University of Ottawa

Faculty of Engineering

GNG2101 Introduction to product Development and management for Engineers

Project Deliverable F

Submitted by:

Talk Box. C01, Team C13

Tia El Masry, 300160596

Zainab Badawi, 300034146

Kain Mozafarian, 300138481

March 7, 2021

University of Ottawa

Table of Contents

List of Figures

List of Tables

1 Introduction

In this deliverable, group C13's goals are to demonstrate their understanding and showcase their knowledge by creating a second iteration of prototypes, with regards to the client feedback. At this stage of the project, the group is orienting themselves around doing software prototypes for the most part, whilst refining older prototypes such as readjusting the shape of the input method, using a different screen size display, and using a Raspberry Pi for our microcontroller instead of an Arduino. In this stage of the project, we are focused on prototyping and testing out methods regarding the program that will be running on the microcontroller, it will support phrases being displayed, text to speech, and images that are in accordance with their respective phrases. A method to connect the microcontroller to another device node, which is able to update data on the microcontroller wirelessly is another scope of our prototyping procedure.

2 Client Feedback

Interpret client comments and feedback into meaningful criteria to be used to make or assist in making prototype 2. Aspects that are liked or are given counterexamples are clearly extracted.

2.1 Client Feedback Keynotes:

- XLR cable to power the the arduino
- the wheelchair runs on 24 volts
- The analog stick should move left and right horizontally, one direction for navigation and the other for selection
- Use a “gooseneck” design for the support of the display
- The Google Assistant is not as necessary as the main text to speech functionality, probably shouldn’t be pursued any further
- Use a larger screen for the user display
- The software is adaptable to smaller screen sizes for displaying the python app (smaller compared to tablet sized screens)
- Limited space on a wheelchair, so a different input interface needs to be thought of
- Reduce the length of the joystick mold to reduce stress on the joystick circuit itself (about ~5 cm)
- Create a panelling system/ include tabs where phrases are shown in the most user-friendly fashion possible

2.2 Feedback Analysis:

Extracts interesting and useful insight from feedback while identifying how this information can be used to improve the solution.

The main key takeaways from the client meeting were in regard to the software and the input method of the overall system. Whilst demonstrating the physical layout of the Talkbox, the client showed concern regarding screen size, screen mounting mechanism, and space pollution

caused by multiple devices being present. A further explanation was given by us regarding some aspects of the hardware; first of, we displayed that the monitor that will be used will be different than the one demonstrated in the first prototype, that the original monitor was used in accordance to using an Arduino for our microcontroller, since now it has been decided that further prototyping will be done in regards with a Raspberry Pi. A button and a joystick turned out to be a hectic solution according to the client, so the suggested workaround is to have only the joystick which enacts as a bidirectional input response. Furthermore, the client had suggested a Gooseneck mount for our display in order to allow free movement of the display with some rigidity in mind. As for the software part of the project, we showcased our progression from in-line command interface from prototype 1, to a work in progress version in prototype 2 where the program reads a text file and showcases phrases in a button that's displayed on a GUI. The program can go backwards and forwards in terms of what category of phrases it is wanted by the user. Furthermore, the array of phrases is updated in the program without needing to terminate it after updating the text file is edited. A few concerns regarding the software from the client was how the program’s GUI would fit on a small screen and how would multiple phrases fit on one panel. These are issues that we will tackle in the next few weeks.

3 Prototype 2 Diagrams/Pictures

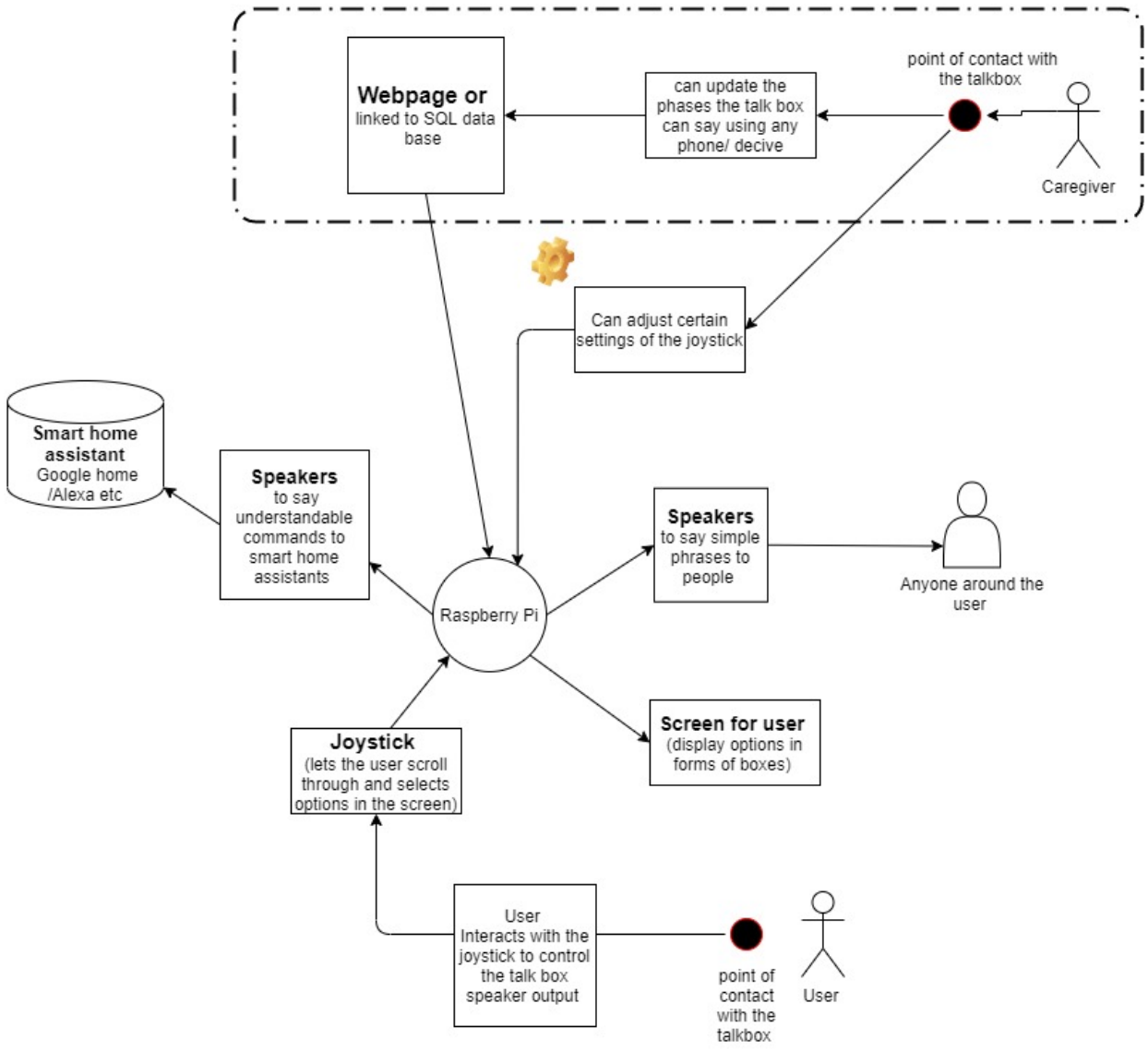
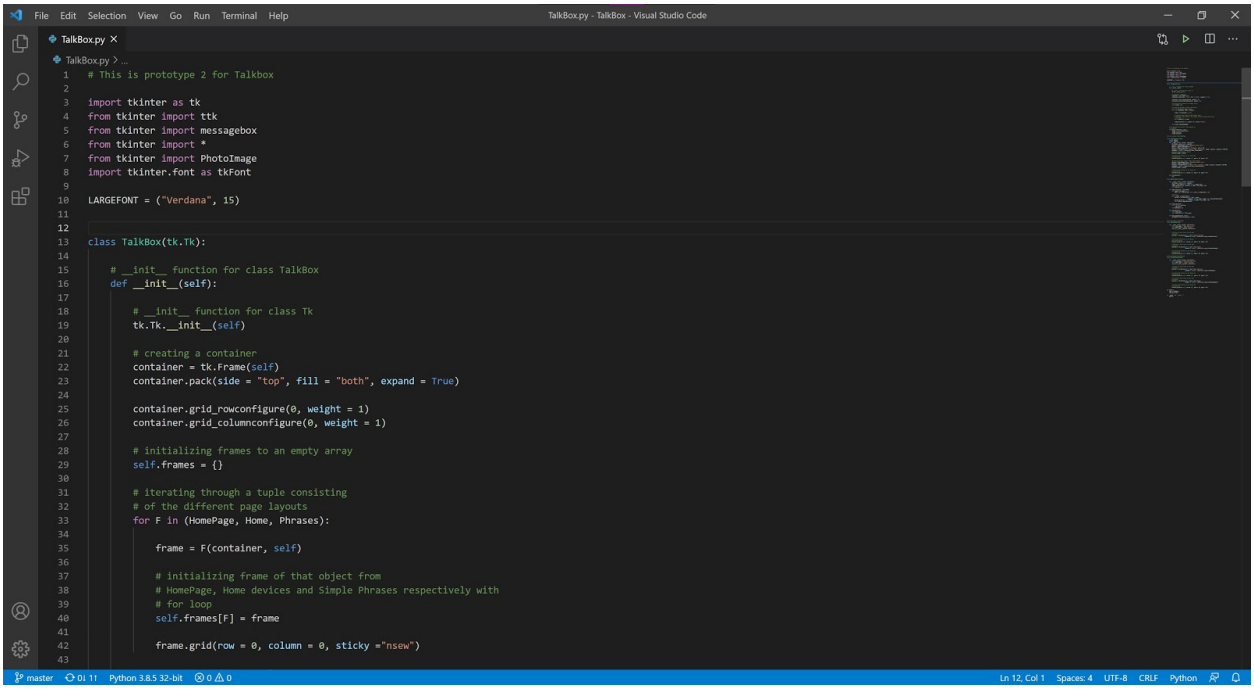


