

ASLBoT: Assistive Sign Language Bot Translator



University of Central Florida

*College of Engineering and Computer Science
Department of Electrical and Computer Engineering*

Final Report

Senior Design II
Dr. Samuel Richie
Group 14

Members

Gustavo Camero	Computer Engineering	gcamero@knights.ucf.edu
Luis Hurtado	Electrical Engineering	luis.hurtado@knights.ucf.edu
Michael Loyd	Electrical Engineering	michael.loyd@knights.ucf.edu
Jared Spinks	Computer Engineering	spinks.jared@knights.ucf.edu

Client/Advisor

Dr. Chung Yong Chan chungyong.chan@ucf.edu

Table of Contents

1. Executive Summary	1
2. Project Description	2
2.1 Project Background	2
2.2 Motivation	2
2.2.2 GPU Based ASLBoT	3
2.2.3 GPU-FPGA Based ASLBoT	3
2.2.4 SBC-MCU Based ASLBoT	3
2.3 Goals and Objectives	4
2.4 Requirement Specifications	5
2.5 Marketing Requirements.	5
3. Research Related to the Project	7
3.1. Existing Projects and Products	7
3.1.1. Google Translate	7
3.1.2. Amazon Translate	8
3.1.3. Stratus Video	8
3.2 Relevant Technologies	8
3.2.1. American Sign Language Typography	9
3.2.1.1. SignWriting	9
3.2.1.2 Si5s	10
3.2.1.3 SignFont	11
3.2.1.4 Stokoe Notation	11
3.2.1.5 Gloss Notation	11
3.2.1.6 Extended Linear Stokoe (ELS) Notation	13
3.2.2 Machine Learning	19
3.2.2.1. Speech to Text	19
3.2.2.2 Text to Text Translation	24
3.2.2.3 Text-to-Speech	33
3.2.3 Processor Technologies	35
3.2.3.1. CPU	36
3.2.3.2. MCU	36
3.2.3.3. FPGA	37
3.2.3.4. GPU	38
3.2.3.5 GPU versus FPGA	39
3.2.4 3D Graphics API Technologies	40
3.2.4.1 OpenGL	40
3.2.4.2 Direct3D 12	41
3.2.4.3 Metal	42
3.2.4.4 Vulkan	42
3.2.5 Internet Connectivity	42
3.2.6 Voltage Regulation	43
3.2.6.1 Linear Voltage Regulator	43
3.2.6.2 Switching Voltage Regulator	44
3.2.6.3 Considerations for Voltage Regulators	44
3.2.7. 3D Rendering Software Platforms	45
3.2.7.1. Unreal Engine 4	46
3.2.7.2. Unity	46
3.2.7.3. MikuMikuDance	47

3.2.7.3. Blender	47
3.2.7.4 Godot	48
3.3. Initial Components and Part Selections	49
3.3.1 Sound System Selection	49
3.3.1.1 Amplifier	49
3.3.1.2 Speaker	51
3.3.1.3 Microphone	52
3.3.2. FPGA Selection	54
3.3.2.1 Altera Cyclone IV – EP4CE22E22xxx	55
3.3.2.2 Spartan 3E - XC3S500E-xPGx208C	55
3.3.2.3 Artix 7 – XC7A35T-1CPG236C	55
3.3.3 Computer-on-Module (COM) Selection	56
3.3.3.1 NVIDIA Jetson Nano Developer Board	56
3.3.3.2 Nvidia Jetson TX2 Developer Board	57
3.3.3.3 ASUS Tinker Board	58
3.3.3.4 Raspberry Pi 4 Model B	58
3.3.4 Display Selection	59
3.3.4.1 10.1" Display & Audio IPS Panel	60
3.3.4.2 Sceptre E205W-16003S LED Monitor	60
3.3.4.3. UPERFECT 12.3" Touch Monitor	60
3.3.5 Controller Selection	60
3.3.5.1 IR Remote	61
3.3.5.2 RF Remote	61
3.3.5.3 Push Button	62
3.3.6 Wi-Fi Module Selection	62
3.3.6.1 Intel Dual Band Wireless-AC 8265	62
3.3.6.2 Ultra USB Wi-Fi Adapter	63
3.3.7 Voltage Regulator Selection	63
3.3.7.1 LP5900	63
3.3.7.2 LP38500-ADJ	64
3.3.7.3 LM7805	64
3.3.8 Initial Summary of Selected Parts	64
3.3.8.1 Speaker	64
3.3.8.2 Microphone	65
3.3.8.3 FPGA	65
3.3.8.4 COM	65
3.3.8.5 Display	65
3.3.8.6 Controller	66
3.4 Final Summary of Selected Parts	66
3.4.1 Final Microcontroller	66
3.4.2 Final Single Board Computer	66
3.4.3 Final Microphone	67
3.4.4 Final LCD Display	67
3.4.5 Final IR Sensor	67
4. Related Standards and Design Constraints	68
4.1 Related Standards	68
4.1.1 Wireless Communication Standards	68
4.1.1.1. Wi-Fi Standards	68
4.1.1.2. Consumer Infrared Standards	70
4.1.1.3 Bluetooth Standard	70
4.1.1.4 Zigbee	70
4.1.2. Unicode 5.0 Standard	71
4.1.2.1. American Standard Code for Information Interchange (ASCII)	72

4.1.3 Python Programming Language Standards	72
4.1.4 Compute Unified Device Architecture (CUDA).....	73
4.1.5 Hardware Description Language - Verilog Standards	73
4.1.5.1 HDL	73
4.1.5.2 Verilog	73
4.1.5.3 System Verilog	74
4.1.6 Machine Translation Benchmarking Models	75
4.1.6.1. Bi-Lingual Evaluation Study (BLEU) Score.....	75
4.1.6.2. Prediction Performance (PRED) Score.....	75
4.1.6.3. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) Score	77
4.1.7 Privacy and Data Storage Standards.....	77
4.1.7.1 Health Insurance Portability and Accountability Act (HIPAA).....	78
4.1.8 Serial Communication Standards	78
4.1.8.1 Inter-Integrated Circuit (I2C) Standard	79
4.1.8.2 Serial Peripheral Interface (SPI).....	80
4.1.8.3 Universal Asynchronous Receiver/Transmitter (UART) Standard.....	81
4.1.8.4 Inter-Integrated Circuit Sound (I2S) Standard.....	81
4.1.8.5 Universal Serial Bus (USB) Standards.....	82
4.1.9 Audio/Video Connection Protocols	83
4.1.9.1 High Definition Multimedia Interface (HDMI).....	83
4.1.9.2 Digital Video Interface (DVI)	83
4.1.9.3 Video Graphics Array (VGA).....	84
4.2 Design Constraints.....	84
4.2.1 Economic and Time Constraints.....	84
4.2.2 Manufacturability and Sustainability Constraints.....	85
4.2.3 Moral and Ethical Constraints	86
4.2.3.1 Automation Displacing Human Labor.....	86
4.2.4 Environmental, Health and Safety Constraints	88
4.2.5 Social and Political Constraints	88
5. Initial Project Hardware Design	90
5.1 Initial Project Design and Component Diagrams	90
5.2 Power Supply Design	91
5.3 Hybrid COM with FPGA	94
5.3.1 I/O Board Description	94
5.3.1.1 Mode Selection Module	95
5.3.1.2 Voice Recording Module	96
5.3.1.3 Status Module	96
5.3.2 General Layout of the I/O Board	96
5.3.3 Communication for Hybrid System	97
5.3.3.1 Audio Transmission	97
5.3.3.2 Status Transmission.....	98
5.4 LCD Interface	98
5.5 Initial Hardware Design	98
5.5.1 Initial Hardware Block Diagram.....	98
5.5.2 Initial Hardware Schematics	100
5.5.2.1 FPGA Diagram.....	100
5.6 Initial Hardware Design & Bill of Materials (BOM)	102
6. Project Software Design	104
6.1 Initial Software Functionality	104

6.1.1 User Options	104
6.1.1.1 Option A: Speech-to-Speech Translation Mode	104
6.1.1.2 Option B: Speech-to-Sign Mode	105
6.1.2 Software Procedure	105
6.1.2.1 Step 1: Speech Capture	105
6.1.2.2 Step 2: Speech-To-Text Conversion	105
6.1.2.3 Step 3: Neural Machine Text-to-Text Translation	106
6.1.2.4 Step 4: Mapping ASL-Gloss to Custom ELS Notation	106
6.1.2.5 Step 5: Rendering ASL Gestures in 3D Graphics Platform	106
6.2 Changes to Software Functionality in Final Design	108
6.2.1 Obsolescence of ELS Notation	108
6.2.2 Revision of User Options	109
6.2.3 Revision of Software Procedure	109
6.2.4 Unity Engine Script Functionality	110
7. Project Testing and Prototyping	115
7.1 Initial OpenNMT Model Testing	115
7.2 Changes to OpenNMT Training Approach	124
7.3 Changes to PCB Design and Hardware Functionality	144
7.3.1 Hardware Prototyping	144
7.3.2 PCB Design	146
7.2.3 Final Hardware Block Diagram	150
7.2.4 Final Bill of Materials	151
8. Operator's Manual	152
9. Administrative Content	156
9.1 Budget and Finance	156
9.1.1 Initial Budget Cost	156
9.1.2 Final Budget Cost	157
9.2 Project Milestones	157
9.2.1 Important Deadlines	159
9.3 Work Distribution	159
<i>Appendix A: Copyright Permissions</i>	<i>160</i>
<i>Appendix B: NMT Training Corpora</i>	<i>161</i>
<i>Appendix C: Acknowledgements</i>	<i>165</i>
<i>Appendix D: Works Cited</i>	<i>166</i>

List of Figures.

Figure 1. House of Quality of ASLBoT.6

Figure 2. Examples of Different ASL Notation Styles. (Public domain.)9

Figure 3. Example of a SignWriting Sign Box. (Public domain.)10

Figure 4. Example of a Sentence in ASL-Gloss.12

Figure 5. Comparison of Original Stokoe Notation (left) and ELS Notation (right)14

Figure 6. Example of a Sentence in ELS. Note that this sentence is the same as the ASL-Gloss sentence in Figure 4.14

Figure 7. Break-Down Diagram of a Single-Handed ELS Gesture.18

Figure 8. Break-Down Diagram of a Bimanual ELS Gesture.18

Figure 9. Example of a basic speech recognition system. (Public domain.)20

Figure 10. Example of the statistical model (a) Hidden Markov Model with (b) state sequence and (c) symbol sequence. (Public domain.)20

Figure 11. Example of Dynamic Time Warping for (A) Two Sequence Representation of a Person Spelling out “Pen” in Sign Language, (B) with the Dynamic Time Warping Finding the Proper Alignment. (Public domain.)21

Figure 12. Wav2letter++ Architecture using Convolutional Layers. (Public domain.)22

Figure 13. Deepspeech2 Model using the Deep Neural Network Approach which includes Convolutional Layers during Training. (Public domain.)23

Figure 14. Example of hybrid rule-based machine translation architecture. (Public domain.) 25

Figure 15. Example of basic statistical machine translation. (Public domain.)26

Figure 16. Example of the Neural network used in Neural Machine Translation. (Public domain.)27

Figure 17. Diagram depicting Moses’s factored translation model approach. (Public domain.)27

Figure 18. Neural network algorithm representative of the OpenNMT model. (Public domain.)28

Figure 19. The pipeline architecture of the OpenLogos system. (Public domain.)29

Figure 20. Diagram depicting how Google’s AutoML Translation performs the custom training for the user. (Public domain.)30

Figure 21. General Representation of a Text-to-Speech system. (Public domain.)33

Figure 22. Diagram of Causal Convolutional Layers. (Public domain.)34

Figure 23. Example of NVIDIA GPU Architecture. (Public domain.)39

Figure 24. Basic flow of Machine Learning on a GPU. (Public domain.)39

Figure 25. An example of the architecture of OpenGL. (Public domain.)41

Figure 26. Organization of the OSI Model. (Public domain.)69

Figure 27. Zigbee Stack Architecture. (Public domain.)71

Figure 28. Standard Testing Environment for System Verilog. (Public domain.)74

Figure 29A. Calculation of the Regression Factor. (Public domain.)76

Figure 29B. Calculation of the Adjusted Regression Factor. (Public domain.)76

Figure 30. Calculation of the Normalized Root-Means-Squared Error. (Public domain.)76

Figure 31. Calculation of the Matthews Correlation Coefficient. (Public domain.)76

Figure 32A. The I2C Bus. (Public domain.)79

Figure 32B. Master-Slave I2C Communication. (Public domain.)80

Figure 33. SPI Communication Setup. (Public domain.)80

Figure 34. SPI Modes of Operations. (Public domain.)81

Figure 35. UART Communication Scheme. (Public domain.)81

Figure 36. Different Configurations for the I2S Protocol.82

Figure 37. Hardware Block Diagram of the Second Initial Project Design.90

Figure 38. Power Supply Circuit Schematic.93

Figure 39. Block Diagram of I/O Board.95

<i>Figure 40. State Machine for Mode Selection.</i>	95
<i>Figure 41. State Machine for Voice Recording.</i>	96
<i>Figure 42. General Board Layout.</i>	97
<i>Figure 43. Block Diagram of I2S Controller IP.</i>	97
<i>Figure 44. Initial Hardware Block Diagram.</i>	99
<i>Figure 45. Breakdown of Conversion of ELS Notation into 3D Graphics Rendering.</i>	106
<i>Figure 46. Initial Software Flow Chart.</i>	107
<i>Figure 47A. Final Software Diagram.</i>	113
<i>Figure 47B. Final Animation Flow Diagram.</i>	114
<i>Figure 48. Test English and Gloss Corpora used in Attempt 1.</i>	116
<i>Figure 49. Validation English and Gloss Corpora for Attempt 1.</i>	116
<i>Figure 50. Example Results of Attempt 1 Post-Training.</i>	117
<i>Figure 51. Test English and Gloss Corpora used in Attempt 2.</i>	118
<i>Figure 52. Validation English and Gloss Corpora for Attempt 2.</i>	119
<i>Figure 53. Example Results of Attempt 1 Post-Training, Part 1.</i>	119
<i>Figure 54. Example Results of Attempt 1 Post-Training, Part 2.</i>	119
<i>Figure 55. Sample of Test English and Gloss Corpora used in Attempt 3.</i>	121
<i>Figure 56. Sample of Validation English and Gloss Corpora for Attempt 3.</i>	121
<i>Figure 57. Example Results of Attempt 3 Post-Training, Part 1.</i>	122
<i>Figure 58. Example Results of Attempt 3 Post-Training, Part 2.</i>	122
<i>Figure 59. Example Results of Attempt 3 Post-Training, Part 3.</i>	122
<i>Figure 60. Example Results of Attempt 4 Post-Training, Part 1.</i>	122
<i>Figure 61. Example Results of Attempt 4 Post-Training, Part 2.</i>	122
<i>Figure 62. Sample of Test English and Gloss Corpora used in Attempt 5.</i>	123
<i>Figure 63. Example Results of Attempt 5 Post-Training.</i>	123
<i>Figure 64. Example of breaking down a sentence from the corpus into smaller phrases for redundancy.</i>	125
<i>Figure 65. Two translation examples from English to Gloss using the first training corpora modification approach. Neither of the attempts were satisfactory translations.</i>	125
<i>Figure 66. Here the first and second approach compared for two translation examples of English to Gloss.</i>	126
<i>Figure 67. Prototyping with LEDs and MSP430FR6989.</i>	145
<i>Figure 68. Final Schematic - MSP430FR6922 connections.</i>	147
<i>Figure 69. Final Schematic - Component Connections.</i>	147
<i>Figure 70: Final printed circuit board layout.</i>	149
<i>Figure 71: Final Hardware Block Diagram.</i>	150

List of Tables.

<i>Table 1. Requirement Specifications.</i>	5
<i>Table 2. Orientations used in ELS Notation.</i>	15
<i>Table 3. Movements used in ELS Notation.</i>	15
<i>Table 4. Hand Shapes used in ELS Notation.</i>	16
<i>Table 5. Locations used in ELS Notation.</i>	16
<i>Table 6. Other Modifier Characters used in ELS Notation.</i>	17
<i>Table 7. Comparison of Ratings for CPU, MCU, FPGA, and GPU.</i>	40
<i>Table 8. Comparison of Hardware/Licensing between 3D Graphic-Rendering Platforms.</i>	49
<i>Table 9. Comparison of Significant Parameters of Amplifiers.</i>	51
<i>Table 10. Comparison of Significant Parameters of Speakers.</i>	52
<i>Table 11. Comparison of Significant Parameters of Microphones.</i>	54
<i>Table 12. FPGA Comparison.</i>	56
<i>Table 13. COM Comparison.</i>	59
<i>Table 14. Total Cost of Final Device.</i>	67
<i>Table 15. Summary of the Various I2C Standard Data Transfer Speeds.</i>	79
<i>Table 16. Summary of Major USB Revisions.</i>	82
<i>Table 17. Overall Power Requirements.</i>	92
<i>Table 18: Power Supply BOM.</i>	94
<i>Table 18. I/O Board BOM.</i>	101
<i>Table 19. Initial BOM.</i>	103
<i>Table 20. ASLBoT OpenNMT Set of BLEU Scores.</i>	127
<i>Table 21. Full OpenNMT Training Corpus, Part 1.</i>	127
<i>Table 22. Full OpenNMT Training Corpus, Part 2.</i>	128
<i>Table 23. Full OpenNMT Training Corpus, Part 3.</i>	129
<i>Table 24. Full OpenNMT Training Corpus, Part 4.</i>	130
<i>Table 25. Full OpenNMT Training Corpus, Part 5.</i>	131
<i>Table 26. Full OpenNMT Training Corpus, Part 6.</i>	132
<i>Table 27. Full OpenNMT Training Corpus, Part 7.</i>	133
<i>Table 28. Full OpenNMT Training Corpus, Part 8.</i>	134
<i>Table 29. Full OpenNMT Training Corpus, Part 9.</i>	135
<i>Table 30. Full OpenNMT Training Corpus, Part 10.</i>	136
<i>Table 31. Full OpenNMT Training Corpus, Part 11.</i>	137
<i>Table 32. Full OpenNMT Training Corpus, Part 12.</i>	138
<i>Table 33. Full OpenNMT Training Corpus, Part 13.</i>	139
<i>Table 34. Full OpenNMT Training Corpus, Part 14.</i>	140
<i>Table 35. Full OpenNMT Training Corpus, Part 15.</i>	141
<i>Table 36. Full OpenNMT Training Corpus, Part 16.</i>	142
<i>Table 37. English to ASL-Gloss Validation Corpus, Part 1.</i>	143
<i>Table 38. English to ASL-Gloss Validation Corpus, Part 2.</i>	144
<i>Table 39. Final Bill of Materials.</i>	151
<i>Table 40. Original Business Budget Approximations for each Component.</i>	156
<i>Table 41. Final Budget Cost of the ASLBoT system.</i>	157
<i>Table 42. Senior Design I – Milestones.</i>	158
<i>Table 43. Senior Design II – Milestones.</i>	158
<i>Table 44: Senior Design I Deadlines.</i>	159
<i>Table 45: Senior Design II Deadlines.</i>	159
<i>Table 46: Final Work Distribution.</i>	159

1. Executive Summary

Translation is the medium for maintaining a meaningful conversation between two individuals who speak different languages. Human-based translations are widely applied in many aspects of society including the financial, medical, legal, and travel industries. The process of translating through human interpreters, however, proves to be erratic, difficult to arrange, and expensive. Deaf students enrolled in a typical classroom environment require robust translation rather than an abridged translation, just like a child of hearing requires a dynamic language environment to learn their language. Research has shown correlation in school the performance of deaf students versus hearing students in terms of grades. The research concluded that deaf students performed poorly due to poor-quality interpretation services provided by the school. Moreover, in recent years, deaf students have been facing the issue of fast responsiveness as the request of ASL interpretation services enabled by the Services for Students with Disabilities requires at least a three-day notice. Thus, the need for a more accurate, responsive, read to use, and low-cost translation system is required for such applications.

Neural machine translation (NMT) provides the next generation of real time translation with minimal errors. NMTs have proven to be useful for their flexible deployment and their ease of use. Despite requiring large data sets to function properly, there are open-source NMTs that can be used for product commercialization. However, most NMTs only perform the translations for spoken languages and often neglect sign languages such as American Sign Language (ASL). People who are congenitally deaf or have never developed an understanding of spoken language use ASL as their primary language. These people lack the same understanding of the language as an individual who is able to listen and speak the language. Thus, an any-to-text translation is insufficient for the Deaf. This is where the NMT algorithms' primary ability to translate any text-to-text falls short of providing the service to ASL users. In this document, we are proposing a real-time translator with virtual ASL interpretation. This device will be designed to require minimal overhead, low budget, and be accurate on speech acquisition and translation delivery. In addition, the device will deliver real time translations such that conversations between both users are continuous with no wait on the translation. The functionality of the translator is to provide spoken English-to-ASL translation.

Our proposed real-time translator with ASL interpretation bridges the gap between universal machine translation and physical human translation. This is critical in modern society where the greater population, including those with disabilities, rely on advanced artificial intelligence to enrich their daily lives. The goal of our Assistive Sign Language Bot Translator (ASLBoT) is to provide a user-friendly experience, effective sign language rendering and a high level of accuracy. Ultimately, ASLBoT aims to provide students with hearing disabilities the capability to provide and receive effortless and natural communication within classroom environments.

2. Project Description

This section provides a brief overview of our project, including the overall design and user interface. This overall design was modified from the original project design, which included a built-in speaker, microphone, and integrated display. This original design also included internal battery power. These components of the project have since been removed, as explained throughout the paper.

2.1 Project Background

The product consists of a single-board computer (SBC), a monitor, an IR remote, an MCU-controlled circuit board, an LCD, and peripheral LEDs. The external microphone will record the user's speech, the speech will be translated into text, and then the speech that was recorded will output on the built-in speakers. The user will be able to control the system by using the MCU-controlled circuit board. They will be able to choose when to start recording, stop recording, reset the board, and wake up the system. The software will also display the original and translated text on the LCD monitor behind the ASL translation to show insights to the user on how the translation is being done. The product will utilize the LCD monitor to display the sign language output using a 3D modeling software. The SST mode will also include functionality from the SAT such as displaying the original and translated text. For powering the product, we have several approaches. Our first approach would consist of a portable battery supply. This internal battery would supply power to the display, the GPU, etc. We are considering the use of a lithium-ion battery that will be able to power the console for at least three to four hours on a single, full charge. The battery will be rechargeable from a standard US AC wall outlet (120V at 60 Hz). Our other approach would be to have the unit be plugged into the wall; that is, the power supply to the system would be from an external source, which would be a standard US AC wall outlet. This is possible since the unit is being developed for medical use, such as in hospitals. Most machines and stations inside of hospitals are installed into rolling carts; and when needed, they are rolled into the patient's room and plugged into the wall. Our product will utilize Wi-Fi to connect to translation services such as Watson API. These services include speech recognition, real time translation, and speech to text. We decided that utilizing APIs for the speech to text translation will allow us to focus more on the text to sign language translation. With the speech-to-speech translation processing offloaded to cloud services, we would have more data storage and processing power available for the speech-to-ASL translation.

2.2 Motivation

The motivation of the product originated from one of our members who works in a hospital where medical professionals see patients regularly who cannot speak English. In the beginning, the project was proposed as a standalone machine translation, since in this way hospitals can save money from buying in-person translators. With enough research, we figured out that no machine translator has been able to translate from one language to sign language. Furthermore, there have been court cases since 2011 where there were multiple

deaf hospital patient complaints that the hospital interpreting services for ASL were poor. Most of the cases reached a settlement of \$70,000 due to the poor services that the hospitals were offering to their deaf patients [2]. Hence, the proposed device can do both speech-to-text and speech-to-sign translations. In addition, we decided to market our product to school settings since the language complexity in such environments are simpler and allows us to create a more precise NMT model given the timeframe. Moreover, the use of small corpora is not present in NMT model development. We want to be able to show that we can achieve effective results using small corpora. Finally, working in a group setting can be very challenging as communication is the most important aspect which can serve as a very valuable tool before graduating as we head to graduate school or industry jobs.

2.2.2 GPU Based ASLBoT

To design a device that generates machine translations using NMTs with the GPU as the main system to handle every component on the device.

- The GPU will manage and control the touchscreen LCD, the microphone, and the speaker
- The GPU will store the model generated by the NMT in its memory for machine translations.
- The GPU will also generate the sign language translation by applying image generation techniques from the output file created by the machine translation model.
- The GPU will send the image generated translation to the touchscreen LCD.

2.2.3 GPU-FPGA Based ASLBoT

To design a device that generates machine translation using NMTs with an GPU-FPGA architecture to handle every component on the device. The reason behind this design is because the performance of the device may be greatly affected with the GPU as the sole controller of the system.

- The GPU will manage and control the touchscreen LCD, the microphone and the speaker, basically a CPU.
- The GPU will also store the machine translation model inside the main memory and send the output of the translation as an input to the GPU.
- The FPGA will then proceed to create the pertaining sign language translation using image generation techniques.
- The GPU will then send the file to the FPGA so that it will get displayed on the touchscreen LCD.

2.2.4 SBC-MCU Based ASLBoT

To design a device that generates machine translation using NMTs with an SBC-MCU architecture to handle every component on the device. The reason behind this option was because the initial design of GPU-FPGA based ASLBoT proved to be too difficult to implement given the time frame of Senior Design 2. We instead opted for something

feasible but still challenging as we never had done intercommunication between an MCU and an SBC.

- The SBC will handle all the 3D rendering animations.
- The SBC will also store the machine translation model inside the main memory.
- The SBC will host the NMT model and the vocabulary developed by the team.
- The SBC will also host other processes such as access to cloud services.
- The MCU will control the SBC by user buttons input and IR remote.
- The MCU will communicate with the SBC to know in which step the SBC is currently in.
- The MCU will display the current status of the system through LEDs and an LCD

2.3 Goals and Objectives

Based on our research findings, the main objective and overall goal of the ASLBoT is to be able to translate from English speech to American Sign Language (ASL). To achieve this the system will use a neural machine translation model using only small corpora and a novel training framework. The system must perform speech-to-text and speech-to-ASL language translations. To achieve speech-to-text translation, the system requires wireless capabilities to access cloud-based services. The system will indicate the current status of the system through LEDs and an LCD. To provide reliable translations, the system will capture audio recording from the user on real-time. To make the system more interactive, the system allows the user to dictate when to start the recording process and when to stop it. Finally, the system will display real-time rendering animation of the sign language gestures of the user-input sentence. Additionally, to show transparency of the system, the system will show the original English sentence and the ASL translated sentence. Additionally, as aforementioned, the design chosen for the product was the SBC-MCU architecture approach since this is the most viable approach with minimal error currently possible.

Upon completion of our project, our device will be able to:

- Translate English speech into American Sign Language.
- Connect wirelessly to access cloud-based services.
- Perform speech-to-text, speech-to-speech, and speech-to-ASL language translations.
- Indicate the current status of processors and peripherals via LED status lights.
- Capture audio at frequencies around natural human pitches that can be translated into text via speech-to-text algorithms.
- Display real-time rendering of an avatar making sign language gestures.
- Have enough storage to contain the rendering software, avatar animations, and other peripheral programs.
- Receive input from the user regarding intent to record voice and wake up functionalities.

2.4 Requirement Specifications

This section covers the requirement specifications that are necessary to ensure proper functionality of the final device.

Table 1. Requirement Specifications.

Component	Parameter	Design Specification
Microphone	Frequency Response	300 Hz to 3kHz
Memory	Size	32 GB
Power Supply	Power	32 W
System	Translation Accuracy	20% (BLEU Average)
System	Cost	\$500
System	Response	< 5 seconds
Vocabulary	Model Complexity	> 50 words
Neural Machine Translation	Size	< 1000 Entries

2.5 Marketing Requirements.

We are marketing the ASLBoT as an accurate, energy-efficient and cost-effective device that can potentially replace human interpreters in hospital settings. Based on customer complaints on accuracy, the device will focus on delivering and displaying accurate translations that can support speech-to-speech, speech-to-text and speech-to-sign language translations. We will perform one-to-one translation so that no word or context is lost between translation. To ensure continuous exchange of information between the doctor and patient, we will access cloud-based services to deliver real time translations through Wi-Fi connectivity. In addition, we will offer quality service to the deaf patients by providing real time sign language translations using 3D generated ASL models. Moreover, our advisor, Dr. Chan, gave us the following requirements for the device:

- Real Time Translations
- Usage of Small Corpora
- Novel Training Framework for NMT
- Accuracy of NMT Translation
- Cost-effective
- Display ASL Models on an LCD
- Speaker Clarity
- Screen Clarity

All of the aforementioned requirements can be found in Figure 1. In addition, Figure 1 also possesses the engineering requirements that we tasked ourselves with. It can be noted that we decided to go for a power supply system that can support both battery and electrical outlets in case of a power outage so that the device can still be operational. Every requirement has a pertaining level of polarity (for positive or negative impact on the device) and correlations (how one requirement can benefit or hinder another requirement).

		Engineering Requirements								
Polarities		-	+	-	+	-	-	+	+	+
Polarities	Customer Requirements	Electronic Enclosure Dimensions	Power Consumption	Cost	NMT Model Complexity	Storage Capacity	NMT Size	Microphone's Frequency Response	Translation Accuracy	System Response
	+ Real Time Translation					▼				▲▲
	+ Usage of Small Corpora				▲▲	▲	▼▼		▼▼	
	+ Novel Training Framework for NMT					▲▲	▼▼		▼▼	
	+ Accuracy of NMT Translation		▼▼		▲▲		▼▼		▲	▲
	- Cost Effective	▼		▼▼	▲▲	▼▼	▲	▲	▲	▲
	+ ASL Models					▲▲				▼▼
	+ Speaker Clarity		▼	▲						
	+ Screen Clarity	▲	▼	▲		▼				
	Targets for Engineering Requirements	12 in x 12 in x 12 in (except display)	32 W	\$500	Greater than 50 words	32 GB	Less than 1000 entries	300 Hz to 3k Hz	20% BLEU Score	Less than 5 seconds

Polarity	
Positive	+
Negative	-

Correlations	
Strong Positive	▲▲
Positive	▲
Neutral	
Negative	▼
Strong Negative	▼▼

Figure 1. House of Quality of ASLBoT.

3. Research Related to the Project

In this section, we will discuss research regarding the project. The researched found below will range in a wide range of topics ranging from existing projects to different part selections. We will start by first researching the different projects that exists on the market that are similar to our project. It can be noted however that our device is the first attempt of commercialization of Speech-to-Sign Language translation by using NMTs in a hospital setting. Afterwards, we will discuss possible parts for the overall device and explain in depth the specific selection of certain parts over other parts. We will consider peripherals such as microphones and speakers that interface nicely with the GPU. Finally, we conclude the section by giving a summary of the parts pick that will be used in the device.

3.1. Existing Projects and Products

In the market for interlingual translation, there are several available programs in place. Although these programs provide translations between a multitude of spoken languages, very few of them support translation between spoken and manual languages, such as English-to-ASL translation or Japanese to JSL (Japanese sign language), for example.

3.1.1. Google Translate

Google Translate is a multilingual machine translator created in 2006 using statistical machine translation (SMT). This is a free service that started by translating any source text to a designated target text as requested by the user with a support of 105 languages. In its early stages, however, it would translate first into English then to the target language. This service was novel at translating from source word-by-word while also providing disambiguation between the original word and the translated word by providing possible alternate translations. When translating phrases or sentences, Google's SMT would translate erroneously and would often have grammatical errors. In 2016, Google redeveloped Google Translate using neural machine translation (NMT) instead of the previous SMT used. Called "Google Neural Machine Translation (GNMT)", the new, updated Google Translate uses deep learning techniques to translate phrases or sentences and provides accurate context of the sentence. The free service provides many free functionalities that are embedded with Google products in which the service translates more than 100 billion words a day. This GNMT can translate text, speech, images and videos. Some of its functionality is used in translation of websites, documents, mobile applications, and handwriting. Moreover, the service provides a technique called "zero-shot translation" in which the service can translate between two given languages that have not been trained previously by the NMT.

In addition, software developers can integrate their service into their products for a paid fee. Google Translate can be requested through Google Cloud services at the range of prices from \$20 to \$80 for every 1 million characters, depending on the features desired by the software developer. However, even with the implementation of the GNMT, the service still suffers greatly from trying to disambiguate homonyms and translate free of

grammatical errors. There have been instances that these services have been used to replace a human translator, such as in court hearings.

3.1.2. Amazon Translate

Amazon Translate is a cloud service provided by Amazon Web Services (AWS) that uses an NMT to deliver fast, high quality, and affordable language translation. It uses deep learning techniques similar to Google to provide accurate and contextual-rich translations. The service is not open to the public since it is catered towards software integrations in application development due to its simplified API calls. This enables straightforward localization of applications and enables the program to process multilingual data for the ease of workflow data in workspaces.

Their NMT also uses a continual-learning model that improves its translation constantly by learning new datasets to deliver the highest level of accuracy and fluency for various cases. Unlike support of 105 languages in Google Translate, Amazon Translate supports 137 languages ranging from Arabic to Vietnamese, encompassing about 125 language pairs. However, there are some exceptions that Amazon are still trying to fix such as Korean to Hebrew, Chinese (Simplified) to Chinese (Traditional).

Amazon Translate supports the following extra features: secure machine translations for secure communications between web pages and applications using SSL encryption, automated language identification, named entity translation customization, etc. Amazon Translate offers a free tier of 2 million characters per month for 12 months. Afterwards, the user must choose a premium tier which costs \$15 per million characters. The advantage of Amazon Translate over Google Cloud is the easy integration that offers through AWS. Companies such as One Hour Translation, that benchmark NMTs, have praised Amazon Translate for its overall best performance, pricing and easy integration.

3.1.3. Stratus Video

Stratus Video is a language-translation service company that gives clinics and hospitals access to on-demand language translation for a variety of languages. In their Video Remote Interpretation (VRI) service, the clinic is lent hospital-grade metal stands and iPads with which they can use the VRI. After subscribing to their services, medical staff can use the iPad to reach certified medical interpreters for over 35 different languages. The Stratus company also offers alternative translation solutions including audio translation, text translation, and in-person translation. With all of their translation services, however, there is always the implementation of human language translation; that is, there is no use of machine translation in their services.

3.2 Relevant Technologies

This section covers the current technologies related to our project off of which the final design was created.

3.2.1. American Sign Language Typography

American Sign Language (ASL) is a mode of communication based solely on visual gestures; that is, a statement or concept is transferred between speaker and listener through a specific movement of the hand(s) and face. Generally, the head, torso, and legs remain stationary while signing. The translation from English to sign language, however, poses several problems. The use of an NMT requires that both languages have some form of typography. Since ASL is entirely gesture based, there have been several attempts at transcribing ASL from gesture-based to text-based while maintaining the meaning of the original ASL.

Each ASL typographical system makes use of several radicals that, when written in one block, form one morpheme; this is similar to the structure of pictographic languages such as Korean and Chinese. The variety between the different notation styles is the uniqueness of the character registry of each system and the placement of each character to represent the face, hands, and motions.

A few examples of typographical ASL are displayed below in Figure 2; some of these notation styles are discussed below. Note that with most of these notation systems, single-hand gestures are assumed to be for the right hand.

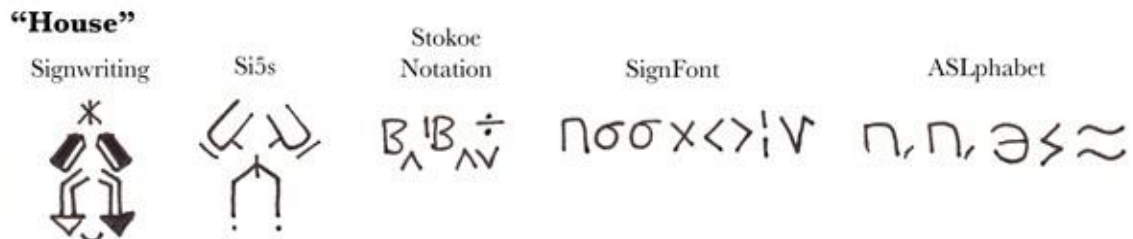


Figure 2. Examples of Different ASL Notation Styles. (Public domain.)

3.2.1.1. SignWriting

SignWriting is an iconic typography where each symbol represents the shape, direction, and orientation of the hands, face, and body. Each gesture is arranged within a "sign box", and these sign boxes are arranged from left to right. The order and the location of each symbol within each sign box begins with the facial expression at the top, followed by the relative orientation of each hand. Within each sign box, the starting gesture is located at the top and the final gesture at the bottom, with the middle gesture(s) signifying important intermediary gestures. Motion radicals are added to the hands and face to signify the major movements of the respective body part. SignWriting originated as a handwritten ASL notation style, as shown in Figure 2, but a digitized version of SignWriting is available as shown in Figure 3.



Figure 3. Example of a SignWriting Sign Box. (Public domain.)

A major drawback of SignWriting is the requirement of specialized characters that are not available in the ASCII library. Also, the arrangement of sign boxes requires a specialized program to create and share sentences.

As each gesture requires one sign box consisting of several symbols, and each sign box often represents only one or two words in the English equivalent, SignWriting requires a lot of space; likewise, each sign box would require a high data requirement for each word, making this typographical system ineffective in our project.

We considered representing the various aspects of the symbols in SignWriting using ASCII characters: the face would be the first encoded, then the left hand and right hand, then the motion. This encoding style would be similar to taking each radical of a Chinese character and using current ASCII characters in a certain order to represent the position of each radical in the Chinese character.

However, due to the complicatedness of this process, and the availability of other notation styles that already employ such linearization, we decided that this approach would be too time-consuming.

3.2.1.2 Si5s

Si5s is a simplified version of SignWriting; the simplification of SignWriting to si5s is similar to the transition between Traditional Chinese and Simplified Chinese, where the overall meaning of the word is preserved, but the simplified version is not as detailed as the traditional version. The radicals in this system give the same meaning as a SignWriting gesture but uses a simpler radical.

For example, the face is reduced only to its significant expressions; the movements are reduced to short lines or diacritics. Since this system is relatively simpler, more gestures can fit on one page; however, since this system still employs symbols not used in the extended ASCII library, this typographical system would be difficult to transcribe for use in the NMT.

Similar to the potential circumvention for SignWriting, we could derive an encoding method that represents the SignWriting symbols using ASCII characters; however, because of the complicatedness of the free-form placement of the symbols, even the use of ASCII

characters would render using this notation style difficult. An example of Si5s is shown in Figure 2.

3.2.1.3 SignFont

SignFont is a linearized transcription of ASL; whereas SignWriting and Si5s utilize blocks for gestures, SignFont writes gestures in the following order: handshape contact point for the right hand, handshape and contact point of the left hand, right hand location, left hand location, and movements. Other symbols are used as superscripts; that is, these act as diacritic modifiers for the superscript character. An example of a SignFont glyph is shown in Figure 2.

The major drawback for SignFont is the lack of support and information regarding this system: the explanation of the writing system and its grammar is not available. Another drawback, like the aforementioned systems, is the complicatedness of the symbols; very few of the characters are available in the ASCII library.

3.2.1.4 Stokoe Notation

Stokoe notation is a more linearized transcription of ASL, similar to SignFont. The most significant differentiation between Stokoe and SignFont notation is the use of Unicode characters in the notation. In this writing system, the order of the symbols are as follows: the location of the gesture, the shape of the left hand, the shape of the right hand, and the motion(s) as a superscript.

Motions done simultaneously are stacked, while motions done successively are placed adjacently. Subscripts are added to the handshapes to signify their orientation. Symbols are added between the two hand positions to signify their relative positions. An example of a handwritten form of Stokoe notation is shown in Figure 2.

The major drawback for Stokoe notation is its use for mapping single words only. Another drawback is the lack of non-manual transcriptions; only the hand gestures are mapped. This would pose a problem during the 3D model rendering, since there would be no information encoding for facial expression. Also, the use of superscripts and subscripts also poses a problem, since standard text files do not support the use of such; that is, the characters must progress on one line.

However, a method could be made in which the superscripts and subscripts are indicated using special characters while keeping the entire string on one line. A further drawback is the use of diacritics that would not be applicable to standard ASCII characters.

3.2.1.5 Gloss Notation

Gloss is a universal form of writing a certain language in another language known by the user. For example, someone learning German would write the direct translation of a sentence into English but using the grammar, syntax, and special rules of German. In this

Stokoe notation, however, since each diacritic modifier is applied only to a single character, we can represent these by placing the respective character after the character.

Because of the readily available Unicode library used by the Stokoe notation, the ease of linearizing its notation for use in the NMT, and the ease of including modifiers to each character, we decided that Stokoe notation would be preferred over ASL-Gloss notation. Although Stokoe notation omits the use of non-manual expressions, due to the flexibility of the notation we may be able to include non-manual expressions by appending its corresponding information to the end of each gesture.

Despite the removal of subscripts, superscripts, and diacritics, the potential drawbacks of using the Stokoe notation in its original form would be the ambiguity of the various symbols used in Stokoe. For example, the character “∅” refers to the empty space in front of the signer, but there is no differentiation between signs being done in front of the chest or in front of the face.

Likewise, there is no differentiation between how the hands are positioned: for example, the original Stokoe notation would suggest a handshape/orientation of “S^”, which implies the handshape “S” pointed upwards; however, there is no indication whether this upward orientation refers to the fingers or palms. Also, with the removal of subscripts and superscripts, there is a requirement to distinguish between symbols that are orientations (originally in the subscript), movements (originally in the superscript), or hand positions.

3.2.1.6 Extended Linear Stokoe (ELS) Notation

ELS is our proposed continuation of the Stokoe notation: the overall form of the notation is preserved. However, many new symbols are included to disambiguate between different hand orientations, spaces, and motions. The list of proposed alterations to the Stokoe notation can be found in Table 2 below. This notation would also linearize the original notation while maintaining the significance between subscript and superscript characters.

Symbols between two periods (e.g. “.⊥!”) signifies the hand position for the handshape indicated before it (called a period-pair), while those between two commas (e.g. “,>,”) signifies the hand motion for that handshape (called a comma-pair). The motion can be indicated for both hands by the use of the vertical bar character ‘|’ placed before the second handshape.

Another alteration of the Stokoe notation implemented in ELS is the disambiguation of the various spaces utilized in ASL. As aforementioned, the empty set character ‘∅’ is not sufficiently explanatory. In ELS, this character is placed before a location to signify using the space around that location; e.g. ∅} would signify the space around the ear. There are six regions which correspond to either the lateral or middle space of the face (upper), chest (middle), or abdomen (lower).

Disambiguation of the hand orientations is also achieved by the combination of identifying the palm and finger orientations. Within each period-pair are two symbols: the symbols

correspond to the palm orientation then the finger orientation, respectively. (For simplicity, the finger orientation refers to the direction the fingers point when extended fully.)

The complete list of orientations, movements, hand shapes, and locations used in ELS notation are shown in Tables 2, 3, 4, and 5, respectively. Symbols not found in the original Stokoe notation or those whose original meaning were altered from the original Stokoe notation are highlighted in yellow. Supination and pronation orientations from the original Stokoe notation were abandoned with its replacement with palm and finger orientations only. These antiquated symbol usages are grayed out in Table 2.

A comparison between the original Stokoe notation and ELS is shown in Figure 5. Note that in the original Stokoe notation, the chin space ('U') is notated, then a supinated hand in a 'B' shape with a prominent forearm ('√', 'B', and 'α', respectively), then a "downward" movement. This description of the first gesture alone is not descriptive enough: for example, the fingers can either be pointing up or to the side and have the same notation. This is disambiguated with the palm position towards the user ('T') and the fingers upward ('!') in ELS.

Likewise, the general downward movement should be a motion toward the chin of the user; this is disambiguated with a symbol for contacting the chin ("x[U]"). (Here, the character "U" is used instead of "u" for simplicity of typing.) Note the complete linearization of the ELS notation versus the superscript and subscript characters in the original Stokoe.

Also note the lack of spaces for a single gesture in ELS versus the use of separation of the different components in the original Stokoe. The removal of spaces is crucial for the computer to recognize and distinguish individual gestures (e.g. the "split" function in C). An example of a sentence in ELS is shown in Figure 6.



Figure 5. Comparison of Original Stokoe Notation (left) and ELS Notation (right)

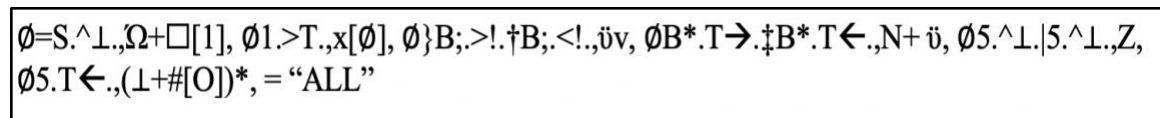


Figure 6. Example of a Sentence in ELS. Note that this sentence is the same as the ASL-Gloss sentence in Figure 4.

Table 2. Orientations used in ELS Notation.

Symbol Type	Char.	Description	Char.	Description	Char.	Description
ORIENTATION	a	Sup. wrist	T	F/P toward user	v	P down
	Ⓟ	Pro. wrist	⊥	F/P away from user	^	P up
	<	P left	>	P right	!	F up
	→	F right	←	F left	j	F down

Abbr. in Table 1: “F” → “fingers”, “P” → “palm” for orientations.

Table 3. Movements used in ELS Notation.

Symbol Type	Char.	Description	Char.	Description	Char.	Description
MOVEMENT	^	Upwards	N	Side-side	e	Wiggling fingers
	v	Downwards	>	Rightward	□	Opening of hand
	<	Leftward	Z	Up-down	t	Crossing of hands
	T	Towards self	⊥	Away from self	⊞	Linking of hands
	I	Back-forth	a	Wrist supination	÷	Divergence of hands
	Ⓟ	Wrist pronation	ω	Alt. sup/pro of wrist; “twist”	#	Closing of hand
	%	L. tilt of wrist	\$	R. tilt of wrist) (Convergence of hands
	ó	Elbow extension	Ω	Elbow flexion	x	Contact of hands
	ō	Elbow pro.	@	Lateral circulation	⊞	One hand inside another
	ü	Elbow sup.	ı	Sagittal circ.	‘	Alt. hand signs
	η	Wrist ext.*	&	Coronal circ.		
ή	Wrist flexion	ə	Hand waving			

Abbr. in Table 2: “sup.” → “supination”, “pro.” → “pronation”, “circ.” → “circulation”, “ext.” → “extension”, “alt.” → “alternation”.

Table 4. Hand Shapes used in ELS Notation.

Symbol Type	Char.	Description	Char.	Description	Char.	Description
HANDSHAPE	B	Flat hand, fingers together	A	Fist, thumb at side of palm	L	Fist, thumb and index at right angle
	S	“A” with thumb on fingers	1	“G” with thumb on fingers	K	“V” with thumb touching index and middle
	G	Fist, extend thumb and index in same direction	C	Curved hand	R	Cross index and middle
	5	All fingers spread apart	4	“5” but thumb on palm	3	Spread index, thumb, and middle
	V	Fist, extend index and middle	O	Curved fingers touching thumb	ϵ	“E” but fingers are looser
	D	“O” with index extended	F	Touch index and thumb, extend other fingers	Y	Pinky and thumb extended
	8	“F” but ring to thumb	7	“F” but middle to thumb	I	Fist, extend pinky
	X	Fist, index extended but bent	H	Index and middle extended and together, thumb touches both	W	Thumb holds pinky, other digits spread out
	E	Fingers and thumb folded in				

Table 5. Locations used in ELS Notation.

Symbol Type	Char.	Description	Char.	Description	Char.	Description
LOCATION	∅∩	Middle upper space	∅}	Lateral upper space	ω	Breasts
	∅	Center space*	∅=	Lateral mid. space	N	Flanks (side of abdomen)
	∅ε-	Middle lower space	∅N	Lateral lower space	∅	Pronated wrist
	○	Eye	Δ	Nose	√	Elbow/Forearm
	}	Ear]	Temple or Eyebrow	ε	Abdomen
)	Cheek	U	Chin	a	Supinated wrist
	∏	Neck	=	Shoulders	\	Upper arm

* When '∅' is used as a primary location, defined as center space in front of the user. When used as a destination, i.e. x[∅], indicates touching the chest (sternal area).

Table 6. Other Modifier Characters used in ELS Notation.

Symbol Type	Char.	Description	Char.	Description	Char.	Description
Relative Hand Positions	-	Left hand under right	+	Right hand under left		Parallel position and movement of both hands
	‡	One hand behind another	†	Crossed hands	~	One hand lags behind another
Indicator	·___·	For orientation	,___,	For motion		
Hand Shapes	;	Thumb next to palm	'''	Curled fingers	*	Ext. thumb
Motions	+	Simultaneous	*	Repeated		
Fingerspelling	= "___"	Spell letters within quotes				

Our proposed method of an ELS gesture follows the following format, which is also illustrated in Figures 7 and 8. One to two characters comprise the space in which the gesture

is performed. Afterwards, one handshape character represents one hand shape; this character can be followed by another modifier character to denote an alteration of the original hand shape.

For single-hand gestures, this character denotes the right-hand shape, as shown in Figure 7; for bimanual gestures, this character denotes the left-hand shape only, as shown in Figure 8. A period-pair follows the handshape character containing two characters. The first character denotes the direction in which the palm is pointing, while the second character denotes the direction of the fingers on that hand.

For simplicity, the finger orientation describes the direction pointed by the fingers when the fingers are fully extended. There must be only two characters within each period-pair. If two hands are used in a particular gesture, the right handshape character is given next. If there is a relationship between the two hands, another modifier character is placed before the second handshape.

Like the first handshape character, another handshape modifier character can be placed after this second handshape. Another period-pair after the second handshape denotes its orientation. At the end of each gesture string, a comma-pair denotes the motion of the hands. There can be any number of motions denoted between this comma pair; in Figure 7, this example shows two different, consecutive motions. The modifier before the second handshape character explains whether both hands move or only the right hand.

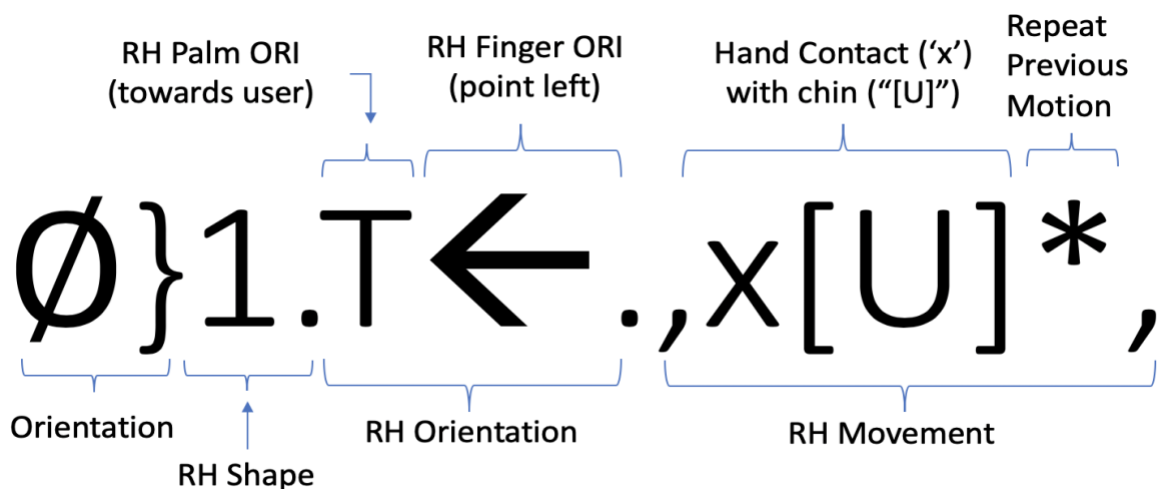


Figure 7. Break-Down Diagram of a Single-Handed ELS Gesture.

