System Design for Mechatronics

Team #20, OpenASL
Robert Zhu zhul49
Zifan Meng mengz17
Jiahui Chen chenj194
Kelvin Huynh huynhk12
Runze Zhu zhur25
Mirza Nafi Hasan hasanm21

January 18, 2023

1 Revision History

Date	Version	Notes
January 18, 2023	1.0	Initial System Design Draft

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
Mechatronics	Explanation of program name
[... —SS]	[... —SS]

Contents

List of Tables

List of Figures

3 Introduction

The purpose of this document will give an overview of the system components for how the user interacts with the system, and the communication between the design of the hardware, software, and any electrical components.

4 Purpose

The purpose of our project is to create a device that will translate sign language gestures into their corresponding words or phrases. This will require the creation and development of a computer vision system alongside a machine learning model that will be used to recognize the hand motions, as well as a Raspberry Pi that will speak the word or phrase. The user will perform the sign language motion that will be captured by our computer vision system through a camera, and processed by our machine learning model and spoken through our Raspberry Pi.

5 Scope

OpenASL is primarily designed to assist the hearing impaired who use sign language to communicate. The following goals listed describe the key requirements for OpenASL to efficiently translate gestures and motions for any individual who does not know sign language to understand. More detailed explanations for each goal can be found in the SRS (REF SRS 2.2.1).