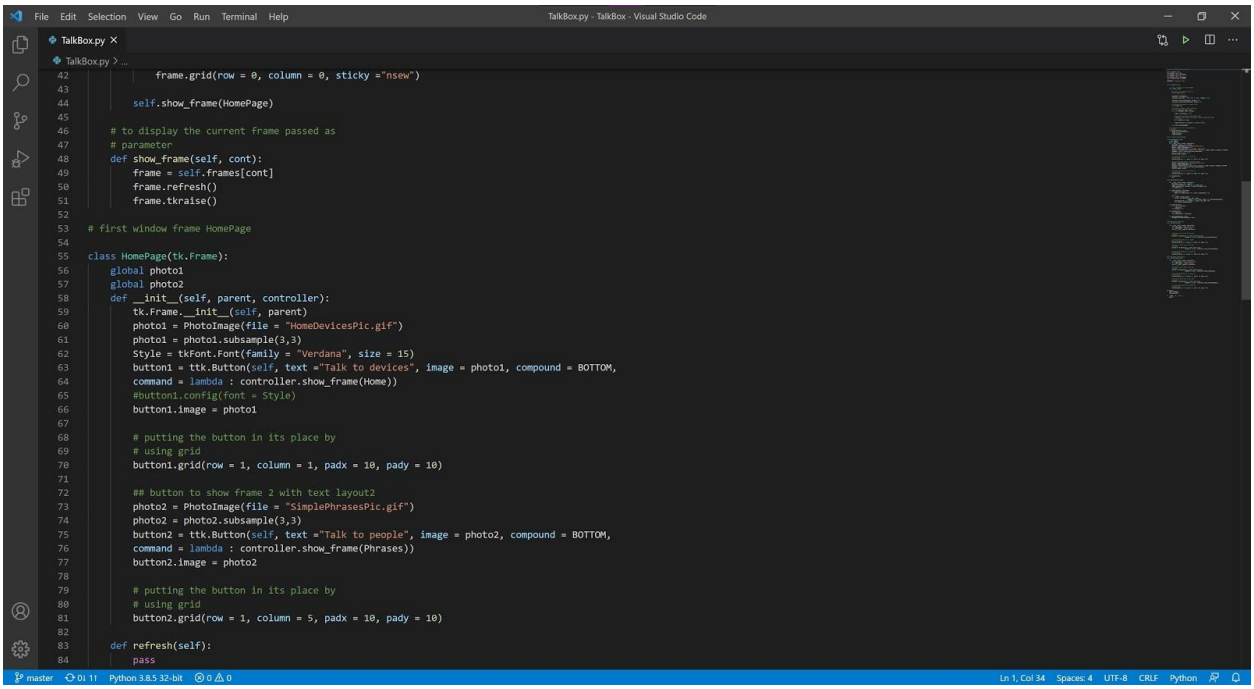
Figure 1: Talk Box Functionality Breakdown

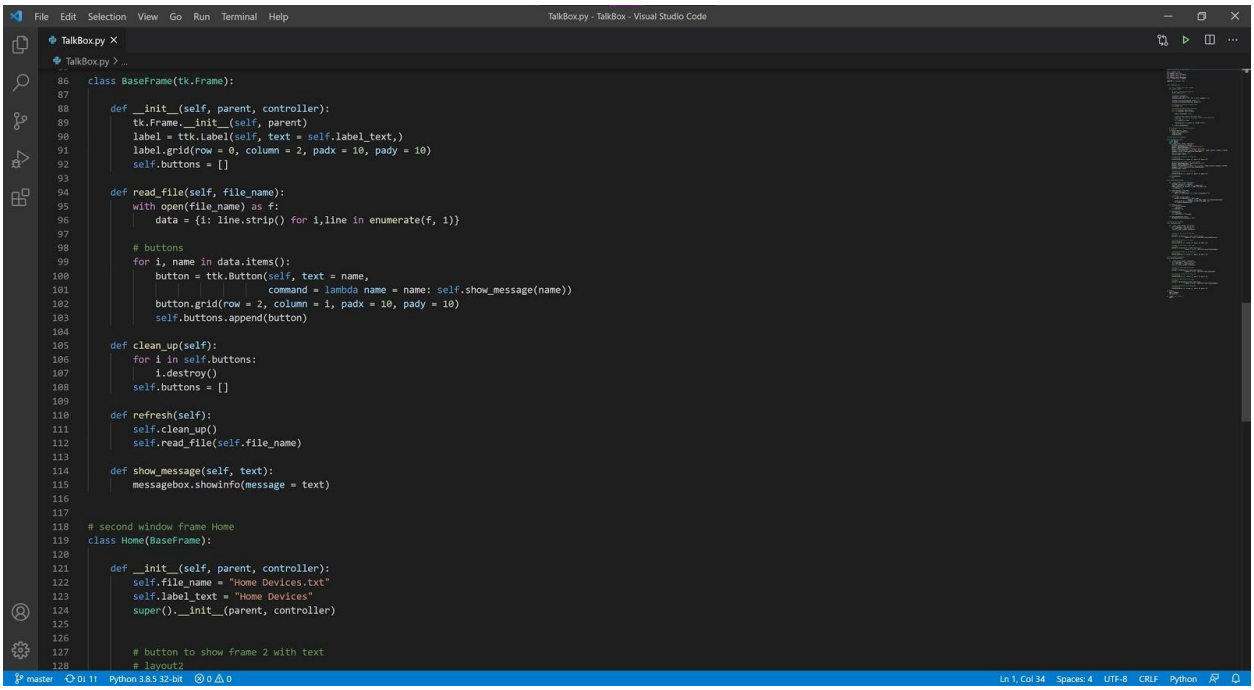
This Prototype objective is to establish functionalities that allow the raspberry Pi to output sound, and display options for the user to pick between speaking with devices or people (shown in **Figure 1**: user main display panel). In addition is to allow the user to navigate between options by moving the joystick in the x-axis then selecting the option by pushing on the joystick downwards. Finally, caregiver is able to add phrases to the sql data phase through a webpage that is hosted in one of the group members existing webpage server. (**Figure**: Talk Box Functionality Breakdown) demonstrate a detailed breakdown of the talk box functionalities.