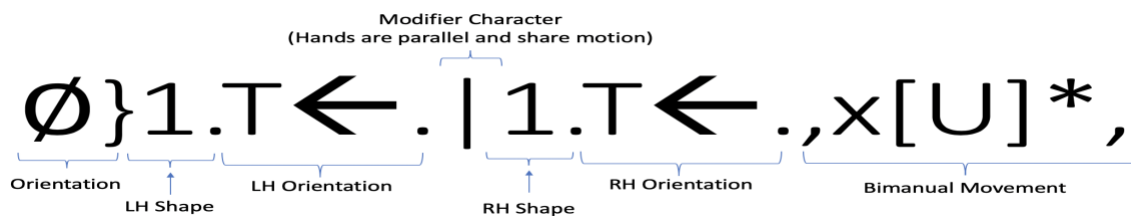


Figure 8. Break-Down Diagram of a Bimanual ELS Gesture.

As of the final project design, this process of encoding ELS into the final code was considered too costly and overgeneralized for the scope of the number of gestures used in this project. However, this notation would definitely have been considered in the future in the sense of scaling up the project.

3.2.2 Machine Learning

This section of the report focuses on the utilization of machine learning techniques for the entire process once the user speaks into the microphone. The process begins when the system recognizes the input voice as its digital and discrete version, then uses machine learning to perform speech to text processing. Once the speech is processed into text, the machine translation algorithm is employed to translate the input source text into the target text. The output target text is in the language of choice, and this process is referred to as text-to-text translation. Finally, the translated text will then have to be processed again to turn the text back into speech using the final machine learning technique, called speech-to-text. These three steps, when done in succession, define the methodology for the machine translation process. For each of these techniques, there are various approaches and algorithms that can be utilized depending on the application. There are open-source toolkits and APIs that can be applied to perform the various machine learning techniques; alternatively, there are proprietary APIs that are offered by certain companies and offer their services through a paid subscription. In the following subsections a closer look is given at the various options in terms of approach and various programs, both open-source and proprietary, that have been developed as solutions for each step in the machine translation process.

3.2.2.1. Speech to Text

Speech-to-text, or speech recognition, is vital in obtaining the most accurate representation of the voice input as text. The basic component to the speech recognition systems are statistical models with the representation of the distinct sounds pertaining to the language, or phonemes, that need to be recognized. There are two models associated with speech recognition, an acoustic model and language model. Furthermore, there is not a universal method in obtaining the best speech recognition, as the models are tweaked for any given language or application. The specialization of these models for any language is what allows speech recognition to become accurate for converting the spoken words into text. The errors that arise from speech recognition are attributed to the speaker, style of speech, and the environment therein. Therefore, it is critical to select the most suitable model available for use in speech recognition to ensure that the user receives the most accurate intended translation. The following are the more popular and most used models for speech recognition, each model delivers and lack certain features. Figure 9 demonstrates a basic block diagram of a speech recognition system. This diagram depicts the process starting from the preprocessing stage after the signal is retrieved and ends with the language modeling.

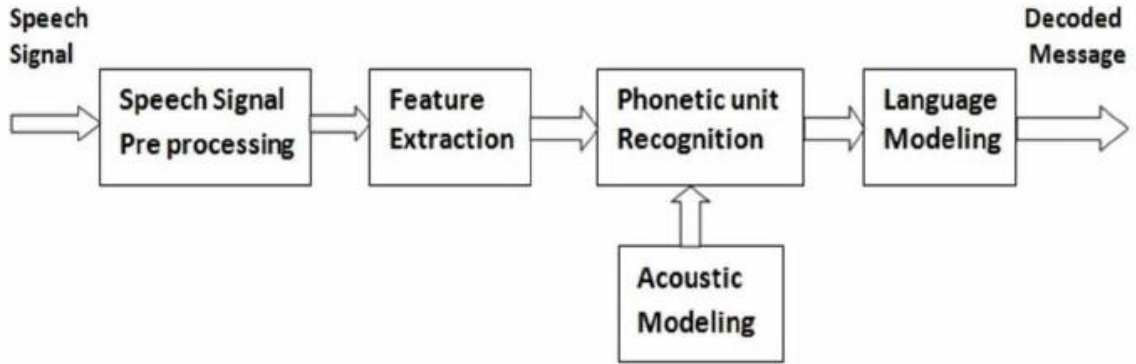


Figure 9. Example of a basic speech recognition system. (Public domain.)

3.2.2.1.1. Hidden Markov Models (HMMs)

HMMs are based on statistical models that assign symbols or quantities for the output as shown in the Figure 10. HMMs administer a much simpler groundwork for the modeling of time-varying spectral vector sequences; as a result, most continuous speech-recognition systems are based on HMMs. Moreover, the popularity of HMMs for speech recognition stems from the statistical framework, availability of training algorithms, flexibility of fine-tuning parameters specific to words, and ease of application. Some of the issues that affect the usability of HMMs involve the intensive memory and computation consumption, the large training data, the time to execute the training, difficulty in choosing the correct seed sequence.

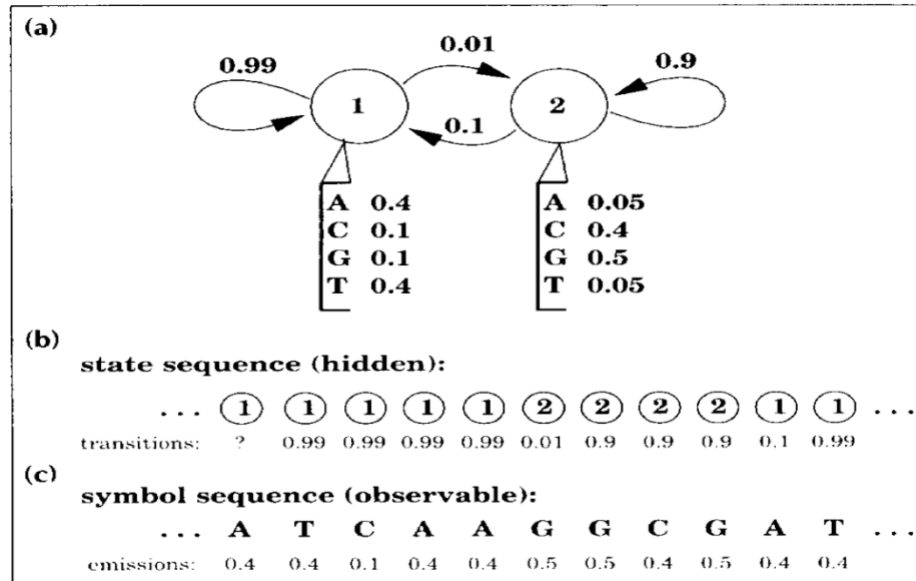


Figure 10. Example of the statistical model (a) Hidden Markov Model with (b) state sequence and (c) symbol sequence. (Public domain.)

3.2.2.1.2. Dynamic Time Warping (DTW)

Based on HMMs and neural networks (NNs), DTW can be utilized for its facile implementation with embedded systems; this is due to its simple hardware integration and

training speed. DTW is based on the mechanism of template matching, in which classic DTW utilizes one template for each word recognized. In this process the DTW compares the parameters of an unknown word with the template; with this, increasing the templates for that particular word would result in improved recognition of that word in speech recognition. However, increasing the number of available templates leads to a longer computation time and memory usage. In Figure 11, an example of the DTW matching the alignment of two sequences is shown.

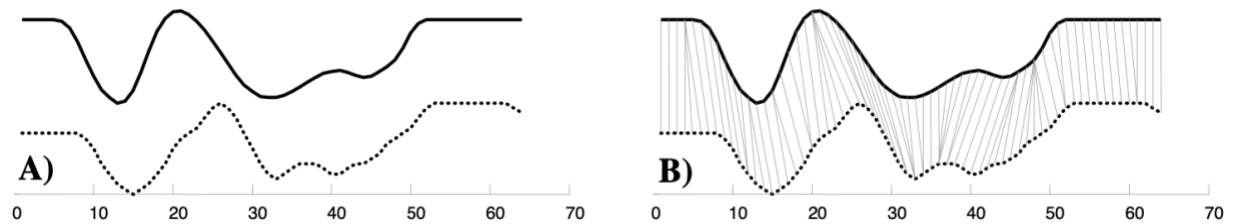


Figure 11. Example of Dynamic Time Warping for (A) Two Sequence Representation of a Person Spelling out “Pen” in Sign Language, (B) with the Dynamic Time Warping Finding the Proper Alignment. (Public domain.)

3.2.2.1.3. Deep Neural Networks (DNNs)

A much newer approach to speech-to-speech recognition is the use of artificial neural networks. In comparison to HMMs, neural networks allow for discriminative training to be more efficient and can be utilized to provide preprocessing for continuous speech signals. Neural networks have demonstrated better results in terms of speech recognition. An example is the MAVIS system developed by Microsoft which yielded a reduction in error by 30%. For speech recognition, neural network algorithms are trained similarly to other applications. During the training of the neural networks the system finds features by adjusting the weights of its hidden layers, and the system uses the features extracted as inputs to the next layer.

3.2.2.1.4 End-to-End Automatic

This approach also uses deep neural networks; however, it provides a simplified model, joint training, direct output which presents a contrast to HMMs that utilize direct deep neural networks (DNNs). This approach detects an input sequence as a corresponding label sequence, and the audio is mapped to characters or words. Since this process relies entirely on DNNs, it does not require expertise in domains to construct a large training database, thus allowing for training to be much easier.

3.2.2.1.5 Open-Source Speech-to-Text Algorithms

CMU Sphinx is an example of an open-source platform for utilizing the HMM approach. This program can be used on low-resource platforms, has flexible design for practical development, a large language support database (English, UK English, Mandarin, German, Dutch, Russian), a BSD license which allows for commercial distribution, and an active development and release schedule.

Julius is another example of an open-source platform for utilizing the HMM N-gram approach. The N-gram approach refers to the number of words used as a single entity when passed through the machine translation algorithm, as explained in Section 4.1.7. Julius offers a real-time, high-speed, accurate recognition through its use of a two-pass strategy. This program utilizes less than 32MB for its workspace, supports rule-based grammar, isolated word recognition, and is highly configurable with various search parameters and alternate decoding algorithms.

Kaldi is an example of an open-source platform for utilizing the DNN approach. This program provides code-level integration with Finite State Transducers (FSTs), extensive linear algebra support, extensible design, and its intended use is for researchers, as the source code for Kaldi is written in C++.

Project DeepSpeech is an open-source speech-to-text engine which uses end-to-end deep learning and Google's TensorFlow software library to simplify its implementation. This program currently only supports 16-bit, 16-kHz, mono-channel WAV audio files. The model needs to be trained for its target language; however, a pre-trained English model is available on GitHub.

Wav2letter++ is an open-source speech processing toolkit for end-to-end speech recognition developed by the Natural Language and Speech research team at Facebook AI Research. The source code is C++-based and utilizes the ArrayFire tensor software library, and flashlight machine learning library. This toolkit also must be trained by the developer. The design is intended to operate effectively when trained with thousands of hours of voice. Facebook states that Wav2letter++ is the first fully convolutional speech recognition speech, utilizing convolutional layers for the learning aspect of the architecture. The architecture for the Wav2letter++ is demonstrated in Figure 12.

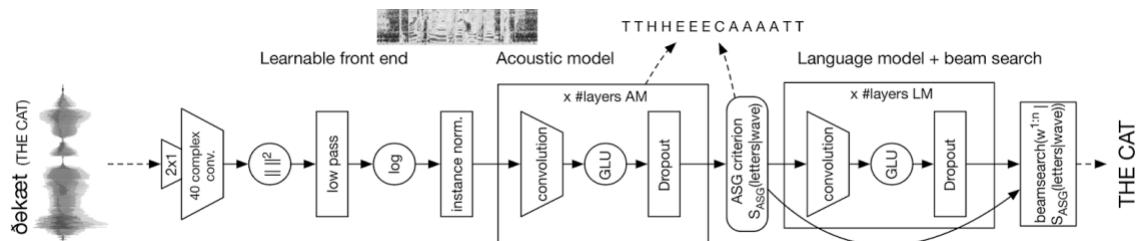


Figure 12. Wav2letter++ Architecture using Convolutional Layers. (Public domain.)

DeepSpeech2 is another open-source speech recognition model that is based on Baidu. The model takes the input waveform, preprocesses it, converts it into a log-spectrogram, which is then applied with the convolutional layers for training and verification. The model utilizes a Word Error Rate (WER) as its main evaluation metric, and the model is trained with a stochastic gradient descent. The architecture for the DeepSpeech2 is demonstrated in Figure 13.

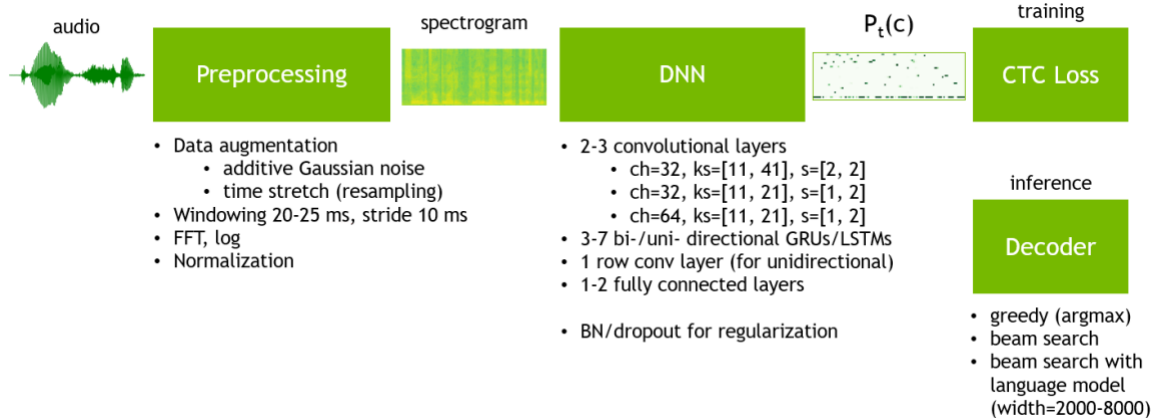


Figure 13. DeepSpeech2 Model using the Deep Neural Network Approach which includes Convolutional Layers during Training. (Public domain.)

3.2.2.1.6 Proprietary Speech-to-Text Algorithms

Hidden Markov Model (HTK) is a free-to-use platform which utilizes the HMM approach. HTK is primarily used for speech recognition; however, it has been used in other applications such as speech synthesis research, character recognition and DNA sequencing. The software provides proper speech analysis, HMM training, testing, results, and it supports the HMMs using both continuous density mixture Gaussians and discrete distributions.

RWTH ASR is another free-to-use platform for speech recognition decoding with the tools necessary for acoustic models. This software consists of several libraries written in C++ and has a decoder for a large vocabulary, feature extractions, acoustic modeling, speaker adaptation, lattice processing, language modeling, and input/output formats. The newer versions of this software also feature neural networks.

Dialog Flow is a speech-to-text software that utilizes Google's machine learning algorithms and Google Cloud. This program is intended to be used as a browser extension, allowing its use on multiple platforms such as Facebook, Skype, Twitter, and other text-centered websites. Dialog Flow supports more than 20 languages.

Dragon Dictation is another speech-to-text software that utilizes a DNN speech engine to detect voice accurately while filtering background noise, accent, and voice distortion through extreme (soft/loud) volumes. This program is used primarily for dictating, editing and formatting text on several word-processing applications such as Microsoft Office.

Google Cloud Speech-to-Text is paid-subscription API developed by Google for voice recognition. This software utilizes neural network models, recognizes 120 languages, and can process real-time or pre-recorded audio. The user can customize the speech recognition to specify common words used in a designated application. The API supports automatic language detection and automatic punctuation.

Microsoft Azure Speech-to-Text is another paid-subscription API developed by Microsoft for voice recognition. This software utilizes the same voice recognition technology that is used for Cortana, but with a more limited library of available languages (compared to Google). The interface allows for user-training of acoustic, language, and pronunciation models for a unique environment and distinct vocabulary. Voice input can be transcribed directly from a microphone or from stored audio files.

Amazon Transcribe is another paid-subscription API developed by Amazon for voice recognition. This API is capable of transcribing stored audio files or live audio stream in real time. The service transcribes WAV and MP3 files, with the transcriptions having an added time stamp to make locating dialogue simpler. The algorithm utilizes neural network models to include punctuation and formatting to the transcribed text. The program also allows for a user to include a custom vocabulary; that is, to add commonly used words for each user. The voice recognition algorithm is capable of differentiating a change in speaker and will transcribe accordingly.

3.2.2.2 Text to Text Translation

The text generated from the processed speech input (with the output being the text from the input speech-to-text algorithm) is directly translated into the equivalent representation in another language such as Spanish, Chinese, ASL, etc. For proper execution of the translation, the system must be able to recognize the context of the sentence or paragraphs. For example, translation word-by-word would not be sufficient to produce an accurate translation. Thus, we decided that machine learning will be required to perform the complex task of text-to-text translation.

The use of machine learning-based software to compute the translation of source text to target text is termed “machine translation”. Machine translation brings together linguistics, computer science, artificial intelligence, translation theory, and statistics to provide translation between two different languages [V]. Within machine translation, there are several different approaches with each approach providing a different algorithm for achieving the translation. The most common approaches are rule-based, statistical, and neural machine translation. Each machine learning method aims to translate the source sentence into a target sentence while accounting for the necessary context of the original sentence to produce an accurate translation. In other words, the meaning of the input text in one must be fully realized in the output text in another language [W]. The following approaches were compared to determine which approach would best suit the needs for the project.

3.2.2.2.1. Rule-Based Machine Translation (RBMT)

The rule-based machine translation (RBMT) is a system composed of a large database of linguistic rules and millions of dictionary entries in multiple languages. Therefore, it is a database that produces these rules and provides the linguistic resources between the two languages of focus. While RBMT allows for users to follow the linguistic rules used in the translator, this processing is deemed to be very time-consuming during implementation

and debugging. Since these translations are also provided by user-defined rules, the text may be translated completely but may lack fluency. An example of a hybrid rule-based machine translation architecture is shown in Figure 14, which is a hybrid RBMT utilizing Moses.

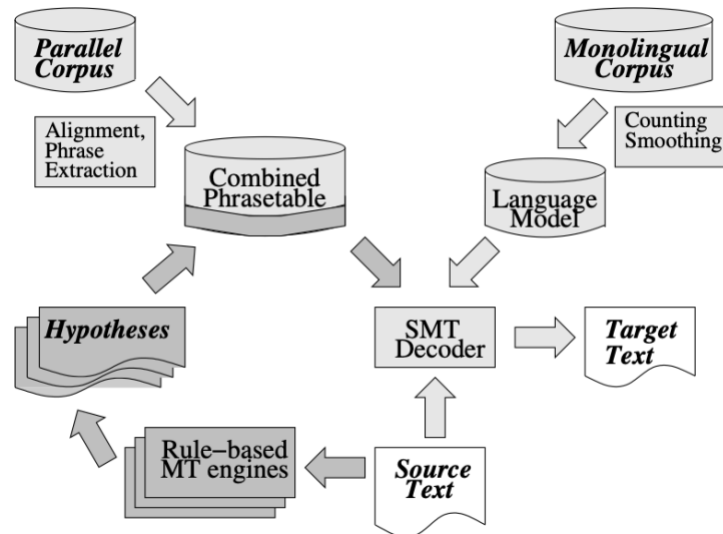


Figure 14. Example of hybrid rule-based machine translation architecture. (Public domain.)

The process of producing fluent translations requires the use of post-processing techniques to refine the translated text. Attempts to mitigate these issues is the use of different hybrids between RBMT- and SMT-based translation systems. In a paper by Dove et al., they used the RBMT output as the standard translation and subsequently refined the translated text by comparing it to a translated model developed through SMT techniques. Overall, to maximize the use of RBMTs for newly discovered languages or for developing new language translation pairs, the quality of the RBMT method will need to be improved. For the use of RBMT in text to ASL text translation, a massive database with the rules and resources will need to be created. However, even with the necessary extensive database, an RBMT can run successfully on ideal hardware.

3.2.2.2.2. Statistical Machine Translation

The statistical machine translation (SMT) system utilizes statistical models with the framework based on the analytical observation of bilingual text corpora. While the process of RBMT is word-based, SMT processes phrases for its translation as well as overlapping phrases during its training. An extensive database is required for SMT with a minimum of 2 million words required for successful translation for a specific language domain and even more for creating a general language translation. SMT requires extensive CPU usage even for simple translations. An example of a statistical machine translation representation is demonstrated in Figure 15.

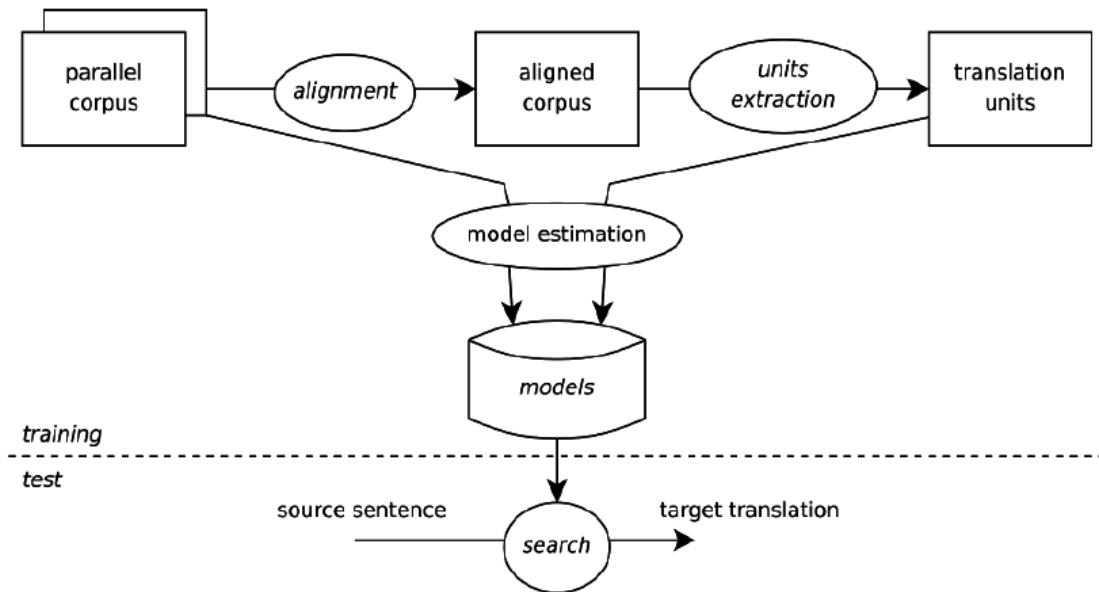
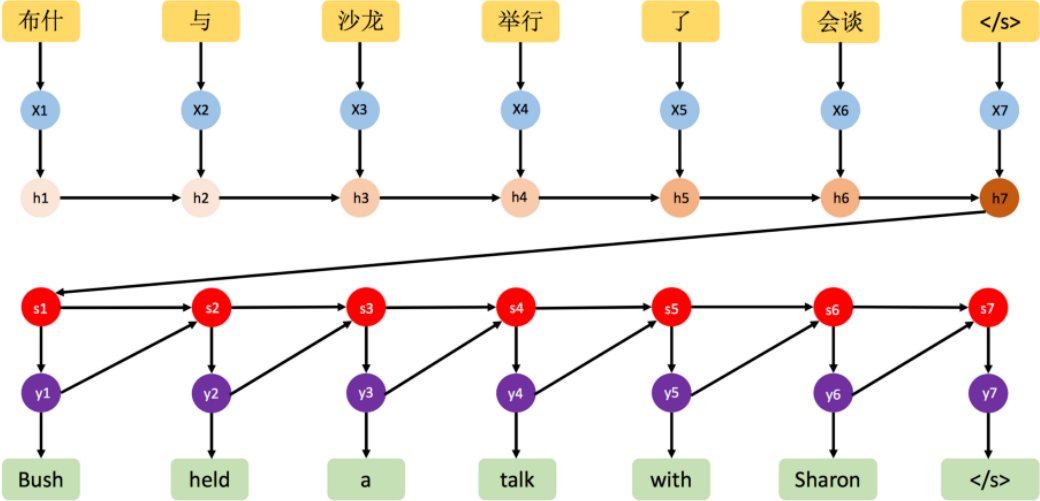


Figure 15. Example of basic statistical machine translation. (Public domain.)

3.2.2.2.3. Neural Machine Translation (NMT)

Compared with the recently mentioned approaches, RBMT and SMT, a more modern approach to machine translation is neural machine translation (NMT). This approach has driven machine translation to exceptional improvements, considerably in human assessment. In comparison to SMT, the NMT approach produces a single neural network that is tuned through the training process, which differs from how SMT works with various models. NMT handles its interaction with input sentences through the use of vectors; that is, input sentences are defined by their magnitude and direction. This approach ultimately leads to a more streamlined algorithm. Moreover, NMT utilizes a bidirectional recurrent neural network (encoder) that processes the sentence into the vector, and then utilizes another recurrent neural network (decoder) to develop the word prediction.

NMT was based on the development of sequence-to-sequence models and then combined with the advancement in attention-based variants. Similar to the previously aforementioned approaches to machine translation, NMT requires a large training set to produce accurate predictions. These datasets are more straightforward to compose since the only contents in these dataset corpora are the source text in one language and the target text in the other language. The NMT will then tweak its weights to predict the word that was used in the translation. As a result, much of the language-translation industry, including Google, has moved towards the usage of NMT for their online translators. A representative example of the neural network used in NMTs is demonstrated in Figure 16.



(Sutskever et al., 2014)

Figure 16. Example of the Neural network used in Neural Machine Translation. (Public domain.)

3.2.2.3.1 Open-Source Text-to-Text Translation Algorithms

Apertium is an open-source platform for utilizing the rule-based machine translation approach. This program is being applied extensively to machine translation systems for various language pairs.

Moses is another open-source toolkit which utilizes the SMT approach and supports linguistically motivated factors, confusion network decoding, and efficient data formats for translation and language models. It allows for the user to automatically train the system with various translation models for the desired language pairs. The algorithm then finds the translation with the highest probability utilizing a search system. Moses offers phrase-based, and tree-based for the translation models. It also features a factored translation model that allows the integration linguistic at the word level. Furthermore, it allows for the decoding of confusion networks and word lattices, this enables speech recognizers or morphological analyzers. Figure 17 shows the diagram of Moses’s factored translation model.

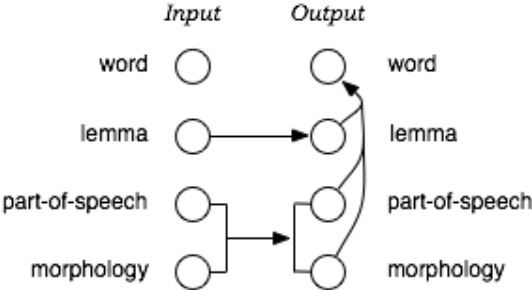


Figure 17. Diagram depicting Moses’s factored translation model approach. (Public domain.)

Apache Joshua Home is a Java-based, open-source toolkit that utilizes the SMT approach, allowing for phrase-based, hierarchical, and syntax-based machine translation.

Great is an open-source toolkit which utilizes the SMT approach, is based on the bilingual language modeling approach, and offers phrase-based translation while utilizing a reduced amount of model parameters and decreased response time in comparison to Moses.

OpenNMT is an open-source toolkit which utilizes the NMT approach. This toolkit emphasizes efficiency, modularity, and extensibility. OpenNMT utilizes neural machine translation as the basis model to deliver the most optimum translation through its training from large datasets. When first realized, OpenNMT was utilized for standard sequence to sequence modeling applied to machine translation; however, it has now been developed to employ additional models and features. These features include image to text, language modeling, sequence classification, sequence tagging, sequence to sequence, speech to text, and summarization. The models included with OpenNMT are convs2s, deepspeech2, gpt-2, im2text, listen attend and spell, RNN with attention, transformer. Depending on which OpenNMT implementation used, either with PyTorch or with tensor flow, certain features will be available. Furthermore, the toolkits mentioned (PyTorch, TensorFlow) provide various configurable models and features that include other types of generation such as summarization, speech to text, and image to text. The OpenNMT also includes CTranslate2, which is a custom inference engine that explores model quantization, it also includes Tokenizer that allows for a fast and customizable tokenization library. According to the developers, OpenNMT is able to reach high translation quality and speech that matches with the other translation competitors and they provide full support. Figure 18 demonstrates a simplified diagram of how the neural network of OpenNMT functions.

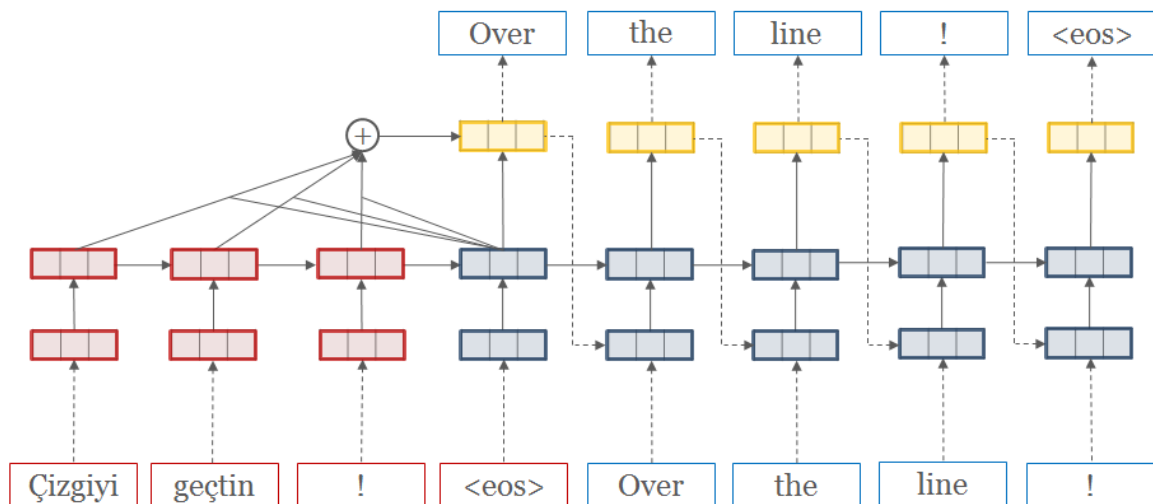


Figure 18. Neural network algorithm representative of the OpenNMT model. (Public domain.)

To train the OpenNMT model, the user needs to utilize the source file and the target file. These files contain sentences per line with each word of the sentence separated by a space. It is recommended to find these files online that are open source and have built on to include

more than millions of sentences. When training the OpenNMT with these datasets, it is recommended to use a GPU to process the algorithm; otherwise, a CPU will not be able to quickly process it and therefore greatly prolong the process.

NiuTrans is an open source translation system that utilizes the statistical machine translation approach. This translation system is fully developed using C++ language, and it's advertised to run fast and use less memory. NiuTrans currently supports phrase-based, hierarchical phrase-based and syntax-based models that is used for research-based purposes. Other features that it provides include multi-thread support, APIs, compact n-gram language models, multiple SMT models.

OpenLogo is an open source machine translation system that is based on the rule-driven machine translation system, as well as the syntactic-semantic taxonomy SAL. The architecture of the system is that of a pipeline, that allows for the modularized, and incremental approach to the source language analysis and the synthesis of the target language.

The input to the system for OpenLogo, with the stream and rules, are based on SAL and therefore the interactions throughout the pipeline are in terms of the SAL pattern. This allows for the more efficient surmounting of the issue that occurs with the large rule-based dataset. The languages that are included with OpenLogo includes English, German, French, Spanish, Italian, and Portuguese. The pipeline architecture of the OpenLogo system is demonstrated in Figure 19.

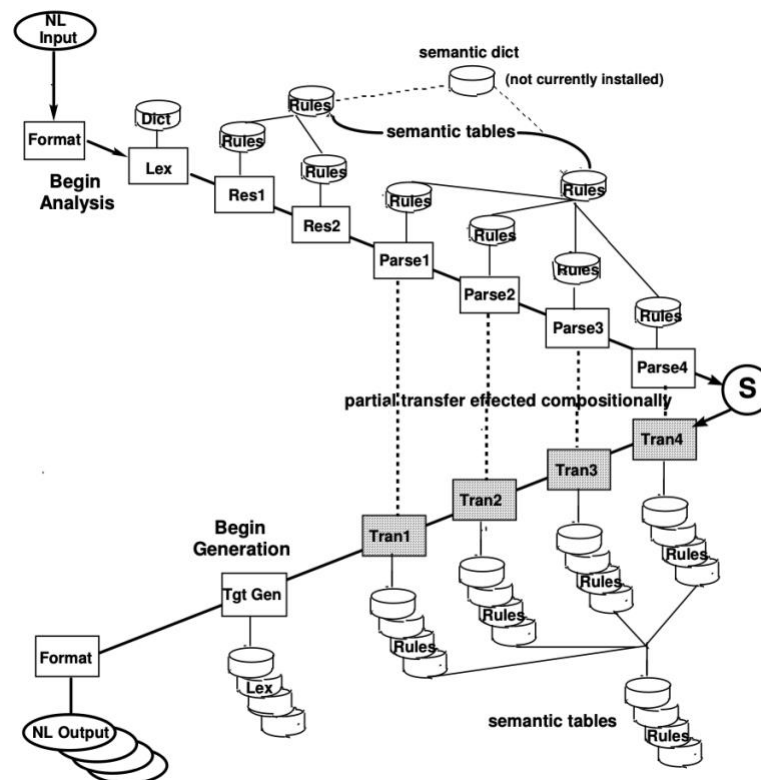


Figure 19. The pipeline architecture of the OpenLogos system. (Public domain.)

3.2.2.3.2 Proprietary Algorithms

IBM Watson language translator is IBM's proprietary machine translation that is available through APIs. This translator identifies the language of the input text and completely translates it through the use of the available translation models. It also provides the user the ability to create custom models that are helpful when using the translation for specific languages or applications. It is based on neural machine translation, allowing the translation to be much faster with improved accuracy. Allows the user to translate entire documents while preserving the formatting and type of file. The custom models that it allows users to manipulate are done through the use of forced glossary or parallel corpus.

Google Translate is Google's proprietary machine translation that is available through APIs. Google states that the translator dynamically translates between thousands of available language pairs. The system utilizes Google's pre-trained and allows user to create custom learning models. This customization is made at a level that is facile for developers, translators, and localization experts with limited expertise to quickly develop production-ready models. This is accomplished through uploading translated language pairs, and the AutoML Translation then trains the custom model that allows the user flexibility when in specific domains. The diagram depicting the AutoML functionality of Google Translate is shown in Figure 20.

Google Translate also offers an advanced API that provides much faster and dynamic results in contrast with the Basic API, and it also provides additional features that support further customization. The Basic version of the API allows for the instant translation and uses the neural machine translation that Google pre-trains.

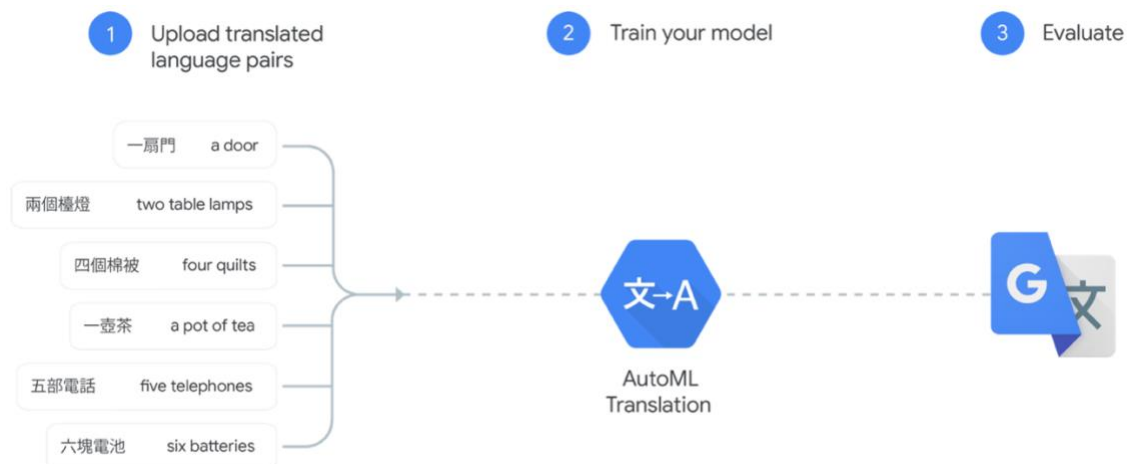


Figure 20. Diagram depicting how Google's AutoML Translation performs the custom training for the user. (Public domain.)

Microsoft Translator Text API is Microsoft's proprietary machine translation system, a cloud-based service that features more than 60 languages. It is branded as a proven, customizable, and scalable technology for the use of machine translation in translating text.

This is the same translation model that Microsoft utilizes for its other software such as Microsoft Office, Edge, SharePoint, Yammer, etc. The API also comes with additional features such as transliteration and bilingual dictionary that aid the functionality of multilingual apps and workflows.

Some of its other main API features include natively neural, translating multiple languages at once, and translation detection. Natively neural means that the system is based on neural machine translation (what most modern systems use) and this allows it to provide higher quality translation. The transliteration feature converts words and sentences from a specific script to another, and the bilingual dictionary provides alternative translations from or to English.

Amazon Translate is Amazon's proprietary machine translation API that utilizes neural machine translation to deliver the fast, and high-quality language translation. It utilizes deep learning models to provide a more accurate translation and translation that is more natural in contrast to rule-based and statistical algorithms. A of the features that Amazon Translate includes is the easy integration of the system into applications, that allows users to use the system to perform the real time translation within their application by simply calling the API.

Another feature is the customization of the terminology for specific domain texts, and it also has a scalable feature that allows for large volumes of text. Additional usage of this system is for enabling multilingual sentiment analysis of social media content, providing on demand translation of user generated content, and added real time translation for communications applications.

Yandex Translate is a proprietary machine translation system that utilizes statistical machine translation. Therefore, the system compares thousands of texts in order to learn the language and develops the system so that it is able to identify parallel texts by their web address. Because this is based on statistical machine translation, the system is much less accurate and efficient than those based on neural machine translation. Nonetheless, this system requires hundreds of millions of phrases in different languages to obtain quality translation standards.

The three key components of this machine translation system are the translation model, language model and the decoder. Within the language model is the list of all the words and phrases that are known to the system for one of the languages, these are also linked to the possible translations and there is a probability assigned to each. The decoder of the translation system then performs the actual translation through the selection of the translation options, combination of phrases, and sorting thereafter.

SYSTRAN is a proprietary machine translation system that allows the user to translate both structured and unstructured multilingual content. These contents include user generated content, social media, web content, and others. It is branded as being scalable and reliable, providing the best-of-breed language processing technologies to apps and websites. This system utilizes SYSTRAN machine translation, with natural language

processing. The natural language processing includes language identification, named entity recognition, segmentation and tokenization, transcription, morphological analysis, and part of speech tagging.

The named entity recognition is a neat feature that recognizes and displays the person's name, locations, numbers, dates organization names, etc. The translation system also provides resource management that includes dictionary management, dictionary lookup, corpus management, and corpus match. Furthermore, the system includes multimodal text extraction that includes text extraction, speech recognition, and optical character recognition.

GramsTrans is a proprietary machine translation system that allows for high quality and domain-independent machine translation for the Scandinavian languages. The product set that is offered with GramsTrans are the free version, personal, commercial lite, commercial standard, and schools. The free version allows for translation with the limitation of 70 words maximum, 10 translations, and not for commercial use. The personal version includes a user subscription that allows for the translation of unlimited amounts of text.

The main approach that GramsTrans utilizes is rule-based for the translation. The core linguistic features that are included are robust source language analysis, morphological and semantic disambiguation, large linguist-made grammars and lexica, high degree of domain-independence, name recognition and separation, dependency formalism for deep syntactic analysis, context-sensitive selection of translation equivalents, insertion, deletion, and splitting of words, word and phrase reordering, and terminology customization. Some of the acceptable data types include plain text, formatted documents, web pages, movie subtitles, arbitrary XML structures, SMS, WAP (mobile protocols) and remote API access.

Prompt is a proprietary machine translation system that is based on advanced neural machine translation approach. This translation system also includes confidential translation, more than 25 languages, high-speed translation, customization, and integration. The features of this machine translation product include the translation of documents (preserves the formatting and allows for various file types), allows for immediate translation through word selection, user tools that allow for the customization of the system for domain specific texts, and integrated reference resources.

Babylon is a proprietary machine translation system that is free software, it allows for the automatic translation of single words, full texts, phrases, and other features. It features over 77 languages, over 1,700 dictionaries, glossaries, thesauri, encyclopedias, and lexicons that cover a wide range of subjects. Furthermore, it has the capability to translate single words, and sentences.

IdiomaX is a proprietary machine translation system that provides several features and tools for businesses but has a limited amount of languages. This software works closely with word processing applications in order to translate the document immediately before emailing. It also has the ability to translate webpages.

3.2.2.3 Text-to-Speech

The final technique in the translation process is text-to-speech, also known as speech synthesis, shown in Figure 21. This technique converts the translated text into human speech, allowing the user to listen to the resulting translation. The process of human created speech can be modeled using the source-filter model and then applied in more complex models. Therefore, the synthesis of speech can be accomplished digitally through the linear prediction model and concatenative speech synthesis.

The linear prediction model is currently the most utilized when it comes to digitally synthesized speech. The modeling of time series for the speech signal allows for the approximation of short-time speech spectra. Therefore, the poles are considered to be related to the speech spectral representation and it has led to extensive studies for modeling speech. Ultimately, the speech signal is approximated through the linear combination of the previous samples with a predicted error.

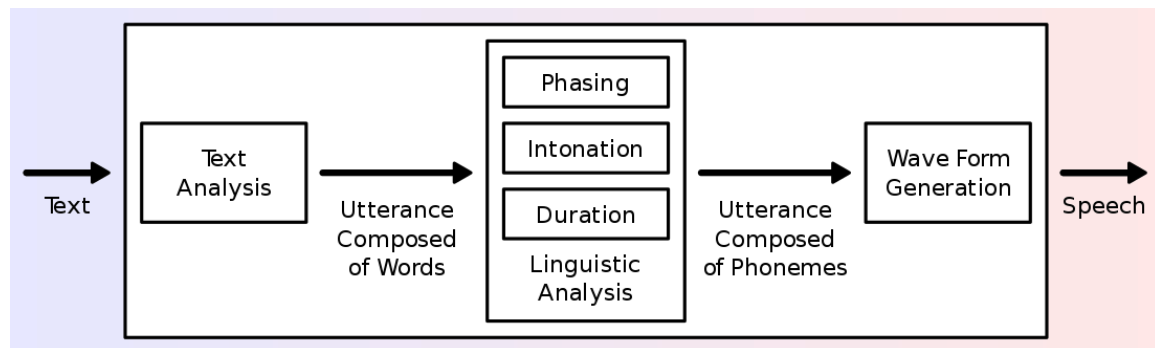


Figure 21. General Representation of a Text-to-Speech system. (Public domain.)

The concatenative speech synthesis accomplishes the generation of speech from text by associating pronunciation models to text through analysis and does this through the segmentation of real speech into units. The synthesis parameters from the units are taken and concatenated according to the pronunciation specific to each text. This approach, however, depends on the use of pre-recorded speech material to be available. The following are other approach synthesizes that need to be considered when deciding open source or proprietary speech-to-text toolkits.

The Formant synthesis is a source-filter based model where the source is modeled after the glottal pulse train and the filter is modeled after the formant resonances of the vocal tract. The formant model also utilizes the linear prediction model mentioned to estimate the parameters but could also use the short-time spectrum for the estimation. Therefore, it is typically used in cascade or parallel second order filter sections, and the model is mostly used by rule-based text to speech systems.

The Articulatory synthesis provides an alternative approach that provides the direct simulation of speech production principles. In contrast to the formant synthesizer, the articulatory approach produces the characteristics of the vocal tract filter through the implementation of a vocal tract geometry and manage potential sound sources throughout

the geometry. This approach has huge potential; however, it is difficult to understand and to implement.

The HMM approach is a statistical time series model used for speech-to-text has also been recently applied to text-to-speech. The model is a finite state machine that creates sequences at discrete time, with each time altering according to a state transition probability and then generates the observed data. This is applied to the speech generation by statistically modeling the spectrum, fundamental frequency, and phoneme duration which are the speech parameters, the HMM's maximum likelihood criterion then generates these models. Therefore, for this approach a sequence of HMM parameters are used to model the transitions of sound, and it results in a smoother output in comparison with concatenative synthesis. Utilizing a large collection of data sets that contain speech, the HMM can more easily estimate these parameters.

The sinewave synthesis approach models the speech signal with three or four time-varying sinusoids. The goal of utilizing the sinusoids is to clone the estimated frequency and amplitude pattern associated with the resonance peaks in order to create the synthetic acoustic. While there are current numerical methods used to compute the frequency and amplitude values, there are estimate errors that need correction for the actual synthesis parameters.

In contrast to the previously mentioned approaches to speech synthesis, the DNNs utilize the use of large recorded data training sets to adjust weights and provide high accuracy output. There have been challenges in using DNNs for speech synthesis, however, there have been many successful implementations as demonstrated below.

WaveNet is based on the DNN approach, the model is probabilistic based and autoregressive with the prediction depending on the previous audio sample condition. Therefore, WaveNet is made with stacks of convolutional layers that outputs a waveform as shown in Figure 22 below.

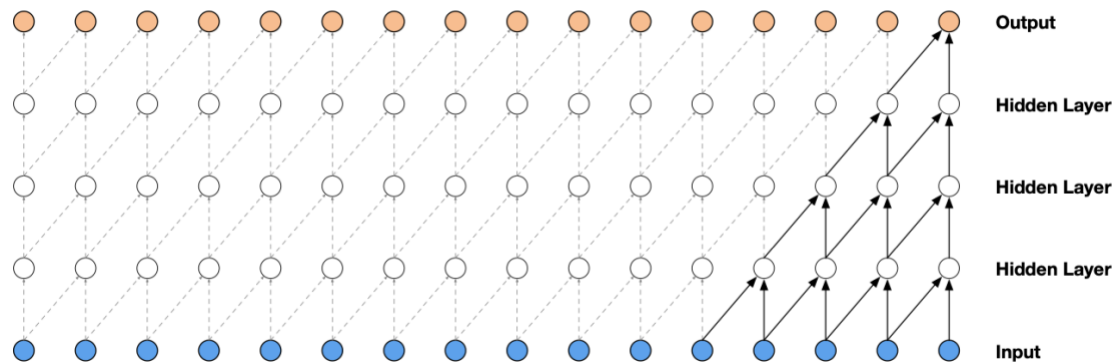


Figure 22. Diagram of Causal Convolutional Layers. (Public domain.)

The model was also conditioned by WaveNet team, providing it with vocoder parameters that allowed the resulting speech to be natural and high quality. However, this method of

utilizing convolutional layers led to extensive computation and it was reported that to generate one second of audio it would take four minutes of computation.

3.2.2.3.1 Open Source Text-to-Speech Algorithms

The MARY Text-to-Speech System (MaryTTS) is an open-source platform capable of producing multiple languages and is written in Java. The structure of the MaryTTS relies on the preprocessing, natural language processing, the calculation of acoustic parameters, and the synthesis when the parameters are converted to audio.

eSpeak Text-to-Speech System is another open-source platform that utilizes the formant synthesis that was mentioned previously. The developers of this program claim to provide many languages in a smaller size, and the output speech is clearer but lacks the natural aspect of human speech. Features include difference voices, production WAV files, file compression, conversion of text to phonemes, and developmental tools available for tuning phoneme data. This program is written in C language.

The Festival Speech Synthesis System is another open-source platform that offers text-to-speech in English and Spanish, is written in C++, and utilizes the Edinburgh Speech Tools library for the low-level architecture with a scheme-based command interpreter.

3.2.2.3.2 Proprietary Text-to-Speech Algorithms

Google Text-to-Speech is Google's prime API for text-to-speech system that utilizes WaveNet voices and Google's neural networks. It allows for human-like speech in 180 voices, with 30 languages available. The development of this program into an API allows for its use in devices that can deliver a REST or gRPC request, including IOT devices. Furthermore, the API allows users to customize the speech with SSML tags, tune the speaking rate, tune the pitch/inflection of the voice, adjust the volume, and customize audio profiles.

Microsoft Azure Text-to-Speech is Microsoft's API system that utilizes neural text-to-speech to produce highly natural speech. This API allows the user to produce the audio in real time and allows the option to save the audio files. Neural and standard versions are available. The standard text-to-speech provides 75 voices and 45 languages.

Amazon Polly is Amazon's API for text-to-speech that also utilizes neural text-to-speech to deliver more natural voices. It provides a dozen voices and eight languages.

3.2.3 Processor Technologies

When starting this project, we are deciding what we would use as the brain of our unit. We need a piece of technology that would allow the device to perform many different tasks and that would do these tasks in a real-time, spoken conversation setting. Since we had chosen to offload much of the spoken language translation to cloud processing, we would need something that would allow us to interface with a Wi-Fi module. Furthermore, we

needed the technology to allow us to run a custom NMT for translating to a written form of sign language. Finally, we needed a technology that had enough processing power and storage space to be able to hold models and render graphics onto an interactive screen. Once we had these requirements, we were able to look at the current technology on the market and choose the best fit for our product.

3.2.3.1. CPU

Current central processing units (CPUs) are extremely powerful pieces of technology that allow us to simulate most things very easily. This means that CPUs can perform several different processes seamlessly. However, due to the high flexibility of CPUs the performance of simulations that we run will suffer. We would receive longer run-times on the translations and graphical processing that we need to do, which in a real-time setting is not something we can afford.

Furthermore, CPUs are a single part of a computer system. They are designed to be installed into motherboards, which would have to be included in the design of our system. While this is a feasible option, we would also have to include all other parts of the computer system, including RAM, Storage, and other I/O devices increasing both monetary and dimensional cost. These setbacks along with CPUs having slower processing speeds make it an unattractive option for “the brain” of the product.

While it is possible to perform machine learning on a CPU it is extremely slow and inefficient. The CPU architecture was not built to handle these sorts of tasks as they are designed with speed and flexibility in mind. They have fewer cores than other technologies, however these cores are highly complex and designed for speed. Therefore, they excel in most modern applications, but they cannot efficiently perform tasks that require an extremely high and parallelized throughput such as graphic processing and machine learning.

3.2.3.2. MCU

Microcontroller units (MCUs) are very similar in concept to a computer system, but all of the necessary components are all contained in the chip. This allows for MCUs to be created with very small profiles which would allow use a much smaller housing for our project. Furthermore, MCUs are designed to be used in embedded and mobile, battery powered applications. For this purpose, MCUs have extremely low power consumption. However, because MCUs have such small form factor and use so little energy, they suffer from extremely slow processing speeds. They are designed to be used in applications that do not require heavy computational work.