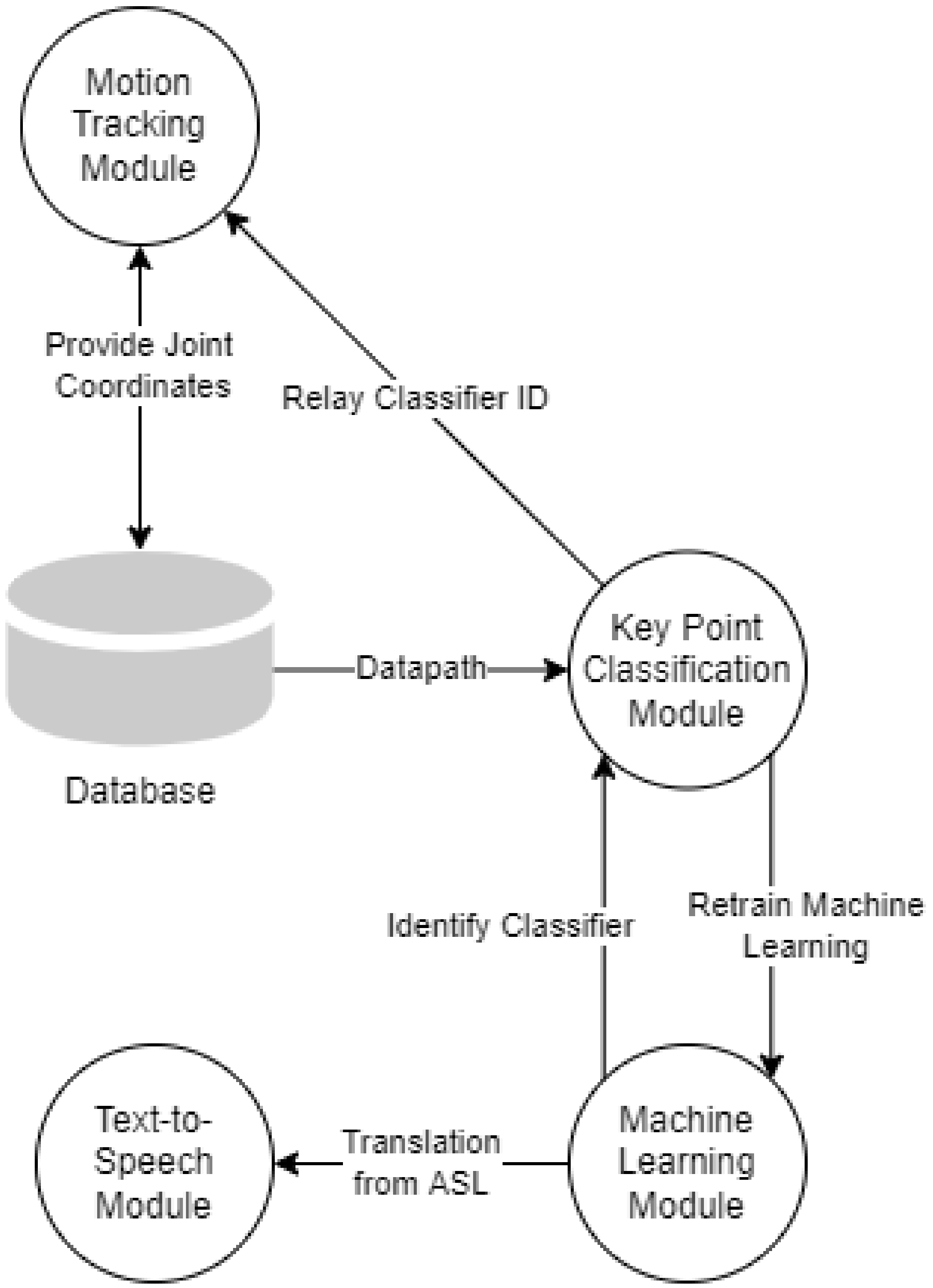

Figure 1: Hand Variables

[I made two tables, choose the one you like —SS]

Goals
Reliable and Accurate Translations
Real Time Translations
Ease of Use
Affordability
Customizable to User

Goals
Reliable and Accurate Translations
Real Time Translations
Ease of Use
Affordability
Customizable to User

6 Project Overview

6.1 Normal Behaviour

OpenASL acts as a medium for sign language to spoken language to help the hearing impaired communicate without the need of a human translator. Under normal operations, the user would perform ASL gestures in front of a camera that would detect motion, which would begin having the Raspberry Pi start classifying the movement of the user with its database and output the corresponding English word/phrase through speakers for the other person to understand. The user is also able to train the algorithm to learn any of the user's subtle differences in gestures from the standard ASL language to improve the accuracy of classification for words/phrases.

6.2 Undesired Event Handling

In the event of an undesired error during the translation, the system should stop the translation being spoken through the speaker and display on the user interface that an error has occurred to let the user know that something has happened to the system. In the case that an error occurs during the process of the user training the model, such that the system is unable to classify the gesture, an error message should display on the interface to tell the user to retry. This would help prevent any incorrect data from being entered into the classification database for more accurate results.