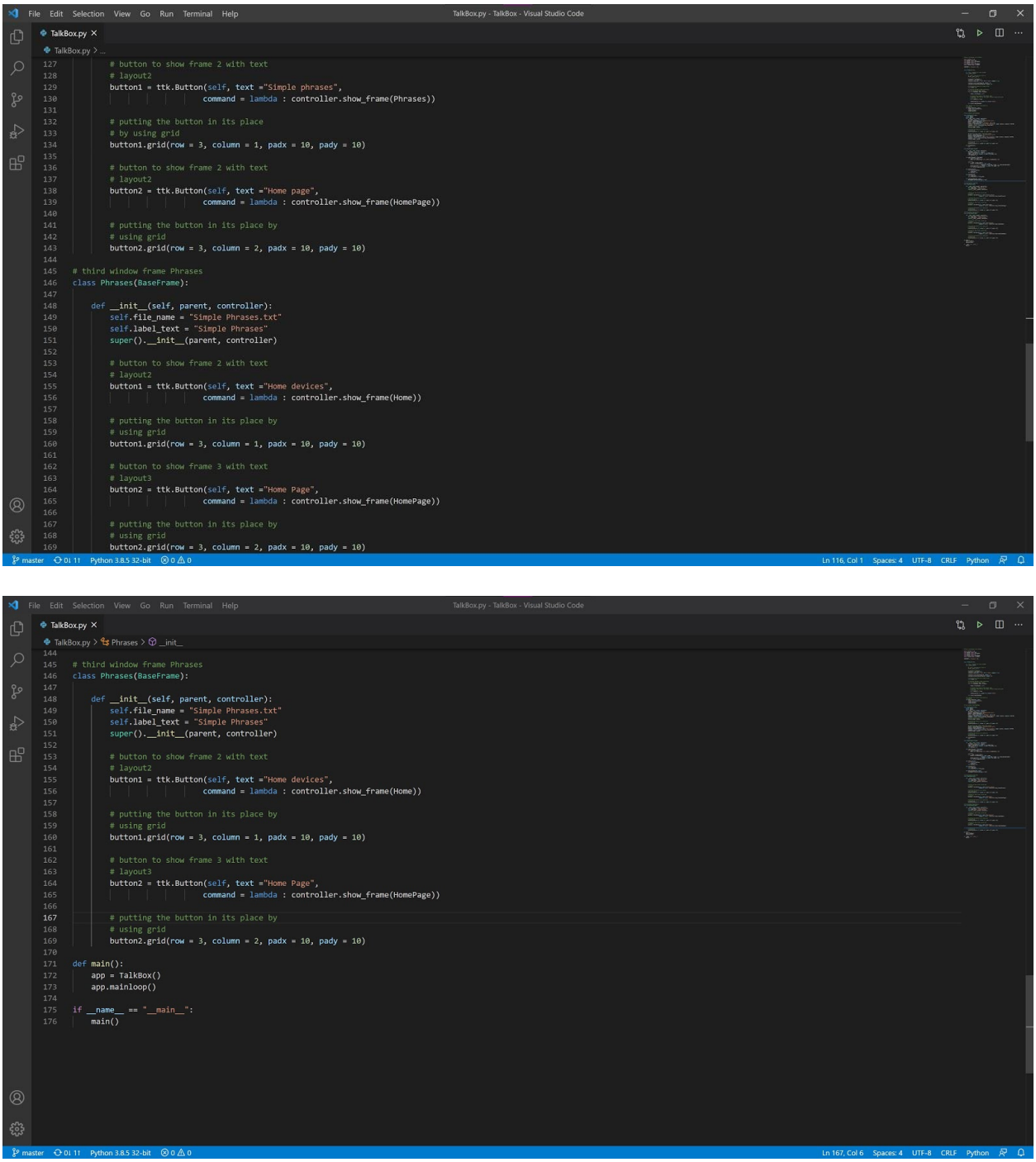
3.1 Software:

In terms of the software prototype, we have created a python script that creates a graphical user interface with the Tkinter package. We have created 5 different classes that interact with each other: the TalkBox class, the HomePage class, the BaseFrame class, the Home class and the Phrases class. The TalkBox class acts as the initializer for the frame of the entire program which creates the frames for every page created in the class. The HomePage class contains two widget buttons that have photo images, where their commands are to go to their respective pages: “Talk to devices” which opens the Home class and “Talk to people” which opens the Phrases class. The most important class is the BaseFrame class which acts as a parent to both the Home class and the Phrases class since it contains all the common functions between both classes. This class contains the function that reads the text files and creates widget buttons from all the entries from those files. The commands for those buttons are currently creating a message show box that returns the specific sentence selected. For the future, we are planning to replace those current commands with code to send to the text-to-speech API which will voice those sentences. In our BaseFrame, we also have a refresh function which allows the pages to get automatically updated with buttons when the text files are updated, without having to kill the program.