Unfortunately, these downsides are something that we cannot afford in our product, as in a real-time conversational setting, we need to ensure that speech strings are processed, translated, and outputted as fast as possible. We also do not have a huge concern for battery life, as our product will stay plugged into the wall. We chose this design feature, because

we plan to use the product in a hospital setting which are connected to grids of generators to ensure that they will never lose power.

With the current technology used inside of MCUs, it is not feasible to use an MCU for machine learning applications. To begin, they do not have the memory required to hold data sets nor training sets and parameters. They also do not have the processing power to complete an algorithm such as those used in machine learning. Other technologies bring a larger number of simpler cores or a very complex and powerful cores to the table while MCUs have neither. MCUs definitely have their place in the market of computing, but it is not in the realm of machine learning.

3.2.3.3. FPGA

Field-programmable gate arrays (FPGAs), as they have started to make their way into the market, have shown that they are a very competitive and powerful product that allows for the user to tailor fit the chip to their needs. Customizing the chip allows the user to eliminate a lot of the overhead associated with other technologies. This makes the FPGA chip an extremely inviting technology to implement into our product. Furthermore, FPGA chips are highly parallelizable as they can be programmed to have multiple blocks that perform tasks completely independent of each other, allowing for the space on the chip to be used efficiently and for higher data throughput to be achieved. The simulation of digital circuits that FPGAs achieve using LUTs makes the technology useful in a variety of applications.

In terms of flexibility, FPGAs also have a useful feature for embedded programming that allows them to connect to any I/O device needed. This would allow us to use many of the modules that we need in our product without needing any additional hardware.

FPGAs are also highly reprogrammable because of their architecture. Typically, even after they have been deployed, FPGAs can be reprogrammed to make better use of their logic blocks or to avoid using damaged sections of the board. This would be beneficial to have in our product as it would allow us to update the board layout after it has been deployed if it were somehow damaged. When looking at various technologies used in hospitals, we noticed that hospital equipment is designed to be as rugged as possible; they are constantly being moved around by hospital staff and must be able to maintain dings and falls. If the board were to sustain damage, it would be possible for us to reroute the board to avoid the damaged parts and continue working.

As promising as FPGA technology is, it is still an extremely costly technology to implement into a product with higher-end chips running for thousands of dollars. This detracts from the attractiveness of the technology, as we are on a limited budget and must try to create a low-cost system.

FPGAs also do not use conventional C or python languages to program. FPGA chips are programmed using Verilog or VHDL, which changes the way the programmer must think. These languages are hardware description languages which are used to model electronic circuits. It is a register-level programming language that allows the programmer to describe

electronic systems. This presents an obstacle that must be overcome to use the technology efficiently. Programmers must think in parallel, instead of sequentially as used in standard programming languages such as C and Python. While this makes modeling digital circuits simpler, this presents problems for accomplishing tasks that would normally be simple in general purpose programming languages.

3.2.3.4. GPU

GPUs were originally created to be paired with a CPU as the graphics processing loads became heavier and increased into 3D space. Recently, there has been a trend of using GPUs in applications such as machine learning, bitcoin mining, and other processing tasks. There have been several different APIs and programs developed to facilitate the use of GPUs in these tasks. Programmers have been learning to program with GPUs to perform tasks that would take CPUs a much longer time to complete. GPUs are able to perform these tasks so efficiently because of their data path architecture.

GPUs are a Single Instruction, Multiple Thread (SIMT) technology that allows large sets of data to be processed in parallel. When using a GPU to train NMT models, memory is one of the biggest concerns. Modern GPUs do not have the memory size of other technologies such as CPUs and must be handled efficiently. Users often must choose between memory allocation space and speed. This is due to the special data paths that GPUs have. While this data path allows for increased efficiency in parallel computing, some tasks and data sets are not the perfect matches for parallel computing and creates bottlenecks in the GPU.

As training begins, the CPU loads as the GPU with learning parameters so that the GPU has the necessary information to begin the process. The CPU then starts the training loop, where it gathers groups of data to send to the GPU. These groups reduce the amount of transfers that need to be initiated between the two technologies which is integral in reducing the overall transfer costs. The GPU will then take this data and perform the computationally heavy algorithm that typically involve large scale matrix multiplication between data and weights. These operations are easy to perform on the GPU as it has a parallelized data path that is designed for these types of operations.

Shown in Figure 23 is a basic GPU architecture from NVIDIA which shows the hardware components of the device along with its interaction of a CPU. When using a GPU for machine learning, unless the architecture is set up to be an embedded system to work with machine learning, the GPU is connected to a CPU system. These two components then communicate to each other with the data path from CPU to GPU being much larger than the opposite direction. The CPU sends the data, parameters and other necessary information to the GPU so that it can be processed off the CPU. The GPU then sends regular updates back to tell the system how far it has gotten and pass along other necessary information. The interface flow between the CPU and the GPU is explained in Figure 24.

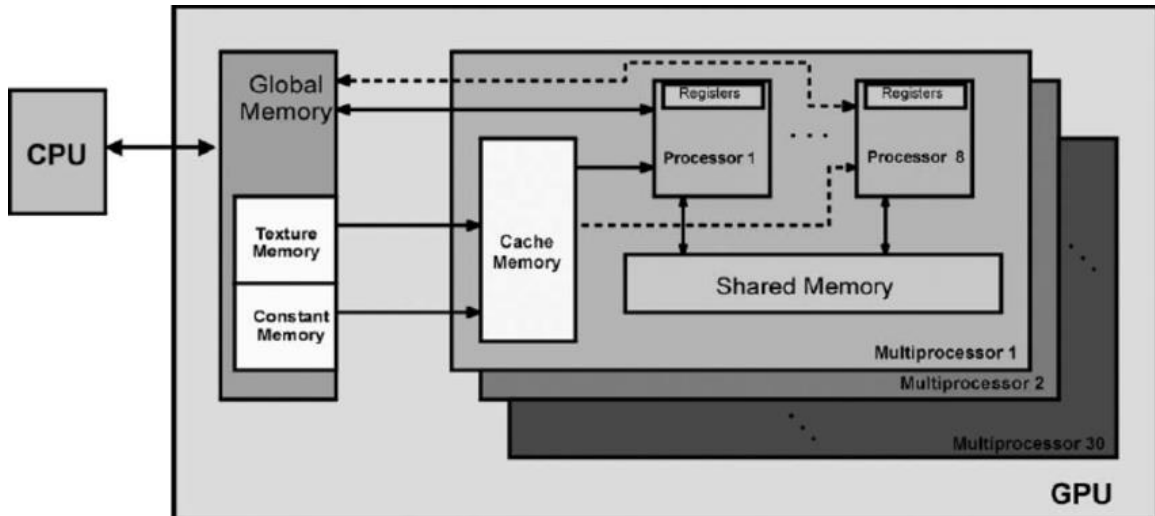


Figure 23. Example of NVIDIA GPU Architecture. (Public domain.)

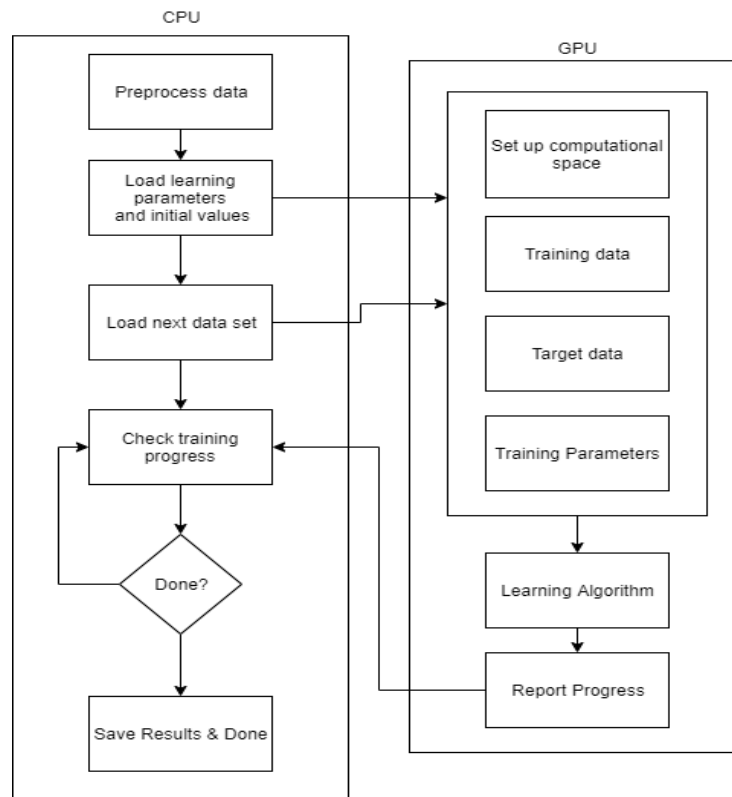


Figure 24. Basic flow of Machine Learning on a GPU. (Public domain.)

3.2.3.5 GPU versus FPGA

In machine learning, GPUs are advertised as being the best on the market for these tasks, but FPGAs are also making their way into the spotlight. FPGAs are much more energy efficient than GPUs, making FPGAs a great way to increase the life of battery powered machines. While we do not have to worry about the battery life of our product as it will be plugged into the wall, we would still like to reduce the amount of energy that is being used.

Furthermore, the flexibility of the FPGA makes it extremely attractive for machine learning applications. The structure of modern GPUs is a multitude of single instruction pipelines that work in parallel with each other. This highly parallelized structure is both a positive and a negative for GPUs: it allows the GPU to execute the instructions extremely fast; but in the case of machine learning, it can only map a certain amount of the workload efficiently. On the other hand, FPGAs allow programmers to create a customized platform around the workload that they have. This removes the restrictions that a lot of GPUs face around parallelization and fixed data paths. FPGAs have been shown to actually perform better than GPUs in some applications. Table 7 shows a comparison of the all previously discussed technologies in multiple different categories. Scores are given between 1 to 4, with 1 being the best performing in that category.

Table 7. Comparison of Ratings for CPU, MCU, FPGA, and GPU.

	CPU	MCU	FPGA	GPU
Power Efficiency	2	1	3	4
Flexibility	1	3	2	4
Cost	2	1	4	3
Speed	3	4	1	2
Processing Power	3	4	2	1

3.2.4 3D Graphics API Technologies

Although we are using a 3D rendering engine software to create our sign language models, we also require 3D graphics APIs that allows us to fully manipulate the GPU to generate smooth models. Since we are using a COM as the main processor of our device, we will need to use 3D graphics API that do not require a lot of area overhead and that can efficiently communicate with the hardware level GPU such that the performance of the COM is not affected. The purpose of these graphic APIs are not for the overall display and animation of a model but rather the implementation of such displays and animations; that is, these APIs function independently of the rendering engine software and simply process polygons. The following APIs are the most popular used in the market for use in low-level, low-overhead hardware-accelerated 3D graphics and computer interface.

3.2.4.1 OpenGL

This graphics API comes in all major operating system platforms including Linux, macOS, and Windows. The Jetson Nano comes with a Linux based operating system; therefore, the focus is with Linux's interaction with OpenGL. The OpenGL is implemented on the Linux strictly through the X windows system. Therefore, in order to use OpenGL on Linux, a GLX extension is needed to the X Server. Linux comes with a standard Application Binary Interface that is already defined for OpenGL; this allows for compatibility of OpenGL in a range of drivers. Furthermore, the Direct Rendering Infrastructure is the driver framework that allows the drives to overwritten and operated in order to allow for hardware acceleration. This DRI that is included in XFree86 4.0 and also needs a specific driver configuration after installation. Before using OpenGL, it must be initialized, and there are two phases to the initialization. The first phase of the initialization is creation of an OpenGL

Context. The second phase of the initialization is the loading of all necessary functions to use OpenGL. During the use of OpenGL does not remember the information regarding an object. Therefore, it is recommended that with OpenGL the user draws everything that needs to be drawn first. The next step is to show the image with a platform-dependent buffer swapping command. Figure 25 demonstrates an example of the architecture of the OpenGL system.

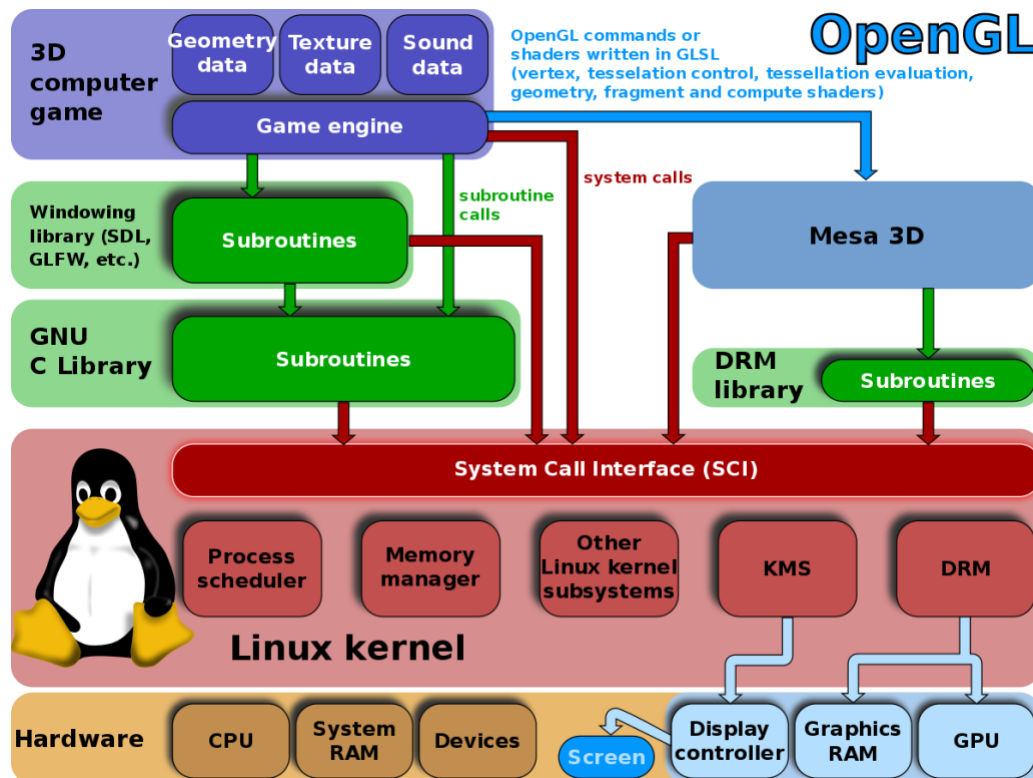


Figure 25. An example of the architecture of OpenGL. (Public domain.)

3.2.4.2 Direct3D 12

This is a graphics API by Microsoft, a platform that allows for apps to fully take advantage of the graphics and computing capabilities of the PC. This new version of the API is much more efficient and faster than the predecessor. In order to improve this efficiency, the platform no longer supports an immediate context associated with a device. This platform now has the apps record and then submit the command lists. This command lists then contain the drawing and resource management calls. The command lists are submitted by a multitude of threads corresponding to one or more command queues. These command queues manage the execution of the commands. This change in the platform increases the single-threaded efficiency by allowing apps take advantage of pre-computing the rendered works and also takes advantage of the multi-core systems. In order to program the Direct3D 12 there are a few components that are needed. The first component is a hardware platform with a GPU that is compatible with Direct3D 12. The second component are the display drivers that have the necessary support for the Windows Display Drive Model (WDDM) 2.0. The set-up of Direct3D 12 is complete after the installation of Windows 10 SDK

software and Visual Studio 2015. The only supported language of Direct3D 12 is C++. The platform must also be initialized by setting up the global variables and classes. There is also an initialize function that prepares the pipeline and assets.

3.2.4.3 Metal

This is a graphics API by Apple, that provides very close access to the GPU that is being used to run the platform. Apple claims this allows the user to maximize the graphics and computation potential of the IOS apps. The platform utilizes a low overhead architecture that has a precompiled GPU shaders, fine grained control, and multithreading support. The platform also provides support for GPU driven command creation and this simplifies working with arrays. Some other features of the Metal include GPU-driven compute encoding, improved ray tracing acceleration, metal for pro apps, simpler GPU families, metal memory debugger, and metal-enabled iOS simulator.

3.2.4.4 Vulkan

This is a graphics API is the new generation of graphics by KHRONOS group, that provides high-efficiency, cross-platform access to modern GPUs. The API focuses on providing portability so that the features of Vulkan can be utilized at native performance across all major platforms. In order to accomplish this Vulkan provides the development specifications, open-source libraries and tools altogether with the introduction of conformance tests that allows the universal performance.

3.2.5 Internet Connectivity

For proper functionality of our device, we require some form of connection to the internet. Internet connectivity is crucial for use of online API services such as the speech-to-text and text-to-speech.

There are both wired and wireless options available for the Jetson Nano. There is an Ethernet port on the device for wired connections. While using an Ethernet cable for internet connection would produce the most reliable and fastest internet connection, the significant drawbacks to feasibility and portability render this option ineffective. In terms of feasibility, there are not a wide availability of Ethernet ports, especially in school and medical settings. In a school setting, classrooms are often limited to one Ethernet wall socket for the teacher's computer. In a medical setting, there is usually not an Ethernet wall socket in patient rooms. In terms of portability, the connection into the wall would immobilize our device, since internet connectivity must be established at all times during operation.

There are two available methods for introducing wireless connectivity to the Jetson Nano. The first method is via a USB port. The other method is via the PCIe slot located under the heat sink. While both methods are available on the Jetson Nano, each approach has its advantages and disadvantages.

The advantage of using a USB port is the simplicity of installing the hardware. Another advantage is the convenience of pre-installed antennae on these adapters; other designs allow for attachable aftermarket antennae to be attached to the adapter. One disadvantage refers to the use of a USB port; since we will need the use of several USB peripherals, the use of a USB Wi-Fi adapter would take up valuable real estate. Another advantage of using this USB-type device is a perk from using USB devices in general; a USB extension cord can be used to have the antennae placed at a distance from the Jetson Nano itself; this not only frees up more room around the crowded USB 3.0 ports on the Jetson Nano, but this also means that the antenna can physically be placed anywhere on our final device, allowing the antenna to be placed at an advantageous location for optimal internet connectivity and reception. Another perk granted through the use of USB-enabled devices is the use of USB port hubs; while this brings up the risk of excessive power drain to the Jetson Nano, the use of USB port hubs for low-power devices (such as a mouse-and-keyboard setup, a common application for USB port hubs), this power drain risk is usually nominal.

The advantage of using the PCIe slot for the Wi-Fi adapter is the added flexibility; since this port is intended solely for installation of Wi-Fi adapters, using a PCIe Wi-Fi adapter would free up one USB port on the Jetson Nano while still granting internet access to the device. One disadvantage is the high cost of these parts; since USB Wi-Fi adapters are geared for convenience, the USB version is in higher demand and thus less expensive. Another disadvantage is the complication of installing antennae on these devices; the only method for antenna installation is by soldering the antennae directly to the module. There is a general risk of destroying the Wi-Fi module with the introduction of soldering, including overheating the component.

Since this socket is located under the heat sink, there is also a size limitation for the Wi-Fi adapter. While the adapter module itself is generally the same size and can fit underneath the heat sink easily, the attachment of an antenna (especially for installing multiple antennae) to these components is limited to what can fit in the available space there. Also, taking up space beneath the heat sink also reduces the effectiveness of the heat sink, leading to higher risk of overheating; however, this risk would be minimal and could also be mitigated through the use of an external fan.

3.2.6 Voltage Regulation

Voltage regulation is the process of producing and maintaining a constant output AC or DC voltage from a power source to any electronic device with various load currents. Linear and switching are the two types of voltage regulators. The switching regulators are based on a varying duty cycle of a pulse, and linear voltage regulators work at a constant operation point.

3.2.6.1 Linear Voltage Regulator

The linear voltage regulator utilizes the change in the differential voltage of the operational amplifier (op-amp) and compares the output voltage to a reference voltage. Furthermore,

the output voltage will be adjusted automatically to match the reference voltage. For the linear regulator, the input must be greater than the output voltage. The dropout voltage, which is the minimum difference between the input and the output, is around 2 V. The dropout voltage may be decreased to around 100 mV by using low dropout regulators; however, a lower dropout results in a decreased ability to reduce noise and ripple on input supply. The linear regulator usually consists of an input, output and ground pins with external capacitors used for filtering noise and achieving a better transient response.

Overall, the linear regulator takes advantage of the transistor and op-amp feedback loop by acting as an automatic variable resistor. The advantages of using a linear voltage regulator are that they are cheaper, and simple while maintaining a constant voltage output. The switching regulator maintains the constant voltage much more efficiently in contrast to the linear regulator, and thus the heat dissipated by the linear regulator is much greater and needs to be taken into account.

3.2.6.2 Switching Voltage Regulator

The switching voltage regulator is based on configuring the duty cycle of the pulse to manage a constant voltage. By using such a pulse width modulator (PWM), the average voltage can be varied directly depending on the width/amplitude parameter of the circuit. The pulse width modulator then controls the gate of the MOSFET, and the output voltage is delivered back to the PWM. The increase of the output voltage causes the pulse width modulator to decrease its width and therefore increase the off time of the MOSFET. The energy storage of the inductor allows for the current to continue flowing even when the MOSFET is off. Some of the topologies of the switching voltage regulator circuit include the “buck” (step-down) switching regulators and the “boost” (step-up) switching regulators. The buck converter steps down the higher input voltage to the lower output voltage. Since this converter is more efficient than the linear voltage regulator, this type of regulator is suitable for applications where the input voltage is much higher than the output. A boost converter does the opposite of a buck converter and instead takes a lower input voltage and boosts it to a higher output voltage. When used in combination, these two converters are useful for battery-powered circuits since the battery will decrease over time. Figure 27 shows an example of a common circuit diagram for a switching voltage regulator, while the two figures in Figure 28 show the differences between a buck and a boost converter.

3.2.6.3 Considerations for Voltage Regulators

Determining the linear or switching voltage regulator to be used depends on whether the output voltage will need to be fixed or adjusted. If the output voltage is fixed, then the regulator is selected to match the desired output voltage. For an adjustable output voltage, then a two-resistor voltage divider circuit is used, and the added costs of these components must be taken into account. The regulator selected will also have a minimum and maximum input voltage that will need to be considered when choosing the regulator. Another consideration is the current output, and this is usually limited by the current carrying capabilities of the MOSFET or any other transistor used.

If the circuit is sensitive to input noise, then the output ripple must be taken into consideration. The power supply rejection ratio (PSRR) for a linear regulator determines how well the regulator can reject any ripple at the input, and a higher PSRR is optimal. For the switching regulator, the ripple is based on the switching aspect of the circuit. This ripple is attenuated through filtering. A beneficial circuit design technique is first to step down the input voltage using a switching regulator, then using a linear regulator to remove the ripples. For high-PSRR linear regulators, an extra pin may be available that can be used to place a 10-nF capacitor which may help filter out some of the noise and ripple.

Another significant specification is the load regulation, which determines how well the regulator can maintain the constant voltage when there are changes to the current load. This is often shown in the specifications as “output voltage versus load current”. In the case of a step-change in the output load current, then the specification of load transient is used to determine how well the regulator would react to such a change. Taking into account changes to the input voltage and its effect on the output voltage, the line regulation measures the variation that change in input voltage results in output voltage. Similar to the load transient, the line transient measures the output response to a sudden step change in the input voltage. It is also important to note that the regulators with a high PSRR usually have better transient performance.

As mentioned previously, the voltage drop-out identifies how much higher the input voltage has to be compared to the output voltage for the regulator to properly function. For cases where the input and output voltage difference must be much smaller, it is best to use low drop-out regulators. The efficiency is also very important to keep in consideration when deciding on the voltage regulator. The efficiency is calculated by dividing the output power by the input power. It is typical of the linear voltage regulators to have a much lower efficiency than the switching voltage regulators. The output capacitor that goes with the linear and switching regulators is a very important consideration that requires careful consideration of the recommendation given by the data sheet. Ceramic capacitors are most commonly used since they have very low parasitic capacitance that improves the transient response.

3.2.7. 3D Rendering Software Platforms

After the process of performing speech recognition and English-to-ASL text translation, we intend to use 3D graphics rendering as our output for the ASL gestures. Although each rendering software has different approaches to graphics rendering, the outcome is very similar. In a virtual environment, the user can design the layout of the surroundings and the objects within it. The quality and actions of each object is determined by the user depending on its importance within the environment. In each of the platforms, the entities are allowed to interact; that is, objects can interact with the environment or other objects. The concepts of physics such as collision, temperature, friction, etc. are implied in each platform, but the degree of realism can be modified to suit the user’s intent.

In each of the software platforms, the user has the option either to use pre-designed models or create a new one. For premade models, therein lie copyrights for each model, either expressed outright or implied through the Berne Convention. Most models are created to be open-source and may be readily used, modified, and redistributed. In this case, mere acknowledgement of the creator is sufficient. Some models may be limited use; for these models, the model itself may be readily available, but its use, modification, and redistribution can only be performed with permission from the creator. For some models, the creator has reserved all rights to the model; in this case, such models cannot be used, modified, or redistributed without official licensing from the original producer. Note that in all cases, these copyrights relate to the model itself; the creation of fan-art replications of the design are considered legal.

3.2.7.1. Unreal Engine 4

Unreal Engine is a 3D game development software by Epic Games. The original environment was developed for the first-person shooter game *Unreal* in May 1998. The Unreal Engine was released afterwards as a platform to develop other types of games. Later editions of Unreal Engine were released for higher-resolution graphics rendering and developer capabilities. Unreal Engine 4 was released in 2014, which included the C++ source code and the availability for Python script embedding.

Like the other platforms, the Unreal Engine IDE can be manipulated without the need for prior coding experience. Unlike the other platforms, however, the entire C++ source code is available to the user through GitHub, allowing for full customization of the platform itself. Unreal Engine also has the option to port content to mobile devices. The Unreal Engine IDE can be used on most popular OSes, including Linux, Windows, and macOS, although the application can be made playable on many devices, including VR headsets and gaming consoles. Both development and consumer-end gameplay require at least 8GB RAM for proper functionality of the environment, although a more powerful RAM is recommended.

Use of Unreal Engine 4 for commercial applications is permitted by Epic Games. Royalties are paid at 5% gross income after the first \$3,000 earned from the application; that is, there are no royalties collected for applications earning less than \$3,000.

3.2.7.2. Unity

Unity is a game-development platform created in 2004 with the mindset of readily available, easy-to-use, royalty free game development. Unity has remained an independent company despite potential buyers from larger companies, as Unity insists on their contribution of “democratise[d] game development”.

The Unity Editor was developed more for developer-end simplicity. Included in the editor are premade physics engines and AI capabilities for non-player characters (NPCs). Unlike the other platforms, Unity has an option for 2D application development. The Unity Editor IDE is available for use on Linux, Windows, and macOS, although the application can be

modified to be played on over 25 different platforms, including PlayStation, Nintendo Switch, and VR platforms. Although there is no minimum hardware requirement to run the Unity Editor or Unity applications, Unity recommends that the device running the application has minimal graphics processing capabilities. The hardware requirement is usually specified by the application producer, which depends on the overall graphics intensity of the application.

Unity advertises that applications developed through their platform do not require revenue sharing; that is, there are no royalties for Unity-based applications. However, there is a tier-based system which disallows developers to use certain tiers based on their income. The maximum gross income for Unity Personal users is \$100,000 and \$199,000 for Unity Plus; there is no gross income ceiling for developers using Unity Pro. Content created by the developer through the Unity platform, however, is owned completely by the developer, with their ability to copyright their application.

3.2.7.3. MikuMikuDance

MikuMikuDance (MMD) is a 3D animation platform created in February 2008 by Yu Higuchi under the online alias HiguchiM. The MMD platform was popularized by the creation and sharing of vocaloid music videos rendered with MMD on YouTube. Although HiguchiM has retired from developing MMD in 2011, graphic designer Reggie Dentmore has continued development, with the most recent release, MMD 9.31, being released in December 2017. Although models come preset with the MMD IDE, users have the option of using the PMX/PMD editor software for model creation/editing. (Models are saved as .pmx or .pmd files.) Like Unity, there is no minimal graphics card requirement, but other users have recommended a 1GB cache and 512MB RAM due to the graphics intensity of the program. MMD is advertised as a freeware 3D graphics application. As a freeware, the development software can be edited and shared without much regulation. Ancillary software such as the PMX/PMD editor software are not made by the developers of the original MMD editor software. As such, there are no entities who can claim ownership of the MMD software, and no royalties are collected from MMD-based application content.

3.2.7.3. Blender

Blender is an open-source, 3D graphics rendering software. There is a Python API integration within Blender that allows developers to create game-like script elements in their projects. The Blender IDE is available to run on Linux, Windows, and macOS. In terms of hardware requirements, the most recent, stable release of Blender (Blender 2.80) requires a minimum of a 32-bit, dual core, 2GHz CPU, 4GB RAM, and a 1GB cache. Further releases will not support a 32-bit CPU.

Blender further recommends a more robust CPU setup to run Blender programs such as a 64-bit quad-core CPU, 16GB RAM, and 4GB of cache; or an optimal setup of a 64-bit eight-core CPU, 32GB RAM, and >12GB of cache. Blender advertises that content created through their software platform is completely royalty-free and does not require any paid subscription. In terms of licensing, the funding for our project is intended to be less than

\$1,000. Even if we were to produce our device commercially, we intend to maintain a local presence. As such, our project can be used well with the \$3,000 royalty-free ceiling for Unreal, as well as the \$100,000 ceiling for Unity. MikuMikuDance and Blender do not have a gross income ceiling for royalty collection. All of the platforms discussed here provide free downloads of their software, but for higher project budgets Unity requires subscription to paid licenses. As we do not intend to exceed \$100,000 for our budget, we should qualify for the Unity Personal license, which does not require a paid subscription.

In terms of hardware compatibility, the requirements extend to its implementation on our GPU/CPU setup, since our final project must be able to run programs based on the 3D rendering platform. A full comparison of the hardware and licensing requirements are discussed in Table 8 below. At this point, we are planning to use the Jetson Nano as our ARM processor. As the Jetson Nano runs at 1.43GHz, running Blender 2.80 will not be possible. Also, since the Jetson Nano runs Ubuntu, which is a Linux-based OS, MikuMikuDance, which only runs on Windows, cannot run on the Jetson Nano without compatibility compensatory programs such as WineBottler.

In terms of applicable platforms, Unity and Unreal Engine are two possibilities; however, in terms of the recommended hardware requirements, Unity seems to require a less-robust hardware setup. This is optimal for the Jetson Nano, which is advertised as a lower-power GPU. As aforementioned, we are intending to use these 3D rendering platforms as the output of our sign language translation; as such, we require some way of programming when certain gestures will be performed. As MikuMikuDance is geared towards video playback and not video game development, this platform has no support for dynamic ability. Blender, Unreal Engine, and Unity all have Python API integration, which is suitable for game development (or gesture management in our case).

3.2.7.4 Godot

Godot is an open source 2D and 3D rendering platform developed by MIT for game development. Although the software is supported by C# and C++, the software can also be supported by community contributions of Python, Nim, and D, among other programming languages. There is also support for scripting via GDScript, which is a unique programming language similar to Python. Like the other rendering platforms, there is an online asset library with purchasable and free-to-use models and APIs. Compared with the other platforms, Godot has the smallest file size for its editor software, comparable with that of MikuMikuDance. Another upside of this platform is its advertisement of royalty-free development, like that of Blender and MikuMikuDance. Although, as aforementioned, that our group would not be developing a device that would receive more than \$3,000 without further commercial marketing and development, the lack of collected royalties from the software provider provides a better peace of mind for the possibility of such an advancement. Similar to other software like Unity, there is a recommended hardware setup to run games made with their software without major buffering or overheating but using a setup with minimum hardware capabilities would be sufficient for running small, low-profile games.

A summary of the hardware and royalty specifications are shown in Table 8. It is significant to note that although each platform has published their respective minimum hardware requirements to run their editor software and/or published games using that software, the minimum requirements to run a specific game depends on the complexity of the published game itself. For example, a simple linear game without any environmentally interactive elements would require a less robust hardware setup than a game with a detailed environment with fully interactive, physics-enabled elements. This is crucial for the application of these “games” into our device, since our COM used in our device, the Jetson Nano, has minimal hardware capabilities suitable for portability and cost-effectiveness.

Table 8. Comparison of Hardware/Licensing between 3D Graphic-Rendering Platforms.

	Total RAM	GPU Cache	Royalties	Notes
Unreal Engine 4	8GB rec. 64GB optimal	Minimal	5% after first \$3000 earned	CPU >2.5 GHz recommended
Unity 2019.2.10	Minimal	Minimal	<\$100k for Personal <\$199k for Plus N/A for Pro	
MikuMikuDance 9.31	512MB rec.	Minimal	N/A	Windows OS only
Blender 2.80	4GB minimum 16GB rec. 32GB optimal	1GB min. 4GB rec. >12GB opt.	N/A	CPU >2GHz
Godot 3.0	4GB recommended	Minimal	N/A	

3.3. Initial Components and Part Selections

Identified in this section are the list of possible or definite key components that will be utilized in this project.

3.3.1 Sound System Selection

This section addresses the amplifiers, speakers, and microphone. Although we have decided to use a display monitor with built-in speakers, we have included our initial debate on speakers and amplifiers that we considered had we used a video-only display.

3.3.1.1 Amplifier

Amplifiers are an essential component of a sound system since they are required to increase the amplitude of the signal that is sent into it. In terms of a speaker, an amplifier is essential for the speaker to produce a sound that is audible to the user. In terms of a microphone, an amplifier is essential to amplify the microvolt signals obtained from a voice into millivolt or volt-level signals that are distinguishable and meaningful to the receiving-end processor.

With amplification of the signal we also must ensure that the integrity of the signal is maintained as it changes without distortion or other data loss. Our product requires an amplifier to produce meaningful signals from both the speaker and the microphone. Some microphones and speakers come pre-installed with an amplifier, but our selection of amplifiers here are discussed for speakers we will discuss that do not include embedded amplifiers. When we were selecting the amplifier to use in our project, we looked at wattage, impedance, gain, communication protocol and the number of channels.

3.3.1.1.1 Max98306 Amplifier

The Max98306 is a small and powerful class D amplifier. It is able to deliver 3.7 W to 2 channels into 3Ω impedance speakers. It can run from 2.7 to 5.5 VDC and comes with thermal and over-current protection. Inputs are fully differential with $1.0\ \mu\text{F}$ capacitors and output is a 360KHz square wave PWM.

Gain is adjustable between 6 and 18 dB. This amplifier would be a great choice for our project as can produce a powerful signal while still being inexpensive and maintaining a small footprint. It also would allow us to add an additional speaker if we choose to do so as well as adjust the gain to ensure that users of the system could easily hear the outputted speech from the machine and adjust the volume of the speakers accordingly. The parameters of the Max98306 amplifier are summarized in Table 9.

3.3.1.1.2 Max98357A Amplifier

This miniature amplifier is perfect for smaller projects as the Max98357A is advertised for its sufficient functionality despite its small size. It is a combination between an I²S DAC and an amplifier. It takes in I²S digital audio and amplifies it directly into the audio output connected to it. It is capable of delivering 3.2W into a single 4Ω impedance speaker. It is a class D amplifier and runs at 2.7V to 5.5V DC. Input from I²S can use either 3.3V or 5V logic data and are bridge connected.

Output is a 300KHz square wave PWM and gain can be adjusted from 3 to 15 dB. This amplifier would be perfect for our needs in the project, as we are wanting to send audio out from the I²S pins on the GPU/CPU/COM board that we will have chosen. This amplifier can only support one speaker, but for the purpose of our project the use of one speaker should be sufficient. The parameters of the Max98357A amplifier are summarized in Table 9.

3.3.1.1.3 Max9744 Amplifier

The Max9744 is a powerful amplifier that can also produce a large sound despite its small size. It is capable of driving two channels with 20W into four 8Ω impedance speakers. It is powered by a 5 – 12 V DC source that can be inserted into its on-board DC power jack. This breakout board also has a 3.5-mm audio jack preinstalled for audio input. However, it can take digital signals as well from I²C communication protocol pins. This amplifier

would be an amazing choice for our project because of how flexible it is in terms of inputs, gain, and channels. The parameters of the Max9744 amplifier are summarized in Table 9.

Table 9. Comparison of Significant Parameters of Amplifiers.

Feature	Max 98306	Max 98357A	MAX9744
Wattage	3.7 W	3.2 W	20 W
Ohms	3 Ω	4 Ω	4 – 8 Ω
Channels	2	1	2
Gain	6 – 18 dB	3 – 15 dB	Up to 29.5 dB
Class	D	D	D

3.3.1.2 Speaker

Speakers are also an essential part of every sound system as these devices are required to convert electrical signals into sound waves that are audible and meaningful to the user. Speakers achieve this by vibrating a cone at high speeds, which results in the cones producing sound waves. We will be using speakers for playing the text-to-speech sound bites that are obtained post-cloud processing of text received from translating in the translated speech mode. When looking at potential speakers for our project we looked at price, wattage, impedance, and size.

3.3.1.2.1 Stereo-Enclosed Speaker Set

This class of speakers which consists of a dual set of speakers would be great for our project and come in a pair to allow for stereo audio. The pair of speakers is enclosed for better sound quality and come with speaker wire preinstalled in them with a single four-port connector. These speakers are low-priced at \$7.50, are powered at 3W, and have an impedance of 4 Ω . The parameters of this part are summarized in Table 10.

3.3.1.2.2 3" Diameter Speaker

This single speaker would also be a reasonable yet frugal choice to add sound to our project as it is priced at \$1.95. This speaker, however, does not come with wires or connectors preinstalled. The speaker is powered at 3 W and has an impedance of 4 Ω . This would pair perfectly with the Max98306 or Max9744 as we could connect one of these to the amplifier for a robust sound system. It is also 3" in diameter with mounting tabs 60mm apart which we could install easily into the case of our device. The parameters of this part are summarized in Table 10.

3.3.1.2.3 XS-GTF1027 Speaker

This speaker is much more powerful than the other aforementioned speakers, being powered off 20 W of power with 4 Ω impedance. This speaker has a wide decibel range and also has a frequency range of 60 Hz to 24 KHz. However, with these speakers being priced at \$14.50, and with its extreme level of volume production, we believe that this

speaker is not the optimal choice to be implemented into our project. The parameters of this part are summarized in Table 10.

Table 10. Comparison of Significant Parameters of Speakers.

<i>Feature</i>	Stereo Speaker Set	3" Diameter Speaker	XS-GTF1027
Wattage	3 W	3 W	20 W
Ohms	4 Ω	4 Ω	4 Ω
Speakers	2	1	1

The HDMI protocol includes both audio and video output. With this, some of the LCDs discussed in section 3.3.4 include displays with embedded speakers. Because of the HDMI standard, there would be no further manipulation of the speakers in order to have the speaker and the display work together correctly. Additionally, many of these display/speaker combinations provide a more cost-effective solution for A/V output. It is because of this simplicity and effectiveness that we decided to use one of these LCD/speaker combinations using an HDMI-HDMI cable which will run from the Jetson Nano to this display (and thus the speaker).

3.3.1.3 Microphone

The use of a microphone is an essential piece of equipment for audio input, which would be voice capture in the case of our project. In terms of microphones applicable to our overall project design, we require either a handheld microphone that can transmit signals from 5 to 10 feet away or a microphone located on the device itself that is able to pick up voice signals from 5 to 10 feet away.

The advantage of having a microphone integrated into the stand is minimizing peripherals that the user would need to handle, but the disadvantage is the requirement of a more powerful (thus larger and more expensive) microphone. The advantage of having a handheld microphone is the flexibility of using either a wireless microphone (utilizing Bluetooth or other wireless protocols) or a wired microphone connected directly to the device itself. Another advantage is the ability to choose less powerful microphones, since the user will be less than 1 foot away from the microphone if it is handheld. The disadvantage of a handheld microphone is the implication of several interfaces for the user; that is, holding the microphone would be required in addition to manipulating the display via buttons, touch screen, remote, etc.

In terms of the use of an I2C microphone versus a USB microphone, both types of serial communication are applicable in our project. The benefit of an I2C microphone would be the flexibility of choosing how the signal will be sent to the main processors on our device (FPGA, COM, etc.); that is, we can choose to take the DC/AC outputs from these microphones and send them through an operation amplifier, filter, and so on.

Another benefit of using I2C microphones is the availability of schematic-level descriptions of each microphone; most USB microphones are marketed for ease of use, so

schematics and audio capture methods are not usually included with USB microphones. The drawback of using I2C microphones would be the requirement of soldering the mount header pins onto the board; with soldering in general, there is a risk of burning pins and losing functionality of the board. Another drawback is the reliability of the connections made, since these connections will need to be created by ourselves.

The benefit of a USB microphone is the simplicity of connecting the microphone to the processors via a USB port. A major drawback is the occasional requirement of downloading driver software for proper functionality of the microphone and compatibility between the microphone and the processor.

3.3.1.3.1 Adafruit Mini USB Microphone

This low-profile microphone would be an example of a microphone that would be attached to the device itself. The microphone is about the size of a flash drive and would be able to plug directly into a USB port installed on our device (on the FPGA or the COM, for example). Due to the flexibility of USB-enabled devices, the microphone can be attached to the end of a USB-USB extension cord. The full characteristics of this board are shown in Table 11.

Out of all the other microphones discussed here, this microphone is the least expensive in terms of USB microphones at \$5.95 USD. The major downside is the lack of documentation and specifications on this product. There is a lack of specifications on the frequency range and accuracy of sound capture.

3.3.1.3.2 Adafruit MEMS Microphone (SPH0645LM4H and SPW2430)

This MEMS microphone from Adafruit is very low-profile due to the utilization of MEMS technology. There are a variety of different boards available that provide a small yet powerful microphone such as the SPH0645LM4H and SPW2430 boards. The major differences are the detectable frequency range of the microphone, and the power supply requirement. The full characteristics of this board are shown in Table 11.

In terms of price, these boards are inexpensive at \$6.95 USD and \$4.95 USD for the SPH0645LM4H and SPW2430 boards, respectively. Despite the difference in cost, there is not a significant difference in the potential of these two microphones.

3.3.1.3.3 SparkFun MEMS Microphone (INMP401/ADMP401)

This microphone breakout board from SparkFun utilizes the ADMP401 (now INMP401) MEMS microphone from Analog Devices. Unlike the boards from Adafruit, which feature digital output, this ADMP401 board has an analog output. The full characteristics of this board are shown in Table 11.

This board could be a viable option since an analog output would provide more flexibility on the processing of the sound signal than the already-digitized signals produced by the Adafruit microphones. However, due to the lack of analog input on the Jetson Nano, we would require the inclusion of an analog-to-digital converter (ADC) before transmitting the signal to any processor on our device. This would include our PCB for the FPGA, where we would also require either a peripheral ADC board or an embedded ADC on the FPGA PCB.

Table 11. Comparison of Significant Parameters of Microphones.

<i>Feature</i>	SPH0645LM4H	SPW2430	INMP401/ADMP401
<i>Frequency Range</i>	50 Hz - 15 kHz	100 Hz - 10 kHz	100 Hz - 15 kHz
<i>Power Supply</i>	1.62-3.6 V @ 60 uA	1.5-3.6 V @ 70 uA	1.5-3.3 V @ < 250 uA
<i>SNR</i>	65 dB	59 dB	62 dB
<i>Price</i>	\$ 6.95	\$ 4.95	\$ 10.95

In terms of the different parameters of the three microphones discussed in Table 11, there is not a significant difference in the capabilities of each microphone. Although the SPW2430 has a lack of detection in higher frequencies, the range of human voice is roughly between 80 Hz and 260 Hz. Also, the power supply ranges would be achievable via the 3.3 V power pin commonly found on processor boards. (The Jetson Nano has both 5 V and 3.3 V power pins in addition to having powered USB ports.)

3.3.2. FPGA Selection

Our project will be using an FPGA chip to process and communicate with all I/O devices necessary, except for the monitor visuals, which will be handled via our COM device. We will be creating a PCB onto which the chip will be surface-mounted and then to which the other I/O devices such as the speakers and buttons will be connected. Having a separate board and chip to handle all the I/O allows the COM to focus solely on processing speech and text required for proper execution of our device. We chose to use an FPGA chip to perform this task because it allows us to connect a wide variety of devices and choose how to use the I/O pins as needed.

When it comes to FPGA chip selection, one key element we sought was the low overall cost of the chip. We have not yet partnered with a financial sponsor and therefore must be frugal with our budget. Another key element we looked at was I/O pin count, since we need to ensure that we could communicate with all the components necessary for this project. Next was the number of gates on the FPGA board. We wanted to ensure that we would have enough blocks for our program to fit on the board but not so many that we are wasting time and space on the chip; optimizing the gate number of the FPGA board will also maximize the efficiency of our funds.

Next, we looked at the package type of the chip in question. We wanted to ensure that the chip that we chose was a QFP-style package, as this style of package allows for easier prototyping. Furthermore, a QFP-style package reduces the number of layers required in the PCB design as this style results in a less complex package type.

We also looked at the operating frequency of potential chips, as we would prefer the smallest latency times achievable. The frequency, however, is not as significant as the other elements as there are no timing-critical systems in our project. This allows us to use a slower-frequency board without much loss in quality while reducing the cost of the final product. Finally, we looked at the vendor of the FPGA chip in question as this would determine the software and interfaces that we could use to program the chip. We are used to interfacing with Xilinx chips and using the Vivado software suite, so having a suitable vendor would be easier to begin interfacing with the chip. Furthermore, the UCF laboratories have Basys 3 boards available for student use. These boards run the Vivado software, and access to this software allows us to develop and test our programs more efficiently regarding cost and time.

3.3.2.1 Altera Cyclone IV – EP4CE22E22xxx

The Cyclone IV EP4CE22E22xxx family of FPGA chips is a great choice for our device as it meets all of the specifications required. It has a total of 22,320 logical elements on the board and 594 Kb of embedded memory. For I/O, this family of chips has 79 user I/O pins and 17 LVDS pins. The chip is affordable as well, depending on the speed of the chip selected, it can range from \$35 USD to \$53 USD. This family of chips comes in 2 different core voltages of 1.0 V and 1.2 V. We will not choosing a low-voltage device and therefore, will have a core voltage of 1.2 V to power the chip itself. This would allow us to power the device easily through a USB connection to the COM board that we choose. A downside of this board is that it is an Altera chip, which means that we must use Altera software to interface with the chip, which is not what we have experience with. The full characteristics of this board are explained in Table 12.

3.3.2.2 Spartan 3E - XC3S500E-xPGx208C

The Spartan 3E XC3S500E-xPGx208C family of FPGA chips is also a great choice for our project we are aiming to create as this family of chips also meet all the specifications required. These chips have a total of 10,476 logic cells, 73K distributed RAM bits, and 360K Block RAM bits. For I/O, this family of chips has 158 user I/O pins and 65 differential pins. The chip fits into our budget as well, costing between \$37 and \$52 USD, depending on speed grade and lead-free packaging. The Spartan 3E has a supply voltage of 1.2 V. This would allow us to power the FPGA chip from a USB connection to the COM board that we choose to use. On the downside of the board, this is an older chip from Xilinx and therefore is not supported by Vivado. While we have used Xilinx ISE before, we have more experience in Vivado and would have to learn the software to interface with the chip. The full characteristics of this board are explained in Table 12.

3.3.2.3 Artix 7 – XC7A35T-1CPG236C

The Artix 7 – XC7A35T-1CPG236C FPGA chip is a great choice for being the brains of the I/O board in our project. It contains 33,280 logical elements and 1,800 Kb of embedded memory. For I/O, this family of chips has 250 user pins. It is an affordable chip as it only

costs \$42.97 USD. It operates at 1.0 V, allowing for the chip to be powered from a USB connection to the COM. Furthermore, the Artix 7 is programmable with Xilinx Vivado, which we have experience using. This chip also comes installed in the Basys 3 boards that are in student accessible labs on campus. This would allow for us to program and test the chip quicker and cheaper than other options. Unfortunately, this family of chips only comes in a BGA package type and therefore, would be more difficult to implement onto a PCB board. Table 12 shows a comparison between all of the different FPGA chips that will be discussed in the following sections.

Table 12. FPGA Comparison.

FEATURE	CYCLONE IV	SPARTAN 3E	ARTIX 7
PART NUMBER	EP4CE22E22xxx	XC3S500E-xPGx208C	XC7A35T-1CPG236C
COST	\$35.52 – 53.28	\$37.58 – 52.21	\$40.03
VENDOR	Altera	Xilinx	Xilinx
I/O COUNT	79 User I/O; 17 LVDS	158 User I/O; 65 Diff	250 User
LOGICAL ELEMENTS	22,320	10,476	33,280
EMBEDDED MEMORY	594 Kb	360 Kb	1,800 Kb
SPEEDS	6, 7, 8	Standard, High	1

3.3.3 Computer-on-Module (COM) Selection

In our project, we will be using the Computer-On-Module (COM) for running the screen of the terminal and for processing and rendering the sign language graphics. We believe that this will be rather resource-intensive, which will require the COM to contain a GPU unit capable of handling the graphics demand. Furthermore, we will be using the COM to run an OS that will allow us to run the tasks and programs that we need, such as the 3D rendering software (such as Unity) and interfacing with the FPGA.

The first element we looked at when selecting a COM was the cost of the unit, which is the most important element for our team as we lack a financial sponsor for our project and must be frugal in the usage of our limited funds. Another element is the number of computation cores that the GPU unit has. Next, we looked at the number of I/O pins on the device as this is extremely important for connecting our peripherals necessary for proper implementation of our final project.

3.3.3.1 NVIDIA Jetson Nano Developer Board

The Jetson Nano Developer Kit is of greatest interest for our project due to its relatively low cost in relation to its performance capability. At only around \$100, it is both low-cost and high-performance. The board contains a quad-core ARM A57 which has a clock frequency at 1.43 GHz. Furthermore, the board contains 128 Maxwell computation cores

that allow for robust performance in parallel computing, such as with our neural machine translation (NMT) models. It comes with 4 GB of high-speed internal memory.

For storage, the Jetson Nano uses microSD cards for its memory modules. NVIDIA sets the minimum size as 16 GB for the microSD card, but with the size of OS images and other necessary files, a 32 GB or larger microSD card is recommended. The board is powered through a 5V power supply, which can be delivered through the micro-USB port or the 2.1mm barrel port depending on the available power supplies. Higher power usage is directly correlated to excess heat production, but the Jetson Nano is prepared for the heat since it comes preinstalled with a large heat sink. This heat sink also has 4 mounting holes for mounting a single 140mm fan that can be plugged into the 3-pin PWM fan header on the board.

When it comes to I/O connections, it has 40 pins and includes support for I2C, SPI, and UART communication protocols which will allow us to communicate with any the FPGA board and any other peripherals that we choose to use. There also comes a Gigabit ethernet port that comes pre-installed on the board. This is favorable for our application as we will need to have our board connected to the Internet to send out our audio snippet files and have them converted from speech to text and receive this text snippets back. If we choose to not use the ethernet port, it also offers a M.2 E key slot for Bluetooth and Wi-Fi Module extensibility.

Furthermore, the Jetson Nano has four USB 3.0 ports, which allow us readily to connect our peripheral components such as a microphone, speakers, etc. as needed. Finally, the board features two display connectors, one being display port and one being HDMI. This will allow us to output to our chosen display which is essential for our application. These ports allow for simultaneous use, so we can drive 2 displays from the board natively. In the chance of further extensibility of our product, the board also features a CSI video camera connection that will allow us to possibly implement sign language detection functionality. A full description of the Nvidia Jetson Nano developer board and its comparison to the Jetson TX2 development board are summarized in Table 13.

