if the gesture is written to the library file you can enter 3 to check the list or you can even open GUI.py file to check if button is correctly generated. As you can see in the window, the Example 1 gesture has been added to library and a button has also been created automatically.

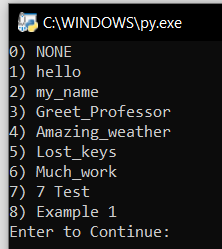


Figure 31 Add Gesture Step 13



Figure 32 GUI after Gesture is added

Adding emotions to library manual

This project has another library which contains set of emotions to show on fritz robot head. These emotions do not require audio or time series of postures. It is a single posture and therefore no timestamps needed. It is simple to add an emotion to the library. The figure 10 shows format of emotion library. Simply add a comma before the last emotion parentheses and past the same format as the previous emotions. Change the key name to whatever you want to call your gesture and enter values from 10-99 that are needed to be set to each motor to show the desired expression. Refer tables in previous sections to check the corresponding motor motion change for percentages.

Csv file working and usage

csvReader.py file helps to interface project files provided from Richard Romano to this project to replicate postures shown in Richards project video. Richard’s project provides two files, one audio file and csv file. Place the audio file in audioFiles folder and csv file in csvFiles folder. After placing the files correctly in the folder, run the csvReader.py script. First, it will ask for name of csv file.

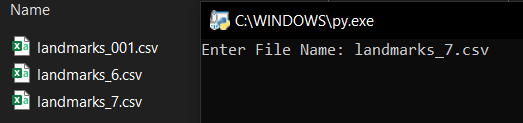


Figure 33 Add Gesture from CSV 1

Enter the file name correctly with extension. Next enter the corresponding audio file name with extension.

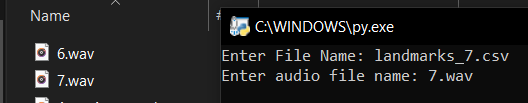


Figure 34 Add Gesture from CSV 2

Lastly, it will ask for button name you want to give that will be displayed in the gui button. After you have entered the name, it will convert csv files coordinates to posture motion and update the library file.

The working of this file might not be accurate, but it can be modified by future developers. The csv files contain timestamp and as sequence of x y coordinates.

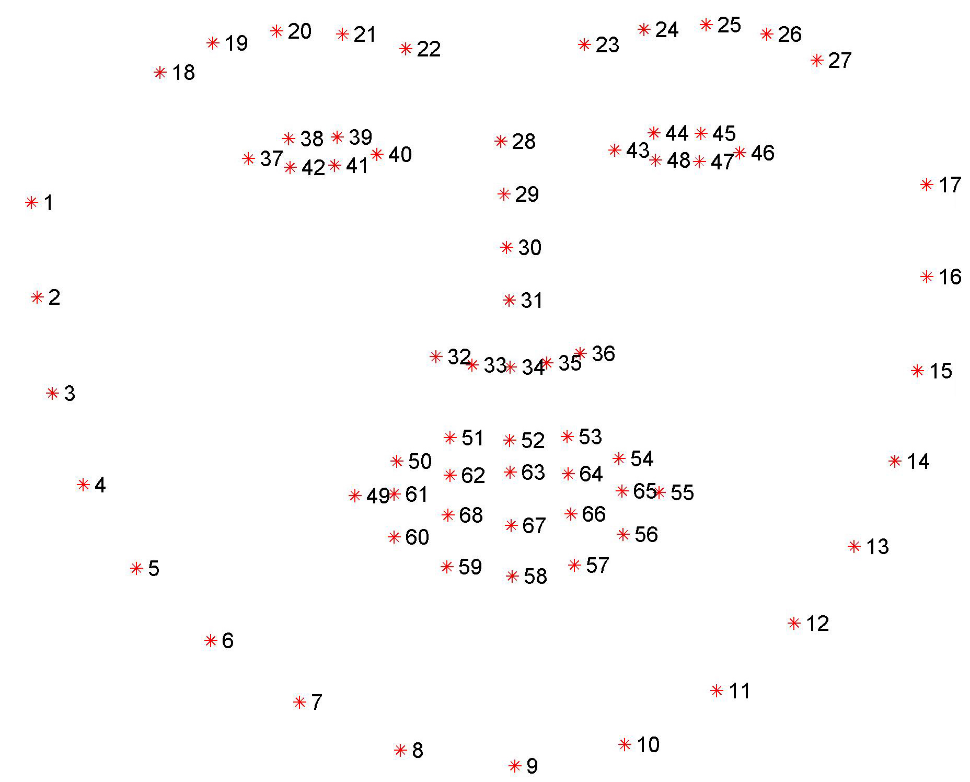


Figure 35 Landmark Locations

The image shows landmarks for each point on face. For the conversion, the script takes point 20 for left eyebrow, 25 for right eyebrow, 38 and 41 for left eye lid, 45 and 48 for right eye lid and for lips, 63 and 49 for left lip, 63 and 55 for right lip, and 63 and 58 for bottom lip. The script takes the 1st time stamp posture as normal values and calibrates all other time stamp posture according to this 1st timestamp.

Additional documentation- Software from other sources description

Visual Studio Code

Visual Studio Code is source-code editor made by Microsoft for windows, Linux and macOS. It helps in editing, debugging and syntax highlighting. It also features git commands for easy use.

It has many modules for python, c, c++ and even Arduino programming to suggest quick syntax, variables auto fill.

OBS

This software is simply used to be able to show video camera on bigger screen during the video lectures. It has no other purpose in this project. It works similar to zoom screen share only difference being creates another sharable window which contains required windows, screens, video cameras or text at required position.

The only reason to use this software was to show the video camera feedback on a bigger screen for demo purpose.

Hardware Troubleshooting

Power Connection Problem

There is a power connection problem that connects the power adapter and Arduino motor shield. This might be due to improper contact with wire and motor shield. A suggestion would be replacing the power adapter or Arduino motor shield. Checking if there are any loose soldered power contacts, fixing it might also solve the problem.

**Motor and moving part connection problem:**

If any DOF moves to extreme position, there is a possibility for the parts to break. If it breaks, it is difficult to replace these parts as casing parts protecting these parts are joined with hot glue. Therefore, it is best to rotate the motors within safe limits. As of now, left eye vertical motion connector is disconnected. And its contact is difficult to access and therefore I was not able to rejoin the motor and eye part.

Software troubleshooting

Gui Arduino disconnect while running.

Due to power connection problem, the Arduino board resets and loses serial communication with PC. This also causes the software program to loose control over serial communication with the Arduino board. To reconnect the software to Arduino board, it is necessary to restart the python software.

Gui crash due to library improper format.

The GUI might crash when a button is pressed. If this occurs, there is high chance that contents of library file are not in proper format. Remember that library file format is in json. Any excess ‘,’ can cause exception to convert the library to python dictionary and crash the software. To fix this, it is required to go through raw library file contains and find improper syntax. The expression library is well maintained and therefore, it can be easy to find for errors. The gesture library is not well maintained and therefore, it is suggested to only use addDeleteGesture.py script to modify gesture library. This will avoid this problem.

Conclusion

It was a challenging project to write different scripts for ease of the user. There were many things to learn from the project like serial communication transfer and reading serial data without losing any data and avoiding delays for data capture. It was also a challenging task to interface this project and Richard Romano project. The calibration of coordinates contained in csv file to replicate postures on the fritz head might not be perfect but still does a considerable good job. With the progress I made and the scripts I wrote for this project have a considerable strong logic so that new users can use them easily.