File Edit Selection View Go Run Terminal Help
TalkBox.py - TalkBox - Visual Studio Code
TalkBox.py
TalkBox.py > ...
frame.grid(row = 0, column = 0, sticky ="nsew")
self.show_frame(HomePage)
# to display the current frame passed as
# parameter
def show_frame(self, cont):
frame = self.frames[cont]
frame.refresh()
frame.tkraise()
# first window frame HomePage
class HomePage(tk.Frame):
global photo1
global photo2
def __init__(self, parent, controller):
tk.Frame.__init__(self, parent)
photo1 = PhotoImage(file = "HomeDevicesPic.gif")
photo1 = photo1.subsample(3,3)
Style = tkFont.Font(family = "Verdana", size = 15)
button1 = ttk.Button(self, text ="Talk to devices", image = photo1, compound = BOTTOM,
command = lambda : controller.show_frame(Home))
#button1.config(font = Style)
button1.image = photo1
# putting the button in its place by
# using grid
button1.grid(row = 1, column = 1, padx = 10, pady = 10)
button to show frame 2 with text layout2
photo2 = PhotoImage(file = "SimplePhrasesPic.gif")
photo2 = photo2.subsample(3,3)
button2 = ttk.Button(self, text ="Talk to people", image = photo2, compound = BOTTOM,
command = lambda : controller.show_frame(Phrases))
button2.image = photo2
# putting the button in its place by
# using grid
button2.grid(row = 1, column = 5, padx = 10, pady = 10)
def refresh(self):
pass
master 0↓ 1↑ Python 3.8.5 32-bit 0 0
Ln 1, Col 34 Spaces: 4 UTF-8 CRLF Python


File Edit Selection View Go Run Terminal Help
TalkBox.py - TalkBox - Visual Studio Code
TalkBox.py
TalkBox.py > ...
class BaseFrame(tk.Frame):
def __init__(self, parent, controller):
tk.Frame.__init__(self, parent)
label = ttk.Label(self, text = self.label_text,)
label.grid(row = 0, column = 2, padx = 10, pady = 10)
self.buttons = []
def read_file(self, file_name):
with open(file_name) as f:
data = {i: line.strip() for i,line in enumerate(f, 1)}
# buttons
for i, name in data.items():
button = ttk.Button(self, text = name,
command = lambda name = name: self.show_message(name))
button.grid(row = 2, column = i, padx = 10, pady = 10)
self.buttons.append(button)
def clean_up(self):
for i in self.buttons:
i.destroy()
self.buttons = []
def refresh(self):
self.clean_up()
self.read_file(self.file_name)
def show_message(self, text):
messagebox.showinfo(message = text)
# second window frame Home
class Home(BaseFrame):
def __init__(self, parent, controller):
self.file_name = "Home Devices.txt"
self.label_text = "Home Devices"
super().__init__(parent, controller)
# button to show frame 2 with text
# layout2
master 0↓ 1↑ Python 3.8.5 32-bit 0 0
Ln 1, Col 34 Spaces: 4 UTF-8 CRLF Python



```
        # button to show frame 2 with text
        # layout2
        button1 = ttk.Button(self, text ="Simple phrases",
                             command = lambda : controller.show_frame(Phrases))

        # putting the button in its place
        # by using grid
        button1.grid(row = 3, column = 1, padx = 10, pady = 10)

        # button to show frame 2 with text
        # layout2
        button2 = ttk.Button(self, text ="Home page",
                             command = lambda : controller.show_frame(HomePage))

        # putting the button in its place by
        # using grid
        button2.grid(row = 3, column = 2, padx = 10, pady = 10)

# third window frame Phrases
class Phrases(BaseFrame):

    def __init__(self, parent, controller):
        self.file_name = "Simple Phrases.txt"
        self.label_text = "Simple Phrases"
        super().__init__(parent, controller)

        # button to show frame 2 with text
        # layout2
        button1 = ttk.Button(self, text ="Home devices",
                             command = lambda : controller.show_frame(Home))

        # putting the button in its place by
        # using grid
        button1.grid(row = 3, column = 1, padx = 10, pady = 10)

        # button to show frame 3 with text
        # layout3
        button2 = ttk.Button(self, text ="Home Page",
                             command = lambda : controller.show_frame(HomePage))

        # putting the button in its place by
        # using grid
        button2.grid(row = 3, column = 2, padx = 10, pady = 10)
```



```
# third window frame Phrases
class Phrases(BaseFrame):

    def __init__(self, parent, controller):
        self.file_name = "Simple Phrases.txt"
        self.label_text = "Simple Phrases"
        super().__init__(parent, controller)

        # button to show frame 2 with text
        # layout2
        button1 = ttk.Button(self, text ="Home devices",
                             command = lambda : controller.show_frame(Home))

        # putting the button in its place by
        # using grid
        button1.grid(row = 3, column = 1, padx = 10, pady = 10)

        # button to show frame 3 with text
        # layout3
        button2 = ttk.Button(self, text ="Home Page",
                             command = lambda : controller.show_frame(HomePage))

        # putting the button in its place by
        # using grid
        button2.grid(row = 3, column = 2, padx = 10, pady = 10)

def main():
    app = TalkBox()
    app.mainloop()

if __name__ == "__main__":
    main()
```

Figure 2: Python script that creates the GUI using Tkinter

3.2 Hardware:

3.2.1 Initialization

For the initialization process, the Raspberry Pi Imager was used to flash the SD card with the “Raspberry Pi OS (32-bit)”. Following that, the Pi was connected to a monitor using a HDMI cable and the country, language, time zone, password, and wifi were set up.

3.2.2 MYSQL Database

The team used MYSQL to set up a database on the Raspberry Pi. A tutorial made by PiMyLife was used to complete the database setup. The tutorial guided through the process of installing “mysql_secure_installation”, followed by creating the database user with the following commands: “sudo mysql -u root -p” , CREATE “DATABASE TalkBox;” , “CREATE USER zainab@localhost IDENTIFIED BY 'pimylifeup';” and all the privileges were granted to the zainab@localhos user.

3.2.3 PHPMyAdmin

PHPMyAdmin was used to administer the MySQL database created, using a similar tutorial by PiMyLife. A summary of the steps done include: on the Raspberry terminal PHPMyAdmin “sudo apt install phpmyadmin”. Secondly, the apache was configured for PHPMyAdmin. Thirdly, NGINX was configured for PHPMyAdmin. The current prototype uses PHPMyAdmin database that is hosted on the local server. However, the team is working on hosting it on a GoDaddy server to allow users and carveger to edit the database from any device on any network.
The figure “PhpMyAdmin to login page” shows the initial page to access the database.

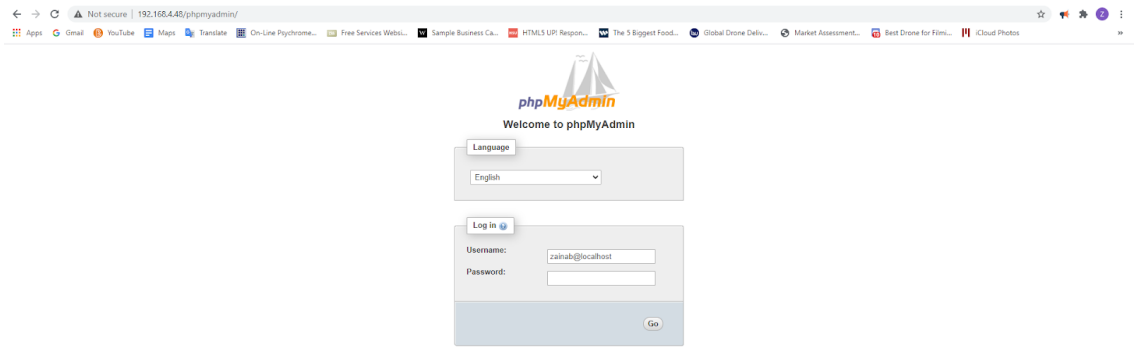


Figure 3: PhpMyAdmin to login page

Two tables were created, “HomeDevices” and “SimplePhrases” and were populated with phrases that are directed to either home assistant devices or people.