3.3.3.2 Nvidia Jetson TX2 Developer Board

The Jetson TX2 is the bigger brother of the Jetson Nano and brings a lot more power to the table. It costs a significant amount more at \$449, but it also comes with a quad-core ARM Cortex A57 processor clocked at 2GHz as well as a dual-core NVIDIA Denver2 processor clocked at 2 GHz. Its graphics processing unit has a 256-core PASCAL GPU built in. Furthermore, the TX2 has 8GB of high-speed DDR4 internal memory. For storage, the Jetson TX2 Development Board has 32 GB of eMMC flash storage. For power delivery, the TX2 is powered off of a single 19V AC adapter that is plugged into the board with a 2.1 mm barrel jack. The TX2 comes preinstalled with a large heatsink and fan to ensure that the unit does not overheat when operating at or near maximum operating conditions.

When it comes to I/O connectivity, the Jetson TX2 has 40 pins and includes support for I2C, SPI, and UART communication protocols. The board also comes preinstalled with a

Gigabit ethernet port and has Wi-Fi and Bluetooth modules pre-installed. This would allow us to connect to the Internet either way that we choose without any additional part installation as well as have the option to Bluetooth microphones or remotes for our final product. Furthermore, the TX2 has a single USB 3.0 Type A port and a single USB 2.0 Micro AB port.

While this isn't much in terms of the number of USB ports, there is also a PCI-e expansion slot to add more USB slots, or we could use a simple USB hub. There is a single HDMI port for a single display terminal. Next there is a microSD reader built into the board for memory expansion and an M.2 E key slot. Finally, there is a 5-Megapixel fixed focus CSI camera which comes pre-installed on the board, which would allow us to expand our project in the future to allow for sign language detection. A full description of the Nvidia Jetson TX2 developer board and its comparison to the Jetson Nano development board are summarized in Table 13.

3.3.3.3 ASUS Tinker Board

The Tinker Board by ASUS is another option as a single-board computer. At \$65, this COM would be the most economical choice. This board comes standard with a Rockchip Quad-Core RK3288 SOC and an ARM Cortex Mali-T764 GPU with 4 pipelines clocked to 600 MHz. This board has 2 GB of DDR3 RAM and microSD card slot for system storage.

Much like the very popular Raspberry Pi SBCs, the Tinker board has 4 USB 2.0 connectors and Gigabit Ethernet port. Furthermore, the board has a MIPI camera, a MIPI display connector, and an HDMI 1.4 connector, capable of 4K at 30 frames per second. It has a 3.5 mm Stereo audio jack capable of good quality audio that uses the extra sleeve for microphone input. The Tinker Board itself is powered by a micro-USB port at 5V and 2/2.5 A. For I/O, the tinker board has a 40-pin color coded GPIO header, which would allow for quite a bit of connectivity for our project. Finally, the Tinker Board does come packaged with a heatsink that can be installed onto the board to ensure the board does not bottleneck due to heat issues or damage itself. The ASUS Tinker board does have an enhanced version (ASUS Tinker Board S) available for purchase that includes 16 GB of eMMC storage preinstalled, out of box Bluetooth capabilities, and various other small upgrades. However, this upgraded version does not improve upon the SOC, GPU, or RAM that comes with the board. Therefore, it is not attractive enough for our use to warrant the price jump from \$65 USD for the standard Tinker Board to \$90 USD for the S version.

3.3.3.4 Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B is the newest edition to the line of extremely popular Raspberry Pi single board computers. This COM is an affordable option for a COM to use in the project as it has a variety of features and comes with a ton of software, drivers, and other features created by the huge community of followers that this board has accumulated. It costs only \$55 US dollars for the model with the most pre-installed RAM.

The Pi 4 Model B comes with a quad-core Cortex – A72 processor clocked at 1.5 GHz. Furthermore, the Model B comes with 4 GB of DDR4 RAM. It has 2 USB 3.0 ports, 2 USB 2.0 ports, and 40 GPIO header pins allowing for a wide variety of connectivity to different peripherals. The board also has 2 micro-HDMI ports, a MIPI DSI display port and a MIPI CSI camera port, as well as stereo audio and composite video ports. The COM can be powered with 5V DC with a minimum of 3A via a USB-C connector or by the GPIO headers.

It comes installed with a 2.4 and 5.0 GHz wireless adapter, a Bluetooth module, and a Gigabit Ethernet port. While this COM unit seems to be an extremely attractive option for our project, it has a rather large downside since it lacks a dedicated graphic processing unit. This unit comes with an integrated graphics chip on the processor which limits the amount of graphical work that can be done on the unit.

While it might be possible to modify the board to include a PCI-e slot and connect the board to an external GPU, this would be far out of the scope of this project and would require us to make our own drivers for the project as well as do quite a bit of fine soldering to modify the board.

Table 13. COM Comparison.

Feature	Jetson Nano	Jetson TX2	ASUS Tinker Board	Raspberry Pi 4 Model B
Cost	\$99	\$449	\$65	\$55
GPU	128-core NVIDIA Maxwell™ GPU	256-core NVIDIA Pascal™ GPU	ARM Cortex Mali-T764	Integrated Graphics
I/O Pins	40 GPIO pins	40 GPIO pins	40 GPIO pins	40 GPIO pins
Communication Protocols	I2C, SPI, UART	I2C, SPI, UART	I2C, SPI, UART	I2C, SPI, UART
Memory	4GB 64-bit LPDDR4 Memory 1600MHz - 25.6 GB/s	8GB 128-bit LPDDR4 Memory 1866MHz - 59.7 GB/s	2GB DDR3 memory	4GB LPDDR4 memory
Power	5 – 10 W	7.5 – 15 W	10 – 12.5 W	15 W
Storage	microSD card slot for storage	32GB eMMC 5.1	microSD card slot for storage	microSD card slot for storage
CPU	Quad-Core ARM® Cortex®-A57 MPCore	Dual-Core NVIDIA Denver 2 64-Bit CPU and Quad-Core ARM® Cortex®-A57 MPCore	Rockchip Quad-Core RK3288	Cortex – A72 processor clocked @ 1.5 GHz
Wi-Fi Enabled	No	Yes	Yes	Yes
OS	Ubuntu	Ubuntu	TinkerOS - Debian	Raspbian OS, Ubuntu, Etc.

3.3.4 Display Selection

The display is an integral component of the final product as the display will be the medium upon which the GUI will be interfaceable. The LCD will be able to display the ASL animation so deaf users can read the sign language as well as the input and translated text.

The display will also tell users in which mode they are currently using, when they are recording, and what the device read as their dialogue (this last part will be included to allow for debugging on our end and troubleshooting on the consumer end as needed). The most integral factor is choosing a display is the price, since LCDs are expensive. We have yet to partner with a financial sponsor and must use our limited funds in the most efficient manner possible. The next factor is the size of the display. Users should be able to see the models on the screen easily and be able to read text from a reasonable distance without having to strain their eyes. Another factor we considered was touch-compatible displays to have a device that is sleeker and more portable; that is, the device can be handheld.

3.3.4.1 10.1" Display & Audio IPS Panel

At 10.1 inches, this 1280-by-800 in-plane switching (IPS) display is one option for a terminal in our product. It is an all-in-one option that includes the screen, HDMI driver, and the ability to connect audio output peripherals. Furthermore, it has a button keyboard to open a monitor menu with options such as brightness and contrast. It has inputs for video of HDMI and VGA and a 3.5 mm audio jack for audio input. However, at \$154.95, it is rather expensive for a small screen size in comparison to other options on the market.

3.3.4.2 Sceptre E205W-16003S LED Monitor

This 20" monitor would be great for our project as its large size is still suitable for our device design. It would make the user viewing experience much better, as the resolution for this monitor is 1600x900. This display is also \$59 which fits our budget well. The monitor has inputs for HDMI and VGA and has built in speakers. With a refresh rate of 75 Hz, models being displayed on the screen would be able to keep up with the output of the graphics processor of our device and would allow the user to view non-choppy animations.

3.3.4.3. UPERFECT 12.3" Touch Monitor

The UPERFECT monitor would allow the user to touch the screen directly to control the terminal. This is an attractive option as it would increase the ease of use for users by being more intuitive and would allow for a sleeker and more portable final design. The monitor is 12.3" which is about the size of a tablet computer, which is an optimal size for our project. It has a high resolution of 1600x1200 and has built-in speakers. For input options, it has HDMI, DVI, and VGA. It uses a microUSB cable to achieve the touchscreen capabilities and is designed to be used with a huge variety of devices including COMs. It is priced at \$149 US Dollars which is rather expensive for a 12.3" monitor but would add touch capability to our terminal.

3.3.5 Controller Selection

When it comes to the user interaction with our device, we had several different options for how the user could do so. The device needs to be able to perform the actions of mode selection and voice recording with input from the user. For voice recording, we wanted the user to press a button to begin the voice recording process and press the button again to

stop the recording process and have a sound file ready for processing. For mode selection, the user would be able to press a button to switch between the speech-to-speech translation mode and the speech-to-ASL mode. This would require the use of at least two buttons, one of which would be for voice recording and the other for the mode selection.

3.3.5.1 IR Remote

One option for the user to notify the device about their input is for the device to have an IR communication method. IR remotes are common devices seen mostly to help users interact with devices such as TVs and DVD players. These remotes work by sending data in the form of pulses of modulated infrared light. Each button on the remote sends a code that is unique to that button that is then demodulated and decoded by an IR sensor in the receiver.

This implementation for indicating when to record the user's voice would be a great choice as it is extremely inexpensive to implement, at \$6.95 USD. Furthermore, this implementation would be easy to program as it is as simple as decoding the code that is received from the IR sensor and telling the controller what to do when certain codes are received. However, this implementation would be subject to interference in the form of sunlight and other light sources, such as light bulbs. While most modern-day LED light bulbs emit less IR light than older incandescent light bulbs, this still imposes a risk that a signal would be impeded by other IR signal noise and the user would have to press the button multiple times for the system to pick up the signal. Furthermore, IR sensors are very sensitive to the position of the transmitter and they must have line-of-sight to send and receive signals properly; that is, without the use of IR relays or amplifiers, the user must point the remote directly to the receiver for the signal to be transmitted and received successfully. This could lead to the same problems as interference and lead to a suboptimal user experience.

3.3.5.2 RF Remote

Another option would be to use an RF remote and receiver. RF remotes are common devices seen in everyday devices such as car keys, garage door openers, and radio-controlled toys. They work by sending out radio frequency waves that represent the binary code for the respective button that was pressed on the remote. The receiver takes in the signal and then decodes the signal back into a usable binary code.

This implementation would be a great choice for our project as it is inexpensive to implement, costing around \$12 USD for an RF remote and receiver. It is also extremely easy to implement as the receiver and remote that we have chosen are a pair that know how to communicate with each other already. There are five output pins on the receiver that correspond to an interrupt pin and four pins corresponding to four unique buttons. When any button is pressed, the interrupt pin goes high and the respective pin for that button also goes high. When the button is released, all pins go low.

A benefit of this approach is that the remote is invariant to line of sight with the receiver and typically has a much larger range than an IR approach. Furthermore, the remote

transmits at a specific frequency, 315 MHz, that is not used by many devices and is therefore less subject to interference and will not interfere with any other devices that are not within this range. Issues might arise from there being no error checking built into these remotes, as the remote has no way of knowing if the signal was received correctly or at all. Furthermore, these devices have no way of addressing the signals, and therefore, all devices that operate at 315 MHz will receive the signals from the remote.

3.3.5.3 Push Button

By far the simplest approach for the user to interact with the console, another option would be to install a momentary push button on the console for the user to press whenever they want their voice to be recorded. If the button is pressed, the microphone will be activated and recording. This implementation would be cost effective and easy to implement and it would only require a single button to be added to the PCB board and would require no extra programming or fiddling. It would also decrease the complexity of user interaction by only having a couple of buttons housed in the console itself that users could easily access. We would not have to worry about users dropping, breaking, or losing the remote. Furthermore, we would not have to worry about a remote losing power or running into interference as it transmits. However, this would make users have to sit right next to the console and remain at the console to use it; otherwise, a long cable would need to extend directly from the device to the user, which would result in a cumbersome and non-user-friendly setup.

3.3.6 Wi-Fi Module Selection

To determine the proper Wi-Fi module suitable for our device, the adapter must be able to provide sufficient speeds and reliability while also maintaining a small overhead and profile. The types of connectivity available to the Jetson Nano include PCIe and USB. For PCIe-enabled Wi-Fi modules, we have the option of soldering external antennae to the adapter to increase its range. The USB-enabled Wi-Fi adapters are usually designed as a single package; rarely do these USB devices give the user the option of installing one or more antennae. Below we discuss briefly one available PCIe Wi-Fi module and one available USB Wi-Fi module.

3.3.6.1 Intel Dual Band Wireless-AC 8265

The Intel Dual Band Wireless-AC 8265 is a PCIe-enabled Wi-Fi adapter that provides both Bluetooth and Wi-Fi support with speeds up to 867 Mbps and offers extended battery life in comparison to the legacy 11ac devices. The PCIe connection provides a reliable channel for sending and receiving data. This card can also provide connectivity through 2.4 and 5 GHz channels. The module weighs around 2.6 grams and can operate in the temperature range of 0°C to 80°C. These specifications would work with the current project design since the design does not have portability constraints. With this module an additional antenna may be needed to extend the range and reception of the adapter. The antenna that is available for this card is the RP-SMA Dual Band that provides the additional functionality needed. One disadvantage of this card is the excessive price at \$140.92 USD.

3.3.6.2 Ultra USB Wi-Fi Adapter

The Ultra USB Wi-Fi Adapter provides up to 600 Mbps and has access to both the 5 GHz and 2.4 GHz channels. It also provides an external high gain long-range 2 dBi omnidirectional antenna. The integration of the antenna in one package provides more simplicity to the Wi-Fi connectivity aspect of the project, while also maintaining the required specifications. The Ultra USB Wi-Fi adapter is the route of choice because of the one in all design and the portability of the product is not a top priority.

The adapter we would be using for our design was donated by one of our team members. However, comparable USB Wi-Fi modules with an integrated antenna range in price from \$20 to \$50 USD. Although the PCIe module we listed was out of our price range, there are other PCIe Wi-Fi modules that are also around the \$20 to \$50 range. Also, the wireless connectivity is also comparable between the USB and PCIe protocols. Therefore, the selection of the Wi-Fi module for our project merely depends on availability and feasibility; since we already own USB-enabled Wi-Fi modules, we have chosen to proceed with this method of internet connection. Should we need a faster or more reliable connection, we may consider purchasing a PCIe card or upgrading to a more robust USB module.

3.3.7 Voltage Regulator Selection

For the selection of the voltage regulator, there are several considerations to consider when choosing the component that it has the best performance while also being low-cost. Several of these considerations include the required output voltage, input voltage, current output, output ripple, load regulation, load transient, line regulation, line transient, voltage drop-out, and efficiency. The purpose of the voltage regulator in our project is in the development of our power supply unit, where the voltage supplies to the Jetson Nano and the Altera Cyclone IV require specific voltages to be available for extended periods of time to maintain proper functionality of both devices.

3.3.7.1 LP5900

For maintaining a constant voltage output of 2.5V, we chose a linear voltage regulator since the input voltage would need to be 5 V. Therefore, since the difference between the input voltage and output voltage is around 2.5V, this linear voltage regulator is the preferred choice. The linear voltage regulator is a more energy-efficient and cost-efficient option, since the voltage difference does not amount excessive energy loss. The input voltage range of the LP5900 is 2.5V to 5.5V and the output voltage range is 1.5V to 4.5V which matches the specification needs of the project with an additional margin of error for the input and output voltages. Since it is a low-drop linear voltage regulator (80mV), a noise bypass capacitor is not required and thus simplifies our circuit layout. The maximum output current possible with this regulator is 150mA which is sufficient for the purposes of this project.

3.3.7.2 LP38500-ADJ

For maintaining a constant voltage output of 3.3V and 1.2V a linear voltage regulator was chosen since the input voltage would only be around 5V. Therefore, the difference between the input voltage and output voltage is around 1.7-3.8V a linear voltage regulator is the preferred choice. As mentioned above, the linear voltage regulator is much simpler and cheaper option, and since the voltage difference isn't too much the amount of energy lost, as a result of the efficiency of linear regulators, is acceptable.

The input voltage range of the LP38500-ADJ is 2.7V to 5.5V and the output voltage range is 0.6V to 5V which matches the specification needs of the project with additional margin. A 10uF capacitor is needed at the input and output for stability of the circuit. The maximum output current possible with this regulator is 1.5 A which is enough for the purposes of this project. The general wiring diagram for the LP38500-ADJ is explained in Figure 30.

3.3.7.3 LM7805

The electrical characteristics of the LM7805 demonstrate that the output of the voltage regulator is fixed at either 5V, 12V, and 15V options depending on the input. Therefore, this regulator was not considered for the possibility of being used to regulate the voltage within the circuit design. In terms of the LM7805, although the output voltage is 5V, which we could definitely use for our design, the input voltage required for this voltage regulator is at least 7 V up to 25 V, which is out of range for our power supply unit. This also applies to the LM340, LM340A and LM7805 classes of voltage regulators.

3.3.8 Initial Summary of Selected Parts

This section describes the decisions on specific parts that were to be implemented in the original design. Note that although these components had been chosen for the initial design, many parts were omitted or replaced in the final design after component and device testing.

3.3.8.1 Speaker

When looking into different speaker systems for our project, we had investigated having a separate sound system to play our audio through. After doing some research into this, we found that we would need to have 3 parts to make up our sound system: a digital-analog converter (DAC), an amplifier, and the speaker itself. The DAC would convert our I2S digital sound signal to analog signals, an amplifier would increase the amplitude of the outputted sound signal, and then the speakers would take input analog signal and output audible sound waves. However, when looking into displays for our project, we found that many of the displays came with a connection for audio output or built in speakers. Since we are outputting video through an HDMI cable which carries audio with the video, it would be easier and more cost effective to use a display that has included speakers.

3.3.8.2 Microphone

For our project, we have decided to use the I2S MEMS microphone – SPH0645LM4H. We chose this microphone because of the many benefits of the I2S standard. Not only would this microphone be simple to implement into our project, the output signal is also purely digital. This eliminates the need to use an ADC in the design and would allow us to quickly send our audio signal to the COM module and sent to the cloud transcription service.

3.3.8.3 FPGA

For the brains of the I/O board, we have selected the Altera Cyclone IV – EP4CE22E22C8N. This FPGA has a large number of logical elements to ensure that we can implement our design. Furthermore, the Cyclone has many I/O pins which would be perfect for its implementation in our project. We can run all the necessary peripherals in and out of it as needed with many more unused pins to be used as needed. The Cyclone also comes with 594 Kb of embedded memory, which will easily allow us to implement FIFOs for communication to the COM unit. The chip is also rather affordable, costing only \$35 USD. The only downside that we could see for this chip was that it is from Altera, which is a vendor that we are not used to working with, meaning we will have to learn how to use the Quartus II software to work with the chip. However, even with this downside, the Cyclone IV – EP4CE22E22C8N was the best choice for the application that we have.

3.3.8.4 COM

For the COM unit in our design, we have selected the budget-friendly Jetson Nano from NVIDIA. We decided to purchase the Jetson Nano as it is a powerhouse of a COM for the price of around \$100 USD. It has a quad-core ARM processor and 128 GPU computational cores which allow this COM to deliver a lot of power despite its small form factor. Furthermore, the Jetson has an unmatched selection of I/O possibilities that make it simple to interface with some of the peripherals necessary for our device. It has 40 GPIO pins that we can use to communicate with the I/O board that we plan to design, as well as having 4 USB 3.0 pins that can help interface with peripherals as needed.

Our design will also require quite a bit of storage space, for the NMT model and well as the sign language models that we will be creating. Because of this, the expandable SD card storage is perfect for our application as it will ensure that we have adequate storage for everything that we need. Additionally, we can add external hard drives via the USB 3.0 ports as needed. Finally, the Jetson Nano has HDMI connectors that will allow us to natively pass the video of the sign language models and translated text. Since HDMI protocol includes both audio and visual data transfer, we are also able to transfer the machine-generated text-to-speech to the speakers on our display.

3.3.8.5 Display

For the display of our project, we chose the Sceptre E205W-16003S LED Monitor. We chose this display because of the many features that it brings to the table for such a low

price. It is a 20-inch display with a resolution of 1600x900 and a refresh rate of 75 Hz, which is perfect for our application. It will provide clear video output for users to see the sign language models and translated text. Furthermore, this monitor has built in speakers that will allow us to output the audio that we receive from passing our translated text through a text to speech service. This display is perfect for our application and our budget, costing only \$59 USD.

3.3.8.6 Controller

For users to control our project, we have chosen to go with a remote style control. This will consist of two parts: the first part is the transmitter, the Adafruit 1391. This is a two-button RF remote that will transmit a signal at 315 MHz. The second part is the Adafruit 1097, a four-pin toggle type receiver attached to the I/O board that will capture the signals from a remote at 315 MHz and turn on and off the respective pins. We will have the first button control whether the microphone is recording, and the second button will control the mode in which the device would be currently operating. We will also have buttons on the device itself that will allow for the device to be reset and turned on/off. Table 14 shows a full summary of all of the parts that we have chosen for our project and their cost.

3.4 Final Summary of Selected Parts

The system is composed of many different components that connect and interface together to form the final product. This section provides technical details for each one of these components. Given the time frame, implementation of the GPU-FPGA based ASLBoT was not feasible. Also, the I2S microphone was not able to be implemented in the final product since the MCU was incompatible with the I2S serial protocol.

3.4.1 Final Microcontroller

The brain of the printed circuit board is a TI MSP430FR6922IG56R. The MSP430 family was chosen for its universal serial communication (USC) port, high capacity memory, and large number of general-purpose input and output pins. The chip runs at 16 MHz allowing for multiple operations to be done in quick succession, which is a vital quality for our system. The Universal Serial Communication port supports I2C, SPI, and UART serial communication. The integrated development environment, Code Composer Studio, along with an MSP-EXP430FR6989 programmer, was used to develop, debug, and program the chip.

3.4.2 Final Single Board Computer

The powerhouse of the system is the UDOO x86 II Advanced Plus, which is a single board computer. This board was chosen for its x86_64 architecture which was required to run Unity-based applications such as ASLBoT, the application used in this project. The board features an Intel Celeron quad-core processor, 4G of RAM, an Intel HD Graphics card, and an Arduino Leonardo MCU that is connected to the processor by an internal USB

connection. In addition, the board supports serial communications through the pins with protocols such as I2C, UART and SPI. The UDOO x86 II has 32 GB of internal storage with a microSD card slot to expand the memory, if desired. Furthermore, the board features several interfacing options such as USB 3.0, HDMI, and Arduino pinouts. The integrated Arduino unit allows for easy communication between the microcontroller unit and the Unity application running on the UDOO board.

3.4.3 Final Microphone

The Blue Snowball microphone is used to record audio snippets to be processed and translated. This microphone was chosen because of its easy-to-use digital interface and seamless integration into the system. Its wide frequency range of 40 Hz to 18 KHz ensures that the user's voice could be picked up. It also features different audio detection patterns that allow for this microphone to be used in a variety of situations. The Snowball is connected and powered via USB and is plugged directly into the UDOO. The microphone does not require drivers to use, making setup effortless.

3.4.4 Final LCD Display

The LCD display for this project is a 2-line, 16-character display with blue backlighting. It takes a 5-V power supply and is used in 4-bit mode. Built into the module is a HD44780 controller, which allows for the microcontroller unit to program it. The display is used to indicate the status of the microphone to the user.

3.4.5 Final IR Sensor

The IR sensor for this project is a TSOP38238. This sensor can be powered at any voltage between 3 and 5 V and is tuned to 38 KHz making it perfect for the product. Also featured is improved ambient light and noise immunity, improving the final product's use in classroom settings. Furthermore, the IR codes are automatically demodulated as they arrive at an IR diode, allowing the signals to be much easier for the microcontroller unit to decode.

Table 14. Total Cost of Final Device.

Component	Part Selected	Cost
<i>SBC</i>	UDOO x86 II Advanced Plus	\$ 176.00
<i>Power Supply</i>	12V 3A Banana Plug AC Adapter	\$ 8.90
<i>MCU</i>	MSP430FR6922	\$ 5.33
<i>Microphone</i>	Blue Snowball	\$ 49.99
<i>IR Remote</i>	Adafruit Mini Remote Control	\$ 4.95
<i>IR Receiver</i>	TSOP38238	\$ 1.95
Total Cost		\$ 247.12

4. Related Standards and Design Constraints

In this section, we will discuss standards and design constraints that may pertain to our project. The first part of this section will cover a wide range of standards from wireless communication standards to serial communication standards. The second part will cover the design constraints from aspects such as economic impact to sustainability. We will start describing the standards that can be featured on our device for wireless connectivity, programming languages, power supply, and serial communication protocols. Then we will proceed to discuss the different constraints that the device will face based on the chosen hospital setting such as environmental, economic, and health constraints.

4.1 Related Standards

This section gives an overview of the many standards related to the creation of our device. These standards must be considered such that our device is compliant with other similar devices on the market and do not interfere with them. Also, some of these standards are related to data handling, from which how the final device manages and stores data is paramount in terms of data privacy.

4.1.1 Wireless Communication Standards

Wireless communications empower any electrical device with the ability to communicate between different devices and/or access to different variety of products such as a smartphone trying to access Facebook over the Internet using Wi-Fi connectivity. As stated before, our device requires some form of wireless connectivity to execute certain functionalities. This can enhance the user experience plus gives more utility to the device as it may access a wide range of external functionalities that pertain to language translations.

4.1.1.1. Wi-Fi Standards

Wireless fidelity, or commonly known as Wi-Fi, is considered a family of radio technologies that are commonly used in wireless local area networking (WLAN). Wi-Fi was introduced in 1998 with its base foundation on the IEEE 802.11 family of standard LAN protocols which cover both the medium access control (MAC) and physical (PHY) layers. The wireless communication technology allows a user to navigate through the Internet at broadband speeds with the use of a wireless access point. Wi-Fi is always being regulated and thoroughly tested by the non-profit organization called the Wi-Fi Alliance which overviews electrical devices that are “Wi-Fi certified” by passing a series of testing standards.

This regulation ensures that the user can interconnect between Wi-Fi certified electrical devices. Due to its ease of use and the extensive use of the Internet, Wi-Fi has become one of the biggest wireless communication technologies in the market having approximately 580 Wi-Fi integrated circuits chips ship annually. Furthermore, Wi-Fi can be found in a

wide range of devices from Personal Computers (PC) to smartphones. Finally, there are a wide range of versions of Wi-Fi that are dictated by the IEEE 802.11 standard, specifically the Physical Layer and MAC, which will further explore in this section.

IEEE 802.11 Physical Layer, based on the Open Systems Interconnection (OSI) model of the computer networking as seen on Figure 31, is the lowest layer. Composed of direct hardware transmissions, the physical layer defines the overall means of transmitting raw bits. The physical layer provides an interface in which the transmission medium can transmit bitstreams grouped into code words and converted to a physical signal. The requirement specifications of frequencies, the line code and electrical connectors are defined in this layer. Its overall purpose, on the OSI model, is to provide a means of translation from logical communications requests, provided by the data link layer, to hardware-specific operations to allow transmission or reception of signals.

IEEE 802.11 MAC, based on the OSI model, is one of the sublayers that belong to the data link layer, the other one being logical link control (LLC) sublayer. The MAC provides flow control and multiplexing for the transmission medium that the physical layer employs which allows for a smooth transition between high-level frames from the LLC sublayer to the physical layer. Its overall purpose on the OSI model is to translate higher-level frames into frames that are appropriate for the transmission medium. In addition, it adds control abstraction so that the upper layers are unaware of the complexities of the physical link control. However, when it comes to the Ethernet and Wi-Fi, MAC additionally is required to provide both full and half duplex communications with the addition of error frame protection.

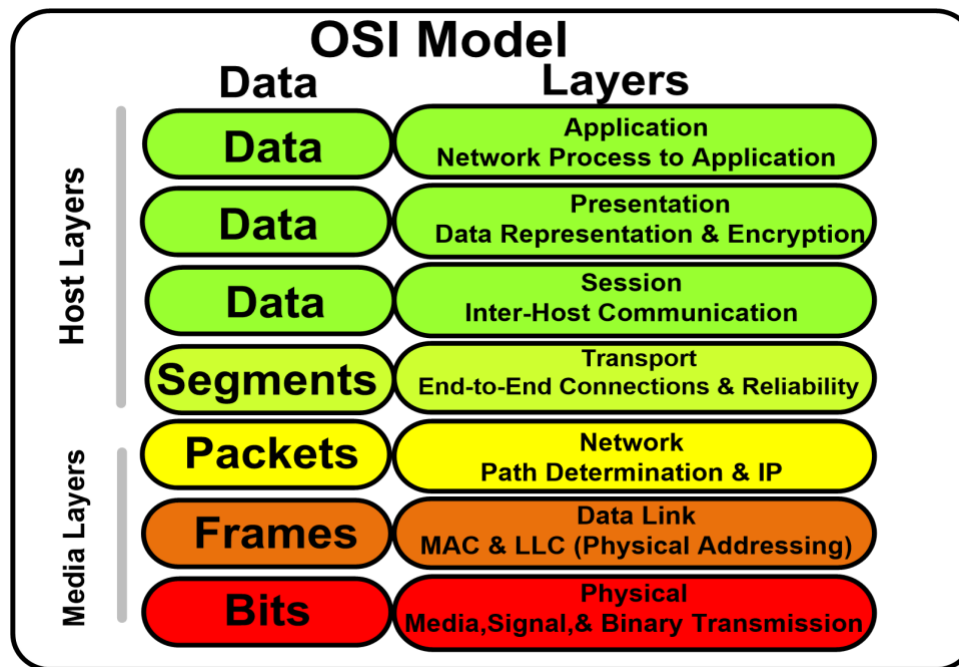


Figure 26. Organization of the OSI Model. (Public domain.)

4.1.1.2. Consumer Infrared Standards

Consumer Infrared (CIR) is a category of electrical devices that can wireless communicate using the infrared section of the electromagnetic spectrum. It can be found in most common electrical devices such as remote controls, laptops and PCs. The functionality, performance and utility of CIR varies from the protocol being employed since they are not standardized. This gives the CIR a lot of freedom in functionality as it can serve as a type of command to the television by remote controls or can be used to navigate through the web by PCs. CIR devices can bring a plethora of issues due to lack of standardization such as:

- Universal remotes that do not adequately control the target device
- Inability to control more than one device of the same type unit
- The need to own multiple distinct remotes

4.1.1.3 Bluetooth Standard

Developed as a “short-link” radio technology, Bluetooth was created by a group of engineers from Ericsson Mobile in Lund, Sweden. Another wireless technology that uses the 2.4GHz to 2.485 GHz ISM bands, Bluetooth is known for data exchange between electrical devices in a personal area network (PANs) using UHF radio waves. Bluetooth is one of the most popular wireless technology devices on the market as approximately 920 million units are sold as Bluetooth IC chips annually. It used to be regulated under the IEEE 802.15.1 Standard, but management move to the Bluetooth Special Interest Group (SIG) whose main interests is to oversee the development, management of Bluetooth profiles while protecting their trademarks.

Used primarily for low power consumption, Bluetooth employs a certain profile that compatible devices are able to interpret for management of the communication and parameters between the devices. This allows for ease of re-connectivity and wide range of different applications for devices. This feature allows Bluetooth to be applied to a wide range of different devices such as wireless headsets, wireless control of electrical devices such as smartphones, and portable wireless speakers, and wireless streaming of data between devices between other numerous applications.

4.1.1.4 Zigbee

Zigbee is a low-cost, low-power, low-bit rate communication wireless mesh networks (WMNs). Developed by the Zigbee Alliance, The Alliance also is in charge of maintaining and publishing the Zigbee Standard. As seen in Figure 32, its architectural stack is based on the IEEE 802.15.4 short-range communication standard for low-rate wireless personal networks (WPANs). While the IEEE 802.15.4 standard provides the physical layer and the MAC sublayer, Zigbee provides coverage to the upper layers of the stack. Due to its intended functionality as a low-throughput, low-power, low-cost applications, Zigbee is much simpler than other wireless protocols such as Bluetooth and Wi-Fi. This simplicity allows for Zigbee to be used in sensor networks, cyber-physical systems, home automation and smart buildings.

There are three different types of Zigbee devices which are the Coordinator, the Router, and the End Devices that are present in every Zigbee network. The Coordinator is the main device in which its purpose is to create the network and allow other Zigbee devices to join in. Other tasks are to gather all the data that is being transmitted and assign short addresses to newly joined devices. The Router works as intermediate devices in which join already existing networks and relay packets. Finally, the End Devices which only acquire data and usually enter sleep mode to save energy. The Zigbee Standard three different network topologies which are star, mesh and tree topologies which allows different setups for data transmission between devices.

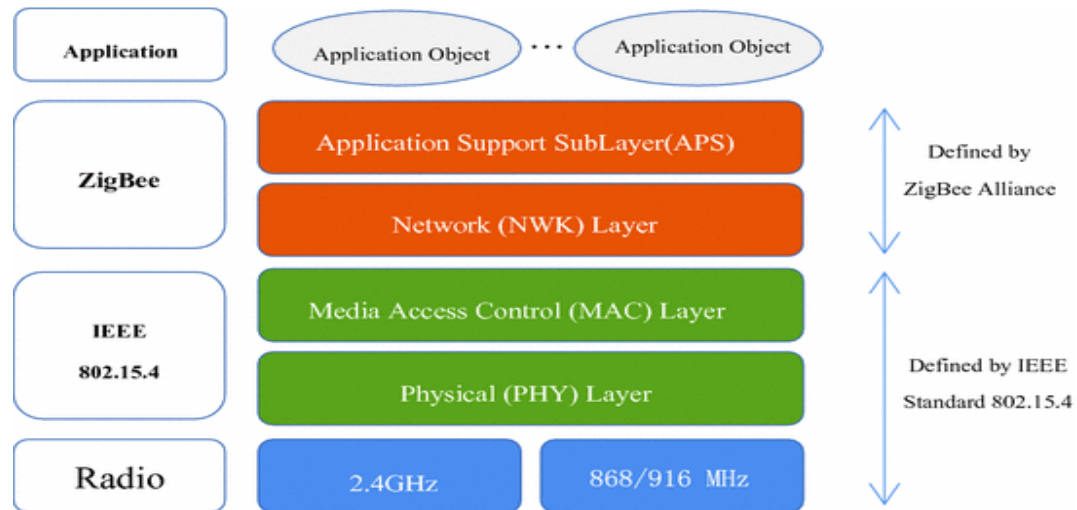


Figure 27. Zigbee Stack Architecture. (Public domain.)




4.1.2. Unicode 5.0 Standard

Unicode 5.0 is a universal character encoding standard for characters that can be used on any computer and transferred internationally. Each character is assigned a specific numerical value; this value can be compressed into hexadecimal (or octal) bases for ease of use. There are three encoding forms for ASCII: UTF-32, UTF-16, and UTF-8. The number associated with each encoding form corresponds to its bit size. Most computers are preset with UTF-8, which can utilize up to 2⁸ (256) characters, but the extended bit sizes are required for international use, as certain characters cannot be displayed using UTF-8 alone.

The Basic Multilingual Plane (BMP) comprises the entirety of the UTF-16 encoding form, which contains the majority of Unicode code points used internationally. The current standard, Unicode 5.0, contains 99,024 unique characters, with 70,229 characters coming from East Asian languages alone. The Unicode library is constantly expanding with the inclusion of many ancient and experimental languages.

To save space, code points are given to diacritics alone, from which multiple diacritics and a single alphabetic character can be combined into one glyph, for example “ȧ” individually with hex codes [0061 0308 0303 0323] can be combined into the glyph “ă”.

However, many common characters with diacritics are given their own unique code point. Although Unicode is a mapping of unique characters, the displayed character may differ between computers.

This is due to the use of different fonts and typesets: although the same character may have the same code point, the letter ‘a’ in Times New Roman, for example, will show up differently in Wingdings (‘’) or MS Mincho (‘a’), but its Unicode code point remains the same. Likewise, the same code point can be represented in different font sizes: the character ‘a’ in 12-pt font would have the same code point as a character in 5-pt. (‘’) and 17-pt. (‘’) font sizes. Also, within the BMP, over 6,000 unassigned points may be used for special characters not already in the Unicode library. Another 131,000 unassigned points within the UTF-32 may also be used. These unused points are often used for logos or privately used characters commonly used on a specific computer/network [86].

Unicode 5.0 conforms to a variety of standards, including ISO/IEC 6937, 8859, and 8879. International standards are also covered for Unicode for that country’s respective contribution to the Unicode library.

4.1.2.1. American Standard Code for Information Interchange (ASCII)

ASCII is a specific substandard of Unicode 5.0 that deals primarily with characters used in the English language. ASCII was introduced in the Unicode 1.0 and has not changed much since its implementation. ASCII utilizes the UTF-8 encoding protocol, which enables the computer to read each character byte-wise. From this, a computer will read a text file saved as UTF-8 byte-wise. However, when characters not found in the standard ASCII library are used in a text file, the file will need to be saved in UTF-16 or UTF-32 encodings. The computer will read these files two or three bytes at a time, respectively. Converting files from one encoding type to another will display a text file that may seem corrupted but is actually mis-encoded. ASCII complies with the ISO/IEC 8859 standard.

4.1.3 Python Programming Language Standards

Python is a programming language that is optimized for exploratory code; that is, Python is useful for benchmarking code and analyzing the state of the GPU, CPU, etc. Although most operating systems have Python pre-installed, this version of Python is insufficient for development. Installation of Python and its third-party extensions (such as Pip and PyTorch) are all implemented through Command Prompt.

Python, like other programming languages such as C and Java, has its own libraries and syntax. Code written in Python is saved with the “.py” extension and is executed through Command Prompt with its respective arguments.

4.1.4 Compute Unified Device Architecture (CUDA)

CUDA is a computing platform developed by NVIDIA that allows users to execute complex and resource-intensive programs on their GPU instead of their CPU. The utilization of the GPU for these resource-heavy programs instead of the CPU is due to the contrasting architecture between the two; the GPU is specialized for graphics rendering with its high allocation of transistors solely for arithmetic logic units (ALUs) and having a much smaller cache and control in return.

However, the use of the GPU instead of a CPU via CUDA can only be achieved if the GPU is a CUDA-enabled device; that is, AMD GPUs and some NVIDIA GPUs do not contain CUDA cores and will not allow CUDA to run software via those GPUs. The most common CUDA-enabled GPUs contain either the Kepler, Fermi, or Tesla Architectures.

The CUDA software comes preset with C compatibility for interaction with CUDA, although there is support for other high-level programming languages such as C++, Fortran, Java, Python, and directives such as OpenACC.

4.1.5 Hardware Description Language - Verilog Standards

This section addresses the standards set by Verilog in terms of how hardware descriptive languages are utilized, especially in the realm of FPGA programming and design.

4.1.5.1 HDL

Hardware Description Languages (HDLs) are a type of programming language that allows us to describe digital systems. Considered to be a Computer Aided Design (CAD) tool, it allows for designers to create and synthesize modern day digital system. These types of programming languages are easily read by both humans and machine making code writing easier on developers and allows for faster execution and translation times. HDLs make up a huge portion of the tools used to create modern day integrated circuits as they allow engineers to describe the ever increasingly complex digital circuits being created.

Furthermore, HDL programs are easily debugged with test benches and simulations. These test modules can be created from the code allowing for developers to save cost and time when it comes to testing. When the code is ready to be downloaded onto and used in actual hardware, it is synthesized and then implemented onto the hardware. Synthesizing is the process in which the code of the HDL program is translated into the logical gates and flip-flops of the digital circuit it describes.

4.1.5.2 Verilog

Verilog is a widely accepted and used HDL and became an IEEE standard in 2005. Verilog contains many built-in features for IC development such as logic gates. Furthermore, Verilog introduces a concept not seen in many programming languages called nets. These

nets allow values to continuously be assigned to them instead of having to call an assignment for a new variable. Along with nets, Verilog also features classic variables that software designers are used to.

Designs created in Verilog are made up of modules, much like Object Oriented Programs are made up of classes. Modules can be instantiated inside of other modules to make use of the logic inside of them. Furthermore, designs have a set of inputs and outputs that allow interfacing between other designs connected to them. Verilog brings three different types of abstraction levels that allow for developers to focus at the Gate level, Behavioral level and Register-Transfer level.

The Gate level is characterized as a system described by logical links and timing properties. In this type of level, we only witness logical values and usable predefined logic primitives such as AND, OR, XOR gates. The Behavioral level is characterized as a system described by usage of algorithms. These algorithms are sequential in nature which are a set of instructions that are executed one after the other. Finally, there is the RTL which is characterized as a system as a circuit design using data transfers between registers. RTL design is strictly more focused on exact timing bounds which means certain operations can be only executed at certain times.

4.1.5.3 System Verilog

System Verilog is another standard based off of the original Verilog standard as well as several other extensions that were created for Verilog. This extension of the original standard was created to support the ever increasingly complex designs for digital circuits as well as making verification much easier. Originally, Verilog did not support verification and therefore, engineers had to switch between Verilog and a verification language for their verification process. It was made into IEEE standard 1800 in 2008.

Like Verilog, it is a hardware description language that is used to create, test, and aid in the implementation of digital circuit designs. System Verilog also allows for use of test benches that let designers quickly verify their design. Furthermore, System Verilog is extensible with APIs allowing designers to use the language in a way that works for them. In Figure 28, we show a standard testing environment of System Verilog Hardware Description Language. It can be noted in the figure that the testing is done on an expected scoreboard versus the testing scoreboard for accuracy and performance.

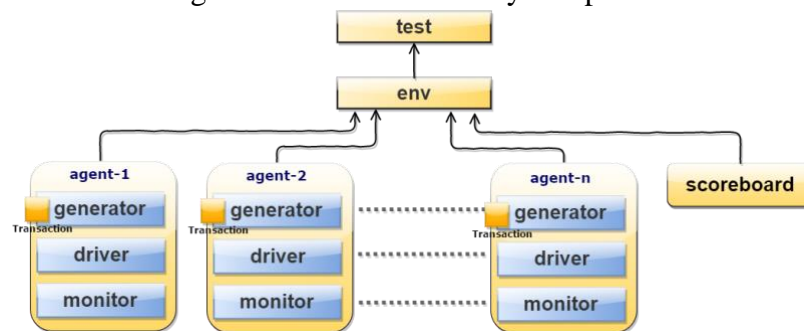


Figure 28. Standard Testing Environment for System Verilog. (Public domain.)

4.1.6 Machine Translation Benchmarking Models

There are several algorithms that are used to evaluate the performance of a certain trained machine translator. Although these metrics are not standardized, the overall algorithm used in the production of these scores are widely accepted in the computer science community. In the benchmarking algorithms discussed here, the scores are given between 0 and 1 which, when used as the power of 10, gives the overall percent accuracy of the machine translation model.

Some scores utilize negative scores (such as in the range from -1 to 1); here, the negative sign is used solely to identify a high dissimilarity between the predicted sentence and the actual sentence. The negative sign is removed to calculate the numerical percent (in)accuracy. Although each of these metrics are universally applied to any form of machine translation (gene sequencing, for example), these scores can also be readily applied to neural machine translation for language processing.

4.1.6.1. Bi-Lingual Evaluation Study (BLEU) Score

The BLEU score is a predictive score that is primarily used in machine translation models for language processing. The calculation of this score is the simplest of the models described here. The metric for this model differs with regard to how many words are considered as one parameter: one-word metrics are called unigram, etc. In this model, there are two general methods of determining the score. For a unigram metric, there is the standard precision; here, the words that are provided in a candidate sentence are matched to reference sentences. If one word in the candidate sentence is matched to a word in the reference sentence in the same position in their sentences, it is considered a match. This can be extended to bigrams and trigrams, in which a pair or trio of words, respectively, are used as the matching parameter between a candidate sentence and a reference sentence.

The overall standard unigram precision score is calculated as the number of matches divided by the total number of words in the candidate sentence. However, with these parameters, the candidate sentence “the the the the the the the” and the reference sentence “the cat is on the mat” would produce a 7/7 (100%) match. To avoid this false-positive score generated from a “high-precision” but low-accuracy sentence, a modified unigram precision is created; here, if one word is matched between the candidate and reference sentences, that specific word in the reference sentence cannot be used again. In this case, the two previous sentences would result in a modified unigram precision of 2/7 (28.57%).

4.1.6.2. Prediction Performance (PRED) Score

PRED is a collection of different performance metrics, but the overall score is a log likelihood of the produced translation being the correct translation through a specific machine translator. Within the PRED score are the regression factor (R_2), normalized root-mean-square error (NRMSE), and the Matthews correlation coefficient (MCC). For the OpenNMT model, there are three different PRED scores produced; the PRED SCORE is the overall log predictive score of the generated sentence. The PRED AVG SCORE is the

average log predictive score of each word in the sentence; here, the number of words in the sentence influence this score. The PRED PPL is the perplexity of the sentence. Here, the perplexity is defined as the exponential of the negative PRED AVG SCORE.

R₂ is a measurement of the prediction accuracy of the MT model with one word in the sentence altered. Because of this dependence on word count within a sentence, this metric is biased toward word count; using the R₂ metric in higher word-count sentences will ultimately result in a lower PRED score. There is an alternative R₂ method, called “adjusted R₂”, which uses the word count as a parameter in calculating the overall PRED score; here, the PRED score can be calculated without a word-count bias. This value ranges between 0 and 1, which correlates to a log score of 0 to 100. The calculation of the R₂ value is shown in Figure 29A. The calculation of the adjusted R₂ value is shown in Figure 29B. In this figure, n refers to the number of words being evaluated, and p refers to the number of parameters having been evaluated.

$$R\text{-squared} = 1 - (\text{sums of squares error} / \text{sums of squares total})$$

Figure 29A. Calculation of the Regression Factor. (Public domain.)

$$\text{adjusted R-squared} = 1 - ((1-R_2) * (n - 1) / (n - p))$$

Figure 29B. Calculation of the Adjusted Regression Factor. (Public domain.)

NRMSE is another parameter which takes the root-mean-square of the differences between the observed and simulated values. The simulated and observed values are compared for each word (or lack of a word), and these values are summated; the summation is then modified with the root-mean-square process to produce the value between 0 and 1.

The general calculation for the NRMSE is shown in Figure 30. Here, *N* refers to the number of terms being compared (word count in a sentence); *S* and *O* correspond to the simulated and observed values, respectively; *nval* refers to a normalized value; this value can correspond either to the range (*O*_{max} - *O*_{min}) or the standard deviation of the observed values.

$$nrmse = 100 \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (S_i - O_i)^2}}{nval}$$

Figure 30. Calculation of the Normalized Root-Means-Squared Error. (Public domain.)

MCC is related to the X₂ (chi-squared) calculation in its production of positive similarity and negative similarity (dissimilarity). Dissimilarity is indicated using a negative sign as explained earlier. Since this metric weighs each entry as a positive or negative correlation, the detection of false positive and false negative results is possible. The calculation of the MCC in general is shown below in Figure 31. In this case, X₂ is the chi-squared value of the data set, and n is the number of observations.

$$|MCC| = \sqrt{\frac{\chi^2}{n}}$$

Figure 31. Calculation of the Matthews Correlation Coefficient. (Public domain.)

4.1.6.3. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) Score

ROUGE is another metric system that is primarily used for the evaluation of language processing machine translation models. Within this package are five different metrics. The first metric, ROUGE-N, is a replica of the BLEU system; here, the 'N' stands for the number of grams being evaluated in the system, so ROUGE-1 refers to unigram measurements and ROUGE-2 refers to digram measurements, and so on.

ROUGE-L awards the highest score to the longest (here, 'L' stands for "longest") sequence of matched words between the candidate and reference sentences. ROUGE-L differs from ROUGE-N in that the longest matching sequence can be found in any position of the sentence. ROUGE-L also only seeks the sequence in which the words appear; that is, this sequence can be separated throughout the sentence yet still be considered a sequence. ROUGE-W is an extension of ROUGE-L. In the comparison between two sentences, the sentence with the sequence of words consecutively is awarded a higher weight (here, the 'W' stands for "weight") than another sequence of words that is dispersed throughout the sentence.

ROUGE-S is the measurement of skip-bigram co-occurrences. A skip-bigram is a pair of words in a sentence; the second word is always after the first word in this bigram, but these two words need not be consecutive in the sentence. For example, in the sentence "I have two dogs," the skip-bigrams would be "I-have," "I-two," "I-dogs," "have-two," "have-dogs," and "two-dogs." Using this as a reference sentence, ROUGE-S would award the highest score to the candidate sentence with the highest amount of these skip-bigrams co-occurring between it and the reference sentence.

ROUGE-SU is an extension of ROUGE-S with the addition of a unigram-weight. This avoids the false-negative awarding of a reference and candidate sentence where one sentence is the reverse-word-order of the other. This unigram weight is different than the ROUGE-1 or the BLEU-1 scores in that the matched words need not have the same position in the sentence. Since "I have a dog" and "dog a have I" would receive nothing in the ROUGE-S score, for example, this pair would still receive a 1 (100%) for the unigram score, since all the words in the reference appear in the candidate.

In our benchmarking of our OpenNMT model, we plan to use a combination of each of these metric systems to evaluate the accuracy of our model. Note that since these metrics are language- and algorithm-independent, we can apply these metrics not only to our English-to-Spanish translation model but also to our English-to-English Gloss translation model.

4.1.7 Privacy and Data Storage Standards

For the transmission and processing of data in a medical setting, there are national standards in place to maintain the confidentiality and protection of patients' rights.

Although different countries may have laws under different names, each country's data privacy laws follow the same principle of regulating the transmission and sharing of confidential patient data.

4.1.7.1 Health Insurance Portability and Accountability Act (HIPAA)

The HIPAA standard is the US standard for maintaining confidentiality in medical settings. This standard is separated into two clauses. One clause called the "HIPAA Security Rule" regulates the susceptibility of unauthorized individuals to obtain private health information and the measures taken by the original data holder to prevent such a breach. In conforming to the HIPAA Security Rule, those with original possession of such private health information are obligated to evaluate potential data breach risks in the information that they choose to transmit or store. For long-term protection of this data, personnel are hired by organizations to perform regular data assessments. Staff who produce or come in contact with this private health information are informed and trained on proper data storage and transmission protocols to prevent or minimize data loss.

The other clause called the "HIPAA Privacy Rule," however, addresses the balance between meaningful use of private health information and how and when this private health information is acquired. Explained in this clause indicates that any form of information that could identify the following are not permitted to be shared: mental or physical state of a patient, whether past, present, or future (prognosis); schedule or access to health care; and payment methods toward health care (insurance).

These potentially identifying parameters include full name and birthdate, social security number, street address, and sex/race/ethnicity. Note that within these parameters, giving sex/race/ethnicity, birthdate, or parts of a name are individually not considered identifiable parameters; it is, however, the combination of two or more of these parameters that can potentially identify an individual.

For the compliance of our design with the HIPAA standards, we intend not to store any conversations. During the training of our NMT model, the data that we use as training may include examples of private conversations. This data will be scrutinized to ensure that no potentially identifiable information is contained within these data sets.