6.3 Component Diagram

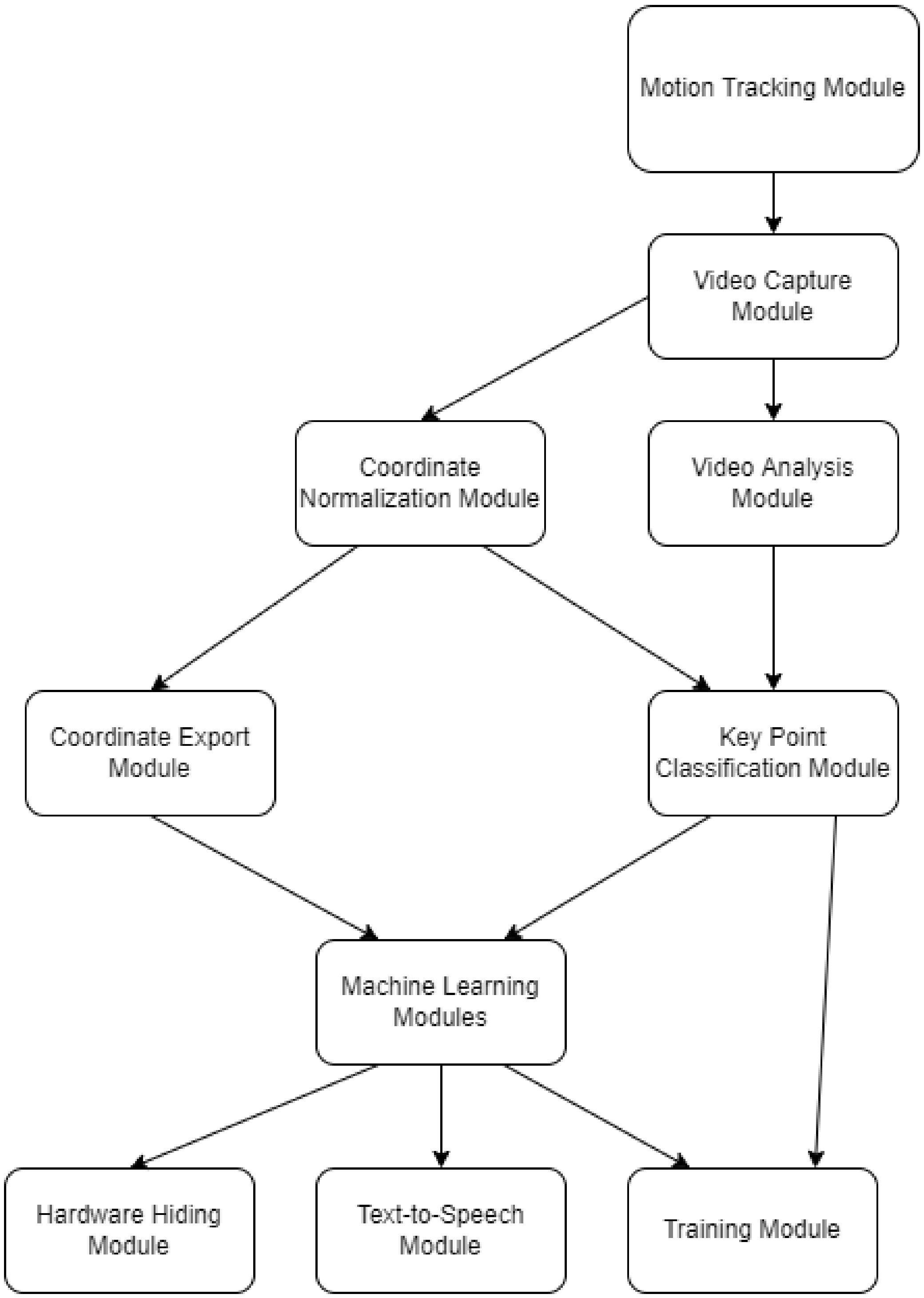

Figure 2: Component Diagram

6.4 Connection Between Requirements and Design

(REF SRS 6.1, 6.2, 8.1, 8.2, 8.3, 8.4 for description of ID).

Requirement ID	**Design Decision**
CFR1	We used a Tensorflow machine learning algorithm already incorporated into the Mediapipe library to recognize hand shape and model its joints through the lens of any camera when it detects motion.
CFR2	The program normalizes the resolution of any camera lens to ensure that the coordinates detected from the camera can be used to classify the vision of a gesture to its corresponding word/phrase.
MLFR1	We used a Tensorflow machine learning algorithm already incorporated into Mediapipe to recognize a hand shape and model its joints. This allows the coordinates of each joint to be identified and recorded into a dataset for the system.
MLFR2	Using the Mediapipe library, we have set up our program to take a set of normalized coordinates based on the location of the hand. This allows us to recognize.
MLFR3	The Mediapipe library has a built-in variable to limit the number of hands that can be tracked. We have limited that number to 2.
MLFR4	We acquired a Raspberry Pi board that we would use to process the machine learning model and set the delay within the hand tracking script to a reasonable amount for conversation.
MLFR5	Mediapipe's built-in machine learning model classifies hand shapes within a specified confidence value between 0 and 1, of which a defaulted value of 0.5 is used. This means that if the model can confidently say with 50% certainty that the video/image being recorded contains a hand, the hand and its joints will be displayed (subject to hand limit defined by MLFR3).
MLFR6	The Raspberry Pi board has the necessary hardware to process the hand gestures at a pace that is understandable to other people.
MLFR7	The program is set up with a training mode, where we can add data points for certain gestures to increase the accuracy of the model.
NFR1	The Keypoint Classifier Module checks the dataset for each gesture and is able to display its confidence as a percentage for an estimate of how often it will be able to classify the correct gesture to support testing.