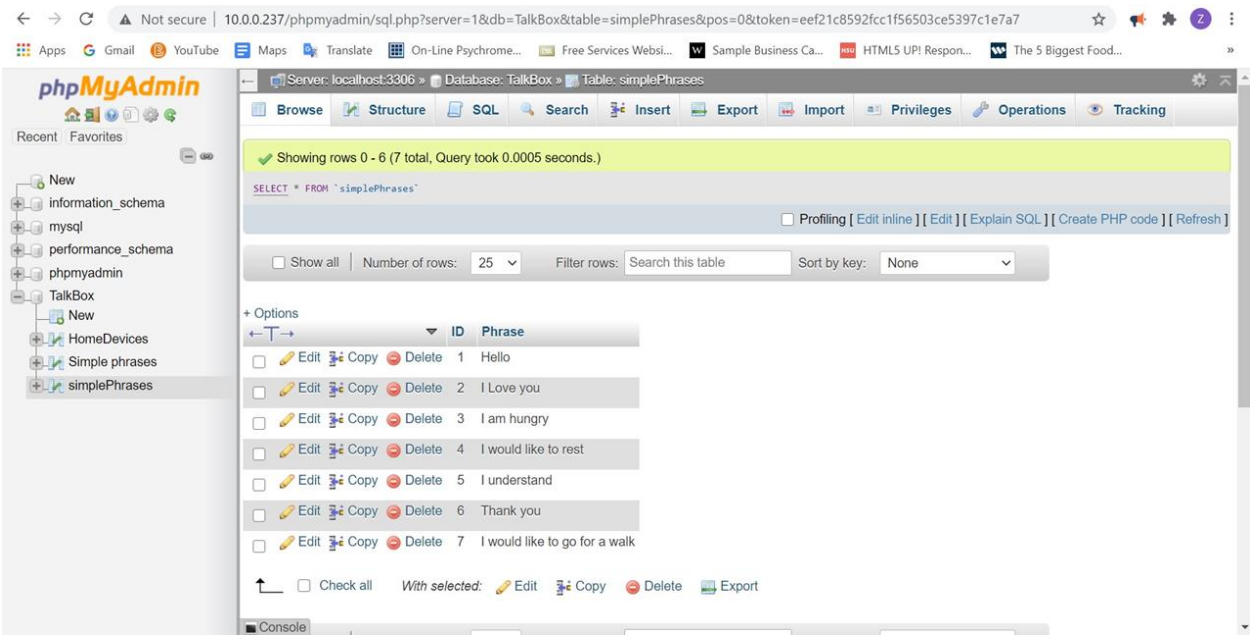


Figure 4: Editing the database

3.2.4 LCD Screen

A 3.5-inch LCD screen is used to display the phrase options to the user. The following five lines were typed in the terminal to set up the LCD screen:

- sudo rm -rf LCD-show
- git clone https://github.com/goodtft/LCD-show.git
- chmod -R 755 LCD-show
- cd LCD-show/
- sudo ./LCD35-show

The TalkBox github repository was cloned to the Raspberry Pi, and the talkbox.py python script was successfully executed. The talkbox.py script outputs the following panel, where the user can first choose if they want to output phrases that talk to people or talk to devices as shown in the **figure 5** "User main display panel". After the user selects one of the options, they will be directed to another panel that provides multiple buttons with phases in the database, as shown in the **figure 6** '"Home Devices" phrases displayed as buttons, ready to be voiced via speaker', and **figure 7** "Simple Phrases" word bank displayed as buttons', ready to be voiced via speaker. Currently the phrases shown are stored in a text file in the TalkBox github repository. But in the following weeks, the team will create a query that fetches data directly from the phpMyAdmin hosted on GoDaddy server.

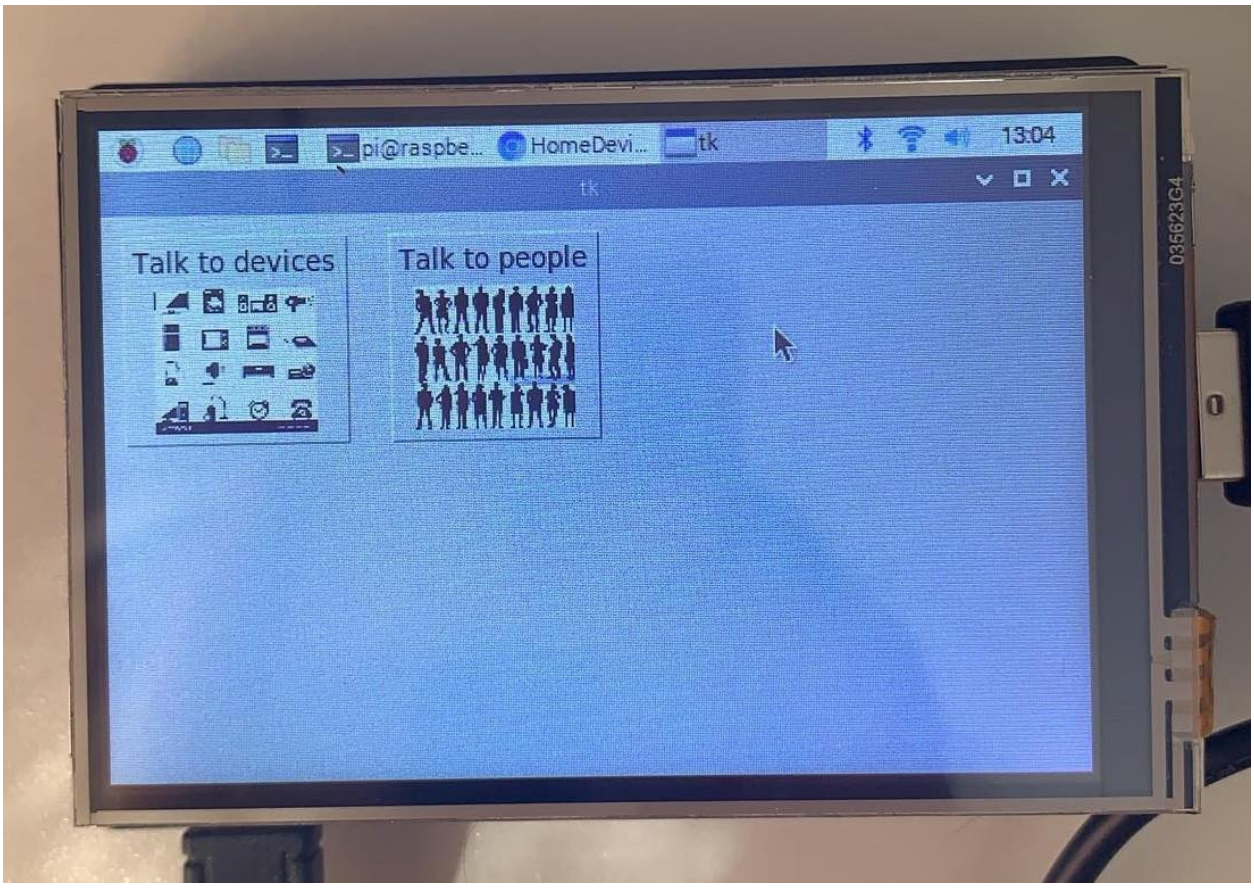


Figure 5: User main display panel

Figure 6: “Home Devices” phrases displayed as buttons, ready to be voiced via speaker