4.1.8 Serial Communication Standards

Serial communication is the means of transferring data between different components in a system. In some instances, the data can consist of one line such as power on/power off or can consist of a bus (such as file transfers). Serial communication is typically communication via hardwiring; communication via electromagnetic waves is considered "wireless" communication, explained elsewhere.

4.1.8.1 Inter-Integrated Circuit (I2C) Standard

The I2C bus specification is registered under the UM10204 standard. This method of serial communication was developed by NXP Semiconductors for bidirectional 2-wire bus communication between integrated circuits. The two wires consist of a serial data line (SDA) and a serial clock line (SCL). There are five different modes within the I2C standard. The summary of the transfer speeds and the directionality of each mode is visualized in Table 15.

Table 15. Summary of the Various I2C Standard Data Transfer Speeds.

Data Transfer Mode	Maximum Transmission Speed	Directionality
Standard Mode	100 Kbps	Bidirectional
Fast Mode	400 Kbps	Bidirectional
Fast Mode Plus	1 Mbps	Bidirectional
High Speed Mode	3.4 Mbps	Bidirectional
Ultra Fast Mode	5 Mbps	Unidirectional

Additionally, The I2C has a bus topology which provides flexibility by allowing the addition of extra peripherals to a single bus. In order for the topology to work, every device that uses an I2C unique 7-bit address so there are differences between each device. The I2C bus can be seen in Figure 32A.

I2C follows a master-slave operation in which the master device starts the transmissions that allows the master to read from and write to other devices. Moreover, the master device is also in charge of driving the clock signal which is necessary for timing the start and stop signal. The signals are transmitted during intervals that transmissions of data bits do not occur such that the signals will not get mixed. The I2C master-slave communication is shown in Figure 32B with the provided example.

Notice that ACK is the shorthand for “acknowledgement” which is an indication pin the signal was received successfully. This notation is used by both master and slave devices. The master will start by sending a signal to signify reading or writing to the target device using the unique I2C address stored inside the target device. An ACK signal will be sent back to the master if the device was able to receive the signal successfully. Afterwards, the data is transmitted or received, and the communication is stopped once the master sends the “stop” signal.

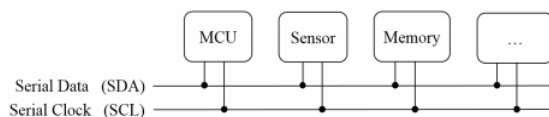


Figure 32A. The I2C Bus. (Public domain.)

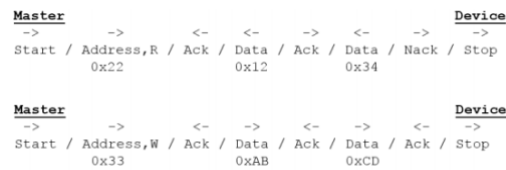


Figure 32B. Master-Slave I2C Communication. (Public domain.)

4.1.8.2 Serial Peripheral Interface (SPI)

The serial peripheral interface is a four-wire serial communication method. There is no formal standard for this type of bus; this method was developed by Motorola in the 1980s and became widely used in the digital communication area. Since there is no formal standard for this type of communication, there have been several different implementations and designs utilizing this communication protocol, such as JTAG, UEXT, and Secure Digital. This communication protocol employs a simple Master-Slave scheme in which the master communicates with one or more devices just like I2C.

Since there are two wires (one for receiving and the other for transmitting), the protocol is considered to be a full duplex. The signal scheme by the protocol is synchronous in nature, since the master generates the clock signal and transmits it to the external devices. In order to synchronously communicate with a single device at a time and not allow miscommunication between devices, the master uses a fourth wire called the chip select (CS) signal. If the CS signal is high, the master establishes a communication path with a device. When the CS is low, a high impedance is placed by the external device to disable communication. This mechanism is also used to ensure that the Master is only communicating with a single external device at a time. In total SPI communication protocol offers four wires: Serial Data Out, Serial Data In, Serial Clock and Chip Select.

However, this may not be the case if there is a master connected with a single device. In this case, a 3-pin SPI can be implemented instead. Figure 33 shows a normal configuration of master and external device using SPI communication. Noticed that, as previously stated, since SPI does not have a formal standard the Serial Data Out and Serial Data In pins have different names. In this case the names are MOSI and MISO, which means Master Output Slave Input and Master Input Slave Output, respectively.

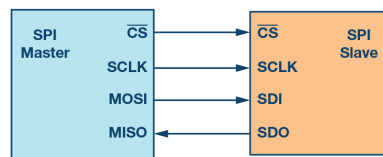


Figure 33. SPI Communication Setup. (Public domain.)

The communication scheme works as follows: the SPI uses two shift registers (SRs), one provided by the master and the other by the target external device. Depending on the number of bits by the two shift registers (usually 8-bit), the master's SR will exchange its contents with the target external device's SR bit by bit an n-bit number of times. To

maintain stability, a D flip-flop is employed to latch the current MSB bit. This mechanism is known as latch/shift actions since we latch the bit being transmitted and then shift the bit between the SRs. Four modes of operation are defined by the SPI, which are 0/0, 0/1, 1/0, 1/1. These modes detail the usage of clock signal (clock polarity) and trigger event for latching and shifting (clock phase). The format used is polarity/phase and is explain in detail in Figure 34.

```
Polarity 0: Clock idle at low
Polarity 1: Clock idle at high
Phase 0:   Latch at trailing edge, communicate at leading edge
Phase 1:   Latch at leading edge, communicate at trailing edge
```

Figure 34. SPI Modes of Operations. (Public domain.)

4.1.8.3 Universal Asynchronous Receiver/Transmitter (UART) Standard

The UART is an asynchronous serial communication standard in the sense that the transmitter and the receiver each has its own clock signal. This communication protocol is considered full-duplex in that data can be sent and received simultaneously through the use of a UART-enabled communication using two wires to allow bidirectional simultaneous transmission. The protocol also allows half-duplex by using one wire to transmit data in one direction. Through the use of a FIFO and sending and receiving simultaneously, the UART is analogous to a bidirectional roadway, where the flow of one set of traffic does not impede the flow of traffic moving in the opposite direction.

The UART communication protocol has a transmission pattern that is shown in Figure 35. The scheme, unlike the other serial communication protocol, is simple but effective. It uses a signal line that represents different modes of the protocol. When the signal is high, it means the transmission is idle. When the signal, after being idle for a while, drops low, this is called the Start Bit in which the UART transmission starts to communicate. After that, each bit transmitted will be considered to be data bits that are sent from the LSB first to the MSB last. Then, once the signal is switched to high again for a long time, called the Stop Bit, this signals the transmission to stop.

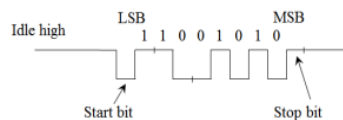


Figure 35. UART Communication Scheme. (Public domain.)

This protocol also behaves in a FIFO or queue data structure; that is, the first set of data to be sent through the UART is also the first set of data to be sent out. FIFO is analogous to a drain or a funnel, where the first amount of liquid inserted is the first to be removed through the orifice at the bottom. The FIFO structure differs from the stack architecture, which is LIFO (last-in, first-out).

4.1.8.4 Inter-Integrated Circuit Sound (I2S) Standard

The I2S bus standard is an extension of the I2C bus standard mentioned previously. This I2S bus consists of three lines: a continuous serial clock (SCK), a word-select line (WS), and a serial data line (SD). There are three different configurations regarding I2S. Either the transmitter, the receiver, or a controller is the master; in any case, there is only one master, with the other components being the slave. In each case, the transmitter sends data to the receiver, but the determination of the WS and SCK lines is determined by the master. The simplified block diagram for each case is shown in Figure 36. The word select line has a high and low level; the high level indicates transfer via one channel, while the low level indicates transfer via the other channel. The SCK frequency is determined by the master component and transfers this frequency to the slave component.

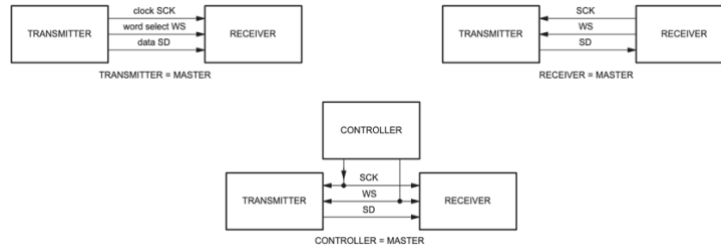


Figure 36. Different Configurations for the I2S Protocol.

4.1.8.5 Universal Serial Bus (USB) Standards

The universal serial bus standards define a serial communication protocol that enables interconnect between various devices, communication between them, and power supply from one device to another. This specification is attributed to a number of companies, including the Hewlett-Packard Company, Intel Corp., Microsoft Corp., Renesas Corp, ST-Ericsson, and Texas Instruments. The various revisions of the USB standard define a series of different connectors and connection speeds, with each revision providing a more well-designed, faster connector. The various USB speeds and their date of documentation are explained in Table 16. The implementation of the USB as a standard for interconnection between devices is meant for a simple yet universal communication protocol. The use of USB in the personal computer realm was intended for the “ease-of-use for PC peripheral expansion,” low-cost implementation, and universal availability for both device manufacturers and consumers.

Table 16. Summary of Major USB Revisions.

	Year Implemented	Data Transfer Speed
USB 1.1	Sept. 1998	12 Mbps
USB 2.0	April 2000	480 Mbps
USB 3.0	Nov. 2008	5 Gbps
USB 3.1	July 2013	10 Gbps
USB4	Aug. 2019	40 Gbps

Out of the USB hardware models, the project will use USB 3.0. One of the main reasons is because the Jetson Nano already provides USB 3.0 ports that can transmit data and charge the FPGA. USB 3.0 provides differential pairs which provide a full-duplex communication between the COM and the FPGA. Additionally, we will use the type-A connectors as we need to deliver both power to the FPGA and data transmissions between devices.

4.1.9 Audio/Video Connection Protocols

The process of sending a display from one device to another is achieved through some form of a cable. There are a wide variety of these cables: some of these cables are developed to be proprietary to a specific company, while most are developed for universal connectivity, similar to USB. In fact, while USB acts as a serial communication protocol, some displays can actually communicate using this protocol.

4.1.9.1 High Definition Multimedia Interface (HDMI)

High Definition Multimedia Interface, or popularly known as HDMI, is currently the most widely accepted and most versatile form of display communication. This cable type was introduced in 2002 and was developed by several major electronics companies, including Sony and Toshiba. As most displays are moving towards integrated visual displays with audio, the popularity of HDMI can be attributed to this ability to transfer both video and audio simultaneously. Since both the audio and video can be transmitted through the HDMI cable at the same time, many audiovisual events can be timed to occur at the same time, allowing users to watch videos with sound or follow along with a MIDI player on the screen.

The initial version of HDMI, HDMI 1.0, was introduced in 2002 as an uncompressed, fully digital audio and video connector. At its inception, the HDMI 1.0 could achieve bitrates of up to 4.9 Gbps. The first update of the HDMI was the 1.1 version in 2004, which added support for DVD audio. The 1.2 and 1.2a updates in 2005 provided support for computers, which greatly contributed to the popularity of this protocol. The 1.3 version in 2006 increased the bandwidth to 10.2 Gbps and allowed for the use of “Deep Color” 16-bit channels. This version was also able to develop reliable synchronization between audio and video, greatly reducing the instances of video lag. The later updates continued to expand upon the resolution and audio capabilities, especially in keeping up with the major technologies; for example, the current HDMI version supports 4K x 2K displays and has a bandwidth of 18 Gbps.

4.1.9.2 Digital Video Interface (DVI)

Digital Video Interface (DVI) is another form of digital display connection that was introduced in 1999. However, unlike HDMI, which is able to transmit both audio and video, DVI is a video-only transmission protocol. This connector was neglected due to the introduction of the HDMI cable shortly after the inception of the DVI.

4.1.9.3 Video Graphics Array (VGA)

Video Graphics Array (VGA) is an analog interface that was introduced in 1987. Although there were other digital connectors before this such as CGA and EGA, VGA provided a much higher resolution of 640 x 480 and was able to show up to 256 colors. Although this form of connector has been quickly replaced by digital connectors that can provide much faster bandwidths and resolutions (see HDMI), many computers still include VGA output as a “standard definition” connection, as opposed to a “high definition” connection like DVI or HDMI.

4.2 Design Constraints

This section describes the constraints that will bound our project into being realistic in design and implementation. These design constraints are formulated based on the requirement specifications of this project, on the client’s need, and outside constraints that we may face as we implement the device. Having the use of realistic constraints provides us having a direct and throughout understanding of exactly what to focus on. These constraints aim to improve society and the quality of life by applying constraint factors such as:

- Economic
- Time
- Manufacturability
- Sustainability
- Health
- Social
- Political
- Moral
- Ethical

4.2.1 Economic and Time Constraints

Our project’s estimated budget is around \$1100. Since no financial sponsorship was obtain from an outside source, all the members had to five an equal amount of \$275. The goal overall was to aim for a COM that was not greater than \$500 and an FPGA that was not greater than \$200 but with the mindset of having fast performance and accuracy of delivering translations. This allows to still pick average quality external peripheral components such as microphones, speakers, buttons and the LCD. An advantage that we possessed was the availability of the HDMI port provided by the COM which enabled us to use an LCD Monitor that had a high refresh rate. This allows for a smooth transition between ASL models. Even with the budget constraint, we are facing, if this product is implemented successfully, places that require human translations would spend less money on translations since human translator require a lot of high upkeep to keep the services. Our device comes with support of 128 different languages plus ASL translations that would be easy to assemble and deploy.

Additionally, the court cases against poor services provided, by establishments against the deaf people, could potentially decrease since our device can produce real-time accurate translations and display accurate ASL models. This can potentially save a lot of money for the establishments as court cases against poor services, for the deaf people, have reached around \$70,000. Moreover, its ease of use could make it more marketable to other settings outside of schools and hospitals. Finally, most of the components can be bought in easy to reach websites that allows for assembly of such device with ease by following our user's manual.

Time is a big constraint that our project currently faces and will keep facing. The time schedule for project completion is divided into two sections; Senior Design 1 and Senior Design 2. Senior Design 1 focuses more on the development, design and documentation of our project whereas Senior Design 2 focuses on the implementation of the device.

The time span of our project is from August 2019 to April 2020, which is basically 8 months. This short amount of time damages the performance of our machine translation, as we are training an NMT to translate from speech to ASL. This training requires a lot of dedication and time as good machine translations usually take large amounts of data sets. To meet the requirements set in this document a milestone table has been set and can be found in section 8 of this document.

4.2.2 Manufacturability and Sustainability Constraints

Manufacturability, the art of design devices for ease-of-construction and mass-production, restricts the selection of the components being used in the device. Manufacturing constraints affect us in the sense of the limited services that are provided to us. Since we are a self-funded project, most components will not be of the highest quality which shortens the range of services that we can potentially use.

Designers should always consider certain functions of manufacturing when developing a device such as assembly, fabrication, testing, acquisition, shipping and repair of the components being used. Sustainability is the art of durability and maintenance of the device. This type of constraints narrows down the component selection even further since cheap components are not as durable as the high-quality components. Ironically, high-quality component may last longer but their maintenance is more expensive than cheap components. Our project heavily benefits from having high quality components that are manufacturable, since high manufacturability brings high sustainability in the maintenance area.

One of the main goals was to deliver real-time accurate machine translations for languages and ASL and since we needed to deliver high quality translations, we based our component selection with high quality components that were manufacturable and preferably cheap to buy. Following the requirement specifications found in this document, we required high quality microphone components since we need the speech translation to acquire the audio accurately. Additionally, we needed to ensure that, if the microphone ever malfunctioned, it could be easily replaceable. The same mindset for selecting the speaker and display was

used. A more detailed selection process was discussed in section 3 in which manufacturability and sustainability were in mind when selecting these components.

However, when selecting the COM and the FPGA we faced manufacturability constraints since they were expensive even the midrange-quality components. To ensure the sustainability of these expensive devices, we decided that an electrical safe environment had to be used to minimize potential electrical damage that can be done to the boards. This can lead to the malfunction of the device and a new one would have to be bought which would cause the expenses of the project to expand. Furthermore, we bought most of the components from third-party vendors that possess fast shipping and large quantities of the components used.

4.2.3 Moral and Ethical Constraints

Our project involves the automation of the occupation of human-performed language interpretation. In this society of the constant evolution of technology and the requirement of automation for the purposes of streamlining, improving, and cost-reducing, there is always the question of whether the implication of this automated technology is truly beneficial to society; for while a new technology may improve the lives of those using the technology and those developing it, this new technology may also negatively affect others.

4.2.3.1 Automation Displacing Human Labor

One aspect of ethicality our group discussed was the potential destruction of others' current occupations. Since we are developing an automated language translation machine, one could assert that we are effectively removing the demand for organic language interpretation done by humans. We counter this with the point that natural language translation is still in its developmental stages due to the complexity not only of individual languages alone but also the complexity of attempting to describe a sentence in one language in the language of another. We also counter this claim with the notion that because this concept of natural language translation is still being developed, the demand for language interpreters has not diminished but rather shifted.

This shift is marked by language interpreters translating in their field, e.g. during press conferences, broadcasting, etc. to collaborating with scientists and engineers for the continued research and application of machine translation and natural language processing.

This concept of the balance of human versus machine labor is described by Pramod Khargonekar and Meera Sampath in four tiers. These tiers start at level 0 as the most unethical level and progress to level 3 as the most ethical tier.

In level 0, called the "cost-focused automation," ethics are completely ignored, and human labor is completely replaced by automation for the sole purpose of "economic gain." This approach to automation is considered not only unethical but also potentially fatal to the

economy, since this displacement of labor would lead to a rise in unemployment and the decrease in demand for goods and services.

In level 1, called the “performance-driven automation,” human labor is not completely forfeited for robotic automation. Tasks that would be too labor-intensive, time-consuming, or monotonous would be given to robots, while tasks too complicated or dynamic for robotic programming would be sustained by human labor. While this level of automation still maintains a certain level of human labor, this approach to automation is desired for the expedition and streamlining of labor without enhancing human labor in any way.

In level 2, called “worker-centered automation,” human labor is left completely alone for a period of time. Since this labor is performed by humans, the humanistic concepts of finding more efficient (or lazier) methods to achieve the same results to the same quality can still be done. Conversely, in levels 0 and 1, since the labor is immediately given to robots without consideration on how the process could be streamlined, this concept of pathfinding would be delegated to the ones working on the machines alone. Since robots are usually programmed to perform a specific set of tasks without room for improvement or intelligence, these robots would never consider improving their current job.

However, for level 2, once this task has been considered to be well-developed and streamlined, the task is then given to a robot that would perform the same task in the same way as the streamlined human method. Since human labor is left to flourish for a longer period of time than the lower levels, this approach to automation is considered more ethical; but since the job is still immediately taken away from humans and given to machines, the result is still as unethical as the lower levels.

In level 3, called “socially responsible automation,” automation is created for the sole purpose of streamlining work done through human labor. This form of automation can also be thought of as making machinery that is operated by humans and teaching them how to use it effectively. This level is considered the most ethical since the creation of automation not only streamlines the production of labor but also sustains the demand for human labor concurrently.

This tier system proposed by Khargonekar and Sampath can be ambiguous depending on the perspective on whose labor is being defined. This tier system only observes the final result of automation but does not consider the development, implementation, improvement, and evaluation of the automation system. Our device would be considered at level 0 or 1 from the perspective of the human language translator but level 3 from the perspective of the potential users of our device. Note that despite the eventual elimination of human language interpretation, those who used to be language interpreters would continue to work alongside engineers in the further development and refinement of our language translation machine.

If we take the example of a clinic visit that requires a translator, the language translator would be replaced by our device entirely, and the demand for language interpreters for clinic visits would cease. In this case, our device would be considered level 0 or level 1.

However, medical providers and staff would now have a new tool that would help them in their provision of medical care. This new staff would need to be trained on how to use the system, but their jobs would not be replaced entirely by our device; thus, our device could be considered level 3.

4.2.4 Environmental, Health and Safety Constraints

When designing new electrical devices, there is a lot of environmental, health and safety concerns when operating such devices. Since it can damage either the environment or even the user. To avoid this, we need to follow the constraints that ensures the device will not be harmful to anything. Our device will market to hospitals and schools, in such establishments a wide range of ages will be exposed to our device. We have to ensure that device, even if it malfunctions, does not damage the surrounding in any way.

Additionally, even if it works as intended, another functionality that we have to ensure is that the display does not hurt the eyes of the viewer. In this section, we will discuss the RoHS complaint parts, and the potential harmful effects of the backup battery, and the display.

Restriction of Hazardous Substances, or RoHS for short, was developed by the European Union and restrictions specific materials that are hazardous to the environment and that can be found in electrical and electronic products. The substances currently banned are lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls and polybrominated diphenyl ethers. These substances are harmful to the environment and pollute landfills. They are also dangerous due to occupational exposure during manufacturing and disposal. We had to select electrical components that were RoHS compliant. However, the providers that we purchased from were kind enough to label which components were RoHS complaint before the purchase of said component. This made the selection process easier.

The use of Lithium-Polymer batteries has been the cause of a lot recent commotion, since a brand line of smartphones developed by Samsung were imploding on the pocket of users. To avoid this, we need to thoroughly follow and apply the benchmarks imposed by the power standards on battery.

Additionally, we will need to safeguard against current leakages that can cause malfunctioning of the system to avoid any unforeseeable happenings with the device. Most providers document a way to safely guard against the leakages. Finally, the monitor cannot hurt the eyes of the viewer since the idea is for the audience to see the ASL translation. A normal brightness rate with a high refresh rate ensures that the viewer will have no problems in viewing the video.

4.2.5 Social and Political Constraints

The project has a lot of implications in the Social and Political constraints aspects. An obstacle that we need to answer is how our device will stand out from the rest while also using cloud services provided by the competition. As started early in the document, no

machine translator has done ASL translations using image processing techniques or anything at all. Many establishments hiring human translators have become the norm for too long and the implementation of our device can bring the end of a job that is highly used in society overall. Another question that is imposed by these constraints is the process of delivery fast accurate translations without facing latency issues.

The main purpose of our system is to deliver accurate and quality service to the deaf people. Competitors such as Google Translate and Amazon Translate do not provide translation services for the deaf people. Human translators are still being employed to do this type of task. However, a high upkeep is required to maintain the human translator and most of them work online in which a limited wireless coverage or low internet speed (since streaming of a video is required) can lead to mistranslations between two people. In hospitals and schools, this can bring a lot of issues.

In order to promote and market our device, we use high-quality cost-effective components which allows the overall cost of the device to be lower than the upkeep required by the human translators. Additionally, since we are also offering normal language translations plus the ASL translation for deaf people, establishments will find our product more affordable and attractive to purchase.

Finally, we need to consider the ease of use that the device will require. This is required because we want the establishments such as schools and hospitals to familiarize fast with the device so that usage can be quickly for real time events such as emergencies. Therefore, the installation and GUI of the device with its modes and functionalities should be easy to explain and user-friendly. This additional feature can make our product become more attractive.

Political constraints can apply to our device in the area of human replacement. Since we are marketing the device as a way to obtain an affordable translator that requires a one-time buy instead of a steady maintenance, the device will be compared in terms of performance to a human translator. This means that we need the device to translate in real-time and accurate.

Additionally, the flow of displaying the ASL models must match, if any, the way human ASL translators do sign language. Machine translations can be deployed for diplomatic use which would require for the device to support a wide range of languages for translations. For this reason, our device can use cloud services to obtain a fast-accurate translation. Moreover, using machines for translations can benefit the government and even the military with concerns over biasness.

5. Initial Project Hardware Design

This section addresses the hardware design plan from Senior Design I. Note that many of these sections, including the hardware diagrams and schematics, may have been outdated, replaced, or modified in the final hardware design, which was explained in a later section.

5.1 Initial Project Design and Component Diagrams

When we first started to design our project, we were overly optimistic with the capabilities of the technology we were using and with our programming skills. Our first idea was to use a GPU as the main brain of the system and have all the components connected to it. It would run both the trained NMT model and the graphical rendering for the product. Furthermore, we expected it to take the inputs from a microphone, remotes, and output the animations and speech.

After taking a harder look at the problem that we were trying to accomplish and the technology we were trying to use, we decided that we would need more than just a single GPU. Not only would a GPU not be able to handle everything that we were throwing at it, but programming everything in CUDA would be a steep learning curve. Therefore, we decided to change the design of the project to use both an FPGA and a GPU to complete this task. The FPGA would oversee running the NMT model and outputting the results on to the display, while the GPU would focus on the peripherals and creating the sign language graphics. Figure 37 shows the initial hardware block diagram. In this diagram, we also included which personnel of our group will work on which aspect of the project. Gustavo will work on the GPU. Jared will work on the FPGA. Michael will be working on the Stokoe notation and graphics rendering. Luis will be working on the PSU and the NMT. Any other aspect of the project will be work upon by all members of the group.

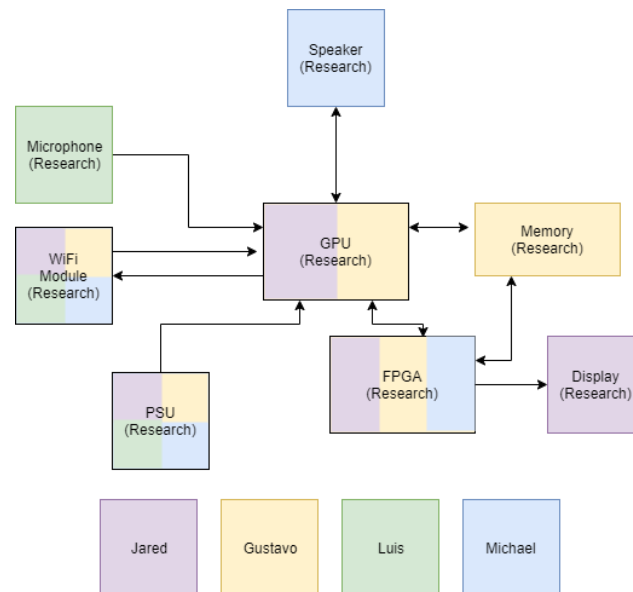


Figure 37. Hardware Block Diagram of the Second Initial Project Design.

This design was always extremely optimistic for our knowledge levels in the technology we chose to work with, mostly because we were going to try to deploy an NMT model to an FPGA board. Deploying python code into FPGA boards is something that is just now starting to emerge in the world of research and implementing an OpenNMT model created in python has never been attempted before.

Therefore, we would be completely on our own in the process of deploying our model to our system. Furthermore, we would still have to learn how to learn in CUDA to be able to use the GPU as the hub of the design and learn to make a GPU take inputs from a microphone and other peripherals. Finally, The GPU would also have to be running an operating system that could run some sort of game engine, like unity, to be able to generate and display the sign language graphical models. Despite doing research, we were unable to find instances of this type of deployment being done and therefore were forced to abandon the idea in the pursuit of a more feasible design.

After we had given it more thought, we settled on the design of having a COM unit as the brain of the project and an FPGA I/O board to handle the audio recording and button management.

With this design, we had originally planned on using a mobile battery to power the system so that we could place the system on a mobile cart. However, we ran into two issues that prevented us from using a rechargeable battery system. The first was that the Jetson would occasionally go into sleep mode with the user telling the system to shut off. We found that this was caused by the lack of a consistent voltage and amperage from the battery pack.

While this issue could have possibly been avoided by purchasing a higher quality battery pack, it would have stretched the budget thin. Furthermore, this problem would make the system extremely difficult to use, as the user would then have to stop their conversation and restore power to the system.

The second issue that we encountered while trying to use a battery pack was finding a method of powering the display that we had chosen for the system. We had chosen to purchase a 20-inch computer monitor that came preinstalled with speakers. This was cost effective as it limited the number of components that we were required to buy and also provided the users with a large, high quality image.

However, since we had chosen a full-size monitor, it has to be powered with 100 - 240 V AC, with a maximum power consumption of 30W. This type of power draw is only sustainable through an outlet plug and so we were resigned to power the system via a PSU that plugged into a wall socket.

5.2 Power Supply Design

In this section, we will talk about the different power supply requirements along with the final power supply of the system. The data gathered in this section is mostly based on the datasheets of each major component of the system plus the selected power supply system.

Table 17 shows the overall power requirements for the major components in our design.

On the DE0-Nano development board for the Altera Cyclone IV FPGA, all of the voltage inputs required for proper functionality of the FPGA are supplied entirely from a single USB 3.0 power input. Given that the DE0-Nano board uses a USB 3.0 protocol, a 5V input voltage is required at up to 900 mA through this pin. From this 5V, 900 mA input, six different voltages will need to be created to accommodate the various voltage requirements for the FPGA, including a power supply for the I/O pins and the phased-locked loop. For the VCCINT and VCCD_PLL input voltages, the input voltage can vary between 1.0 V and 1.2 V.

Table 17. Overall Power Requirements.

Device Name	Input	Voltage	Current	Power
NVIDIA Jetson Nano	USB Micro/ Barrel Jack/ Power Pin	5 V	> 4 A	> 20 W
Altera Cyclone IV - EP4CE22E22C8N	VCCINT	1.0-1.2 V	1.5 A	1.8 W
	VCCA	2.5 V	150 mA	375 mW
	VCCD_PLL	1.0-1.2 V	1.5 A	1.8 W
	VCCIO	1.2, 1.5, 1.8, 2.5, 3.0, 3.3 V	150 mA or 1.5 A	375 mW, 1.8 W, or 4.95 W
Digital MEMS Microphone - SPH0645LM4H	3V Pin	3.3 V	600 μ A	1.08 mW
RF T4 Receiver - Adafruit 1097	5V Pin	5 V	0.1 μ A	0.5 μ W

The overall configuration of this schematic is realized in Figure 38. The schematic will require the use of linear voltage regulators (LP38500SD-ADJ and LP5900SD-2.5) to produce the required voltages and corresponding currents while also maintaining that specific voltage for the I/O peripheral device.

Included with the voltage regulators are the corresponding components such as capacitors and resistors for proper functionality of the voltage regulator. Also included in the circuit are Schottky diodes (PMEG2010AEB) to rectify current and prevent current backflow. For each voltage produced by this circuit, an indicator LED will be used to show that each voltage is available. There will be four different LEDs shown because we are operating

with the following voltages: 1.2V, 2.5V, 3.3V and 5V. Figure A shows the full power supply unit schematic.

The power supply for our project consists of an input voltage provided by the power pins of a USB 3.0 port, denoted as VCC_USB. A typical USB 3.0 port supplies 5 V at up to 900 mA. As the USB is to provide input power and should not leak power the other way, two Schottky diodes (denoted as CR1 and CR2 in Figure 41) are used. The voltage at this node is the 5V node, denoted as VCC_SYS.

From the 5 V power supplies, we will use zero-ohm resistors to safeguard against possible high currents that might affect the power supply unit. To output 3.3V and 1.2V, we are using the LP38500 (U2 and U3) from Texas Instrument as selected in the parts selection section of the report in two different sets of configurations. The 3.3V output voltage is assigned to the VCC3P3 label whereas the 1.2V output voltage will be assigned to the VCC1P2. Additionally, we use the LP5900 (U1) from Texas Instrument to generate an output voltage of 2.5V. Both voltage regulators' configurations were based on the documentation provided by the Altera Cyclone IV FPGA Chip and the DE0-Nano development board.

The VCC_SYS will provide power to the main system which is the Jetson Nano and the RF T4 receiver. The VCC3P3 will be used to power on the MEMS microphone along with the I/O Banks that are configurable in the Cyclone IV (This case the I/O Banks 1, 3, 6 and 8), the banks are necessary for the configurations of GPIOs.

The VCCA voltage will be used to power the Phase-Locked Loop (PLL) analog power supply, which allows use to control the frequency of the clock of the Cyclone IV. Finally, VCC1P2 will power the digital power supply of the PLL and the core chip of the Cyclone IV. A Bill of Materials (BOM) of the PSU Schematic is provided in Table 18 with the components used in the schematic.

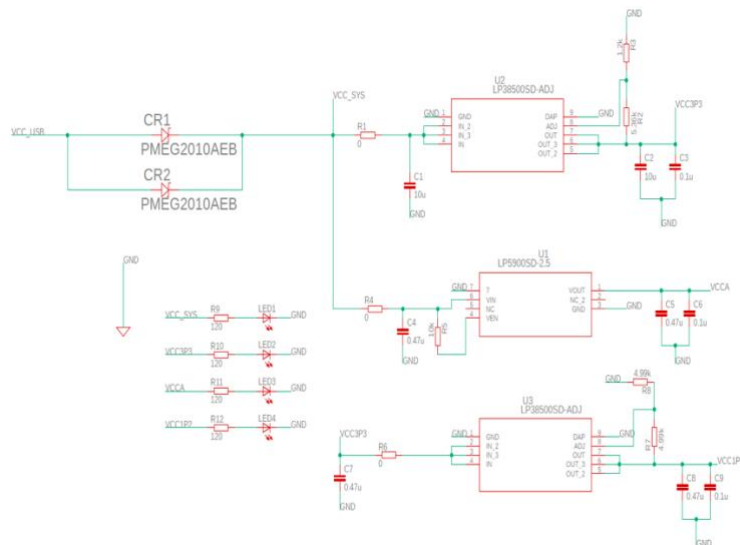


Figure 38. Power Supply Circuit Schematic.

Table 18: Power Supply BOM.

Label	Quantity	Package	Manufacturer	Description
C1, C2	2	C0603	TDK	Capacitor, 10 μ F
C3, C6, C9	3	C0603	AVX	Capacitor, 0.1 μ F
C4, C5, C7, C8	4	C0603	KEMET	Capacitor, 0.47 μ F
CR1, CR2	2	SOD523	NEXPERIA	Rectifier Diode
LED1, LED2, LED3, LED4	4	0603	DIALIGHT	RED LED
R1, R4, R6	3	R0603	VISHAY	Chip Resistor, 0 Ω
R2	1	R0603	VISHAY	Chip Resistor, 5.36 k Ω
R3	1	R0603	NIC COMPONENTS	Chip Resistor, 1.2 k Ω
R5	1	R0603	VISHAY	Chip Resistor, 10 k Ω
R7, R8	2	R0603	VISHAY	Chip Resistor, 4.99 k Ω
R9, R10, R11, R12	4	R0603	YAGEO	Chip Resistor, 120 Ω
U1	1	SDB06A	TEXAS INSTRUMENTS	Voltage Regulator, 2.5V
U2, U3	2	SDA08C	TEXAS INSTRUMENTS	Voltage Regulator, 3.3V, 1.2V

5.3 Hybrid COM with FPGA

This section entails the hardware functionality of the device by implementing a hybrid COM with FPGA architecture. We will discuss the I/O board first and then proceed to describe the different modules using FNMs.

5.3.1 I/O Board Description

Our I/O board will perform the functions of taking inputs from users and processing the data before sending it to the Jetson Nano unit, our COM. The I/O board will perform 3 distinct functions: the first function will be to keep track of its current operating mode and change the mode in which the device operates when the mode button from the RF remote is pressed. Secondly, it will oversee capturing the user's voice whenever they press the record button on the remote. Finally, it will track of the status of the unit, by having a status LED, a recording LED, and buttons for both resetting the unit and for turning the unit on and off. Figure 39 shows the software block diagram of the Verilog code.

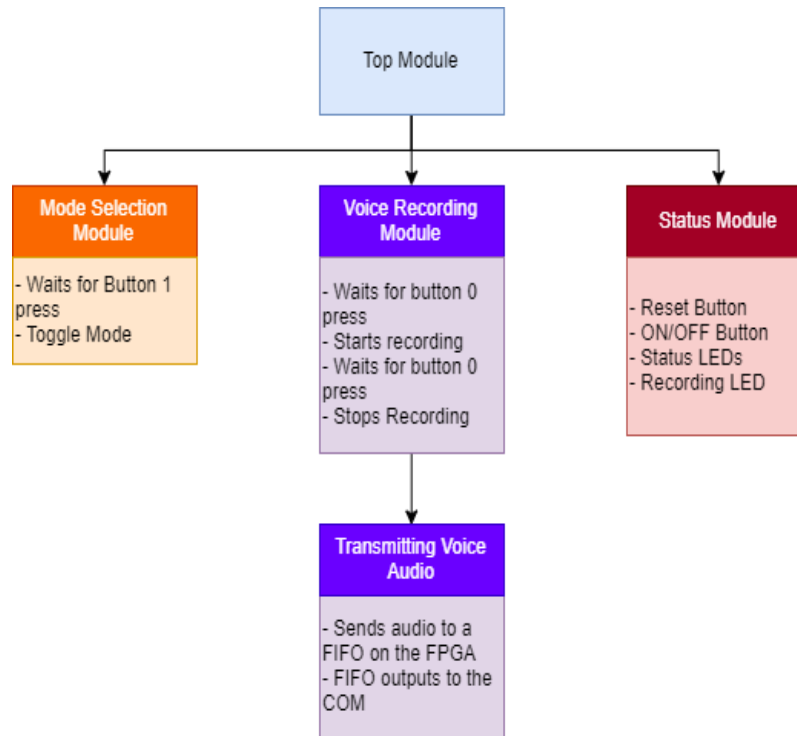


Figure 39. Block Diagram of I/O Board.

5.3.1.1 Mode Selection Module

This module of the board will be implemented using a state machine that has two states; each state representing a functional mode. The program will stay in the current mode until the mode button on the RF remote is pressed. When the mode button is pressed, it will toggle the state of the pin on the RF receiver, which will be read on the positive edge of every clock cycle. If the voltage level of the pin has changed, it will change to the respective state in the state machine and it will pass this information to the Jetson Nano over an I2C connection. Figure 40 shows the state machine of the mode selection.

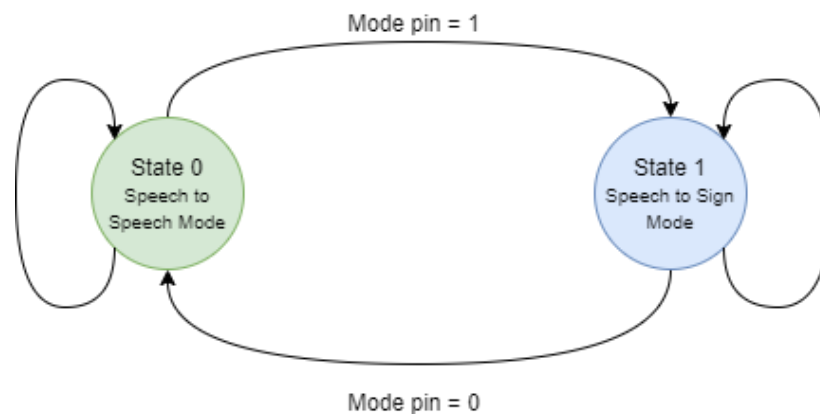


Figure 40. State Machine for Mode Selection.

5.3.1.2 Voice Recording Module

This module of the board will be implemented using a state machine that has 4 states. The state machine will stay idle in state 0 while the recording pin (pin d0) on the RF receiver is low. When the record button on the RF remote is pressed, the recording pin will go high and the state machine will move to state 1, the start state. This will allow initialization of any necessary processes to start, such as communication with the COM unit and setting addresses to the block RAM FIFO. Figure 41 shows the state machine for the voice recording module.

After one clock cycle, the state machine will move to state 2 and start recording, in which the state machine will remain until the record key on the remote is pressed again, setting this recording pin to low. Once this recording pin is low, the state machine will move to state 3, the stop state, which will tell other processes that the machine has stopped transmitting audio signals. After 1 clock cycle the state machine will move back to state 0.

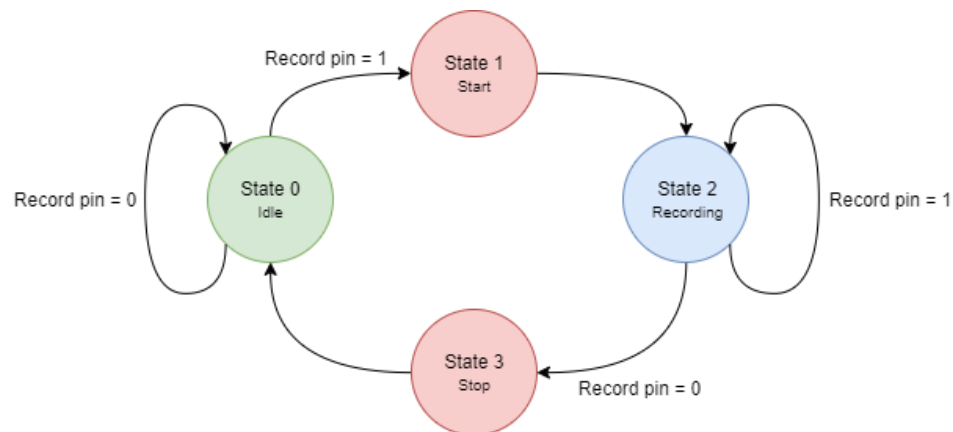


Figure 41. State Machine for Voice Recording.

5.3.1.3 Status Module

This module will not contain any state machines and will be just check for certain conditions to be met on the board. If the board is turned on, then the power LED will be on. Furthermore, if the device is currently recording, the recording LED will be on and will turn off when the recording is finished. The board will also contain a reset and power button that will send signals to the Jetson Nano when pressed to reset the device or to power off, respectively.

5.3.2 General Layout of the I/O Board

Figure 42 shows the general board layout of the PCB we intend to create. This figure is a simplified overview of the major components of the I/O PCB of the initial project design.

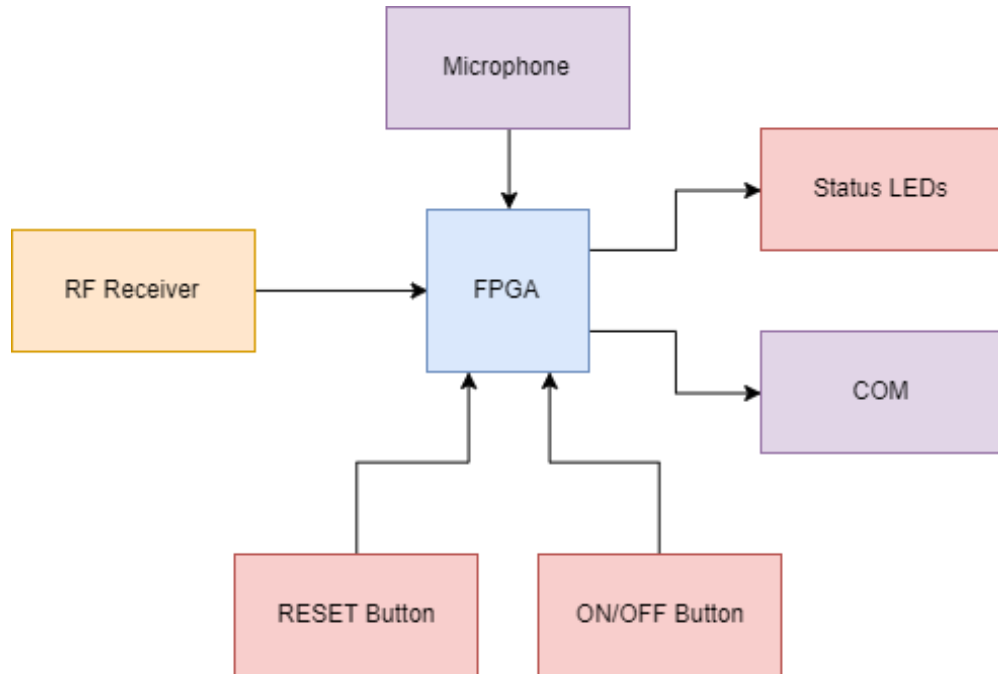


Figure 42. General Board Layout.

5.3.3 Communication for Hybrid System

The connection between the FPGA I/O board and the Jetson Nano was a 6-pin connection. 3 pins will be used for I2S audio transmission and the last 3 pins will be used to connect the recording status, power on button, and mode pins. We chose to connect the board in this fashion to avoid using more complicated and finicky interfaces over PCI-e and USB.

5.3.3.1 Audio Transmission

The microphone we have selected requires no configuration or drivers and therefore can be directly connected to the board. It will be connected to the PCB board with 5 pins; Voltage, Ground, LRCLK, BCLK, and DATA. The board will be running an I2S controller IP block that will handle the audio with two 64 x 32-bit FIFOs and is capable for both transmitting and receiving audio. Figure 43 shows the block diagram of the I2S controller IP block from Intel.

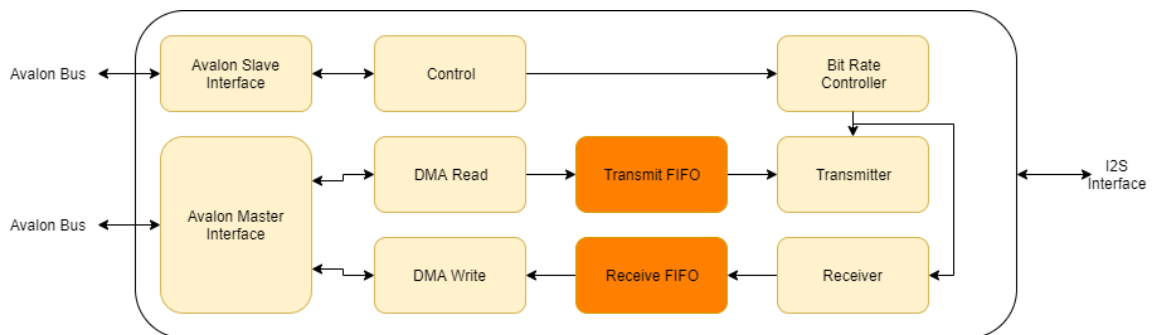


Figure 43. Block Diagram of I2S Controller IP.

It will then send this audio to the Jetson Nano through the 3 pins connecting the boards; LRCLK, BCLK, and DATA. On the Jetson Nano, we will be using the ALSA command line interface to mix and record the audio being received from the I2S input into an audio file that we can send off for transcription.

5.3.3.2 Status Transmission

The other two pins that will connect the FPGA PCB board to the Jetson Nano will be to transmit the status of translator to the Jetson. When the user presses the record button, the respective pin from the RF receiver will go high and this will be transmitted to the Jetson through the recording status wire. This will tell the Jetson to start the process of mixing and recording audio through the command line interface.

When the user presses the mode button, the respective pin coming from the RF receiver will go high and this will be reflected on the wire leading to the Jetson. When the wire is low the system will be in mode 0 and when it is high the system will be in mode 1. Furthermore, we will have a power on button attached to the I/O board that will turn on and off the Jetson Nano. This is done by shorting pins 7 and 8 with a jumper and then momentarily shorting pins 1 and 2. Shorting pins 7 and 8 will disable the auto power on sequence that is on by default, while shorting pins 1 and 2 will restore power to the system. We will use the FPGA chip to create a reset button style signal from a button press.

5.4 LCD Interface

For our project, we are intending to use a Spectre 20" monitor. This monitor utilizes the HDMI protocol, which allows us to connect the Jetson Nano to this monitor directly using a male to male HDMI cable. This monitor runs at 75 Hz and at a 1600x900 resolution which is easily handled by HDMI protocol. The Jetson Nano natively interacts with an HDMI output; therefore, we do not have to implement any drivers or code modules to obtain an image output. Furthermore, because the monitor comes installed with speakers, we will be taking advantage of how the HDMI protocol natively carries audio over the same cable as video. This will allow us to transmit both audio and video with one cable natively without the need of additional hardware.

5.5 Initial Hardware Design

This section shows the diagrams and schematics for the initial hardware design of the device. Although many of these sections have either been modified or removed from the final design, these diagrams were kept in this report to show the original thought process. Given more time and more resources, however, implementation of the FPGA and the Jetson Nano SBC unit could have been possibly implemented.

5.5.1 Initial Hardware Block Diagram

Figure 44 shows the initial hardware block diagram for the overall device design.

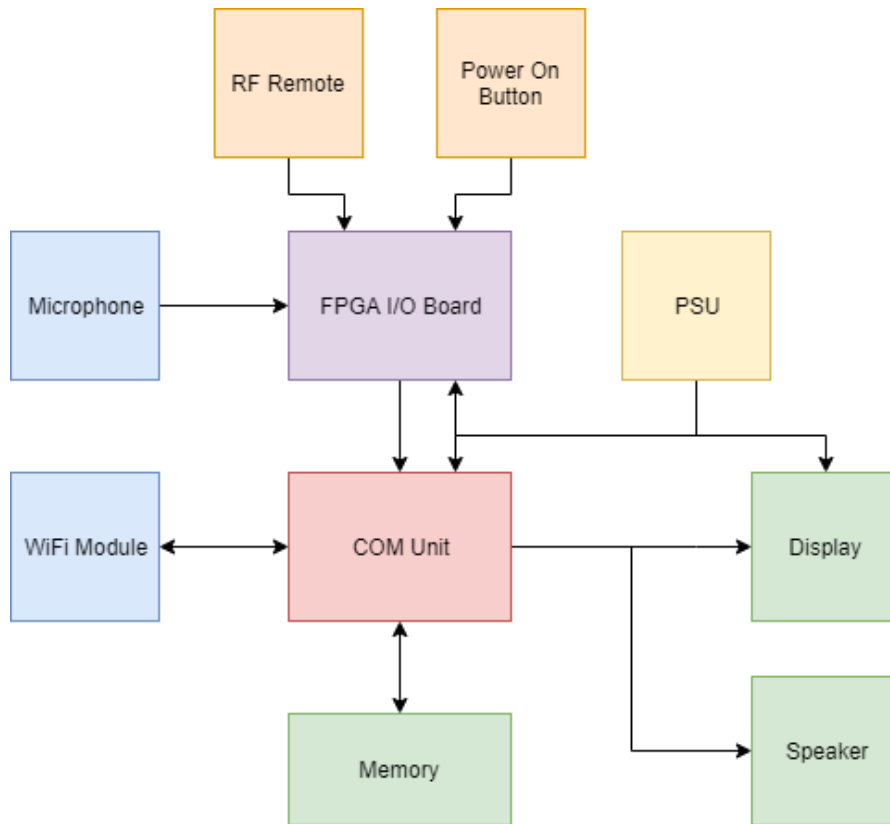


Figure 44. Initial Hardware Block Diagram.

The FPGA PCB board will be responsible for taking the inputs from our system's peripherals. It will send this received data to the Jetson Nano COM unit to be processed. The audio received from the I2S microphone will be sent via Wi-Fi to a cloud-based transcription service and then sent back to the Jetson for input into a trained English to Gloss NMT model.

We will then map this translated Gloss text to our Extended Stokoe Notation and map it to a sign language motion animation. This animation will then be displayed on the display and the translated text will be converted to speech and played through the speakers.

The RF remote will signal to the device for it to start recording and to change modes and the power on button will restore power to the system after it has been turned off. Furthermore, the PSU will provide power primarily to the COM unit and to the display. The COM unit will provide power to the I/O board through one of its USB connections. The final block diagram is shown in Figure 51.

Note that this section included components that were modified or removed completely in the final project design. This section is explained in Section 7.2.

5.5.2 Initial Hardware Schematics

This section will cover the wiring and the interfacing between the different components in our device. Each major component that contains several I/O lines will be given its own schematic. The peripherals will not be given their own schematic, but rather will be included as necessary as the input/output to the other major components. Additionally, power supply systems will be provided as schematics in this section.