NFR2	Mediapipe, an open-source library built from OpenCV for Python, is able to detect the user's hands and highlight their joints accordingly to help the user focus on their hand gestures.
NFR3	The device is already preset with a dataset that contains many common phrases that can be translated, allowing for little set up.
NFR4	The user interface contains only written words and is clearly labeled for translating (normal operation), and can be further adapted into training the machine learning model as incorrect classifications indicate that the model must be retrained.
NFR5	The Motion Tracking module is equipped with the ability to snapshot hand positions as needed to train the relevant classifier label if retraining is needed. These coordinates are then stored into a .csv file which can be used in conjunction with the Keypoint Classifier module to retrain the model.
NFR6	We have chosen to implement our program onto a Raspberry Pi board. The board has a small form factor, making it easy to carry and very portable.
NFR7	The current design is using the universal training model for standard ASL gestures. And the machine learning function for the program will allow for the ease of training in different grammar and phonology in the future.

Module	**Requirements+**
Motion Tracking	CFR1, CFR2, MLFR1, MLFR2, MLFR3, MLFR5, MLFR6, NFR2
Keypoint Classifier	NFR1, NFR5
Machine Learning	MLFR4, MLFR7

7 System Variables

7.1 Monitored Variables

Monitor Name	Monitor Type	Range	Description
mode	Integer	[0, 2]	Program operation mode (normal, keypoint training, point history/motion training)
num	Integer	[0,25]	Determines corresponding ASL classifier label to use
landmark_list	List	[-1.0,1.0]	Set of normalized coordinates to be paired with ASL classifier label

7.2 Controlled Variables

Control Name	Control Type	Value	Description
RANDOM_SEED	Integer	51	Value set to control random shuffling in ML model for reproducible output

7.3 Constants Variables

Constant Name	Constant Type	Value	Description
min_detection_confidence	Integer	0.5	Minimum confidence for hand detection from tracking script
num	Integer	0.5	Minimum confidence for hand joint tracking
landmark_list	Integer	16	Maximum point history entries for motion tracking
NUM_CLASSES	Integer	26	Number of classifier labels in ML model

8 User Interfaces

The user interface is designed to give the user the choice between translating ASL to spoken language and training the machine learning module to adapt to any of the user's habits for any phrases. For translating ASL, the user would interact with the Raspberry Pi Camera that is equipped to register hand movement for the machine learning algorithm to classify.

Either on a PC or laptop, the English translation will be provided to the user to determine if it is accurate or if it requires additional training. For training, the interface will display when to do the gesture in front of the camera to retrain the classification module for more accurate results. For the audience, who do not know sign language, a text-to-speech module will deliver a translation of ASL.

9 Mechanical Hardware

For our project, we will be implementing our program onto a Raspberry Pi 3 Model B+ board. We chose a Raspberry Pi board mainly because of its smaller form factor. We wanted our device to be portable, but we also need enough processing power to run a machine learning algorithm. The Raspberry Pi board allows us to achieve both of these goals. The Raspberry Pi board also has the ability to add external components such as a speaker and camera, both of which will be needed for our project. We will be using the Raspberry Pi Camera v2 to capture the sign language gestures, and we will use the on-board audio socket to output the translation.