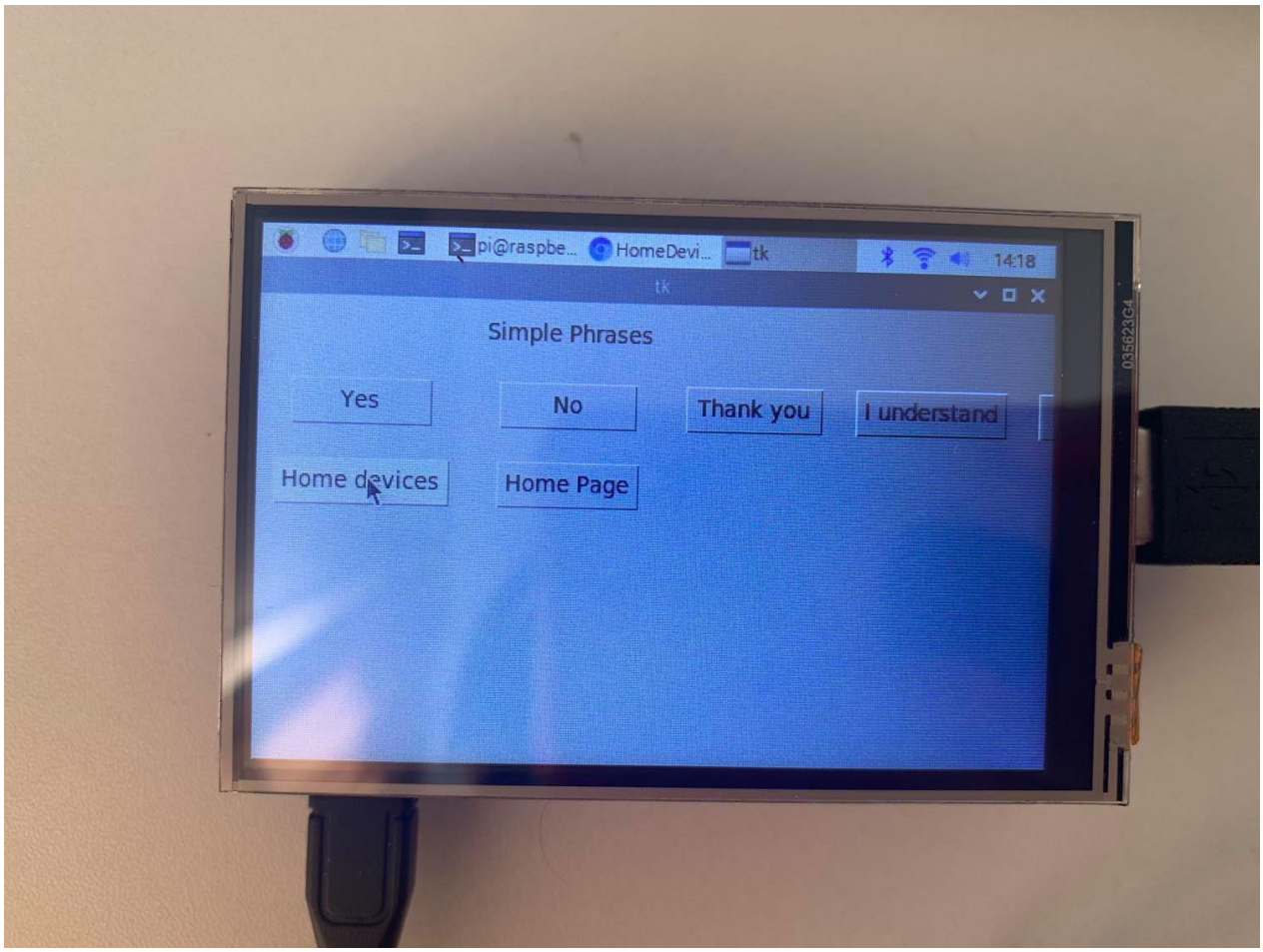


Figure 7: “Simple Phrases” word bank displayed as buttons, ready to be voiced via speaker

3.2.5 Button and Speaker

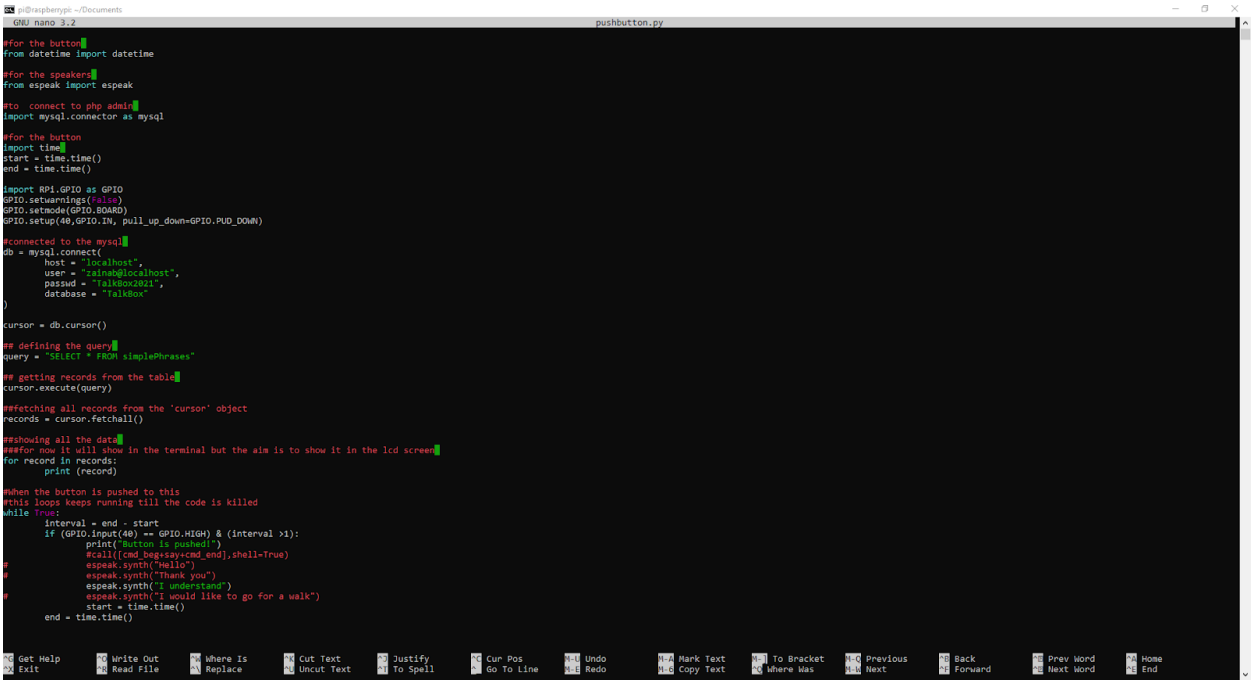


Figure 8: pushbutton.py script

The python script “pushbutton.py”, which is stored on the Raspberry Pi, connects MySQL and queries from the database and outputs the phrases to espeak. This script allows the team to test text-to-speak functionality. By pushing the button, the script sends data to the espeak.synth() method, and then the phrase is to be said out loud. (**Figure 9:** button SolidWord design) demonstrates the SolidWorks design to fit the arcade low force push button and mounting strap. (**Figure 10**: 3D printed button mounted) demonstrates the printed button and wiring soldered to the button to the arcade button. Finally (**Figure 11**: full assembly of the button) demonstrated all the parts of the button.