5.5.2.1 FPGA Diagram

To create the I/O Board schematic, we first had to take into consideration the pin connections necessary to communicate with the Jetson Nano and the peripherals. We decided to use one of the USB ports of the Nano to communicate any non-audio data from the FPGA to the Nano and vice versa. When it came to audio-related data, we use the I2S-enabled pins on the Jetson Nano to receive such data. This audio transmission required three pins from the Jetson Nano and the Cyclone IV chip.

Additionally, the Cyclone IV used two more pins for mode selection to the Jetson Nano, these pins were record mode and translation mode. The final pin for the transmission between the Nano and the Cyclone IV was a power on button feature which turns off the system by pressing a button. Six GPIO pins were used and connected to a 6-pin header to the Jetson Nano and it can be seen in Figure 51.

For the RF receiver, we only required to use four pins which were the VDD, GND, D0 and D1. The VDD which is used to power the system was supplied power from the PSU. We only required D0 and D1 since we are using a two button RF transmitter to control when are we recording and when are we translating. The recording mode will toggle between start and stop whereas the translating mode will toggle between ASL translation and language translation. The D0 and D1 pins will connect to GPIO pins in the Cyclone IV chip so that the Cyclone IV can drive the Jetson Nano to the aforementioned modes.

The Cyclone IV chip also communicates with an external I2S module. This module provides a clean transmission of data between the selected MEMS microphone and the I/O Board. As part of the datasheet and documentation of the module, we only require three more I/O pins. Finally, the Cyclone IV chip needs to be driven with a 50MHz oscillator as required by the datasheet and documentation.

The schematic of the oscillator is included in Figure 51. The overall Hardware design will possess LEDs that display the current mode of operation, the system is in. We will use two LEDs to signify ASL and Language translation, and one LED for recording mode operation. The BOM of this initial hardware design is shown in Table 18.

The RF remote has been wired directly to IO_1 and IO_2. Although the direct wiring of the RF remote to the FPGA seems counterintuitive, this has been done to show the eventual transmission of the RF remote input. The sole reason for using only two IO pins was because the remote controller intended for the usage of the RF transmitter only possesses

two buttons. One for ASL translation and the other for normal translation. Eventually, this schematic will include the RF receiver, to which the RF transmitter will provide a wireless signal.

Table 18. I/O Board BOM.

Label	Quantity	Package	Manufacturer	Description
C1	1	0603	AVX	Multilayer Ceramic Capacitor, 0.1uF
FPGA	1	QFP 144 pin	ALTERA	FPGA, CYCLONE IV, EP4CE22E22C8N
JETSONNANO, I2SMODULE	2	1x6	SAMTEC	Board-To-Board Connector, 6 Contacts, Header, 1 Rows
RECORD, MODE0, MODE1	3	0603	ROHM	RED LED
R1, R2	2	0603	MULTICOMP PRO	Chip Resistor, Thick Film, 100k
RF TOGGLE TRANSMITTER	1	2x8	SAMTEC	Board-To-Board Connector, 8 Contacts, Header, 2 Rows
S1, S2	2	12 mm	OMRON	SWITCH 12MM x 12MM
Y1	1	3.2 x 2.5 x 1.2 mm	ABRACON	Oscillator, 50 MHz, 3.3 V

The microphone has also been wired correspondingly. The digital out (DOUT) pin is wired to IO_5. The 3V pin, although it has not been wired yet, will be wired to the VCC3P3 signal. The microphone can operate from voltage between 1.5 V and 3.6 V, so the 3.3 V power from the FPGA will be sufficient to power the microphone.

The LEDs have been wired to I/O banks 6, 8, 9, and 10. These LEDs will be used as indicators for the various statuses of the overall device. One indication will be whether power is being given to the FPGA, COM, etc. Another indication will be whether the device is currently recording audio. This indicator light will illuminate when the user presses the button on the RF remote and the device is actively listening for an audio signal.

Another indication will be whether the device is successfully connected to the Internet. Connection to the Internet is crucial for all of the cloud-based services and APIs that our device will be using, including the speech-to-text and text-to-speech services. The absence of this light being illuminated will indicate to the user that cloud-based services (and thus the overall functionality of the translator) is not available.

Although there will be many different clocks that will be operating simultaneously throughout the FPGA, only one clock has been indicated on our schematic so far, the

50MCLK. This clock is attached to CLK11. Additionally, S3 and S4 are buttons that will direct the FPGA into changing to different modes that allow certain functionalities. One of the buttons will be the ON/OFF button for the overall device whereas the other button will be a Reset button that cleans the data being processed by the FPGA. The ON/OFF button will shut down or powerup the system completely and the reset button will be used whenever the data being processed to the FPGA is erroneous in nature (early stop of audio acquisition) or if the FPGA stops processing data as intended.

5.6 Initial Hardware Design & Bill of Materials (BOM)

This section describes the overall hardware usage and total cost for our project based on the assembly and parts selected for the device based on the shown electrical schematic designs, as shown in Table 19. The Jetson Nano from NVIDIA and the Cyclone IV FPGA from Altera will both serve as our main processor units. The Jetson Nano will control anything related to the NMT model, language translations, the 3D graphics rendering, and displaying the model on the monitor screen. The Cyclone IV will control the data transmissions from the microphone using an I2S module, the RF T4 receiver, the LED power status board and the LED current operation status board.

Our remote controller will interact with the RF receiver using two functionalities. The first functionality is to switch between ASL translations and normal language translations whereas the second functionality will focus on the audio capturing by using a start and stop recording. A MEMS microphone will be used to acquire speech data and we will use an I2S module to transmit the data correctly into the Cyclone IV. Afterwards, the Cyclone IV will send the data to the Jetson Nano through USB connection.

Our PCB vendor is going to be JLCPCB due to its cheap cost and well-known reputation for delivering quality PCBs. The contents of the PCB will consist of the power supply circuitry required to power both of these processors and the I/O board, driven by an FPGA, functionality that will handle the data transmissions between the peripherals and the hybrid processor. For proper user-end functionality of our product, we included a 20" monitor, microphone, and RF receiver/transmitter set. Note that for speakers, the monitor we chose from Spectre also has built-in speakers and can receive audio input via an HDMI-HDMI cable from the Jetson Nano.

In terms of connectivity, we included an HDMI-HDMI cable to connect the Jetson Nano to the 20" display to show the graphics rendering software. For proper functionality of the Jetson Nano, a microSD card containing the operating system and room for general storage is required; for this reason, we included a SanDisk 32GB microSD card. Table 19 possesses the overall components used in the system.

Table 19. Initial BOM.

Level	Description	Group	Manufacturer	Part Number	Quantity	Unit Cost	Total Cost
1	Electronic Device	Assembly			1	-	-
2	Jetson Nano	Part	NVIDIA	102110268	1	\$95.96	\$95.96
2	Cyclone IV	Part	Altera	EP4CE22E22C8N	1	\$35.52	\$35.52
2	PCB	Part			1	\$40.10	\$40.10
2	MEMS Microphone	Part	Adafruit	SPH0645LM4H	1	\$6.95	\$6.95
2	Simple RF T4 Receiver	Part	Adafruit	1097	1	\$4.95	\$4.95
2	2-ButtonRF Remote	Part	Adafruit	1391	1	\$6.95	\$6.95
2	20" LED AV Monitor	Part	Spectre	E205W-1600	1	\$59.99	59.99
2	4' HDMI to HDMI Cable	Part	Dynex	DX-SF109	1	DONATED	--
2	Ultra 32GB microSD Card	Part	SanDisk	SDSQUAR-032G-GN6MA	1	\$7.49	
2	Voltage Regulators	Part	Texas Instrument	SDB06A, SDA08C	3	\$2.40	\$2.40
2	USB WiFi Antenna	Part	--	--	1	DONATED	
						Overall	\$255.21

6. Project Software Design

Our project has a high emphasis on software and programming. The software for our project requires six different functions in sequence that it must complete for proper translation. This section will deal primarily with the English to ASL-Gloss translation; other translation pairs, however, will be achieved through online cloud services. The six steps in the software design are as follows: speech capture, speech-to-text translation, neural machine translation of English text to ASL-Gloss, mapping the ASL-Gloss to ELS notation, and rendering ASL graphics and displaying them on the final display. The overall software flow chart is summarized in Figure 48.

6.1 Initial Software Functionality

Our software will have two different modes that users can choose between: speech-to-speech translation mode or speech-to-ASL mode. These modes will perform essentially the same steps with the speech-to-ASL mode requiring an alternate process to the speech-to-speech translation mode as the speech-to-ASL mode has a greater emphasis on our project. At the start of the program, the user will be prompted to choose between the two available modes using an IR remote. Once the user has chosen a mode, the main loop of that chosen mode will begin.

6.1.1 User Options

Originally, the user was to be given two options. The first option was the speech-to-speech translation mode, which allowed the user to speak in one language, and the device would output speech in a specified language. The second option was the speech-to-ASL translation mode, which allowed the user to speak in English, and the device would show a display of the corresponding ASL gestures. Although the first option has since been deprecated in the final design, the second option has been modified.

6.1.1.1 Option A: Speech-to-Speech Translation Mode

When this loop begins, the user will be prompted to input the source and target languages for their translations. Once the users have chosen the source and target languages the system will wait until a recording button is pressed. Once the button has been pressed, the software will take in the input soundwaves of speech from the microphone and pass these snippets to an online speech-to-text service. Once this typographic text has been received from the online speech recognition service, the program will pass this through another online service to translate it from the source language to the destination language. Once the system has received the translated text the program will output this text onto the screen.

Additionally, the program will also send the translated text to another cloud service that will perform text-to-speech of the translated text. This sound file will be passed to the speaker so that users can both read and hear the translated communication.

6.1.1.2 Option B: Speech-to-Sign Mode

When the loop begins, the program will wait for the recording button to be pressed. Once the button has been pressed the software will begin taking in the input sound waves of speech from the microphone, pass them through an online speech-to-text service, and then take the received English text and translate it into English Gloss. Once the text is translated to Gloss, the text will be converted into our proposed Extended Linear Stokoe (ELS) notation that will tell our graphic rendering software how to perform the specific American Sign Language gesture. These motions will then be sent onto the terminal monitor along with a log of the text received from the speech-to-text conversion.

6.1.2 Software Procedure

The initial software design procedure consists of several steps which are necessary for conversion of spoken language to a text file containing a different language. This procedure has been sustained for the most part, with the final design specifying which APIs are used and how the software is realized; however, the final step of conversion via ELS has been removed since the ELS notation itself has been deprecated in the final design.

6.1.2.1 Step 1: Speech Capture

This first step in the software process will capture speech from the person speaking into the microphone on the device. The microphone will only capture speech once a button has been pressed; we have not yet decided whether this button will be available on the device itself or if the button will be available on a remote and transmitted via IR.

Once the signal from the button has been received, the microphone will capture the sound waves of the person speaking and convert this signal into a corresponding digital signal. The microphone we have chosen for the project will output a digital signal instead of analog to mitigate the need for an analog-to-digital converter (ADC) system.

6.1.2.2 Step 2: Speech-To-Text Conversion

The function of this second step is to send the digital voice signals to the cloud-based transcription service and to receive the transcribed text. Once the button has been released and we have a digital signal for the input, we can send this signal from the board to the cloud-based transcription service. This will handle the conversion from speech to text and we will receive a stream of text back from service.

Once we have received this stream of text, we will parse through the text and replace any of the pronouns that are present in the sentences with our filler pronoun word. The NMT model does not understand the distinction between pronouns and will weight every unknown pronoun differently. We would have to train the NMT model with every proper noun for the field this product was being deployed in for it accurately be able to pass proper nouns through the model.

We can avoid having weights on the proper nouns by keeping track of all the pronouns in the original sentences and simply replacing them once they have been passed through the NMT model. We can make this substitution since in ASL, most proper nouns are spelled out with individual letters so passing it through the model would yield the same word.

6.1.2.3 Step 3: Neural Machine Text-to-Text Translation

The function of this third step is to translate the text received from the cloud-based transcription service to ASL-Gloss. This will be accomplished by feeding sentences of the transcribed text into a pretrained OpenNMT model that will make predictions on the best match for the English text in ASL-Gloss, along with removing words that are not necessarily used in ASL (such as auxiliary verbs, etc.).

6.1.2.4 Step 4: Mapping ASL-Gloss to Custom ELS Notation

The function of this task is to map the ASL-Gloss text received from the model translation verbatim to our ELS notation. We will parse through the sentences of Gloss, removing any of the empty words along the way. Furthermore, as we parse the sentences, any pronoun fillers found will be replaced with its respective pronoun from the original sentence. We will have a database of all the words that we have trained in the NMT model that have been pre-translated into our ELS notation, from which we can translate the ASL-Gloss word/phrase into the ELS equivalent. The use of the ELS notation is to allow a structured form of rendering the graphics in the next step. We will also translate the proper nouns into the ELS Notation by changing it to individual letters or its gesture equivalent in ASL.

6.1.2.5 Step 5: Rendering ASL Gestures in 3D Graphics Platform

The function of this task is to create ASL graphics for the translated text that can be displayed on the screen for the deaf user to understand what the original English speaker had said. We will be creating models using a 3D rendering software such as Unity and using the ELS notation that we have created as a set of instructions on how to animate the virtual avatar. Figure 45 shows the breakdown of how our rendering software will interpret our ELS.

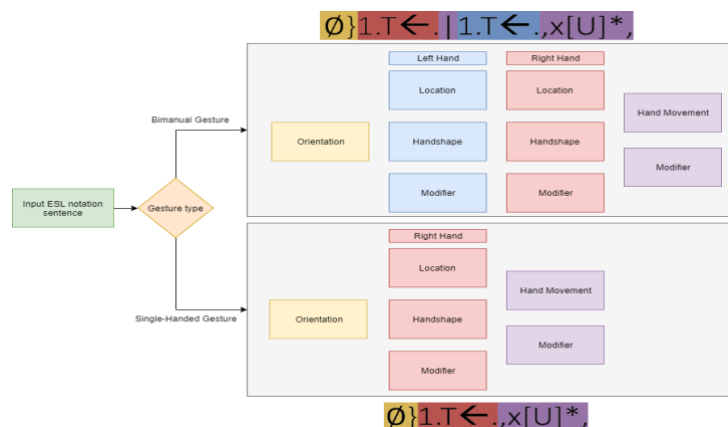


Figure 45. Breakdown of Conversion of ELS Notation into 3D Graphics Rendering.

Each character in the ELS notation will tell the program what the location of the hands will be, the location and handshape of the hands, hand movements, and other elements that are essential in the delivery of a sentence in ASL. This will allow for natural-looking ASL motions that are created in real-time.

After the sequence of gestures have been produced, the final step is to output the ASL graphics created by the software onto the terminal display. These motions will play in the sequence determined by the ELS input so that the sentence can be created by stringing these motion bits together. Figure 46 shows the overall software flow of our device.

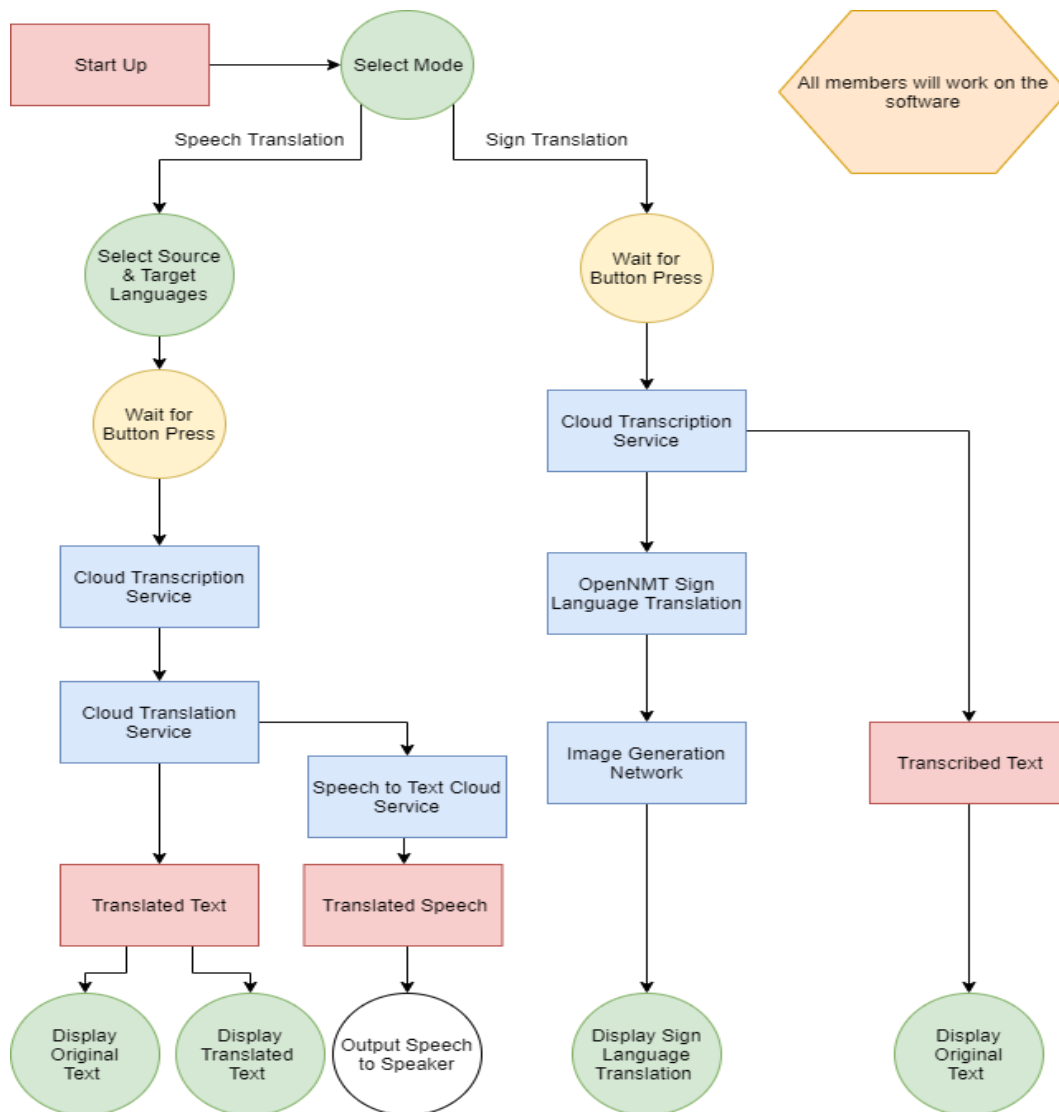


Figure 46. Initial Software Flow Chart.

Once they have finished playing, the users will have the option to restart the animation and replay the translation, start recording another sentence to be translated, or exit the current mode and switch to the other for speech-to-speech translation.

The engine will use a preset of already defined “frame layers”. These layers will possess a unique characteristic based on the symbols provided which will be divided into space, left hand, right hand. Within each major layer, there will be sub layers that will have unique characteristics such as location, handshape and modifier. After all the layers have been selected based on the provided symbol, we will extract the unique characteristics of each sub layer and add them to the pertaining major layers.

Moreover, the major layers, once done, will mesh into the orientation layer. Finally, the layers pertaining to the movements of the hand or any other additional layer will be provided. The extractions explained here can be already done on most game engines. The reason is because modern games such as Telltale Games make non-playable characters or surroundings to change depending on the user’s decisions. We can use this feature and translate it into different “movements” based on the input symbols that we receive.

6.2 Changes to Software Functionality in Final Design

Between the original design and this finalized design of the software aspect of this project, there have been several changes to accommodate the requirement specifications but also address realizability issues along the implementation phase of designing the final product. A summary of these changes is addressed in this section.

6.2.1 Obsolescence of ELS Notation

Originally, the proposed solution for parsing sign language gestures was through our Extended Linear Stokoe (ELS) notation, first described in Section 3.2.1: American Sign Language Typography. This notation was intended to have been used to create a universal and general approach from which any sign language gesture could be created. The purpose of ELS in the scope of this project was to create layered animations such that an animated gesture would be broken down into several categories, including initial/final hand positions and hand movements, for example.

This project contained 100 different animations, of which 80 words mapped to a specific sign language gesture; 26 animations were mapped for alphabetic fingerspelling, and a few animations were reserved for ancillary gestures such as the idle and thinking animations. Given the limited vocabulary of our OpenNMT language corpus, using a generalized approach for an animation was considered too time-consuming for the results produced. Instead, animations were created individually without concern for similarities between animations in the final Unity game design.

If this project were to be scaled up, approaching animations using this ELS notation would be optimal. The entire ASL “lexicon” per se contains thousands of gestures, from which

gestures can be combined and modified as needed to create context and emphasis from a given root gesture. Section 3 of this paper compared the ASL typographic system to pictographic languages such as Chinese. The reason the ASL typographic system could be comparable to a pictographic language is because ASL itself is inherently pictographic; that is, a gesture in ASL can be considered as a composition of several smaller sub-gestures (a handshape, a hand movement, a facial expression, and so on). In fact, ASL can be considered a three-dimensional version of two-dimensional pictographic writing systems such as Chinese. Due to this complexity, simply mapping individual gestures would not be feasible when attempting to scale up the animation map of this project to thousands of gestures.

6.2.2 Revision of User Options

Our final approach of the user interface has remained the same from the initial design first described in Section 6.1.1: User Options, but a few major changes were enacted during the design phase. Originally, the user would have had the option to select between two options upon starting the game. The first option was to enter traditional language translation mode, where the user would speak in one language, and the device would output text-to-speech in a target language of the sentence that the user had spoken into the device. The second option was to enter English-to-ASL mode, where the user would speak a sentence in English and the device would show on the screen the corresponding sequence of gestures in sign language. Due to the time constraints and complexity of the English-to-ASL mode alone, the first option of traditional language translation was not implemented in the final product.

6.2.3 Revision of Software Procedure

The approach for the user interface has remained the same as the original design mentioned in Section 6.1.2: Software Procedure, with the major differences being the removal of the option for translation between two spoken languages and the obsolescence of the ELS notation. The user interface created for this project was created such that the user is given a few simple options but can interface with the device completely. Upon starting the game, the user is shown the avatar in the foreground and a text-displaying canvas in the background. The avatar maintains an idle animation state until the user decides to interact with the device. Once the user presses the “Start” button on the PCB, the canvas will display “Recording...” which indicates the user to start speaking into the microphone. Once the user is finished recording, the user presses the “Stop” button on the PCB, at which point the avatar transitions to a thinking animation state. This thinking animation state is used as a facade while the game processes the audio file to create a string array containing the ASL-equivalent sentence. Once this sentence has been created, two lines are displayed on the background canvas. The first sentence on top is the transcription that was generated from the Watson Speech-to-Text API service. The second sentence on the bottom is the translation that was generated from the OpenNMT translation software. While these sentences are being displayed, the avatar begins making the gestures in the same order as the translated sentence. Once the gestures have been completed, the canvas is refreshed to remove the sentences, and the avatar calmly returns to the idle state. At this point, the user

has the option to pass another spoken sentence into the device or end the game. If the user presses the Escape key on a keyboard (which was intended to be mapped to the “Reset” button on the PCB), the game begins a shutdown sequence. This sequence begins with displaying “Goodbye!” on the canvas and the avatar gesturing a “Hello” animation. (The “Hello” animation was recycled as a substitute to the “Goodbye” animation due to their similarity in ASL.) Once both of these events have finished, the game then completely shuts down.

While the user is interacting with the program, there are two warning messages that appear if the user attempts to interact with the program that could potentially break the game. The first warning message appears if the user presses the “Stop” button if the Start button had not been pressed beforehand. Proper functionality of the game requires the user to press the “Start” button and begin recording on the microphone before pressing the stop button. If this out-of-order button pressing is attempted, a pop-up message in the top-left corner appears stating “Microphone is not recording...”, and the program will not attempt to stop recording from the microphone. The other warning message appears if the user tries to quit the game while animations are currently playing. This message was implemented since there is no GUI for menu options to quit the game. If the player were able to quit the game during animations, the game would terminate without displaying the shutdown sequence.

Originally proposed was a method for replacing pronouns and proper nouns that were given in the sentence to provide a more generalized translation. This approach was considered due to the limitations on translating proper nouns and pronouns in the OpenNMT training algorithm. However, due to modifications to our approach in training the OpenNMT model, this process of pronoun/proper noun replacement has been removed from the final product.

Also, due to the obsolescence of the ELS notation, Section 6.1.2.5 regarding rendering ASL gestures based on this ELS notation has subsequently been considered obsolete in the final design. This process of rendering has been replaced with direct animation mapping.

The process of fingerspelling, however, was kept in the final product. This was mainly done due to the presence of a few words that required fingerspelling, such as “campus” and “ASL”, which are finger-spelled by default.

6.2.4 Unity Engine Script Functionality

The Unity Engine game consists of three main aspects. The first aspect are the game objects whose purposes are for showing the resulting graphics renderings. These game objects possess their actual rendered graphics in the game environment; for example, the avatar game object contains meshes and textures necessary for viewing the avatar in the game. The objects also have child objects that are attached to the main object; these objects serve auxiliary purposes but do not necessarily have a corresponding texture/mesh that is visible in the game environment. An example of child objects is the animator controller which is attached to the avatar object, and the bone/joint map which are attached to different body parts of the avatar object. The animator controller object controls the bone/joint map to

articulate specific joints in a particular way under certain conditions that are set in the game environment.

The second main aspect of the game is the animator controller which was implemented to handle all the animation clips and transitions between such clips. Contained within the animator controller are a set of animations that had been prepared individually. There are default states within this controller which are either references to external events or meta references. One of these states is the “Entry” state, which is triggered when the game starts up; after being triggered, the controller immediately transitions to another animation to which the “Entry” state is connected in the animation map. Another state is the “Any State” state, which allows for dynamicity of the animation map; if any transition from this state is made available, then the controller will transition to that state. The transitions from “Any State” are all blocked using a trigger variable. A trigger variable is a variable unique to the Unity Engine: it is similar to a Boolean variable in that a trigger can only be in two states, “on” and “off”; however, it differs from a Boolean in that once a trigger is set to “on” in the animation map, it is referenced then reset to “off”. If multiple triggers are set at once, the animator controller references each trigger in a FIFO stack process. A trigger variable is attached to each transition from “Any State”, with the variable name matching the name of the corresponding animation clip name to which transition this trigger is attached; the purpose of this is due to the trigger setting method implemented in the C# script, explained later.

The third aspect are the C# script files which handle timing of all events and routines. Each C# script is attached to a game object depending on the purpose that script serves. There are three main scripts and one auxiliary script that are used for the Unity program. The main scripts were made to inherit the *MonoBehavior* interface, from which a *Start()*, *Update()*, and *OnGUI()* method can be inherited. The *Start()* method is called only once before the game environment is loaded. The *Update()* method is called once every frame once the game environment has been loaded. The *OnGUI()* method is called once every frame but only handles GUI events.

The first script, named *update_anim.cs*, is attached to the avatar object and is the de facto main script for this program. In its *Start()* method, a hash table is created which maps specific strings to a corresponding animation clip in the animator controller. Also, in this method is an object reference to the animator controller object which is a child of the avatar object; this is necessary for the script to understand which animator controller is being referenced. This method also includes an initialization of a *Microphone* object, which is a built-in Unity class for handling in-game audio clips.

In the *Update()* method of the *update_anim.cs* script, there are three main conditions, and each condition is mapped to a specific key press. Each button on the PCB has been mapped to be registered as keyboard presses in the Unity game. Pressing the “Start” button on the PCB is used as a conditional to start recording. Pressing the “Stop” button on the PCB is used as a conditional to stop recording, then start a method named *Translate()*. Although not having been implemented in the final design, there is a conditional to stop the game if the Escape key was pressed; whether this button were to be mapped to the “Reset” button

on the PCB button or not was debated and left unresolved by the implementation of the final design.

The *Translate()* method in the *update_anim.cs* script contains all of the data processing required to modify an audio file containing English speech into a text file containing the ASL-Gloss sentence necessary to display the corresponding animations. After the user presses the “Stop” button, the Unity program saves an AudioClip object which contains the speech audio clip. Conversion of this AudioClip object to a usable WAV file was through the auxiliary script named *SavWav.cs* written by GitHub user *darktable* [139] which converts AudioClip objects into WAV files. Once this audio file was made available, it was used as an input parameter to the Watson Speech-to-Text (STT) API call using the *curl* command in a command prompt. Communication with a command prompt local to the Unity program was made possible using the Process class. The output of calling the Watson STT API was a JSON file containing various information parameters, including transcription confidence and the transcribed sentence itself. The JSON file was parsed to extract only the transcription, and this transcription string was written to a text file named *translate.txt*. This text file along with the OpenNMT model produced beforehand was used as the input parameter to the OpenNMT translate call using the *onmt_translate* command in the command prompt. The output of this command resulted in another text file named *pred.txt* which contained the ASL-Gloss equivalent of the transcribed English text. This text file was then parsed into a string array which was used as the input parameter for the next method, *AnimHandler()*.

The purpose of the *AnimHandler()* method in the *update_anim.cs* script is to process the string array produced in the *Translate()* method to display the proper animations available in the animator controller. Each string in the array is potentially matched to the entries in the hash table. If the hash table returns a match, then the corresponding trigger for that animation is set, and the avatar will begin making the specified gesture; however, if there is no match in the hash table, the string is further split in to a character array, and each letter is passed to the animator controller consecutively but separately such that the avatar begin gesturing the corresponding fingerspelling animations. Once every word has been addressed in the string array, then the method is finished, and the game returns to the initial idle state to await the user’s next input.

The other main C# scripts were named *update_canvas.cs* and *update_camera.cs*. These scripts are considered secondary scripts to the *update_anim.cs* script and serve as object reference scripts. The *update_canvas.cs* script is attached to the background canvas object, and the purpose of this script is to reference the TextMeshPro object such that manipulation of the canvas text is possible. The *update_camera.cs* script is attached to the main camera object (the lens through which the user views the game environment), and the purpose of this script is to hold an AudioSource child object. AudioSource objects are used to hold audio clips within the game, and this specific object was used to hold the produced audio clip from the Microphone class until the audio clip can be converted to a WAV file. This AudioSource object was attached to the main camera object such that the user need not speak loudly into the microphone for the game to register voice.

The final built Unity game consists of the default Unity files which include the Unity executable file and the ancillary files that are compiled alongside the Unity game to help with runtime processes. Included with these Unity files is a file folder named “ASL” which consists of two files which are crucial for proper functionality of the game and are included externally from the built game. The first file is the OpenNMT model, which is required to translate the English transcript text file *translate.txt* into the ASL-Gloss translation text file *pred.txt*. The second file is the dictionary text file, *vocab.txt*, which contains a list of words that are also available in the OpenNMT model dictionary and require a unique animation to be played. This file is necessary for the game to initialize the hash table while the game is starting up. The final software diagram is shown in figure 47A. An insight on the animation parsing specifically is shown in figure 47B.

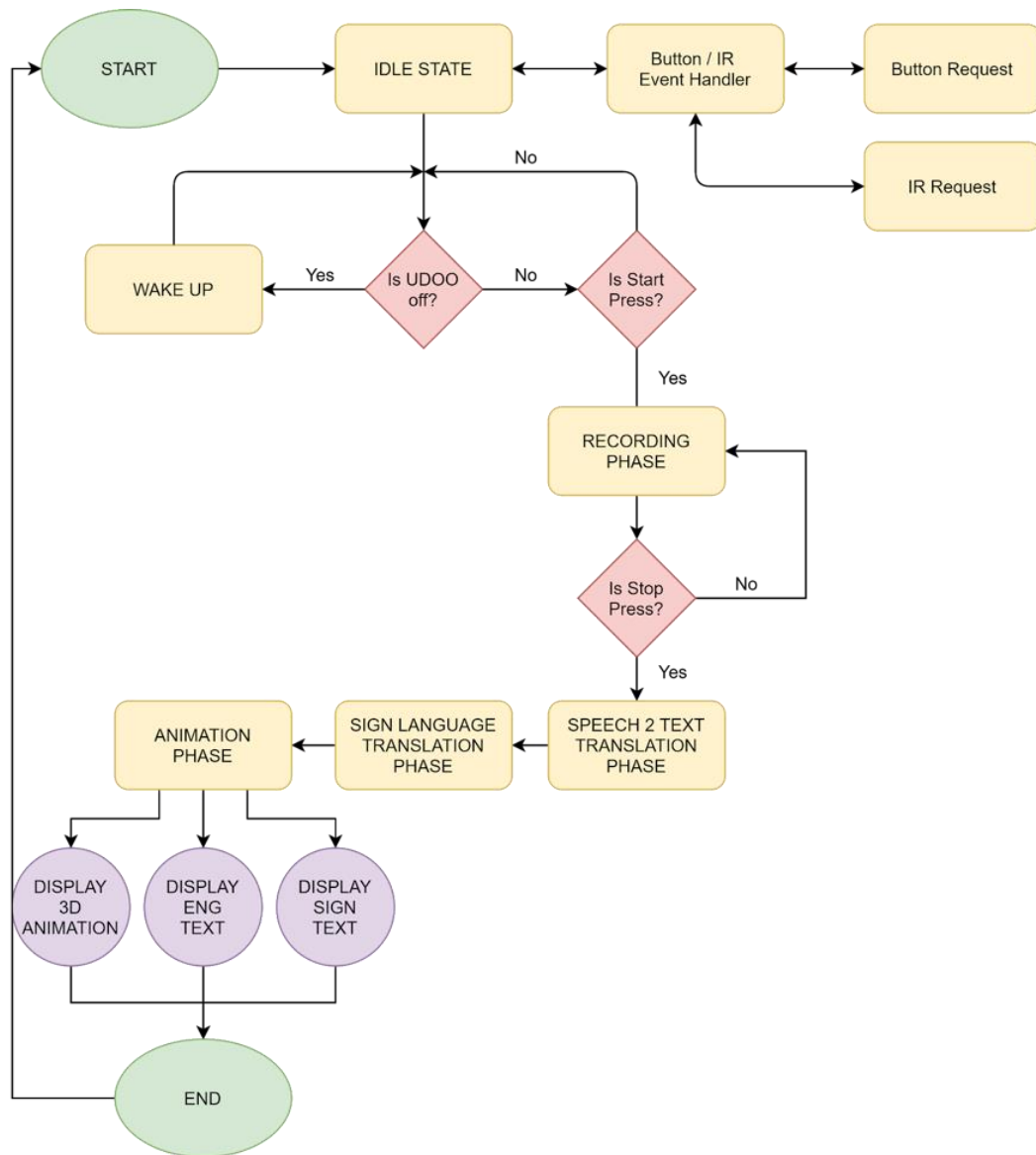


Figure 47A. Final Software Diagram.

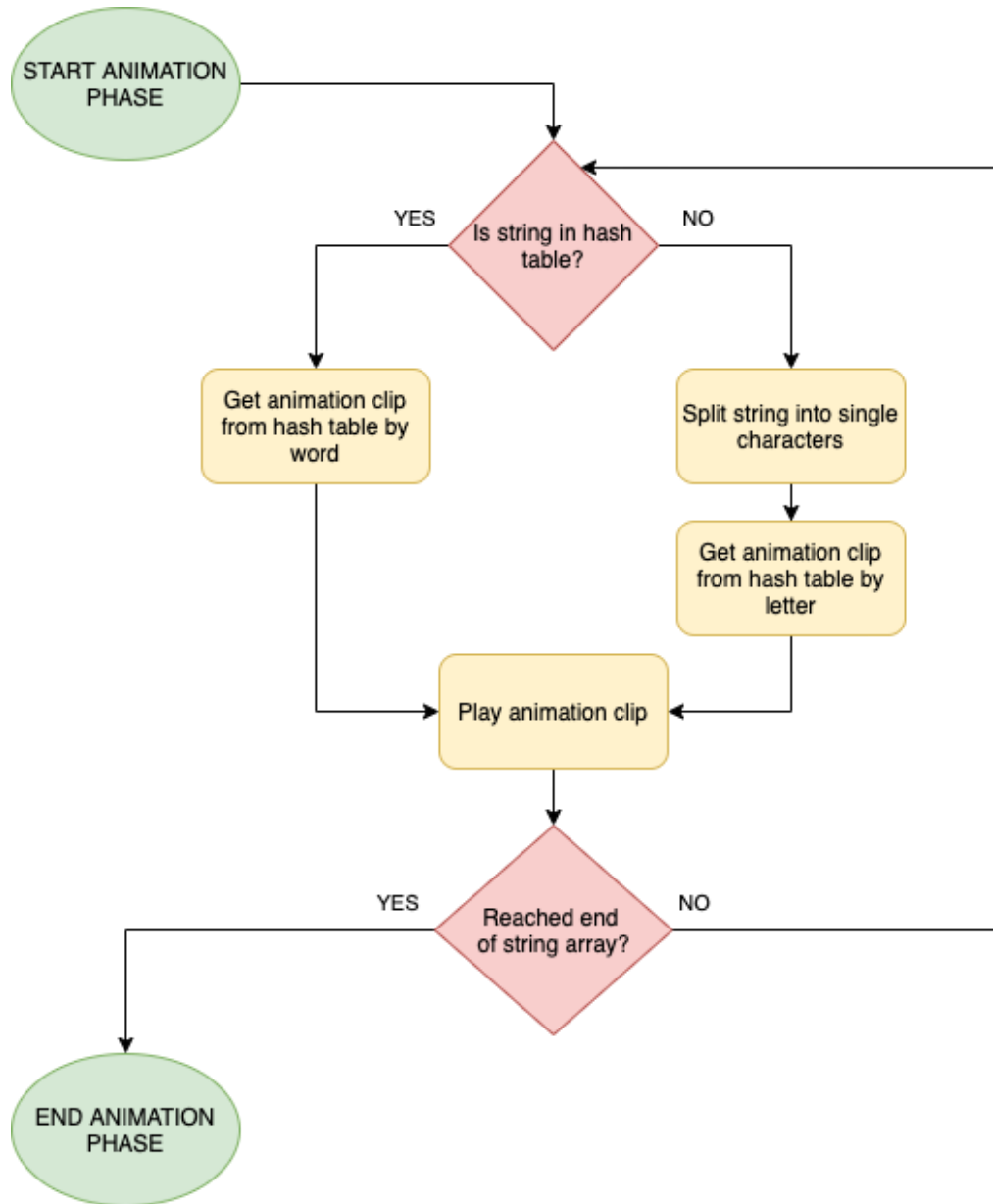


Figure 47B. Final Animation Flow Diagram.

7. Project Testing and Prototyping

This section addresses the aspects of the PCB design procedure as well as the neural machine translation training done through the OpenNMT platform.

7.1 Initial OpenNMT Model Testing

This section is a collection of the various attempts in modifying the training and validation sets used in creating a translation model using OpenNMT. Our first step is the creation of the pair of training and validation files. One file is typed in English while the other is typed in ASL-Gloss. The contents of these files will provide the basis for the creation of our NMT model. After running the steps of preprocessing and training, we made another test file containing sample sentences.

The output of these sentences is shown through the Command Prompt window (or the Terminal window on macOS). Included with the predicted sentence are various statistics represented by their PRED score. (The PRED score can be converted to the approximated percent accuracy by raising 10 to this number.)

Note that each training set consists of four different files: a training source corpus, a training target corpus, a validation source corpus, and a validation target corpus. In each training run, the OpenNMT algorithm pairs each line between the two training corpora and uses the validation corpora to confirm that its training model was successful. For each attempt, we included examples of the contents of each corpus, but the full list for each corpus for each attempt can be found in Appendix C.

Attempt 1 — October 18, 2019

We began training our model for the English-to-ASL Gloss neural machine translation by creating a data set of two text files from which we could benchmark the effectiveness of the NMT model. In each of the text files, we chose a handful of common medical checkup terms and phrases. Included in this dataset are introductory phrases such as “Hello”, “How are you?”, “How old are you?”

In OpenNMT, the translator is trained by having a source file for the start language and a source file for the destination language that must match in length of phrases for proper translation. The entire test and validation corpora are shown in Figures 48 and 49, respectively.

Hello How are you ? Do you need help ? Are you hurt ? Do you feel cold ? How do you feel ? I am fine How old are you ? I do not understand I don't understand I'm sorry I am sorry What is your name ?	HELLO HOW YOU ? YOU NEED HELP ? YOU HURT ? YOU FEEL COLD ? HOW YOU FEEL ? ME FINE HOW OLD YOU ? ME NOT UNDERSTAND ME NOT UNDERSTAND ME SORRY ME SORRY YOU NAME WHAT ?
English Training Corpus	Gloss Training Corpus

Figure 48. Test English and Gloss Corpora used in Attempt 1.

Furthermore, the training algorithm needs to have a validation file for languages used in the translation, which are English and Gloss in our case. These validation files use the same words that are in the original training corpora but contain different combinations of the words that are not found in the original training corpora but the NMT should be able to translate effectively. This process of cross-validation ensures that the weights of the words and sentences will be adjusted properly in the NMT and will translate accurately when someone inputs a sentence that is not verbatim found in either the training or the validation corpora.

Hello John Hello Charles How are you John ? How are you Charles ? What happened John ? What happened Charles ? I am not fine I am not sorry Do you have a dog ? What time is it John ? What time is it Charles ? Feel cold ?	HELLO JOHN HELLO CHARLES HOW YOU JOHN ? HOW YOU CHARLES ? WHAT HAPPEN JOHN ? WHAT HAPPEN CHARLES ? ME NOT FINE ME NOT SORRY YOU DOG HAVE ? TIME WHAT JOHN ? TIME WHAT CHARLES ? YOU FEEL COLD ?
English Validation Corpus	Gloss Validation Corpus

Figure 49. Validation English and Gloss Corpora for Attempt 1.

After training with this data set, we input some test sentences through the NMT using the model created. The results were not as expected and did not produce accurate predictions, even when tested with sentences that were located both in the training and validation files. A few examples of the output test results are shown in Figure 50.

```
SENT 1: ['Hello', 'how', 'are', 'you'] SENT 1: ['Hello', 'how', 'are', 'you']
PRED 1: TIME WHAT ? PRED 1: TIME WHAT ?
PRED SCORE: -0.0873 PRED SCORE: -0.0873
PRED AVG SCORE: -0.0291, PRED PPL: 1. PRED AVG SCORE: -0.0291, PRED PPL: 1.

(base) C:\Users\Jared\Desktop\OpenNMT (base) C:\Users\Jared\Desktop\OpenNMT
[2019-10-25 13:20:59,989 INFO] Transl [2019-10-25 13:20:59,989 INFO] Transl

SENT 1: ['Dog', 'is', 'sorry'] SENT 1: ['Dog', 'is', 'sorry']
PRED 1: ME SORRY PRED 1: ME SORRY
PRED SCORE: -0.0399 PRED SCORE: -0.0399
PRED AVG SCORE: -0.0200, PRED PPL: 1. PRED AVG SCORE: -0.0200, PRED PPL: 1.

(base) C:\Users\Jared\Desktop\OpenNMT (base) C:\Users\Jared\Desktop\OpenNMT
[2019-10-25 13:22:09,480 INFO] Transl [2019-10-25 13:22:09,480 INFO] Transl

SENT 1: ['feel', 'cold'] SENT 1: ['feel', 'cold']
PRED 1: HELLO PRED 1: HELLO
PRED SCORE: -0.0132 PRED SCORE: -0.0132
PRED AVG SCORE: -0.0132, PRED PPL: 1. PRED AVG SCORE: -0.0132, PRED PPL: 1.

(base) C:\Users\Jared\Desktop\OpenNMT (base) C:\Users\Jared\Desktop\OpenNMT
[2019-10-25 13:22:37,894 INFO] Transl [2019-10-25 13:22:37,894 INFO] Transl

SENT 1: ['feel', 'cold', '?'] SENT 1: ['feel', 'cold', '?']
PRED 1: HELLO PRED 1: HELLO
PRED SCORE: -0.1469 PRED SCORE: -0.1469
PRED AVG SCORE: -0.1469, PRED PPL: 1. PRED AVG SCORE: -0.1469, PRED PPL: 1.

(base) C:\Users\Jared\Desktop\OpenNMT (base) C:\Users\Jared\Desktop\OpenNMT
[2019-10-25 13:22:53,855 INFO] Transl [2019-10-25 13:22:53,855 INFO] Transl

SENT 1: ['what', 'time'] SENT 1: ['what', 'time']
PRED 1: HELLO PRED 1: HELLO
PRED SCORE: -0.0564 PRED SCORE: -0.0564
PRED AVG SCORE: -0.0564, PRED PPL: 1. PRED AVG SCORE: -0.0564, PRED PPL: 1.

SENT 1: ['feel', 'cold']
PRED 1: HELLO
PRED SCORE: -0.0132

BEST HYP:
[-0.0132] ['HELLO']
[-5.4834] ['MY', 'FEEL', 'JOHN']
[-7.2005] ['MY', 'FEEL', '?']
[-8.1771] ['MY', 'NAME', 'JOHN']
[-8.3735] ['HOW']
PRED AVG SCORE: -0.0132, PRED PPL: 1.0133
```

Figure 50. Example Results of Attempt 1 Post-Training.

Note that despite the inaccurate sentences produced, the level of confidence in this model is quite high, as exemplified in the top-left box which shows a PRED score of -0.0873, which corresponds to an 81.79% accuracy.

Attempt 2 — October 25, 2019

We experimented further by making several changes to the training files. We made both the training and verification files the same on both sides so that we could see the effect of this. Furthermore, because of how Gloss is structured to represent sign language, its sentence structure is Object Verb Subject (OVS). On the other hand, the sentence structure in English is Subject Verb Object (SVO). We were concerned that this was affecting the training outcomes, so we restructured the Gloss files to follow SVO.

At this point we were more worried about the accuracy of the translation rather than the structure of the translation, so this is something we could sacrifice for the time being. Furthermore, if time had permitted, we could create an algorithm that would convert sentences from SVO to OVS. These training and validation corpora can be seen in Figures 51 and 52, respectively.

Finally, we changed the training files so that all the sentences on both sides of the translation were of the same word length. We have some concerns that the algorithm is attempting to do one-to-one mapping on the sentences in the corpus and so when they are of different length, it might have been leading to the mistranslations. Therefore, we made them the same length throughout the files.

Because Gloss removed many unneeded or unrepresented words out of sentences being translated to Gloss from English, we implemented an “empty word” filler which is simply the underscore character ‘_’. Our reasoning for this approach is that these unnecessary words will be mapped to the empty word, so when we perform our Gloss to ELS transcription after the translation, we can simply ignore these empty words.

Hello	HELLO
How are you ?	HOW _ YOU ?
Do you need help ?	_ YOU NEED HELP ?
Are you hurt ?	_ YOU HURT ?
Do you feel cold ?	_ YOU FEEL COLD ?
How do you feel ?	HOW _ YOU FEEL ?
I am fine	ME _ FINE
How old are you ?	HOW OLD _ YOU ?
I do not understand	ME _ NOT UNDERSTAND
I don't understand	ME NOT UNDERSTAND
--	
English Training Corpus	Gloss Training Corpus

Figure 51. Test English and Gloss Corpora used in Attempt 2.

Hello How are you ? Do you need help ? Are you hurt ? Do you feel cold ? How do you feel ? I am fine How old are you ? I do not understand I don't understand ..	HELLO HOW _ YOU ? _ YOU NEED HELP ? _ YOU HURT ? _ YOU FEEL COLD ? HOW _ YOU FEEL ? ME _ FINE HOW OLD _ YOU ? ME _ NOT UNDERSTAND ME NOT UNDERSTAND
English Validation Corpus	Gloss Validation Corpus

Figure 52. Validation English and Gloss Corpora for Attempt 2.

After training with these modifications, we saw improvements to the results and were able to get much better predictions from sentences that were outside of our training set even with the small training set that we had. We were even able to correctly translate a six-word sentence, as shown in Figure 53.

```
SENT 1: ['I', 'am', 'in', 'need', 'of', 'help']  
PRED 1: ME NEED HELP  
PRED SCORE: -0.6537  
PRED AVG SCORE: -0.2179, PRED PPL: 1.2435
```

Figure 53. Example Results of Attempt 1 Post-Training, Part 1.

However, even with these improvements, we are still facing problems with the NMT model not understanding the weight of certain words or what exactly proper nouns are. An example of a mistranslated sentence that adds excessive weight to a proper noun is shown in Figure 54.

Proper nouns are used in sentences to talk about subjects. If we would like accurate predictions when using proper nouns, we would have to teach the NMT every single proper noun that we believe would be implemented with the system and have several different combinations of said pronoun inside of sentences.

```
SENT 1: ['Charles', 'is', 'in', 'need', 'of', 'help']  
PRED 1: CHARLES  
PRED SCORE: -0.0975  
PRED AVG SCORE: -0.0975, PRED PPL: 1.1024
```

Figure 54. Example Results of Attempt 1 Post-Training, Part 2.

For the scope of this project, it is not feasible to create a dataset like this, so we would have to derive a potential workaround to this problem. Our proposed solution is to filter proper nouns out and replace them with a “filler word”. This filler word would replace any actual proper nouns that are in the transcribed text that we receive from the cloud transcribing service. If we were to use both the filler word and the empty word that we used in this attempt, we would need to distinguish between both words.

When the proper noun is filtered out, we would need to keep track of the position of that pronoun so that we can replace the filler word with the original word after it has passed through the model. Inside the training set for the NMT, we will use this filler pronoun, just like we would a regular pronoun so that the NMT can learn how to use it and weight it properly. We believe this will help us work around the problem our NMT is facing with translating pronouns.

In both attempt 1 and 2 we had not realized that the contents of the training and validation files must be different in some way to produce an accurate NMT model. Not only do the training and validation files need to have different content, the content of the validation model should contain words that are available in the training corpus but should include different orders and combinations of words into phrases that were not originally in the training corpus but should be understood by the NMT.

Attempt 3 — November 6, 2019

Following the success of attempt 2, we decided to maintain this type of training style and to implement the changes that we discussed in attempt 2. We added a “filler” name (CHARLES) that would be used to replace all other names, allowing for easier and more accurate translations from our model. Furthermore, we decided to change the setting of our model and train the model to be used in a school setting instead of a hospital setting. We made this decision because we wanted to reduce the number of proper nouns that we would have to filter as well as make the scope of the project closer to where it will be used.

To discover how to train the NMT model further in an efficient manner, we started to add more variation to the validation files so that we could ensure more accurate translations when non-exact matches were encountered. We took the words from the training sentences and created more sentences combinations to ensure that words are being weighted correctly. The approach of these training sets is similar to the approach in attempt 2 with the use of the empty-word “_” and the word-by-word mapping of words between the English corpora and the ASL-gloss corpora. Like previously aforementioned, with this word-by-word mapping, we lose the grammatical aspects of ASL, but we intend to write another back-end algorithm which will ensure that the output ASL gestures are performed in accordance with correct ASL grammar. Figures 55 and 56 show the test and validation files, respectively, for attempt 3.

Have Homework What grade did you get ? Grade Get When is the test ? What is the test on ? Test Did you study for the test ? Study When is it due ? What did the teacher do today	- HOMEWORK WHAT GRADE BEFORE _ GET? GRADE GET WHEN __ TEST ? WHAT __ TEST ABOUT ? TEST BEFORE _ STUDY __ TEST ? STUDY WHEN __ DUE ? WHAT BEFORE _ TEACHER DO T
English Training Corpus	Gloss Training Corpus

Figure 55. Sample of Test English and Gloss Corpora used in Attempt 3.