Raspberry Pi 3 Model B+ GPU: Broadcom Videocore-IV Memory: 1 GB Storage: Micro-SD Ports: 3.5mm analogue audio-video jack, Camera Serial Interface (CSI)

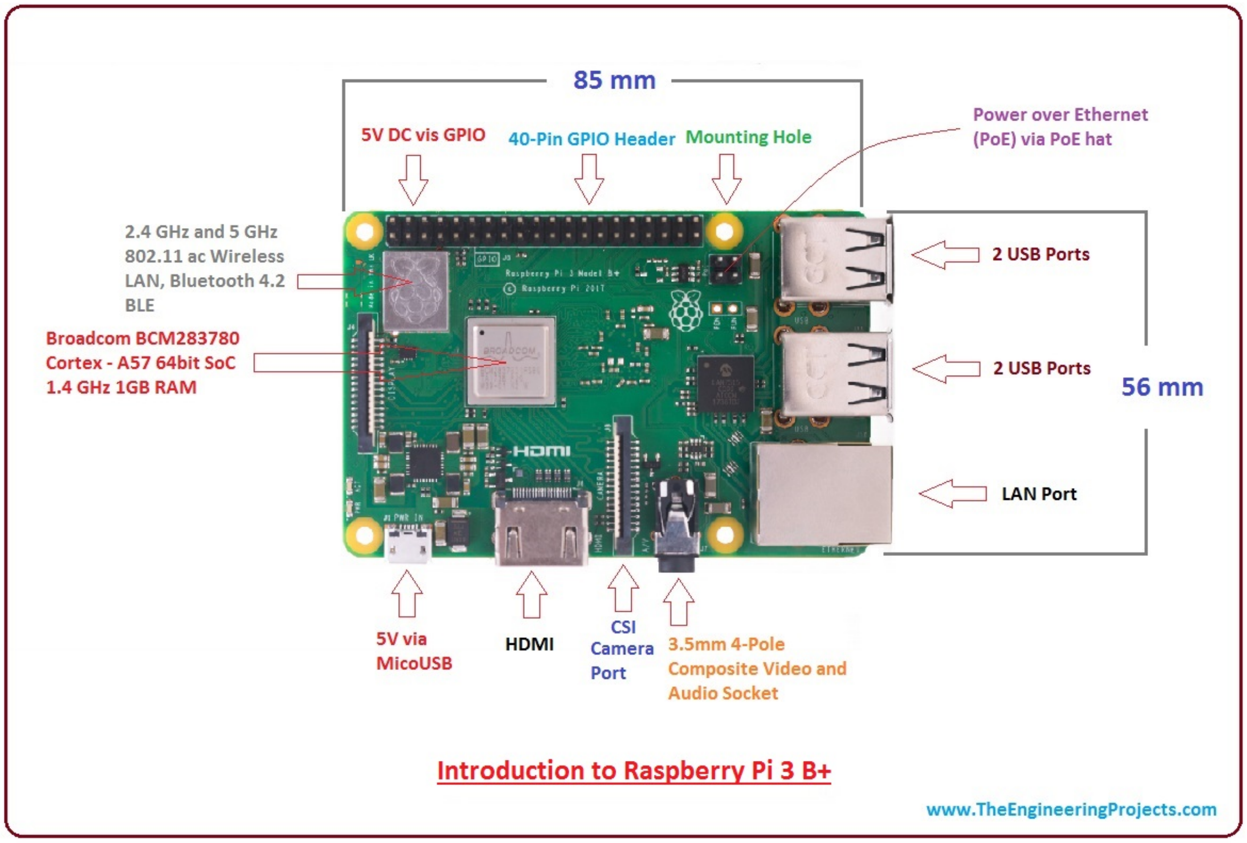
85 mm
5V DC vis GPIO
40-Pin GPIO Header
Mounting Hole
Power over Ethernet (PoE) via PoE hat
2.4 GHz and 5 GHz 802.11 ac Wireless LAN, Bluetooth 4.2 BLE
Broadcom BCM283780 Cortex - A57 64bit SoC 1.4 GHz 1GB RAM
2 USB Ports
2 USB Ports
56 mm
LAN Port
5V via MicoUSB
HDMI
CSI Camera Port
3.5mm 4-Pole Composite Video and Audio Socket
Introduction to Raspberry Pi 3 B+
www.TheEngineeringProjects.com

Figure 3: Raspberry Pi Model 3 B+ diagram

Raspberry Pi Camera v2 Video Resolution: 1080p30, 720p60 and 640x480p60/90 Sensor: Sony IMX219 Image sensor