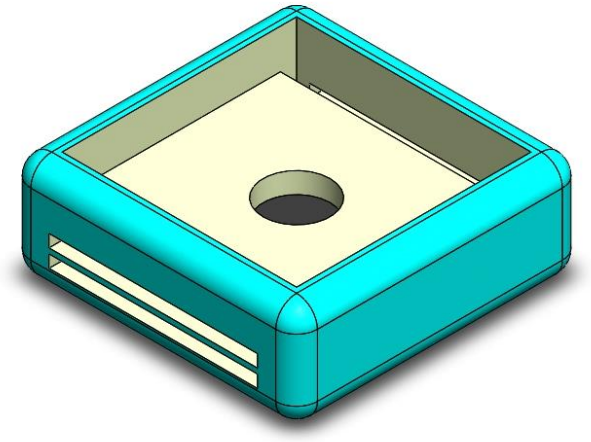


Figure 9: Button SolidWorks Design

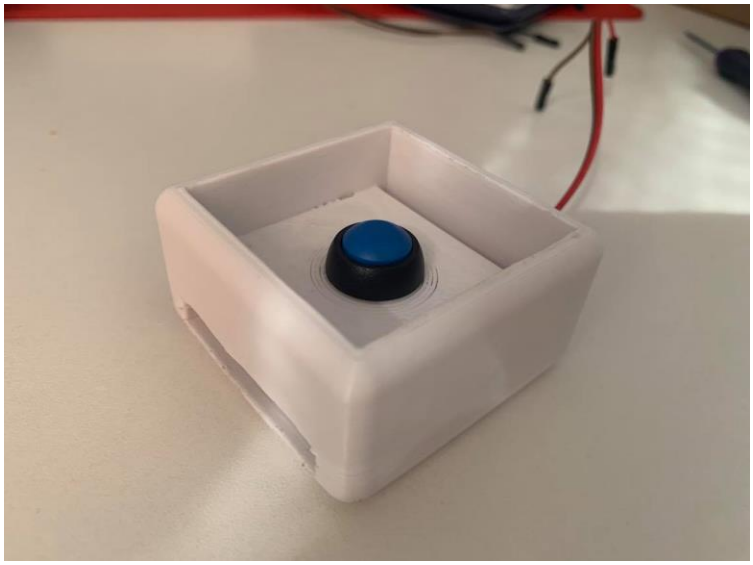


Figure 10: 3D printed and Button mounted

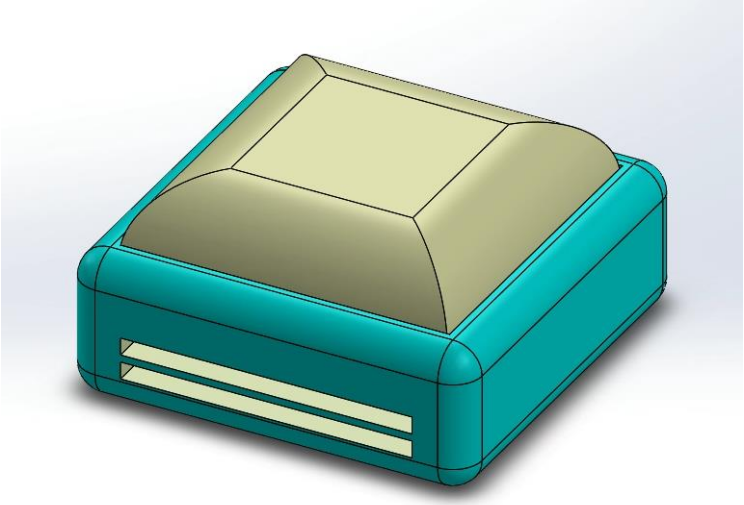


Figure 11: Full assembly of the button

3.2.6 joystick

To create a comfortable grip for the user, a joystick extension was created where the user has more contact area to control the device. A 3D model of the 3-axis joystick that was used in prototype 1, was downloaded from Grabcad and used to create the sketch that would then be used to extrude cut a loft feature of the joystick extension. A cross section of the joystick is shown in (Figure 12: Joystick extension SolidWorks cross section). The joystick extension solidworks files were 3D printed and fitted into the joystick as shown in (**Figure 13**: 3D printed joystick extension).

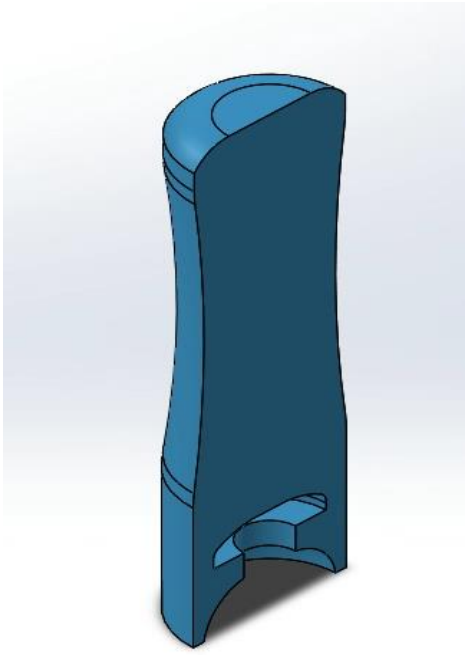


Figure 12: Joystick extension SolidWorks cross section

Figure 13: 3D printed joystick extension

4 Prototype Testing & Evaluation

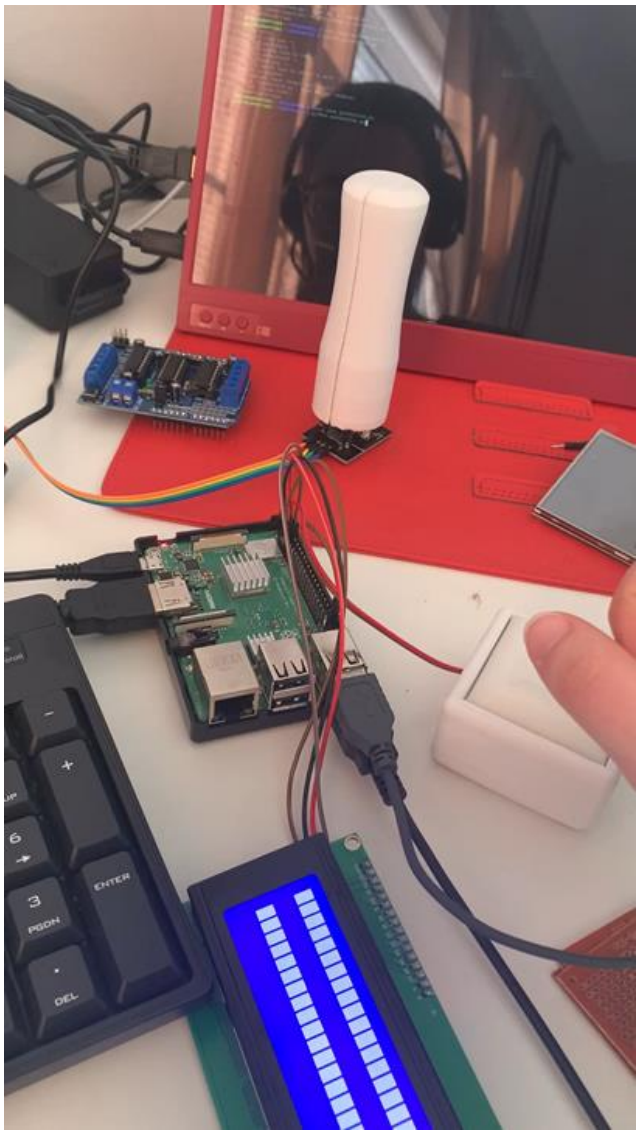


Figure 14: Hardware prototype parts

The test result of the hardware prototype is how the button triggers the talkbox, querying data from MySQL, and then sending commands to the speaker to output phrases. The next iterations of the prototype will use the joystick to scroll as well as select between options, which will eliminate the use of the button. We were able to test the button response time, which is relatively the same for the joystick. The sound quality produced by the speakers were adequate in our prototype testing, however in the next iteration the talkbox sounds will be tested to communicate with a smart home assistant and ensure that the talkbox can produce well enough