Hello how are you ? Do you need help Charles ? Charles is fine Does Charles need help ? Charles does not understand Charles is not sorry What time is it Charles ? Can Charles help you ? Can Charles help me ? I do not feel fine I do not know what time it is Charles knows what time it is	HELLO HOW _ YOU ? _ YOU NEED HELP ? CHARLES _ FINE _ CHARLES NEED HELP ? CHARLES _ NOT UNDERSTAND CHARLES _ NOT SORRY WHAT TIME __ CHARLES ? CAN CHARLES HELP YOU ? CAN CHARLES HELP _ ? ME _ NOT FEEL FINE ME _ NOT KNOW WHAT TIME _ CHARLES KNOW WHAT TIME _
English Validation Corpus	Gloss Validation Corpus

Figure 56. Sample of Validation English and Gloss Corpora for Attempt 3.

After training the NMT using these corpora, we realized that the NMT may be case sensitive; that is, the weighting for the word “What” is not the same as the weighting for the word “what.” This presented us with an inaccurate model. Because of this, we decided to keep the entirety of each English corpus in lowercase characters except for those that are inherently capitalized (such as proper nouns like “Charles” and the pronoun “I”). Figures 57, 58, and 59 show the results of attempt 3 post-training.

```
SENT 1: ['Charles', 'homework']  
PRED 1: CHARLES  
PRED SCORE: -0.0002  
PRED AVG SCORE: -0.0002, PRED PPL: 1.0002
```

Figure 57. Example Results of Attempt 3 Post-Training, Part 1.

```
SENT 1: ['I', 'am', 'in', 'need', 'of', 'help']  
PRED 1: _ YOU NEED HELP ?  
PRED SCORE: -0.8267  
PRED AVG SCORE: -0.1653, PRED PPL: 1.1798
```

Figure 58. Example Results of Attempt 3 Post-Training, Part 2.

```
SENT 1: ['Charles', 'is', 'in', 'need', 'of', 'help']  
PRED 1: CHARLES SORRY  
PRED SCORE: -0.6280  
PRED AVG SCORE: -0.3140, PRED PPL: 1.3689
```

Figure 59. Example Results of Attempt 3 Post-Training, Part 3.

Attempt 4 — November 6, 2019

After realizing the shortcomings of the data set that we had tested, we fixed the issues with capitalization and trained a new model shortly after our third attempt. Figures 60 and 61 show the results of attempt 4 after fixing issues from attempt 3.

```
SENT 1: ['I', 'am', 'in', 'need', 'of', 'help']  
PRED 1: YOU BEFORE HELP GET?  
PRED SCORE: -0.5153  
PRED AVG SCORE: -0.1288, PRED PPL: 1.1375
```

Figure 60. Example Results of Attempt 4 Post-Training, Part 1.

```
SENT 1: ['Charles', 'is', 'in', 'need', 'of', 'help']  
PRED 1: YOU HELP _ ?  
PRED SCORE: -0.3275  
PRED AVG SCORE: -0.0819, PRED PPL: 1.0853
```

Figure 61. Example Results of Attempt 4 Post-Training, Part 2.

While, this model performed slightly better than attempt 3, on closer examination of the data set we realized that we had made several mistakes not only with capitalization but also with words being linked to different translated words in the target files. Furthermore, in this data set, we had tried to replace some words such as ‘did’ with ‘before’ to indicate past tense. However, in doing this, we accidentally would take out the subject of the sentence, such as ‘you’ or ‘me’ and replace them with an empty character. These two issues would affect the weights of those words and skew the predictions of the model greatly. This would have to be fixed in the next data set before trying to train again.

Attempt 5 — November 6, 2019

Attempt 5 takes a detour away from our typical testing at the request of our advisor Dr. Chan. While in our weekly meeting with Dr. Chan, he wanted to see the results of using corpora for training and validation files which consist of single-word entries alone. Therefore, we created a small corpus and validation file to his specifications and trained the model. The goal was to see if the model could be trained to create sentences when being trained with single words. When the model was finished training, we were to see the results of trying to create 2 sentences from this data set: “Where is the classroom?” and “Where is my classroom?”. Figure 62 shows the entirety of the corpus for attempt 5. Figure 63 shows the result of this training set.

the	–
my	MY
where	WHERE
is	–
classroom	CLASSROOM
?	?
English Training Corpus	Gloss Training Corpus

Figure 62. Sample of Test English and Gloss Corpora used in Attempt 5.

```
SENT 1: ['where', 'is', 'the', 'classroom', '?']
PRED 1: –
PRED SCORE: -0.0000

SENT 2: ['where', 'is', 'my', 'classroom', '?']
PRED 2: ?
PRED SCORE: -0.0418
PRED AVG SCORE: -0.0209, PRED PPL: 1.0211
```

Figure 63. Example Results of Attempt 5 Post-Training.

As we can see, the use of single words as entries within the corpora seems to have a negative impact on the success of training the NMT model. While this set was very small, we trained it with the intent of translating these two sentences error-free, but it failed. Furthermore, the scores for these predictions show that the model was fairly confident in

its prediction, suggesting that attempting to train a model using single-word entries is not the correct approach.

Using the results from attempts 3, 4, and 5, we decided we should consider using complete sentences with each word in the source corpora mapped only to one translated word in the target corpora.

7.2 Changes to OpenNMT Training Approach

After deciding to use an NMT as the default machine translation approach for this project, which NMT system would be most viable for the purposes of this project needed to be decided. NMT systems were offered from the larger tech companies; however, these systems remain strictly proprietary which would not prevent modifications the system and train a suitable model to meet the specifications for this project. Open-source NMT systems were also available, and this style of an NMT was the main focus when deciding on which system to choose. After considering various open-source systems, OpenNMT was chosen due to its necessary provided support that other software lacked. Other software that were mainly used for research code did not provide substantial support. OpenNMT also provided support for the PyTorch and TensorFlow frameworks, which provide the ability to train and validate the NMT model from a high-level programming interface. Due to the arbitrary choice between the PyTorch and TensorFlow backbones in the scope of this project, the PyTorch framework was chosen for this project. This OpenNMT software was also based on the aforementioned seq2seq models.

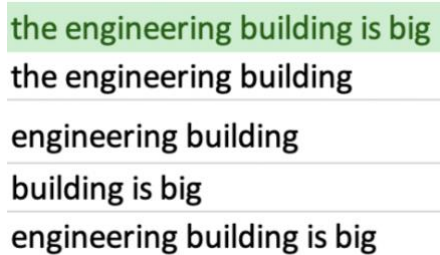
The NMT presented the challenge of finding a GPU capable of training the model in a short amount of time. Training with a CPU would exceed 24 hours for each session and therefore would not be replicable when several models would need to be tweaked and re-trained. Google Research's Colaboratory (abbreviated "Colab") was used to overcome the issue of lacking a capable GPU to train the model. Using Colab allowed code to be written to and executed on Google's cloud servers and also use their available GPUs at no cost. Colab also provides a Python environment and therefore allows for simple integration of the PyTorch framework. Training the NMT model with Google's GPU took about 40 minutes per session.

Training the OpenNMT model requires four different text files. Two of the text files correspond to the English source corpus, and the other two correspond to the ASL-Gloss target corpus. Both the source and target corpora contained 516 entries within each corpus, whereas most training sets for other research teams contain up to hundreds of thousands of entries. A smaller set of corpora was used for the validation files in both English and ASL-Gloss. The validation files had 38 entries for each corpus, and the model used these files to evaluate the convergence of its training.

As aforementioned, NMTs require large training corpora in order to achieve a respectable translation. The unobtainability of large multilingual corpora for English to ASL-Gloss translations resulted in the 516-entry corpora. To overcome the lack of such a large training

corpus, a two-step approach was utilized. The two-step approach included keeping the training corpora to a specific domain and employing redundancy among the sentences.

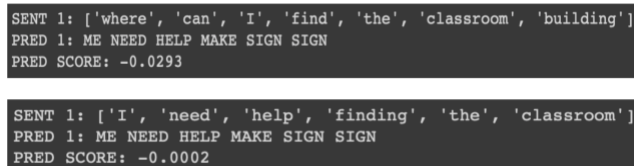
For the first step, the corpora were limited to contain sentences that were related to the school domain. Therefore, this restricted the number of words available throughout both corpora. Choosing the school setting as the primary domain also allowed for the use of a simpler vocabulary set and more commonly used sentences. After implementing the first approach, there showed improvements in the translations; however, the ASL-Gloss translations were still not acceptable for the purposes of this project. Two translation examples English to ASL-Gloss are shown in Figure 64. These two examples demonstrate that some words such as “Me” and “Need” are correctly translated but not the rest of the sentence.



the engineering building is big
the engineering building
engineering building
building is big
engineering building is big

Figure 64. Example of breaking down a sentence from the corpus into smaller phrases for redundancy.

The second step was to add to the training corpora redundancy. This was accomplished by splitting up sentences already in the corpora into smaller phrases as shown in Figure 3. Sentences similar to existing sentences were also introduced to the corpora. This introduction of redundancy greatly improved the performance of the resulting NMT model. This amelioration can be attributed to the neural network needing to force itself to adjust its weights and focus more on the limited and repeated words within the corpora. An example of the vast improvement after implementation of the second approach can be seen from Figure 65. Figure 66A is the previous example still using the first approach and Figure 66B is the same example using the second approach. Between these two figures, the addition of approach two to the training corpora providing a much more accurate and more acceptable translation can be shown. A similar improvement can similarly be seen between Figures 66C and 66D.



```
SENT 1: ['where', 'can', 'I', 'find', 'the', 'classroom', 'building']  
PRED 1: ME NEED HELP MAKE SIGN SIGN  
PRED SCORE: -0.0293
```

```
SENT 1: ['I', 'need', 'help', 'finding', 'the', 'classroom']  
PRED 1: ME NEED HELP MAKE SIGN SIGN  
PRED SCORE: -0.0002
```

Figure 65. Two translation examples from English to Gloss using the first training corpora modification approach. Neither of the attempts were satisfactory translations.

Figure 66A is translated using a corpora that only has approach one, and Figure 66B was translated with the addition of approach two. Similarly, Figure 66C is translated using a

corpora with approach one and Figure 66D was translated with the addition of approach two.

```
A SENT 1: ['where', 'can', 'I', 'find', 'the', 'classroom', 'building']  
   PRED 1: ME NEED HELP MAKE SIGN SIGN  
   PRED SCORE: -0.0293  
B SENT 1: ['where', 'can', 'I', 'find', 'the', 'classroom', 'building']  
   PRED 1: WHERE ME CAN FIND CLASSROOM  
   PRED SCORE: -0.0006  
C SENT 1: ['I', 'do', 'understand', 'the', 'homework']  
   PRED 1: ME NOT WANT KNOW  
   PRED SCORE: -0.2670  
D SENT 1: ['I', 'do', 'understand', 'the', 'homework']  
   PRED 1: ME UNDERSTAND HOMEWORK  
   PRED SCORE: -0.0006
```

Figure 66. Here the first and second approach compared for two translation examples of English to Gloss.

Evaluation of the performance of the model could have been done through several available machine translation scoring techniques. The BLEU score was used to calculate a score for the English to ASL-Gloss translation model. The BLEU score is known to be solution that is simple to implement to retrieve a score; however, this scoring method also include its draw backs [140]. The main objective of the BLEU score is to compare the n-grams of both the human-translated reference sentence and the machine-translated reference sentence [141]. The term “n-gram” refers to a consecutive string of words within a sentence; for example, a unigram is one word, a bigram is a set of two consecutive words, etc. within a sentence. The problem with using the BLEU score arises from the fact that the overall BLEU score for one sentence is calculated as the geometric mean of the all the n-grams calculated for this, where $n \in [1,4]$ for this project’s scoring method. Therefore, for shorter sentences whose word count are less than the highest n-gram (in this case 4-gram), the resulting geometric mean is zero.

Evaluating the BLEU score more accurately was done by utilizing an open-source script, SacreBLEU. The script provides additional smoothing techniques that mitigate the issue when a higher n-gram precision resulted in zero [142]. For the calculation of the bleu score, a human-translation reference corpus and a machine-translated corpus were needed. For this project, 40 entries were used from the available corpora. The overall BLEU score and the subsequent scores for each n-gram match are shown in Table I, where BLEU-1 refers to the unigram-based score, BLEU-2 for digram scores, etc. The BLEU scores shown were multiplied by 100 from their original scores, since the original scores were given between 0 to 1. A score of 100 implies that the entire set of corpora completely match each other. Table I shows that the scores for BLEU-1 and BLEU-2 are much higher than the scores for BLEU-3 and BLEU-4. Since the ASL-Gloss translation is a truncation of English, and shorter English sentences were used to simplify the training, the resulting ASL-Gloss corpus was composed of several sentences containing less than 4 words. Therefore, evaluating the 4-gram BLEU score resulted in a very small number (nonzero as a result of the smoothing provided by SacreBLEU). There is a higher general probability for a unigram match than a digram match, or a digram match compared to a trigram match, and so on; this is why the n-gram BLEU scores show a marked decrease as the n-gram increases. Based on how English to ASL-Gloss translations function for the purposes of

these training sets, the BLEU-1, BLEU-2, and BLEU-3 scores provide a more significant overall score than when combined with the calculated BLEU-4 score.

Table 20. ASLBoT OpenNMT Set of BLEU Scores.

Bleu Score – Corpora of 40 Entries				
<i>Bleu Total</i>	<i>Bleu-1</i>	<i>Bleu-2</i>	<i>Bleu-3</i>	<i>Bleu-4</i>
13.81	62.0	30.2	14.1	1.6

Table 21. Full OpenNMT Training Corpus, Part 1.

English Source Corpus	ASL-GLOSS Target Corpus
I know sign language	ME KNOW SIGN
know sign language	KNOW SIGN
sign language	SIGN
please sign slowly	SIGN SLOW PLEASE
sign slowly	SIGN SLOWLY
please sign	SIGN PLEASE
I need help making these signs	ME NEED HELP MAKE SIGN
need help making these signs	NEED HELP MAKE SIGN
need help making	NEED HELP MAKE
these signs	SIGN
making these signs	MAKE SIGN
help making these signs	HELP MAKE SIGN
help making signs	HELP MAKE SIGN
making signs	MAKE SIGN
I want to learn to sign	ME WANT LEARN SIGN
want to learn sign	WANT TO LEARN SIGN
want to learn to sign	WANT TO LEARN SIGN
I want to learn sign	ME WANT LEARN SIGN
want to learn	WANT LEARN
I want to learn	ME WANT LEARN
I want to learn sign language	ME WANT LEARN SIGN
want to learn sign language	WANT LEARN SIGN
want learn sign language	WANT LEARN SIGN
learn sign language	LEARN SIGN
please sign again slowly	SIGN AGAIN SLOW PLEASE
sign slowly again	SIGN AGAIN SLOW
sign slowly	SIGN SLOW
sign again	SIGN AGAIN
please sign again	SIGN AGAIN PLEASE

Table 22. Full OpenNMT Training Corpus, Part 2.

sorry I don't understand please sign again slowly	SORRY ME NOT UNDERSTAND SIGN AGAIN SLOW PLEASE
I don't understand sign again slowly	ME NOT UNDERSTAND SIGN AGAIN SLOW
don't understand sign again slowly	NOT UNDERSTAND SIGN AGAIN SLOW
I don't understand sign slowly	ME NOT UNDERSTAND SIGN SLOW
don't understand sign slowly	NOT UNDERSTAND SIGN SLOW
sorry don't understand sign slowly	SORRY NOT UNDERSTAND SIGN SLOW
I don't understand please sign again slowly	ME NOT UNDERSTAND SIGN AGAIN SLOW PLEASE
don't understand please sign again slowly	NOT UNDERSTAND SIGN AGAIN SLOW PLEASE
I don't understand please sign slowly	ME NOT UNDERSTAND SIGN SLOW PLEASE
don't understand please sign slowly	NOT UNDERSTAND SIGN SLOW PLEASE
sorry don't understand please sign again slowly	SORRY NOT UNDERSTAND SIGN AGAIN SLOW PLEASE
sorry don't understand please sign slowly	SORRY NOT UNDERSTAND SIGN SLOW PLEASE
do not understand sign again slowly	NOT UNDERSTAND SIGN AGAIN SLOW
I do not understand sign slowly	ME NOT UNDERSTAND SIGN SLOW
do not understand sign slowly	NOT UNDERSTAND SIGN SLOW
sorry do not understand sign again slowly	SORRY NOT UNDERSTAND SIGN AGAIN SLOW
sorry do not understand sign slowly	SORRY NOT UNDERSTAND SIGN SLOW
I do not understand please sign again slowly	ME NOT UNDERSTAND SIGN AGAIN SLOW PLEASE
do not understand please sign again slowly	NOT UNDERSTAND SIGN AGAIN SLOW PLEASE
I do not understand please sign slowly	ME NOT UNDERSTAND SIGN SLOW PLEASE
do not understand please sign slowly	NOT UNDERSTAND SIGN SLOW PLEASE
sorry do not understand please sign again slowly	SORRY NOT UNDERSTAND SIGN AGAIN SLOW PLEASE
sorry do not understand please sign slowly	SORRY NOT UNDERSTAND SIGN SLOW PLEASE
sorry do not understand	SORRY NOT UNDERSTAND

Table 23. Full OpenNMT Training Corpus, Part 3.

sorry I do not understand	SORRY ME NOT UNDERSTAND
sorry I don't understand	SORRY ME NOT UNDERSTAND
sorry don't understand	SORRY NOT UNDERSTAND
please sign again slowly	SIGN AGAIN SLOW PLEASE
please sign again	SIGN AGAIN PLEASE
please sign	SIGN PLEASE
please sign slowly	SIGN SLOW PLEASE
do you want to learn a sign language	YOU WANT LEARN SIGN
you want to learn a sign language	YOU WANT LEARN SIGN
do you want to learn sign language	YOU WANT LEARN SIGN
do you want to learn sign	YOU WANT LEARN SIGN
you want to learn sign language	YOU WANT LEARN SIGN
you want to learn sign	YOU WANT LEARN SIGN
I understand	ME UNDERSTAND
I want to understand	ME WANT UNDERSTAND
want to understand	WANT UNDERSTAND
I want to	ME WANT
I do not understand	ME NOT UNDERSTAND
don't understand	NOT UNDERSTAND
I do understand	ME UNDERSTAND
do understand	UNDERSTAND
do not understand	NOT UNDERSTAND
not understand	NOT UNDERSTAND
I don't understand	ME NOT UNDERSTAND
I do not want to understand	ME NOT WANT UNDERSTAND
I don't want to understand	ME NOT WANT UNDERSTAND
do not want to understand	NOT WANT UNDERSTAND
don't want to understand	NOT WANT UNDERSTAND
not want to understand	NOT WANT UNDERSTAND
why don't I understand how to do this homework	WHY ME NOT UNDERSTAND HOW DO HOMEWORK
why don't understand how to do this homework	WHY NOT UNDERSTAND HOW DO HOMEWORK
why not understand how to do this homework	WHY NOT UNDERSTAND HOW DO HOMEWORK
do not understand how to do this homework	NOT UNDERSTAND HOW TO DO HOMEWORK

Table 24. Full OpenNMT Training Corpus, Part 4.

I do not understand how to do this homework	ME NOT UNDERSTAND HOW TO DO HOMEWORK
I don't understand how to do this homework	ME NOT UNDERSTAND HOW TO DO HOMEWORK
why don't understand how to do homework	WHY NOT UNDERSTAND HOW DO HOMEWORK
why not understand how to do homework	WHY NOT UNDERSTAND HOW DO HOMEWORK
do not understand how to do homework	NOT UNDERSTAND HOW TO DO HOMEWORK
don't understand how to do homework	NOT UNDERSTAND HOW TO DO HOMEWORK
I do not understand how to do homework	ME NOT UNDERSTAND HOW TO DO HOMEWORK
I don't understand how to do homework	ME NOT UNDERSTAND HOW TO DO HOMEWORK
to do homework	TO DO HOMEWORK
how to do homework	HOW TO DO HOMEWORK
how to do this homework	HOW TO DO HOMEWORK
to do this homework	TO DO HOMEWORK
I am going to college	COLLEGE ME GO
going to college	GO COLLEGE
am going to college	COLLEGE ME GO
I go to college	COLLEGE ME GO
go to college	COLLEGE GO
I am going to college this semester	NOW SEMESTER ME GO COLLEGE
am going to college this semester	NOW SEMESTER ME GO COLLEGE
going to college this semester	NOW SEMESTER GO COLLEGE
will go to college this semester	NOW SEMESTER GO COLLEGE
this semester I am going to college	NOW SEMESTER ME GO COLLEGE
I am making new friends in college	AT COLLEGE ME MAKE NEW FRIEND
making new friends in college	AT COLLEGE MAKE NEW FRIEND
I am making friends in college	AT COLLEGE ME MAKE FRIEND
making friends in college	AT COLLEGE MAKE FRIEND
I wanted to make a friend in college	ME WANT MAKE FRIEND AT COLLEGE
wanted to make a friend in college	WANT MAKE FRIEND AT COLLEGE
wanted to make friend in college	WANT MAKE FRIEND AT COLLEGE

Table 25. Full OpenNMT Training Corpus, Part 5.

wanted to make friends in college	WANT MAKE FRIEND AT COLLEGE
I want to make a friend in college	ME WANT MAKE FRIEND AT COLLEGE
want to make a friend in college	WANT MAKE FRIEND AT COLLEGE
want to make friend in college	WANT MAKE FRIEND AT COLLEGE
I want to make friends in college	ME WANT MAKE FRIEND AT COLLEGE
want to make friends in college	WANT MAKE FRIEND AT COLLEGE
want to make	WANT MAKE
I want to make	ME WANT MAKE
I want	ME WANT
make friends	MAKE FRIEND
make friend	MAKE FRIEND
why can't I make friends in college	WHY ME CANNOT MAKE FRIEND AT COLLEGE
why can't make friends in college	WHY CANNOT MAKE FRIEND AT COLLEGE
I can't make friends in college	ME CANNOT MAKE FRIEND AT COLLEGE
make friends in college	MAKE FRIEND AT COLLEGE
why make friends in college	WHY MAKE FRIEND AT COLLEGE
I make friends in college	ME MAKE FRIEND AT COLLEGE
I make friends	ME MAKE FRIEND
I can make friends in college	ME CAN MAKE FRIEND AT COLLEGE
can make friends in college	CAN MAKE FRIEND AT COLLEGE
I can make friends	ME CAN MAKE FRIEND
weren't I supposed to find friends in college this year	THIS YEAR ME SHOULD FIND FRIEND AT COLLEGE
I was supposed to find friends in college this year	THIS YEAR ME SHOULD FIND FRIEND AT COLLEGE
was supposed to find friends this year	THIS YEAR SHOULD FIND FRIEND
was supposed to find friends	SHOULD FIND FRIEND
I was supposed to find friends this year	THIS YEAR ME SHOULD FIND FRIEND
I was supposed to find friends	ME SHOULD FIND FRIEND
supposed to find friends	SHOULD FIND FRIEND
weren't I supposed to find a friend in college this year	THIS YEAR ME SHOULD FIND FRIEND AT COLLEGE
I was supposed to find a friend in college this year	THIS YEAR ME SHOULD FIND FRIEND AT COLLEGE
was supposed to find a friend in college this year	THIS YEAR SHOULD FIND FRIEND AT COLLEGE
was supposed to find a friend this year	THIS YEAR SHOULD FIND FRIEND

Table 26. Full OpenNMT Training Corpus, Part 6.

was supposed to find a friend	SHOULD FIND FRIEND
I was supposed to find a friend this year	THIS YEAR ME SHOULD FIND FRIEND
I was supposed to find a friend	ME SHOULD FIND FRIEND
supposed to find a friend	SHOULD FIND FRIEND
when does this class meet	WHEN CLASS MEET
when does class meet	WHEN CLASS MEET
when will class meet	WHEN CLASS MEET
how are your engineering classes	HOW YOU ENGINEERING CLASS
how is engineering class	HOW ENGINEERING CLASS
how are engineering classes	HOW ENGINEERING CLASS
how is your engineering class	HOW YOU ENGINEERING CLASS
engineering class	ENGINEERING CLASS
how are you	HOW YOU
you should be going to class now	YOU SHOULD GO CLASS NOW
should be going to class now	SHOULD GO CLASS NOW
should be going now	SHOULD GO NOW
should be going	SHOULD GO
going to class now	GO CLASS NOW
class now	CLASS NOW
going to class	GO CLASS
you should be going now	YOU SHOULD GO NOW
you should be going	YOU SHOULD GO
you should	YOU SHOULD
what classes are you taking	WHAT CLASS YOU TAKE
what class are you taking	WHAT CLASS YOU TAKE
what class	WHAT CLASS
are you taking	YOU TAKE
I took that class last semester	PAST SEMESTER THAT CLASS ME TAKE
I took that class	THAT CLASS ME TAKE
took that class	THAT CLASS TAKE
that class	THAT CLASS
I was absent from class yesterday	YESTERDAY ME ABSENT FROM CLASS
I was absent yesterday	YESTERDAY ME ABSENT
was absent from class yesterday	YESTERDAY ABSENT FROM CLASS
last year my classes weren't so difficult	LAST YEAR MY CLASS NOT DIFFICULT
last year	LAST YEAR
my classes weren't so difficult	MY CLASS NOT DIFFICULT
classes weren't so difficult	CLASS NOT DIFFICULT

Table 27. Full OpenNMT Training Corpus, Part 7.

class was difficult	CLASS DIFFICULT
last year class was difficult	LAST YEAR CLASS DIFFICULT
last year my class was difficult	LAST YEAR MY CLASS DIFFICULT
I want to go to class now	ME WANT GO CLASS NOW
want to go to class now	WANT GO CLASS NOW
want to go to class	WANT GO CLASS
I want to go to class	ME WANT GO CLASS
I want to	ME WANT
I want to go	ME WANT GO
want to go	WANT GO
class now	CLASS NOW
I made a friend yesterday	YESTERDAY ME MAKE FRIEND
made a friend yesterday	YESTERDAY MAKE FRIEND
I made a friend	ME MAKE FRIEND
made a friend	MAKE FRIEND
I made	ME MAKE
Do you want to be friends	YOU WANT FRIEND
want to be friends	WANT FRIEND
you want to be friends	YOU WANT FRIEND
I am learning ASL	ME LEARN ASL
learning ASL	LEARN ASL
learn ASL	LEARN ASL
I will learn ASL	ME LEARN ASL
will learn ASL	LEARN ASL
I did not finish my homework	HOMEWORK ME NOT FINISH
I did not	ME NOT
I did not finish	ME NOT FINISH
did not finish my homework	HOMEWORK NOT FINISH
I did not finish the homework	HOMEWORK ME NOT FINISH
finish the homework	HOMEWORK FINISH
was the homework supposed to be done individually	HOMEWORK SHOULD DO INDIVIDUAL
is the homework done	HOMEWORK DONE
homework done	HOMEWORK DONE
the homework was done individually	HOMEWORK DO INDIVIDUAL
when did you finish your homework	WHEN YOU FINISH HOMEWORK
when did you finish	WHEN YOU FINISH
did you finish your homework	YOU FINISH HOMEWORK

Table 28. Full OpenNMT Training Corpus, Part 8.

finish your homework	YOU FINISH HOMEWORK
where can I find a quiet building to do my homework	WHERE ME CAN FIND QUIET BUILDING DO HOMEWORK
where can I find a quiet building	WHERE ME CAN FIND QUIET BUILDING
where to find a quiet building	WHERE FIND QUIET BUILDING
find a quiet building to do homework	FIND QUIET BUILDING DO HOMEWORK
I want to find a quiet building for homework	ME WANT FIND QUIET BUILDING DO HOMEWORK
where to find a quiet building for homework	WHERE FIND QUIET BUILDING DO HOMEWORK
where to find a quiet building to do my homework	WHERE FIND QUIET BUILDING DO HOMEWORK
to do homework	DO HOMEWORK
I want to find a quiet place on campus because I need to finish my homework	ME WANT FIND QUIET PLACE AT CAMPUS BECAUSE NEED FINISH HOMEWORK
I want to find a quiet place on campus	ME WANT FIND QUIET PLACE AT CAMPUS
I need to finish my homework	NEED FINISH HOMEWORK
I want to find a quiet place	ME WANT FIND QUIET PLACE
on campus	AT CAMPUS
want to find a quiet place on campus for homework	WANT FIND QUIET PLACE AT CAMPUS DO HOMEWORK
I want to find a quiet place on campus for homework	ME WANT FIND QUIET PLACE AT CAMPUS DO HOMEWORK
I want a quiet place on campus because I need to finish my homework	ME WANT FIND QUIET PLACE AT CAMPUS BECAUSE NEED FINISH HOMEWORK
I want a quiet place on campus for homework	ME WANT FIND QUIET BUILDING DO HOMEWORK
why don't I understand how to do this homework	WHY ME NOT UNDERSTAND HOW DO HOMEWORK
why don't I understand	WHY ME NOT UNDERSTAND
how to do this homework	HOW DO HOMEWORK
why don't I understand how	WHY ME NOT UNDERSTAND HOW
don't understand how to do this homework	NOT UNDERSTAND HOW DO HOMEWORK
I don't understand how to do this homework	ME NOT UNDERSTAND HOW DO HOMEWORK
why don't I understand the homework	WHY ME NOT UNDERSTAND HOMEWORK

Table 29. Full OpenNMT Training Corpus, Part 9.

how to do	HOW DO
how to do the homework	HOW DO HOMEWORK
I understand the homework	ME UNDERSTAND HOMEWORK
I understand how to do this homework	ME UNDERSTAND HOW DO HOMEWORK
understand how to do homework	UNDERSTAND HOW DO HOMEWORK
where did you find the classroom	WHERE YOU FIND CLASSROOM
where did you find this classroom	WHERE YOU FIND CLASSROOM
where did you find	WHERE YOU FIND
what did you find	WHAT YOU FIND
did you find this classroom	YOU FIND CLASSROOM
did you find the classroom	YOU FIND CLASSROOM
where do you find	WHERE YOU FIND
you find the classroom	YOU FIND CLASSROOM
find the classroom	FIND CLASSROOM
the classroom	CLASSROOM
I need help because I can not find this classroom	ME NEED HELP BECAUSE CANNOT FIND CLASSROOM
I need help	ME NEED HELP
need help	NEED HELP
I need help to find the classroom	FIND CLASSROOM ME NEED HELP
need help because I can not find this classroom	NEED HELP BECAUSE ME CANNOT FIND CLASSROOM
need help because can not find this classroom	NEED HELP BECAUSE CANNOT FIND CLASSROOM
I need help because I can not find the classroom	ME NEED HELP BECAUSE CANNOT FIND CLASSROOM
I can not find this classroom	ME CANNOT FIND CLASSROOM
I can't find this classroom	ME CANNOT FIND CLASSROOM
I can not find the classroom	ME CANNOT FIND CLASSROOM
can't find this classroom	CANNOT FIND CLASSROOM
can not find this classroom	CANNOT FIND CLASSROOM
can't find the classroom	CANNOT FIND CLASSROOM
can not find the classroom	CANNOT FIND CLASSROOM
find this classroom	FIND CLASSROOM
find the classroom	FIND CLASSROOM
the engineering building is big	ENGINEERING BUILDING BIG
the engineering building	ENGINEERING BUILDING
engineering building	ENGINEERING BUILDING
building is big	BUILDING BIG

Table 30. Full OpenNMT Training Corpus, Part 10.

engineering building is big	ENGINEERING BUILDING BIG
do you want to go with me to the engineering building	YOU WANT GO WITH ME ENGINEERING BUILDING
do you want to go with me	YOU WANT GO WITH ME
do you want to go	YOU WANT GO
want to go	WANT GO
you want to go	YOU WANT GO
want to go with me to the engineering building	WANT GO WITH ME ENGINEERING BUILDING
want to go with me	WANT GO WITH ME
you want to go with me	YOU WANT GO WITH ME
did you go to the engineering building with me yesterday	YESTERDAY YOU GO ENGINEERING BUILDING WITH ME
did you go to the engineering building	YOU GO ENGINEERING BUILDING
did you go	YOU GO
go with me	GO WITH ME
did you go to the engineering building yesterday	YESTERDAY YOU GO ENGINEERING BUILDING
go to the engineering building	GO ENGINEERING BUILDING
I wanted to research in the engineering building	ME WANT RESEARCH AT ENGINEERING BUILDING
I wanted to research in the building	ME WANT RESEARCH AT BUILDING
I wanted to do research	ME WANT DO RESEARCH
I want to do research	ME WANT DO RESEARCH
research in the engineering building	RESEARCH AT ENGINEERING BUILDING
research is done at the engineering building	RESEARCH DONE AT ENGINEERING BUILDING
wanted to do research in the engineering building	WANT RESEARCH AT ENGINEERING BUILDING
I wanted to do research at the engineering building	ME WANT DO RESEARCH AT ENGINEERING BUILDING
I want to do research at the engineering building	ME WANT DO RESEARCH AT ENGINEERING BUILDING
in the engineering building	AT ENGINEERING BUILDING
at the engineering building	AT ENGINEERING BUILDING
there are a few ways to get to that building	FEW WAY GO THAT BUILDING
few ways to get to that building	FEW WAY GO THAT BUILDING
get to that building	GO THAT BUILDING

Table 31. Full OpenNMT Training Corpus, Part 11.

ways to get to that building	WAY GO THAT BUILDING
there are few ways	FEW WAY
that building	THAT BUILDING
that is how to get to that building	THAT HOW GO THAT BUILDING
that is how	THAT HOW
to get to that building	GO THAT BUILDING
how many buildings are there on campus	BUILDING ON CAMPUS HOW MUCH
how many buildings	BUILDING HOW MUCH
how many buildings are on campus	BUILDING ON CAMPUS HOW MUCH
what buildings are on campus	WHAT BUILDING ON CAMPUS
are on campus	ON CAMPUS
what buildings are there on campus	WHAT BUILDING ON CAMPUS
are there on campus	ON CAMPUS
how many buildings are there	BUILDING HOW MUCH
how many	HOW MUCH
where is the science building	WHERE SCIENCE BUILDING
the science building	SCIENCE BUILDING
can you help me find this building	CAN HELP ME FIND BUILDING
can you help me	CAN HELP ME
can you help	CAN HELP
help me	HELP ME
find the building	FIND BUILDING
I need to go to the science building now	ME NEED GO SCIENCE BUILDING NOW
I need to go	ME NEED GO
the science building	SCIENCE BUILDING
I need to go now	ME NEED GO NOW
I need to go to the science building	ME NEED GO SCIENCE BUILDING
need to go to the science building	NEED GO SCIENCE BUILDING
need to go to the science building now	NEED GO SCIENCE BUILDING NOW
go to the science building	GO SCIENCE BUILDING
they made so many buildings on this campus	AT CAMPUS MAKE MUCH BUILDING
they made so many buildings	MAKE MUCH BUILDINGS
on this campus	AT CAMPUS

Table 32. Full OpenNMT Training Corpus, Part 12.

they made so many	MAKE MUCH
where can I find a quiet building to do my homework	WHERE ME CAN FIND QUIET BUILDING DO HOMEWORK
where can I find a quiet building	WHERE ME CAN FIND QUIET BUILDING
where can I find	WHERE ME CAN FIND
quiet building to do my homework	QUIET BUILDING DO HOMEWORK
building to do my homework	BUILDING DO HOMEWORK
building to do homework	BUILDING DO HOMEWORK
on campus	AT CAMPUS
why is the campus large	WHY CAMPUS BIG
the campus is large	CAMPUS BIG
where is a quiet place on campus	WHERE QUIET PLACE AT CAMPUS
where is a quiet place	WHERE QUIET PLACE
a quiet place on campus	QUIET PLACE AT CAMPUS
where is the quiet place on campus	WHERE QUIET PLACE AT CAMPUS
where can I do research on campus	WHERE CAN ME DO RESEARCH AT CAMPUS
where can I do research	WHERE CAN ME DO RESEARCH
research on campus	RESEARCH AT CAMPUS
I want to find a quiet place on campus because I need to finish my homework	ME WANT FIND QUIET PLACE AT CAMPUS BECAUSE NEED FINISH HOMEWORK
I want to find a quiet place on campus	ME WANT FIND QUIET PLACE AT CAMPUS
I want to find a quiet place	ME WANT FIND QUIET PLACE
find a quiet place	FIND QUIET PLACE
because I need to finish my homework	BECAUSE ME NEED FINISH HOMEWORK
I need to finish my homework	ME NEED FINISH HOMEWORK
finish my homework	ME FINISH HOMEWORK
I want to start doing research this semester	SEMESTER ME WANT START RESEARCH
I want to start doing research	ME WANT START RESEARCH
start doing research this semester	SEMESTER START RESEARCH
want start doing research this semester	SEMESTER WANT START RESEARCH
why can't I find a place to do research	WHY ME CANNOT FIND PLACE DO RESEARCH
why can't I find a place	WHY ME CANNOT FIND PLACE

Table 33. Full OpenNMT Training Corpus, Part 13.

find a place to do research	FIND PLACE DO RESEARCH
find a place	FIND PLACE
why can't I find	WHY ME CANNOT FIND PLACE
where can I do research on campus	WHERE CAN DO RESEARCH AT CAMPUS
where can I do research	WHERE CAN DO RESEARCH
research on campus	RESEARCH AT CAMPUS
don't you want to study with me later	LATER NOT WANT STUDY WITH ME
don't you want to study	NOT WANT STUDY
don't you want to study with me	NOT WANT STUDY WITH ME
study with me later	LATER STUDY WITH ME
want to study with me later	LATER WANT STUDY WITH ME
tomorrow I want to find somewhere to study	TOMORROW ME WANT FIND SOMEWHERE STUDY
I want to find somewhere to study	ME WANT FIND SOMEWHERE STUDY
somewhere to study	SOMEWHERE STUDY
I want to find	ME WANT FIND
study	STUDY
to	<unk>
I	ME
understand	UNDERSTAND
do	<unk>
sign	SIGN
homework	HOMEWORK
building	BUILDING
want	WANT
the	<unk>
not	NOT
you	YOU
this	<unk>
slowly	SLOW
find	FIND
how	HOW
engineering	ENGINEERING
college	COLLEGE
don't	NOT
go	GO
again	AGAIN
a	<unk>

Table 34. Full OpenNMT Training Corpus, Part 14.

in	<unk>
friends	FRIEND
sorry	SORRY
need	NEED
class	CLASS
please	PLEASE
is	<unk>
classroom	CLASSROOM
can	CAN
campus	CAMPUS
quiet	QUIET
me	ME
learn	LEARN
help	HELP
wanted	WANT
research	RESEARCH
my	MY
on	AT
why	WHY
make	MAKE
language	LANGUAGE
with	WITH
where	WHERE
going	GO
place	PLACE
many	MUCH
did	DO
classes	CLASS
your	YOUR
semester	SEMESTER
finish	FINISH
can't	CANNOT
buildings	BUILDING
because	BECAUSE
are	<unk>
now	NOW
yesterday	YESTERDAY
few	FEW
there	THERE

Table 35. Full OpenNMT Training Corpus, Part 15.

so large	BIG
should	SHOULD
science	SCIENCE
name	NAME
making	MAKE
for	<unk>
be	<unk>
ways	WAY
that	THAT
supposed	<unk>
big	BIG
asl	ASL
when	WHEN
take	TAKE
made	MAKE
learning	LEARN
get	GO
year	YEAR
what	WHAT
weren't	NOT
study	STUDY
new	NEW
meet	MEET
large	BIG
individually	INDIVIDUAL
friend	FRIEND
done	DONE
was	<unk>
they	<unk>
taking	TAKE
signs	SIGN
last	LAST
know	KNOW
five	FIVE
does	<unk>
will	<unk>
what's	WHAT
took	TAKE
tomorrow	TOMORROW

Table 36. Full OpenNMT Training Corpus, Part 16.

these	<unk>
start	START
somewhere	SOMEWHERE
later	LATER
from	FROM
doing	<unk>
difficult	DIFFICULT
absent	ABSENT
my name is	MY NAME
what is your name	YOU NAME WHAT
my name	MY NAME
what's your name	YOU NAME WHAT
what your name	YOU NAME WHAT
your name	YOU NAME WHAT
charles	<unk>
bob	<unk>
miami	<unk>
florida	<unk>
gustavo	<unk>
luis	<unk>
jared	<unk>
michael	<unk>
where is the science building	WHERE SCIENCE BUILDING
science building	SCIENCE BUILDING
I want to go to the science building	ME WANT GO SCIENCE BUILDING
how many science buildings are there	SCIENCE BUILDING HOW MUCH
where are the science buildings on campus	WHERE SCIENCE BUILDING ON CAMPUS

Table 37. English to ASL-Gloss Validation Corpus, Part 1.

English Validation	ASL-GLOSS Validation
I want to know sign language	ME WANT KNOW SIGN LANGUAGE
can you please sign slowly	YOU SIGN SLOWLY PLEASE
I want help for making these signs	ME WANT HELP MAKE SIGN
I need to learn sign	ME NEED LEARN SIGN
Do you need to learn a sign language	YOU NEED LEARN SIGN
I need to understand how to do this homework	ME NEED UNDERSTAND HOW DO HOMEWORK
I will be going to college this semester	NOW SEMESTER ME GO COLLEGE
I need to make new friends in college	AT COLLEGE ME NEED MAKE NEW FRIEND
I am supposed to make new friends in college	ME SHOULD FIND FRIEND AT COLLEGE
when does the class meet	WHEN CLASS MEET
how are your engineering classes this semester	NOW SEMESTER HOW YOU ENGINEERING CLASS
you need to go to class now	YOU NEED GO CLASS NOW
I need to take that class this semester	NOW SEMESTER ME NEED TAKE CLASS
this semester I was absent from class	NOW SEMESTER ME ABSENT FROM CLASS
this semester my classes are not difficult	NOW SEMESTER MY CLASS NOT DIFFICULT
I will go to class now	ME GO CLASS NOW
I made a friend in college	ME MAKE FRIEND AT COLLEGE
do you want to be friends in college	YOU WANT FRIEND AT COLLEGE
I need to learn ASL	ME NEED LEARN ASL
This semester the homework was difficult	NOW SEMESTER HOMEWORK DIFFICULT
sorry don't understand sign again slowly	SORRY NOT UNDERSTAND SIGN AGAIN SLOW
don't understand how to do this homework	NOT UNDERSTAND HOW TO DO HOMEWORK
I wanted to make friends in college	ME WANT MAKE FRIEND AT COLLEGE
was supposed to find friends in college this year	THIS YEAR SHOULD FIND FRIEND AT COLLEGE

Table 38. English to ASL-Gloss Validation Corpus, Part 2.

understand how to do the homework	UNDERSTAND HOW DO HOMEWORK
need help because cannot find the classroom	NEED HELP BECAUSE CANNOT FIND CLASSROOM
where to find a quiet building to do homework	WHERE FIND QUIET BUILDING DO HOMEWORK
I want to do research in the engineering building	ME WANT DO RESEARCH AT ENGINEERING BUILDING
I can't find the classroom	ME CANNOT FIND CLASSROOM
I wanted to do research in the engineering building	ME WANT DO RESEARCH AT ENGINEERING BUILDING
want to do research in the engineering building	WANT RESEARCH AT ENGINEERING BUILDING
can't make friends in college	CANNOT MAKE FRIEND AT COLLEGE
I do not understand sign again slowly	ME NOT UNDERSTAND SIGN AGAIN SLOW
find a quiet building for homework	FIND QUIET BUILDING DO HOMEWORK
did not finish the homework	HOMEWORK NOT FINISH
find this building	FIND BUILDING
can you help me find the building	CAN HELP ME FIND BUILDING
there are a few ways	FEW WAY

7.3 Changes to PCB Design and Hardware Functionality

This section addresses the major changes between the original PCB and hardware design from Senior Design I to the final PCB and hardware design ultimately used in the final product.

7.3.1 Hardware Prototyping

In the initial design for the prototype of the print circuit board for the device, an FPGA was originally chosen as the brains of the board. However, while investigating creating a custom design PCB, the time it would take to design this PCB was decided not to be feasible given the time constraint of the senior design class. FPGAs tend to require multiple different voltage rails to power different parts of its computer. Furthermore, there are multiple I/O banks with clocks to be fed in from an oscillator. Given the complexity and time requirements of the FPGA, using a microcontroller unit was considered to be more feasible instead. The microcontroller used was the MSP430FR6989 Launchpad board as this MCU is part of a widely used and highly documented family of microcontrollers with low power consumption and many pertaining features for our project. Furthermore, our team has had experience working with this family of microcontrollers, so there would be no learning curve for creating embedded code.

Since the board featured status LEDs, seven different LEDs were placed onto a solderless breadboard and were connected to the digital I/O pins on the development board. Seven LEDs were selected to represent power, recording mode, idle mode, transmission indicator, receiving indicator, “translation-in-progress” status, and “animation-being-displayed” status. Different values of dropping resistors were considered that would give us the optimal illumination level. The final set of resistors chosen was rated for 1 k Ω , as these using the LEDs with these resistors provided an output on the LEDs that were bright enough to see easily in multiple lighting conditions. The connections for the prototyped LEDs can be seen in figure 67.

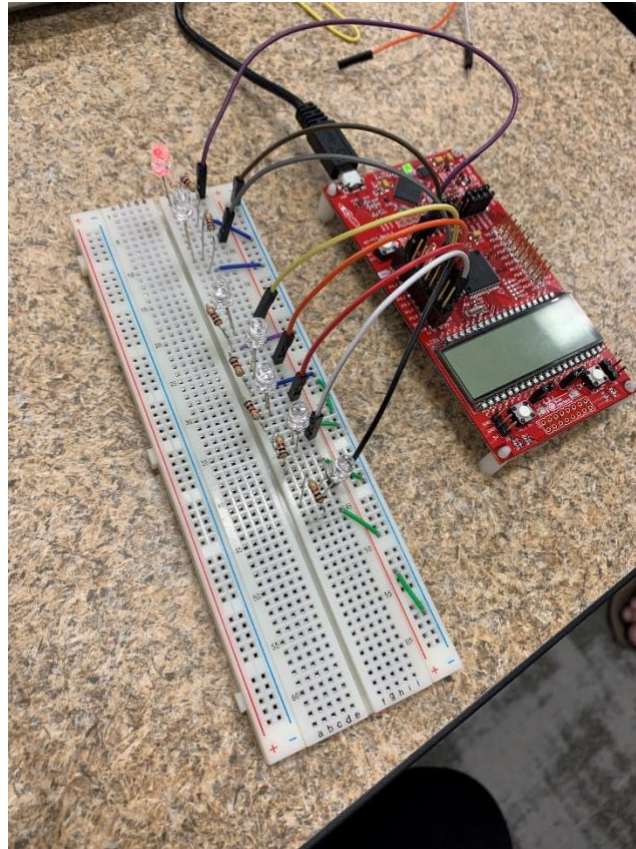


Figure 67. Prototyping with LEDs and MSP430FR6989.

Furthermore, experimentation continued with the connecting push buttons on the launchpad board to control our product. Three push buttons were connected to the solderless breadboard and wired to the board. Two of the buttons were connected to digital I/O pins to receive input from the user on whether to start or stop recording and send control signals to the single board computer. Experimentation with these push buttons showed that these buttons required to be connected to pull-up resistors for the microcontroller to read the inputs from the button properly. Interrupts were also required to interface with the buttons so that the microcontroller could be kept in a low-power mode. The third button was connected to the reset pin of the launchpad board as this button would act as a reset button for our PCB in the case the board locked up while running its software and needed to be hard reset by the user.

During the prototyping stage, an I2S microphone was once component that we had purchased. I2S is a serial bus interface standard used to connect digital audio devices without the need for convertors. This was extremely desirable for our product as we would be able to capture the audio from the microphone on the microcontroller and send it to the single board computer.

However, when investigating the support for this microphone, the I2S protocol was found not to be supported in full by the MSP430 family. While TI provided documentation on interfacing I2S devices with their board's Universal Serial Communication interfaces, this interface requires extra hardware to implement such an interface correctly and completely. This interface uses the USCI SPI port running the SPI communication protocol and connects a 4-bit counter and D flip-flop to create an 'I2S-like Left-Justified SPI' implementation. This implementation seems to be device dependent because the resulting interface is not the official I2S protocol, and certain devices will not be able to correctly interface. Unfortunately, this was the case for the microphone that we had purchased as it did not have enough documentation available to get a working communication line.

To circumvent this another digital device interface, Universal Serial Bus, was implemented to connect a microphone to our system. A USB-enabled Blue Snowball microphone was already owned by one of the group members, and this microphone fit the requirements for the microphone specifications, and the group elected to use this device. The Blue Snowball was connected directly to the UDOO as this single board computer fully supported USB interface and would cut down on the latency of capturing and passing the audio information from the PCB board.

7.3.2 PCB Design

When designing the printed circuit board for the final product there were a handful of features that were to be implemented. The status LEDs were required such that the user could see them and instantly know what the system was executing. Along this same line, the current status of the microphone was also to be represented by having an LCD display that would show if the microphone was recording. Next, buttons were implemented so that the user could control the system easily and have a connection to the single board computer that we are using to run the graphics software. Finally, the user is able to wake the system from sleep by using an IR remote and receiver that was connected to the board. The schematic for our PCB board can be seen in figure 68 and 69.

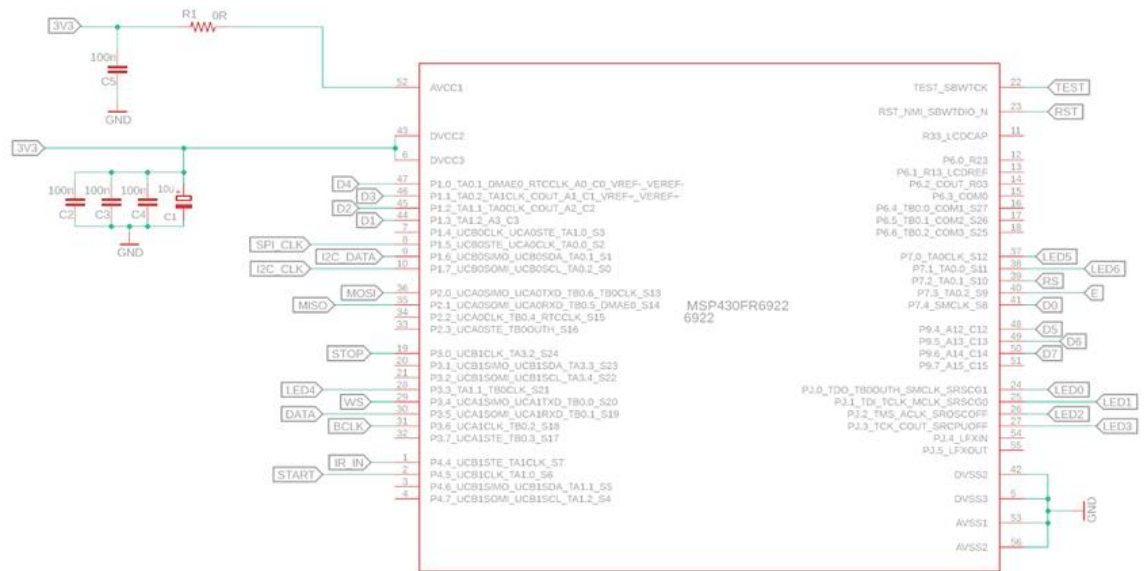


Figure 68. Final Schematic - MSP430FR6922 connections.