10 Design of Electrical Components

N/A

11 Design of Communication Protocols

N/A

12 Timeline

[Schedule of tasks and who is responsible —SS]

Objective	**Date to be completed by**	**Member(s) Responsible**
Create an interface that lets the user switch between training and translating	Jan 20, 2023	Runze Zhu and Jiahui Chen
Add an expanded vocabulary of common phrases into the database	Jan 31, 2023	Robert Zhu
Build and connect Raspberry Pi to OpenCV system to provide real-time translation	Feb 5, 2023	Zifan Meng
Program a text-to-speech algorithm	Feb 5, 2023	Nafi Hasan
Move OpenCV system and machine learning model onto Raspberry Pi	Feb 10, 2023	Kelvin Huynh

A Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

   Robert Zhu: One of the limitations for our solution is that it is unable to capture the full language of ASL through only capturing hand gestures. That is because ASL often uses a range of different body movements to deliver a proper sentence. For example, the phrase for "come here" involves tapping the knee, which our program is unable to categorize since it only tracks hand movement. Grammar is also an issue as face expressions dictate the tone, urgency, and even the meaning of phrases when combined with hand gestures. At the moment, these aspects of ASL are out of the scope for the current plan, however, with enough time and datasets from the ASL community that the machine learning algorithm can read from, more of this language can be translated.

   Zifan Meng: One of the limitations of our solution is we cannot provide customization to each individual user. Every user has their own habits about hand gestures, for our current solution, we only have a universal training model for standard ASL gestures, if a user's hand gesture differs a lot from the standard ASL gesture, it is likely that the translation is incorrect. If we were to have more development time, we could develop a user accounts function, each user has their own account and their own database, some specific gestures of theirs can be stored in the database and the product is able to translate accordingly to the account that's logged in.

   Mirza Nafi Hasan: One limitation of our solution is the amount of time it takes to train our machine learning model. Currently, we have it set up in a way that requires someone to perform the hand sign and record it themselves. Since there are no machine learning models that have data on different sign language gestures, if we want to be able to translate all of ASL, that will require someone to perform every gesture in sign language, which would be a very time consuming process. If we had unlimited resources, we could have someone who is very familiar with ASL do these gestures, or we could try to find a way to automate the training process.

2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select documented design? (LO_Explores)

Robert Zhu: One other design solution involved designing a device that is placed on the hands of the user with sensors that are capable of capturing hand gestures, and transmitting the information into a spoken language. The main benefit of this method compared to our current chosen design would be a very high accuracy in being able to classify the motion. Having sensors on each joint would provide more information for the processing unit by being able to distinctly tell the position of each finger, leading to fewer mistakes compared to using a camera sensor. We did not select this method as it greatly reduced the scope of ASL to only having finger movements. ASL uses dynamic motion that can not be captured using glove sensors, while a camera sensor is able to detect these movements and classify them. A dataset is also easier to gather through using a camera sensor as the hardware required for gloves requires a lot of effort to generate enough datasets to accurately translate.

Zifan Meng: One design solution we considered was implementing a LED screen on Raspberry Pi that outputs the translation results. Our current solution is to have a speaker for audio output, if we have both visual output and audio output, the device is more complete as a portable, individually-working device, and is suitable for all users (users with other disabilities). However this solution significantly increases the complexity of the design and increases the cost of the device. The design that we have right now are targeting users that do not have hearing disabilities, hence the audio output is sufficient enough for helping most of the users to understand ASL languages.

Mirza Nafi Hasan: One other design solution we considered was instead of using a Raspberry Pi board, we considered using a smartphone. Some advantages of this design solution would be that it satisfies the portability requirement we were aiming for. This would also be very accessible considering that everyone has a phone these days. Almost all smartphones have a camera and a speaker meaning we can capture the ASL gestures as input and output the translation through the speaker. We decided against this design because we were unsure whether the average smartphone would be able to run a machine learning algorithm on it. Also, creating a program for one set of hardware would be easier to manage since many things can go wrong if we wanted to test our program on multiple types of hardware.