sound to awaken smart home assistants. By switching the phpMyAdmin to be hosted on a public server, it will allow the caregiver to access the database remotely which gives the device the ability to easily upgrade. Currently, the joystick and the screen are designed to be mounted on either arm of the wheelchair, however a gooseneck mount for the screen will be used to allow the user to comfortably view the screen from any angle. The weight of the device is the sum of the Raspberry Pi, the screen, and the joystick mount.

In terms of the software prototype, we were able to measure the user friendliness of our interface. We believe in terms of appearance and ease of use for the user, that our current prototype is still not up to par since the font is quite small, therefore it is hard to read. The frame of the GUI also changes sizes from switching between reading small and large text files since the buttons are created in a row, which is quite displeasing. The quality of images displayed on the screen of the user was also evaluated and they were clear and visually pleasing and added a nice effect to our home page. Although we could not obtain a hard value for the image quality, we were able to judge them visually. Unfortunately, due to the small screen size, we were not able to see all the buttons that were created, which is an issue that will be addressed in the future prototype by allowing only 2-3 buttons per page and adding the functionality of going between different pages which would encompass the different options of buttons available to the user.

Table 1: Target Specifications Comparison

#	Metric	Desired Value	Actual Value
1	Button response time	Fast Processing time	1s< - immediate
2	Capability of simple dialogue commands	Processing time	NT
3	Sound produced	>60 dB	NT

4	Connection to Home devices (TV, thermostat, Wi-Fi)	# >1	as many as phrases in the database
5	User friendliness	3	1
6	Quality of images displayed	146 PPI	NT
7	Force required to push button	>4 N	~5N
8	Travel Before Activation	cm	3cm
9	Size of button(s)	6.5 cm^2	4cm
10	Space between button(s)	cm	NA
11	Operating temperature	15-30 °C	10~30°C
12	Languages supported	#1-2	1
13	Emergency procedure	Yes	No
14	Capability of upgrade	Yes	Yes
15	Ability to mount on either side of wheelchair	Wire length	Yes
16	Device comfort	Comfort scale	3
17	Electric Consumption	2-5 W	NT
18	Maximum cost	>= $100	$120
19	Screen Size	38 cm	8.89cm
20	Weight	<1000 g	100g
21	Data Storage	16 GB	NT

Table 1: Actual vs Desired values for final target specifications

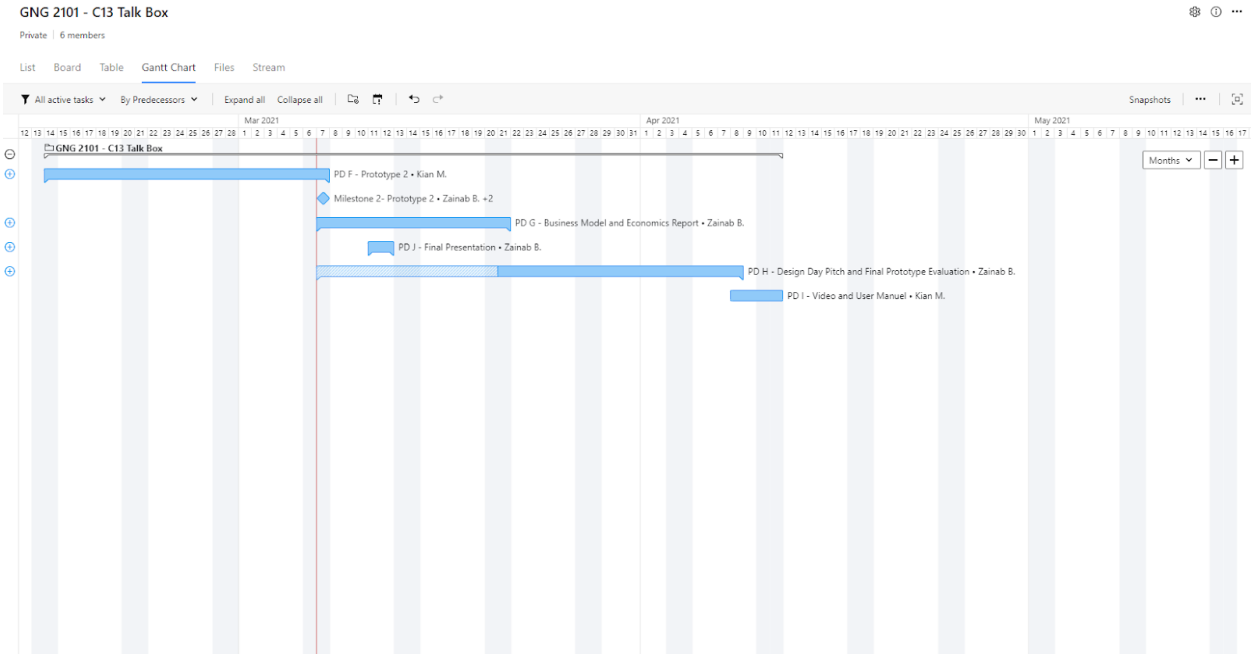
NT = Not tested in this current prototype

5 Conclusions and Recommendations for Future Work

We were successful in being able to implement a basic python software where it offers a simple GUI that has buttons and pictures to go along with it in order to navigate through the distinct categories of phrases. The Chassis for joystick, button, and the Raspberry Pi were reimagined and redesigned to coincide with the client needs and their feedback. There are many tasks that have begun and are work in progress regarding prototype development such as, setting up MySQL and a web page interface for the caretaker to update the python app's data, Test and adjust different fonts and sizes of text and pictures within the python app - which will be showcased on a panel that is also being worked on and is in development. A gooseneck mount is a very viable option which was recommended by the client and inheriting that alongside our mounting design for the microcontroller, will require us to rework our overall design for the wheelchair mount to an extent. Finally, the last aspect of the software prototype, which will advance the project design the most towards completion, is where we implement the selection system, where it doesn't rely on a mouse, but the joystick instead.

APPENDICES

APPENDIX I:



References

Projects.raspberrypi.org, projects.raspberrypi.org/en/projects/raspberry-pi-setting-up.

Emmet. “Setup a Raspberry Pi MYSQL Database.” Pi My Life Up, 9 Feb. 2021, pimylifeup.com/raspberry-pi-mysql/.

Emmet. “How to Install PHPMyAdmin for the Raspberry Pi.” *Pi My Life Up*, 11 Feb. 2021, pimylifeup.com/raspberry-pi-phpmyadmin/.

Hussain, Aamir. “Raspberry Pi 3 Complete Tutorial - Let's Get Started.” *Trick i Know*, 20 Apr. 2020, trickiknow.com/raspberry-pi-3-complete-tutorial-2018-lets-get-started/.