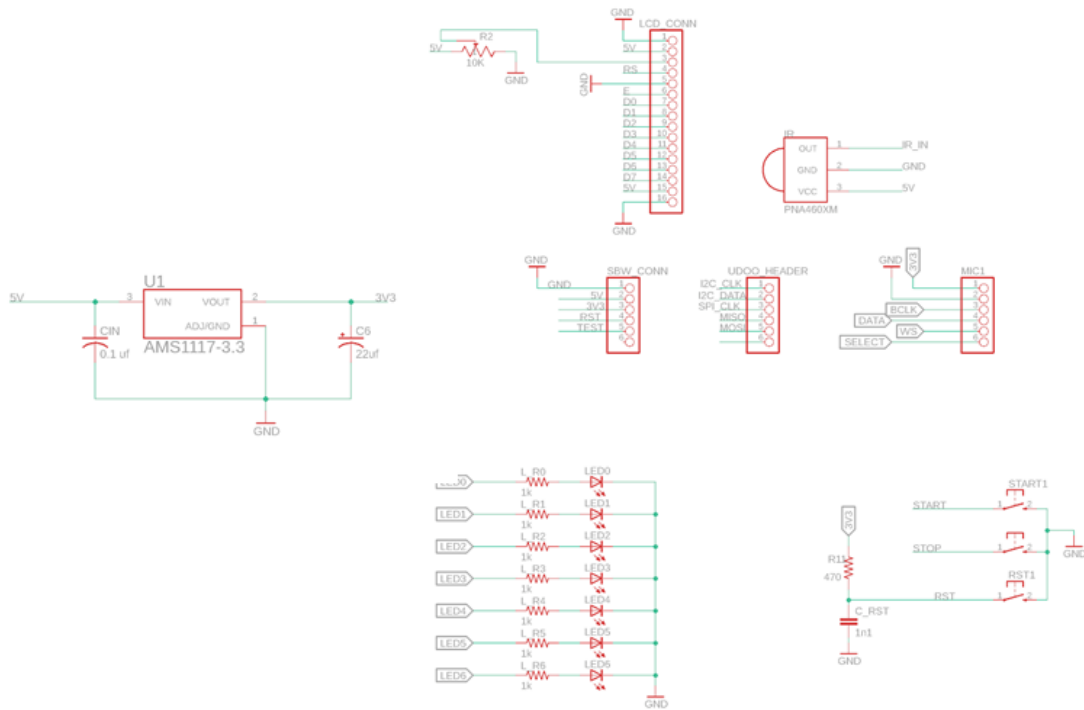


Figure 69. Final Schematic - Component Connections.

These goals were accomplished by using a MSP430FR6922 microcontroller unit as the brain of the printed circuit board. This MCU provided us with the I/O pins required to connect everything to the board and included a small, 56-pin TSSOP package. This package type made the MCU easier to solder to the PCB without much previous soldering experience. Furthermore, due to this group's experience with working on TI's line of MSP MCUs and their embedded development environment, the development process was much smoother. Finally, this microcontroller chip was low cost, running only \$5.33.

Even though the development platform was the MSP430FR6989, when we investigated prototyping a PCB with this unit, we found that the chip only came in large pin packages. The smallest number of pins offered was 80 and came in a LQFP package, which would make soldering the unit to the board manually extremely difficult. Therefore, we chose to purchase the MSP430FR6922 because this MCU offered all the same features but came in a more developer-friendly package. Since these chips were in the same family, there were only few noticeable differences between the development of code between these chips.

Program the PCB's microcontroller unit was done by including a 5-pin header that would connect to the eZ-FET emulator located on the development board. This is provided by TI to program their chips through the 2-wire serialized JTAG programming interface, Spy-Bi-Wire. The header featured 3 pins connected to the 5-volt, 3-volt, and ground pins on the emulator to tie the boards to a common plane and 2 pins that connected to the SBW reset and test pins. These two pins are the data and clock pins for the programming interface. This header is also used to power the board by connecting the 5-volt and ground pins to the Arduino pinouts located on our UDOO x86 single board computer.

Next, a 2-pin header was also included that was connected to the UDOO board. This header would allow the PCB to communicate with the UDOO through the I2C 2-wire serial communication protocol. As shown in figure 70, we can see that the header featured 3 more pins. These pins were originally slated to be used to transmit the audio data from the microcontroller unit to the UDOO board, but after failing to implement the I2S microphone, these pins were left unused. This was also the case for the 'MIC1' header located in the lower left corner of the board layout. The MIC1 was to be where the I2S microphone connected the printed circuit board.

The next header included on the board was the 16-pin LCD display header. This header allowed for connection of the 16x2 LCD display to display the status of the microphone to the user. This header features 5-volt power connections for the display and the backlight, a variable voltage pin to control the contrast of the display, and 11 control pins. 8 of these controls are the data pins used to pass the bits of commands and data to the display. The display was operated in 4-bit mode; i.e. only 4 data pins would need to be used to send commands and text to the display. The 3 other control pins are used for the enable, register select, and Read/Write pin. The variable voltage pin was connected to a 10 k Ω trimmer resistor. This allowed us to control the voltage seen by the pin changing the contrast of the screen so that viewing the characters on the screen could be optimized.

The final header on the board was a 3-pin header that was used to connect the IR receiver to the PCB. It featured a 5-volt power pin, a ground pin, and a data pin. During the assembly

of the board however, the I/O pin connected to the header was damaged or incorrectly connected which prevented receiving data from the IR receiver. To fix this problem, the data pin of the IR receiver was connected to one of the unused data pins on the LCD display header, which allowed the receiver to work properly.

Also located on the board was a USB header that was originally to be used for powering the board through a USB port on the single board computer. However, the surface mounted USB port that we ordered was of low quality and had extremely small legs that would not allow us to properly solder the device to the board. Due to the COVID-19 virus, shipping from most suppliers was massively delayed by the time we encountered this, and we were unable to order new USB ports. Instead, to overcome this challenge, we used the Spy-Bi-Wire header's 5-volt and ground pins to power the board.

The power for the MCU was handled by placing an AMS-1117 voltage regulator on the board that took the 5-volt supply and stepped it down to 3.3-volts. This was within the operating voltage range for the MSP430FR6922. The IR sensor and LCD display were directly powered off the 5-volt power line coming from the UDOO board. The board also featured the 7 status LEDs and 3 push buttons mentioned earlier.

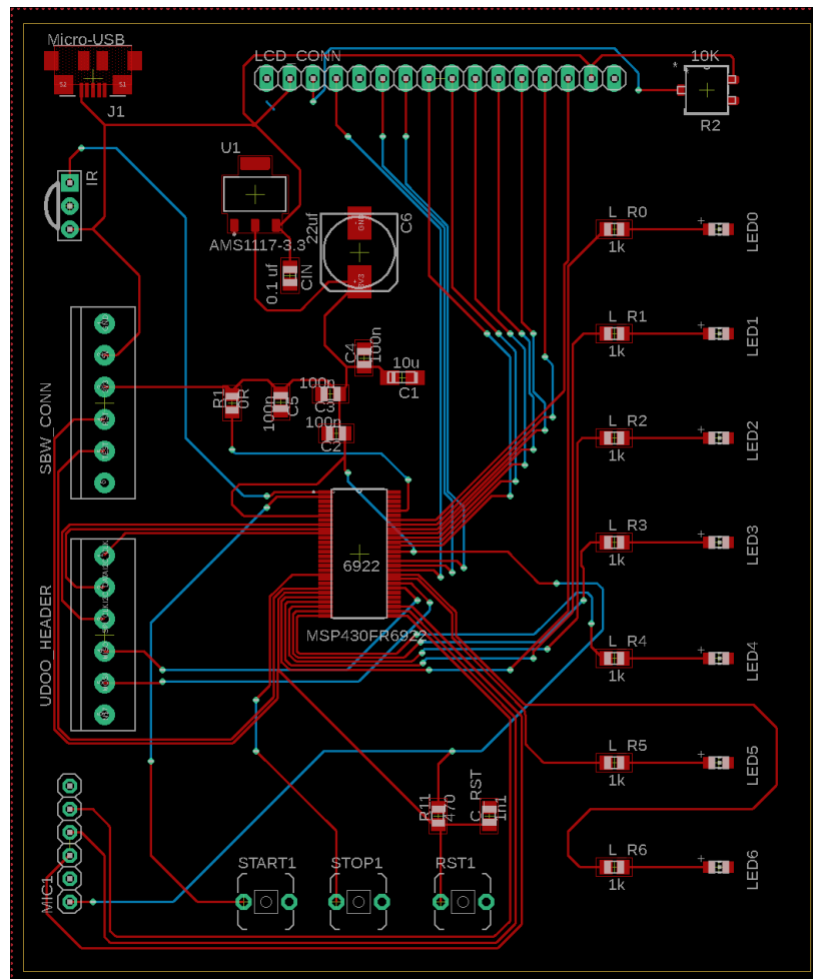


Figure 70: Final printed circuit board layout.

7.2.3 Final Hardware Block Diagram

Figure 71 shows the overall block diagram of our system. The power source is a 12-V, 3-A DC power adapter that plugs into the UDOO x86 II Advanced Plus board. Then, from the single board computer 5 V are delivered to the PCB. This 5-volt line directly powers both the LCD display and the IR sensor. Furthermore, the 5-volt power was run into a 3.3-volt AMS 1117 regulator that provides the MCU unit located on the PCB with 3.3 volts power.

In terms of peripherals on the PCB side, we have a 16x2 LCD, status LEDs, buttons, and an IR receiver connected to the I/O of the PCB. Connected to the UDOO board are the display and the microphone. The display also houses the speakers for the system, allowing for easy connection of both video and audio via an HDMI cable.

Finally, the PCB and the single board computer are connected via a 2-wire I2C connection that allows easy transfer of control signals between the 2 boards.

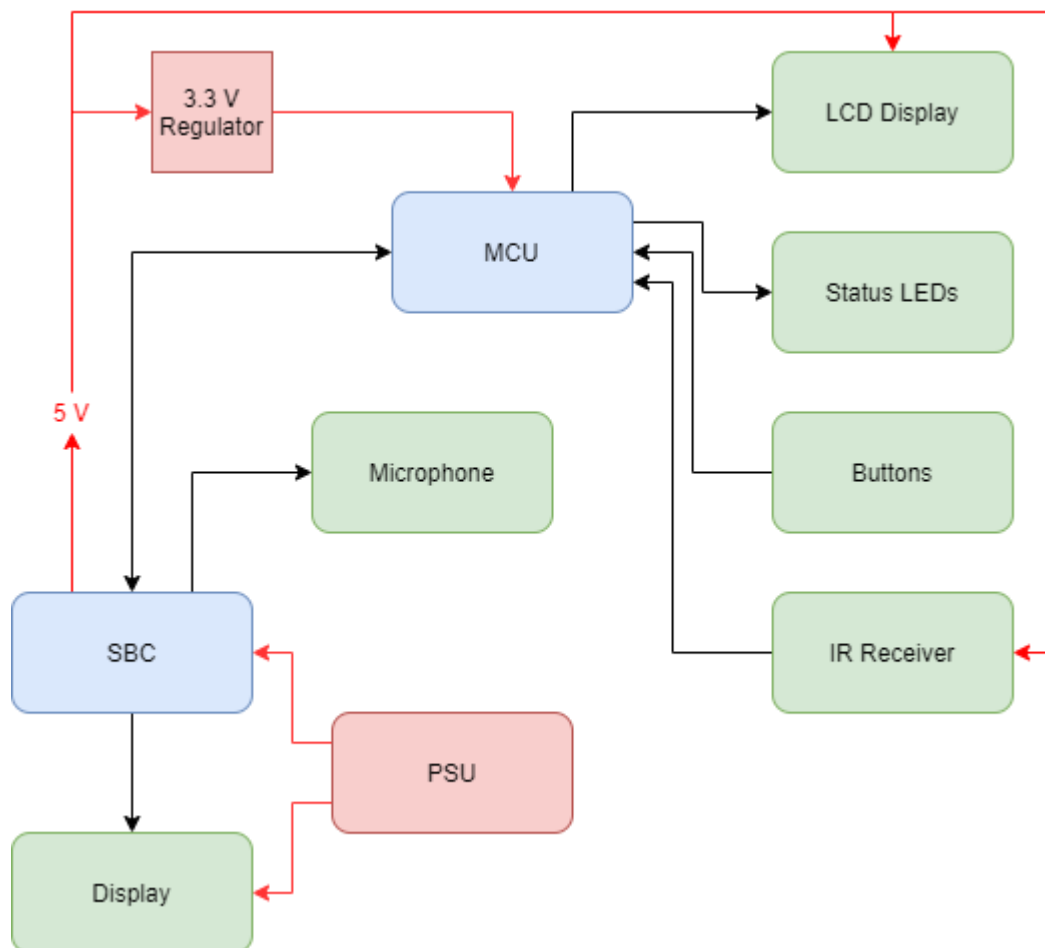


Figure 71: Final Hardware Block Diagram.

7.2.4 Final Bill of Materials

This section includes the final bill of materials for the final design of this project, including unit cost and the total costs for the device.

The UDOO x86 II Advanced Plus serves as our main processor unit, while the PCB/MCU will function as auxiliary processors for user input and alert outputs to the user. The UDOO board will be the sole storage unit and will contain the ASLBoT Unity software as well as the program files necessary for executing the calls for the OpenNMT and Watson Speech-to-Text API. The UDOO will also handle A/V output to the display via HDMI.

The PCB which contains the microcontroller will act as a control board for the user from which the user may interact with the ASLBoT software. The power supply for the UDOO is a standard 12V 3A AC adapter which was purchased directly from the UDOO vendor website.

The PCB vendor for the final PCB was decided to be JLCPCB due to their reasonable costs per board and high-quality PCBs. Despite the current epidemic, these boards were able to be delivered in time such that the board could be implemented in the project. The minor components such as resistors and capacitors are not listed in the table but were factored into the overall cost of the PCB.

Table 39. Final Bill of Materials.

Part	Supplier	Unit Cost	Quantity	Cost
UDOO x86 II Advanced Plus	UDOO	\$176.00	1	\$176.00
Power Supply	UDOO	\$8.90	1	\$8.90
PCB/MCU	JLCPCB	\$19.07	1	\$19.07
IR Receiver	Adafruit	\$1.95	1	\$1.95
Display	Amazon	\$59.99	1	---
Snowball Microphone	Blue Designs	\$49.99	1	---
LCD	Amazon	\$5.99	1	---
IR Remote	Adafruit	\$4.95	1	--
Total		\$326.84		\$205.42

8. Operator's Manual

User's Manual for ASLBoT Version 1.01

This device is intended for translation of spoken English into American Sign Language. This device is intended to be a concept for an out-of-the-box solution for automated sign language interpretation.

Note: This device cannot be used for translation from languages other than English and cannot be used to translate into sign languages other than American Sign Language.

Overview of User Interface:

- Amane Kisora-chan (© SapphiArt Co., Ltd.) has been included as the translation model for this device. While using the program, Amane will act as your sign language interpreter.
- Located on the circuit board are three buttons labeled “START”, “STOP”, and “RESET”.
 - The “START” button is used to begin recording your voice while using the ASLBoT software.
 - The “STOP” button is used to indicate to the software that you have finished speaking and are ready to have your voice translated.
 - The “RESET” button is used to reset the circuit board and the LCD if any issues arise.
 - LEDs are located on this board to indicate if the game is currently recording.
- An IR remote has been included with this device. You can wake up the device by pressing any button on the IR remote. Pressing this button will automatically wake up the device and sign you in.
- An LCD has also been included with this device. When you press the START button on the circuit board, the LCD will display “Recording...”

Contents:

1x UDOO x86 II Advanced Plus single-board computer with pre-installed ASLBoT software

1x Barrel Jack AC-DC Power Adapter (12V, 3A)

1x circuit board with LEDs and buttons

1x IR Remote with 1x CR2025 battery (included)

1x LCD

1x HDMI cable

1x USB-enabled microphone

1x plastic enclosure (sold separately)

Notes about these contents:

- While a microphone has been included with this device, any USB-enabled microphone that can pick up human speech can be used with this device.
- Likewise, with the HDMI cable and power adapter, any working HDMI cable and power adapter may be used with this device. Note, however, that the power adapter must be rated for 12V at 3A.

Operating the Device:

This device is pre-installed with a copy of Windows 10 and a copy of the ASLBoT software.

Turning on the device:

- Insert the power adapter into any 120 VAC wall socket, then insert the barrel jack connector into the power socket of the UDOO.
Note: plugging in the power supply to the UDOO should automatically turn on the device.
- Insert one end of the HDMI cable into the UDOO, then insert the other end of the cable into any HDMI-enabled display (not included).
- Insert any USB-enabled mouse and keyboard (not included) using the USB 3.0 ports located on the UDOO.
Note: the UDOO is not Bluetooth-enabled and will not work with adapter-less Bluetooth devices.
- Insert the USB-enabled microphone into one of the remaining USB 3.0 ports located on the UDOO.

Running the ASLBoT software:

- Log into the default account ASLBoT. By default, the password is blank.
- Find the ASLBoT v1.01 software folder located on the Desktop. Double-click to open the folder.
- Check that all necessary files have been installed:
 - Open the *ASL* folder.
 - Check that the files *nmt_model.pt* and *vocab.txt* are in this folder.
- Run the executable file *ASLBoT.exe*. This will start the ASLBoT software.

Using the ASLBoT software:

- While the program is running, press the START button on the circuit board.
- Once the screen says “Recording...”, you may begin speaking into the microphone.
- Once you have finished speaking, press the STOP button on the circuit board.
- If an audio file has been created successfully, Amane will show you that she is thinking about how to translate that sentence.
- Once she is done thinking, the sentence you said, along with Amane’s predicted sign language translation, will be shown on the display. Amane will also sign her predicted translation.
- The program will also automatically replay the sound file that it has recorded. Please mute your external display if you would like to disable this feature.
- Once Amane is finished with her interpreting, the display will clear and Amane will start waiting again.
- At this point, you have the option of pressing the START button again to begin recording another sentence.
- Otherwise, if you would like to quit the game, press the Escape button on the keyboard. Amane will wave goodbye, and the display will show “Goodbye! Shutting down...”, and the game will automatically quit itself.

Warning messages when using the ASLBoT software:

- Warnings are shown in the top-left corner of the screen while the game is running.
- If you attempt to quit the game while Amane is still interpreting, a warning message will be shown stating “Animations are in progress...”
- If you press the STOP button but have not pressed the START button and began recording, a warning message will be shown stating “Microphone is not on...”

Troubleshooting:

The device is not turning on:

- Check that all the parts have been connected properly. If not, unplug the power supply and try again.
- Check that the correct power supply is being used. Note that using a power supply that is not rated for the UDOO may damage the device.

The device is overheating while the program is running:

- After prolonged use of the ASLBoT software, this is normal. However, if this is a common occurrence, you may need to install a fan on the UDOO. The recommended fan to use on the UDOO is the one from the manufacturer, found here:

<https://shop.udoo.org/cpu-fan-for-udoo-x86-heatsink.html>

The IR remote is not waking up the computer properly:

- If pressing the remote results in waking up the device but not signing in, you may have set a password on the device. Remove the password to allow the IR remote to work again.
- If pressing the remote does not wake the device at all, you may need to change the batteries on the remote.

When using the game, Amane is spelling everything letter-by-letter:

- You may not have the *vocab.txt* file, or it may have been placed in the wrong location. Check that there is a valid text file named *vocab.txt* in the *ASL* folder.

When speaking into the microphone, nothing is being shown:

- Check that the “Start” button has been pressed and the word “Recording...” is shown on the screen.
- Check that a valid USB-enabled microphone has been plugged into a USB port.

Warnings:

Do NOT attempt to install another operating system onto the UDOO computer, as this may result in loss of the ASLBoT software.

Do NOT attempt to modify the Unity game files, as this may result in malfunction of the ASLBoT software.

Do NOT attempt to use the UDOO x86 II Advanced Plus computer other than for running the ASLBoT software; doing so may result in malfunction of the ASLBoT software.

Do NOT attempt to install, move, or remove any files except where specified, as this may harm the device.

Do NOT attempt to connect any external USB devices other than a mouse, keyboard, or microphone. Doing so may harm the device.

Do NOT attempt to create a password for signing into the device. Doing so will prevent the IR remote from waking up the device properly.

Disclaimers:

This device has been produced as a proof-of-concept model and has not been approved for use in legal, medical, or emergency situations. As such, use of this device is not recommended for use in legal, medical, or emergency situations. The provision of this device and its software is “as-is” and does not include any guarantee including suitability for a specific purpose of use and non-infringement of the rights of a third party.

SDF19S20 Group 14 is not responsible for injury or loss resulting from misuse of this device, including alterations to any files in the ASLBoT software folder.

As this device is conceptual, the validity and/or accuracy of translations produced while using default settings of this device cannot be guaranteed.

OpenNMT is an open-source neural machine translation software and is licensed under the MIT license. Your use of this device implies consent with their terms of use. SDF19S20 Group 14 is in no way sponsored or affiliated with the producers of the OpenNMT software.

SDF19S20 Group 14 is in no way sponsored or affiliated with SECO SPA, producers of the UDOO family of single-board computers.

Inclusion of the *Amane Kisora-chan* Unity model in this device is in compliance with the license set forth by SapphiArt Co., Ltd., and your use of this device implies consent with their Terms of Use.

Contact:

Senior Design Fall 2019-Spring 2020, Group 14 (SDF19S20 Group 14)

Affiliated with the University of Central Florida

4000 Central Florida Boulevard

Orlando, FL 32816

Contact the ASLBoT Manufacturers:

For general questions regarding ASLBoT:

- Gustavo Camero: gcamero@knights.ucf.edu

For questions regarding OpenNMT:

- Luis Hurtado: luis.hurtado@knights.ucf.edu

For questions regarding the Unity Engine software:

- Michael Loyd: michael.loyd@knights.ucf.edu

For questions regarding hardware:

- Jared Spinks: spinks.jared@knights.ucf.edu

9. Administrative Content

In this section, we will show the time management and budget constraints that the team had to overcome during the trajectory of Senior Design I and II. In addition, a milestone section will be discussed for both Senior Design I and II. The deadlines and deliverables will be explained in a projected manner. The budget and finance of the team will also be discussed in a projected manner.

9.1 Budget and Finance

In this section, we will discuss the initial budget cost after completing Senior Design I and the final budget cost after completing Senior Design II. The changes made on Senior Design II will be reflected on the final budget cost. Finally, throughout this section, we want to stress that we were self-financed and received no support from a sponsor.

9.1.1 Initial Budget Cost

Our project so far possesses no external sponsorship. The component selection with its pertaining price range is based on the extensive research done in Senior Design I so far. The price ranges are rough estimates from what we have encountered online. The component with their pricing range as shown in Table 39. We are planning to have a budget of \$1000.

Table 40. Original Business Budget Approximations for each Component.

Part	Supplier	Unit Cost	Quantity	Cost
NVIDIA JETSON	NVIDIA	\$95.96	1	\$95.96
Altera Cyclone IV	Xilinx	\$8.90	1	\$8.90
MSP430FR6989	Mouser	\$7.86	1	\$7.86
MEMS Microphone SPH0645LM4H	Adafruit	\$6.95	1	\$6.95
IR Receiver TSOP38238	Adafruit	\$1.95	1	\$1.95
WiFi Module	Amazon	\$15.00	1	\$15.00
Display Monitor	Amazon	\$59.99	1	\$59.99
PCB	JLPCB	\$50	1	\$50
Miscellaneous	--	--	--	\$200
Total	Supplier	\$246.61		\$446.61

9.1.2 Final Budget Cost

Our final budget cost is based on the changes made throughout Senior Design II. We noticed that the original design could not be met given the timeframe of Senior Design II. Therefore, we changed our system from an GPU-FPGA based system to the SBC-MCU based system that we were able to develop during Senior Design II. This heavily raised the cost of the overall product. In the end, we could not get a sponsor for our project, so we ended up being self-funded. Table 40 shows our overall self-funding for this project.

Table 41. Final Budget Cost of the ASLBoT system.

Part	Supplier	Unit Cost	Quantity	Cost
UDOO x86 Advanced Plus	UDOO	\$176.00	1	\$176.00
Power Supply	UDOO	\$8.90	1	\$8.90
PCB/MCU	JLPCB	\$19.07	1	\$19.07
IR Receiver	Adafruit	\$1.95	1	\$1.95
Display	Amazon	\$59.99	1	---
Snowball Microphone	Blue Designs	\$49.99	1	---
LCD	Amazon	\$5.99	1	---
IR Remote	Adafruit	\$4.95	1	--
Miscellaneous	---	---	---	\$132.93
Total		\$326.84		\$338.35

9.2 Project Milestones

In this section, we demonstrate the milestones requirement had to meet in order to complete the project. We broke down the milestones into two sections. One for Senior Design I in which selection of the project, researching and testing was done. The other for Senior Design II in which the integration will be carried out. For Table 21, we assigned the duration of tasks based on the Important Deadline section. For Table 22, we are just giving a rough estimate and projection on how the team will perform at integrating the project together. In addition, we are also taking into consideration the feasibility of running into roadblocks in the integration phase of the project for Senior Design II. We took this consideration also in Senior Design I. It can be noted that a lot of Senior Design I was spent in training the neural machine translation model since we needed to use an accurate model

for Senior Design II. Moreover, we have to train the NMT with a robust data set such that it would understand the ASL-Gloss translation we want to implement.

Table 42. Senior Design I – Milestones.

Process Description	Duration	Dates
Brainstorming & Project Selection	2 weeks	Aug. 30 - Sept. 14
Initial Research / Divide & Conquer	1 week	Sept. 14 - Sept. 20
Continuation of Research	3 weeks	Sept. 20 - Oct. 12
Possible Hardware & Software Selection, Purchase Demos for Preliminary Testing	3 weeks	Sept. 20 - Oct. 12
Cover page/Executive Summary/Technical Contents/Training Neural Machine Translation Model	4 weeks	Oct. 12 - Oct. Nov. 1
Administrative Content/Project Summary/Conclusion/Appendices/ Finalize Training	2 weeks	Nov. 1 - Nov. 15
Final Document Revision + Initial Prototyping	2 - 3 weeks	Nov. 15 - Dec 2

Table 43. Senior Design II – Milestones.

Process Description	Duration	Dates
Complete Prototype/CDR Preparation	4 weeks	Jan. - Feb.
Prototype Testing and Debugging/Mid-Term Demo Preparation/ Work on Final Report	4 weeks	Feb. - March
Product Finalization/ Final Presentation Preparation/ Final Report Revision	4 weeks	March - April

9.2.1 Important Deadlines

Here are import deadlines that we are required to meet during Senior Design I and Senior Design II shown in Table 23 and Table 24, respectively.

Table 44: Senior Design I Deadlines.

Document	Deadline
Initial Divide & Conquer	September 20 th
Updated Divide & Conquer	October 4 th
60-page Draft Senior Design I	November 1 st
100-page Senior Design I	November 15 th
Final Senior Design I Report	December 2 nd

Table 45: Senior Design II Deadlines.

Document	Deadline
Critical Design Review Presentation	February 4 th
Mid-Term Demo	March 19 th
Committee Form	April 3 rd
Final Presentation Video	April 14 th
Final Demo Video	April 14 th
Conference Paper	April 14 th
Final Senior Design II Report	April 21 st

9.3 Work Distribution

Here are the roles of each member of Group 14 shown in Table 25. Everyone in the group had primary and secondary roles. Primary roles were in charge on researching on topics pertaining their role and implementation. Secondary roles were in charge of helping the primary role.

Table 46: Final Work Distribution.

Member	Embedded	Machine Translation	Graphics Rendering	Integration
Gustavo Camero	Secondary	Secondary	Secondary	Primary
Luis Hurtado	Secondary	Primary		Secondary
Michael Loyd		Secondary	Primary	Secondary
Jared Spinks	Primary			Secondary

Appendix A: Copyright Permissions

We have been granted from SapphiArt Co., Ltd. for our usage of their Amane Kisora model in our Unity-based program. This model has been used in this project under a fair-use license. The link to the download for the Unity model is given below, and the link to the developer's website is <http://www.sapphiart.co.jp/>.

<https://assetstore.unity.com/packages/3d/characters/amane-kisora-chan-free-ver-70581>

We have also asked for permission from the Harvard NLP group and SYSTRAN for our usage and manipulation of their open-source OpenNMT program. The OpenNMT information website can be found at <http://opennmt.net/>, and the open-source files can be found on their GitHub here: <https://github.com/OpenNMT>.

We have been granted permission from the Watson IBM group to use their Speech-to-Text API. This permission has been granted in the form of a limited-use license. The description of their speech-to-text service can be found in the link below:

<https://www.ibm.com/cloud/watson-speech-to-text>

Appendix B: NMT Training Corpora

English Training Corpus, 11/8/19

hello	Charles is sorry	what did the teacher do
how are you ?	I do understand	today ?
do you need help ?	I feel fine	teacher
I am fine	I need help	do you know what time it
I do not understand	not	is ?
I don't understand	sorry	know
I'm sorry	Charles	knows
I am sorry	fine	have you done the
what is your name ?	time	homework ?
what happened ?	understand	I have done the homework
my name is Charles	name	I have not done the
what time is it ?	need	homework yet
can I help you ?	like	yet
can you help me ?	looked	do you know what grade
I	did you look at the	you got ?
are	homework ?	your
do	at	did
am	have	does
were	homework	where can I find the
it	what grade did you get ?	classroom ?
a	grade	of
hello <u>charles</u>	get	the classroom is there
how are you Charles ?	when is the test ?	can I go to the bathroom ?
I am not fine	what is the test on ?	may I go to the bathroom ?
I am not sorry	test	the bathroom is that way.
what happened to	did you study for the test ?	the bathroom is closed for
Charles ?	study	today.
what time is it Charles ?	when is it due ?	where is my classroom ?

ASL-Gloss Training Corpus, 11/8/19

HELLO
HOW _ YOU ?
_ YOU NEED HELP ?
ME _ FINE
ME _ NOT UNDERSTAND
ME NOT UNDERSTAND
ME SORRY
ME SORRY
WHAT _ YOU NAME ?
WHAT HAPPEN ?
MY NAME _ CHARLES
WHAT TIME _ _ ?
CAN _ HELP YOU ?
CAN YOU HELP _ ?
HELLO CHARLES
HOW _ YOU CHARLES ?
ME _ NOT FINE
ME _ NOT SORRY
WHAT HAPPEN CHARLES ?
WHAT TIME _ _ CHARLES ?
CHARLES _ SORRY
ME _ UNDERSTAND
ME FEEL FINE
ME NEED HELP
NOT
SORRY
CHARLES
FINE
TIME
UNDERSTAND
NAME
NEED
LIKE

LOOKED
BEFORE _ LOOK _ _ HOMEWORK ?
HOMEWORK
WHAT GRADE BEFORE _ GET?
GRADE
GET
WHEN _ _ TEST ?
WHAT _ _ TEST ABOUT ?
TEST
BEFORE _ STUDY _ _ TEST ?
STUDY
WHEN _ _ DUE ?
WHAT BEFORE _ TEACHER DO
TODAY ?
TEACHER
_ _ KNOW WHAT TIME _ _ ?
KNOW
KNOW
BEFORE _ DONE _ HOMEWORK ?
_ BEFORE DONE _ HOMEWORK
_ BEFORE NOT DONE _ HOMEWORK
YET
YET
_ _ KNOW WHAT GRADE _ GOT ?
YOU
WHERE CAN _ FIND CLASSROOM ?

_ CLASSROOM _ THERE
CAN _ GO _ _ BATHROOM ?
CAN _ GO _ _ BATHROOM ?
_ BATHROOM _ THERE WAY
_ BATHROOM _ CLOSE _ TODAY
WHERE _ MY CLASSROOM ?

English Validation Corpus, 11/8/19

hello how are you ?
do you need help Charles ?
Charles is fine
does Charles need help ?
Charles does not understand
Charles is not sorry
what time is it Charles ?
can Charles help you ?
can Charles help me ?
I do not feel fine
I do not know what time it is
Charles knows what time it is
have you looked at the homework yet ?
do you know what grade you got ?
do you know what grade you got yet ?
what is your grade ?
do you know when the test is ?
did the teacher grade the test?
did the teacher do the test ?
when did you study ?
have you studied ?
did you do the homework ?
is the homework due ?
is the homework due today ?
what is the name of the teacher ?
the name of the teacher is Charles
hello
how are you ?
do you need help ?
I am fine
I do not understand
I don't understand
I'm sorry
I am sorry
what is your name ?
what happened ?
my name is Charles
what time is it ?
can I help you ?
can you help me ?
I
are
do
am
were
it
a
hello Charles
how are you Charles ?
I am not fine
I am not sorry
what happened to Charles ?
what time is it Charles ?
Charles is sorry
I do understand
I feel fine
I need help
not
sorry
Charles
fine
time
understand
name
need
like
looked
did you look at the homework ?
at
have
homework
what grade did you get ?
grade
get
when is the test ?
what is the test on ?
test
have you studied for the test ?
study
when is it due ?
what did the teacher do today ?
teacher
do you know what time it is ?
know
knows
have you done the homework ?
I have done the homework
I have not done the homework yet
yet
do you know what grade you got ?
your
did
does
where can I find the classroom ?
of
where is Charles ?
where can I find the bathroom ?
the classroom is closed for today
I did not find the bathroom
I did not find the classroom yet
where can I find my grade ?
when is the homework due ?
I'm fine
I'm Charles

ASL-Gloss Training Corpus, 11/8/19

HELLO HOW _ YOU ?
_ YOU NEED HELP ?
CHARLES _ FINE
_ CHARLES NEED HELP ?
CHARLES _ NOT UNDERSTAND
CHARLES _ NOT SORRY
WHAT TIME _ CHARLES ?
CAN CHARLES HELP YOU ?
CAN CHARLES HELP _ ?
ME _ NOT FEEL FINE
ME _ NOT KNOW WHAT TIME _ _
CHARLES KNOW WHAT TIME _ _ _
HAVE YOU LOOKED _ _ HOMEWORK YET ?
_ YOU KNOW WHAT GRADE _ GOT ?
_ YOU KNOW WHAT GRADE _ GOT YET ?
WHAT _ YOUR GRADE ?
_ YOU KNOW WHEN TEST _ ?
BEFORE _ TEACHER GRADE _ TEST ?
BEFORE _ TEACHER _ _ TEST ?
WHEN BEFORE YOU STUDY ?
BEFORE YOU STUDY ?
BEFORE YOU _ _ HOMEWORK ?
_ _ HOMEWORK DUE ?
_ _ HOMEWORK DUE TODAY ?
WHAT _ NAME _ TEACHER ?
_ NAME _ _ TEACHER _ CHARLES
HELLO
HOW _ YOU ?
_ YOU NEED HELP ?
ME _ FINE
ME _ NOT UNDERSTAND
ME NOT UNDERSTAND
ME SORRY
ME SORRY
WHAT _ YOU NAME ?
WHAT HAPPEN ?
MY NAME _ CHARLES
WHAT TIME _ _ ?
CAN _ HELP YOU ?
CAN YOU HELP _ ?
_
_
_
_
_
_
HELLO CHARLES
HOW _ YOU CHARLES ?
ME _ NOT FINE
ME _ NOT SORRY
WHAT HAPPEN CHARLES ?
WHAT TIME _ _ CHARLES ?

CHARLES _ SORRY
ME _ UNDERSTAND
ME FEEL FINE
ME NEED HELP
NOT
SORRY
CHARLES
FINE
TIME
UNDERSTAND
NAME
NEED
LIKE
LOOKED
BEFORE _ LOOK _ _ HOMEWORK ?
_
_
HOMEWORK
WHAT GRADE BEFORE _ GET ?
GRADE
GET
WHEN _ _ TEST ?
WHAT _ _ TEST ABOUT ?
TEST
BEFORE _ STUDY _ _ TEST ?
STUDY
WHEN _ _ DUE ?
WHAT BEFORE _ TEACHER DO TODAY ?
TEACHER
_ _ KNOW WHAT TIME _ _ ?
KNOW
KNOW
BEFORE _ DONE _ HOMEWORK ?
_ BEFORE DONE _ HOMEWORK
_ BEFORE NOT DONE _ HOMEWORK YET
YET
_ _ KNOW WHAT GRADE _ GOT ?
YOU
_
_
_
WHERE CAN _ FIND _ CLASSROOM ?
_
WHERE _ CHARLES ?
WHERE CAN _ FIND _ BATHROOM ?
_ CLASSROOM _ CLOSE _ TODAY
_ BEFORE NOT FIND _ BATHROOM
_ BEFORE NOT FIND _ CLASSROOM YET
WHERE CAN _ FIND MY GRADE ?
WHEN _ _ HOMEWORK DUE ?
ME FINE
ME CHARLES

Appendix C: Acknowledgements

We would like to give credit to Jolanta Lapiak and her online ASL database *Handspeak* (handspeak.com). She has taught us not only how the signs are done, but she has also included sample sentences and their ASL-Gloss equivalents.

We would also like to give credit to Dr. William Vicars with his Lifeprint.com website and the European Sign Language Center with their *Spread the Sign* (<https://www.spreadthesign.com/>) ASL databases. Without their help, we would not have been able to produce many of the ASL-gloss sentences necessary for our NMT corpora.

We would like to thank our advising professor, Dr. Chung-Yong Chan, and our senior design coordinator, Dr. Samuel Richie, for their assistance and guidance throughout this project.

Appendix D: Works Cited

- [1] “6 Big Industries that can benefit from language translation services.” Pangeanic. Boston, MA. Feb. 24, 2015. Accessed Sept. 18, 2019. https://www.pangeanic.com/knowledge_center/6-big-industries-can-benefit-language-translation-services/#
- [2] Leila Miller. “I was panicked”: Deaf patients struggle to get interpreters in medical emergencies.” STAT. Boston, MA. May 22, 2017. Accessed Sept. 18, 2019. <https://www.statnews.com/2017/05/22/deaf-patients-interpreters/>
- [3] U.S. Department of Justice. “Communicating with People Who Are Deaf or Hard of Hearing in Hospital Settings.” Civil Rights Division: Disability Rights Section. Washington, DC. August 11, 2005. Accessed Sept. 18, 2019. <https://www.ada.gov/hospcombr.htm>
- [8] “Compare Translate Features for Each Language - Google Translate,” Google. N.d. [Online]. [Accessed Oct. 11, 2019]. <https://translate.google.com/intl/en/about/languages/>
- [9] M. Schuster, M. Johnson, and N. Thorat, “Zero-Shot Translation with Google’s Multilingual Neural Machine Translation System,” Google AI Blog, Nov. 22, 2016. [Online]. [Accessed Oct. 11, 2019]. <https://ai.googleblog.com/2016/11/zero-shot-translation-with-googles.html>
- [10] N. McGuire, “How Accurate is Google Translate in 2018?,” Glenview, IL: Argo Translation, July 26, 2018. [Online]. [Accessed Oct. 11, 2019]. <https://www.argotrans.com/blog/accurate-google-translate-2018/>
- [11] “Amazon Translate - Neural Machine Translation - AWS,” Amazon Web Service, Inc, 2019. [Online]. [Accessed Oct. 11, 2019]. <https://aws.amazon.com/translate/>
- [12] “Amazon Translate - Developer Guide,” Amazon Web Services, Inc, 2019. [Online]. [Accessed Oct. 11, 2019]. <https://docs.aws.amazon.com/translate/latest/dg/translate-dg.pdf>
- [13] A. Mlievski, “Amazon Tops Overall Quarterly Survey by One Hour Translation of Neural Machine Translation Engines,” Washington, D.C: PR Newswire Association LLC, Sept. 26, 2018. [Online]. [Accessed Oct. 18, 2019]. <https://www.prnewswire.com/news-releases/amazon-tops-overall-quarterly-survey-by-one-hour-translation-of-neural-machine-translation-engines-300719593.html>
- [14] “Video Remote Interpreting (VRI),” Stratus Video, 2018. [Online]. [Accessed Oct. 18, 2019]. <https://www.stratusvideo.com/stratus-video/>
- [15] Kwamikagami, “Brief Comparison of ASL Writing Systems,” Wikimedia Commons: Feb. 17, 2014. [Online]. [Accessed Oct. 9, 2019]. https://commons.wikimedia.org/wiki/File:Brief_Comparison_of_ASL_Writing_Systems.jpg
- [16] aslfont, “Symbol Font for ASL.” aslfont GitHub Repository. March 29, 2013. [Online]. [Accessed Sept. 28, 2019.] <https://aslfont.github.io/Symbol-Font-For-ASL/ways-to-write.html>
- [17] Slevinski, “SignWriting-render,” Wikimedia Commons: Apr. 25, 2017. [Online]. [Accessed Oct. 9, 2019]. <https://commons.wikimedia.org/wiki/File:SignWriting-render.svg>

- [18] J.A. Hochgesang, "Introduction to Stokoe Notation," Gallaudet University Linguistics Dept. Fall 2007. [Online]. [Accessed Sept. 23, 2019]. <https://lingdept.files.wordpress.com/2015/08/quickguidestokoenotation-pages.pdf>
- [19] W. Vicars. "Gloss." American Sign Language University. [Online]. [Accessed Oct. 5, 2019.] <https://lifeprint.com/asl101/topics/gloss.htm>
- [20] M. Gales and S. Young, "The Application of Hidden Markov Models in Speech Recognition," *J. Foundations and Trends in Signal Processing*, vol. 1, no. 3, Jan. 2008. pp. 195-304. [Online]. [Accessed Oct. 11, 2019]. https://mi.eng.cam.ac.uk/~mjfg/mjfg_NOW.pdf
- [21] L.R. Rabiner and B.H. Juang, "Hidden Markov Models for Speech Recognition – Strengths and Limitations," in *Speech, Recognition, and Understanding: Recent Advances, Trends, and Applications*, Berlin, Germany: Springer Heidelberg, 1992. pp. 3-29. [Online]. [Accessed Oct. 11, 2019]. https://link.springer.com/chapter/10.1007/978-3-642-76626-8_1
- [22] R. Rivera, "Strengths and weaknesses of hidden Markov models," UCSC Bioinformatics, Univ. California Santa Cruz, Aug. 22, 1996. [Online]. [Accessed Oct. 11, 2019]. https://compbio.soe.ucsc.edu/html_format_papers/tr-94-24/node11.html
- [23] C.B. Kare and V.S. Navale. "Speech recognition by Dynamic Time Warping," *IOSR J. Electronics and Comm. Engineering*, NCIEST, pp. 12-16. [Online]. [Accessed Oct. 11, 2019]. <http://www.iosrjournals.org/iosr-jece/papers/NCIEST/Volume%202/3.%2012-16.pdf>
- [24] S. Xihao and Y. Miyanaga, "Dynamic time warping for speech recognition with training part to reduce the computation," in *International Symposium on Signals, Circuits and Systems, ISSCS 2013, Iasi, Romania, July 11-12, 2013*. IEEE Xplore, Dec. 2, 2018. [Online]. [Accessed Oct. 11, 2019]. <https://ieeexplore.ieee.org/document/6651269>
- [25] A.B. Nassif et al., "Speech Recognition Using Deep Neural Networks: A Systematic Review," *Access IEEE*, vol. 7, pp. 26777-26787. IEEE Xplore, Feb. 1, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://ieeexplore.ieee.org/document/8632885/>
- [26] D. Wang, X. Wang, and S. Lv, "An Overview of End-to-End Automatic Speech Recognition," *Symmetry*, vol. 10. MDPI, August 7, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://www.mdpi.com/2073-8994/11/8/1018>
- [27] "CMUSphinx Open Source Speech Recognition," GitHub, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://cmusphinx.github.io/>
- [28] "Julius: Open-Source Large Vocabulary Continuous Speech Recognition Engine," GitHub, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://github.com/julius-speech/julius>
- [29] D. Povey et al, "The Kaldi Speech Recognition Toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Big Island, HI, December 11-15, 2011*. [Online]. [Accessed Oct. 31, 2019]. https://publications.idiap.ch/downloads/papers/2012/Povey_ASRU2011_2011.pdf
- [30] "DeepSpeech - TensorFlow implementation of Baidu's DeepSpeech architecture," GitHub, Oct. 2019. [Online]. [Accessed Oct. 31, 2019]. <https://github.com/mozilla/DeepSpeech>

- [31] “Wav2Letter++: A Fast Open-source Speech Recognition System,” in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17 2019*. IEEE Xplore, April 17, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://ieeexplore.ieee.org/document/8683535>
- [32] “Wav2letter++, the fastest open-source speech system,” Facebook Engineering, December 21, 2018. [Online]. [Accessed Oct. 31, 2019].
- [33] “DeepSpeech2 - OpenSeq2Seq 0.2 Documentation,” Nvidia GitHub, 2018. [Online]. [Accessed Oct. 31, 2019]. <https://nvidia.github.io/OpenSeq2Seq/html/speech-recognition/deepspeech2.html>
- [34] “HTK Speech Recognition Toolkit,” HTK, June 2016. [Online]. [Accessed Oct. 31, 2019]. <http://htk.eng.cam.ac.uk>
- [35] D. Rybach, S. Hahn, P. Lehnen, D. Nolden, M. Sundermeyer, Z. Tüske, S. Wiesler, R. Schlüter, and H. Ney: “RASR - The RWTH Aachen University Open Source Speech Recognition Toolkit,” in *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2011, Big Island, Hawaii, December 11-15, 2011*. [Online]. [Accessed Oct. 31, 2019]. <https://www-i6.informatik.rwth-aachen.de/rwth-asr/>
- [36] “Dialogflow,” Google, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://dialogflow.com/>
- [37] “Dragon Speech Recognition - Get More Done By Voice | Nuance,” Nuance Communications Inc, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://www.nuance.com/dragon.html>
- [38] “Cloud Speech-to-Text,” Google Cloud, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://cloud.google.com/speech-to-text/>
- [39] “What is speech-to-text?,” Microsoft Azure, July 4, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-to-text>
- [40] “Amazon Transcribe - Automatic Speech Recognition,” Amazon Web Services, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://aws.amazon.com/transcribe/>
<https://engineering.fb.com/ai-research/wav2letter/>
- [41] S. Sreelekha, “Statistical Vs Rule Based Machine Translation: A Case Study on Indian Language Perspective,” Dept. Comp. Sci. & Eng., Indian Institute of Technology, Bombay, India, Aug. 12, 2017. [Online]. [Accessed Oct. 11, 2019]. <https://arxiv.org/abs/1708.04559>
- [42] “What is Machine Translation? Rule Based Machine Translation vs. Statistical Machine Translation,” SYSTRAN, 2016. [Online]. [Accessed Oct. 11, 2019]. <http://www.systransoft.com/systran/translation-technology/what-is-machine-translation/>
- [43] “What is Rules Based Machine Translation (RBMT)?,” Omniscien Technologies, Asia Online Pte Ltd, n.d. [Online]. [Accessed Oct. 11, 2019]. <https://omniscien.com/rules-based-machine-translation/>
- [44] C. Dove, O. Loskutova, and R. de la Fuente, “What’s Your Pick: RbMT, SMT, or Hybrid?,” in *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas, AMTA 2012, San Diego, CA, Oct. 28-Nov. 1, 2012*. Machine Translation Archive, Nov. 7, 2012. [Online]. [Accessed Oct. 11, 2019]. <http://www.mt-archive.info/AMTA-2012-Dove.pdf>

- [45] “What is Statistical Machine Translation (SMT),” Omniscien Technologies, Asia Online Pte Ltd, n.d. [Online]. [Accessed Oct. 11, 2019]. <https://omniscien.com/?faqs=what-is-statistical-machine-translation-smt>
- [46] <removed>
- [47] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush, “OpenNMT: Open-Source Toolkit for Neural Machine Translation.” Proc. ACL, 2017. [Online]. [Accessed Sept. 22, 2019.] <https://arxiv.org/pdf/1701.02810.pdf>
- [48] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*. arXiv, Sept. 1, 2014. [Online]. [Accessed Sept. 30, 2019]. <https://arxiv.org/abs/1409.0473>
- [49] “Neural Machine Translation,” San Francisco, CA: Deep AI, Inc, May 17, 2019. [Online]. [Accessed Oct. 11, 2019]. <https://deeptai.org/machine-learning-glossary-and-terms/neural-machine-translation>
- [50] M.L. Forcada et al., “Apertium: a free/open-source platform for rule-based machine translation,” *J. Machine Translation*, vol. 25, no. 2, June 2011. pp. 127-144. Dordrecht, Netherlands: Springer Netherlands. [Online]. [Accessed Oct. 11, 2019]. https://www.jstor.org/stable/41487458?seq=1#metadata_info_tab_contents
- [51] P. Koehn et al., “Moses: Open Source Toolkit for Statistical Machine Translation,” in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, ACL 2007, Prague, Czech Republic, June 25-27, 2007*. ACL Anthology, 2007. pp. 177-180. [Online]. [Accessed Oct. 11, 2019]. <https://www.aclweb.org/anthology/P07-2045/>
- [52] D. Takamori, “Apache Joshua Home,” Atlassian: Confluence, Mar. 22, 2019. [Online]. [Accessed Oct. 11, 2019]. <https://cwiki.apache.org/confluence/display/JOSHUA/>
- [53] J. González and F. Casacuberta, “GREAT: open source software for statistical machine translation,” *J. Machine Translation*, vol. 25, no. 2, June 2011. Dordrecht, Netherlands: Springer Netherlands, Aug. 28, 2011. <https://link.springer.com/article/10.1007/s10590-011-9097-6>
- [54] C.H. Lee, “Speech Recognition and Production by Machines,” *International Encyclopedia of the Social & Behavioral Sciences*, ed. 2. pp. 259-263. [Online]. Elsevier: Science Direct, 2015. [Online]. [Accessed Oct. 31, 2019]. <https://www.sciencedirect.com/science/article/pii/B9780080970868520236>
- [55] P. Birkholz, “About Articulatory Speech Synthesis,” VocalTractLab, 2017. [Online]. [Accessed Oct. 31, 2019]. <http://www.vocaltractlab.de/index.php?page=background-articulatory-synthesis>
- [56] J.O. Smith III, “Formant Synthesis Models,” *Physical Audio Signal Processing*. Stanford Univ.: Center for Computer Research in Music and Acoustics, July 30, 2019. [Online]. [Accessed Oct. 31, 2019]. https://ccrma.stanford.edu/~jos/pasp/Formant_Synthesis_Models.html
- [57] J. Yamagishi, “An Introduction to HMM-Based Speech Synthesis,” Univ. of Edinburgh,

- Oct. 2006. [Online]. [Accessed Oct. 31, 2019].
<https://wiki.inf.ed.ac.uk/twiki/pub/CSTR/TrajectoryModelling/HTS-Introduction.pdf>
- [58] R.E. Remez, “Sine-wave speech,” Scholarpedia, 2008. [Online]. [Accessed Oct. 31, 2019].
http://www.scholarpedia.org/article/Sine-wave_speech
- [59] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A Generative Model for Raw Audio,” Cornell University: arXiv, September 12, 2016. [Online]. [Accessed Oct. 31, 2019]. <https://arxiv.org/abs/1609.03499>
- [60] U. Saxena, “Speech Synthesis Techniques using Deep Neural Networks,” Medium, October 21, 2017. [Online]. [Accessed Oct. 31, 2019].
<https://medium.com/@saxenauts/speech-synthesis-techniques-using-deep-neural-networks-38699e943861>
- [61] I. Steiner et al. “Mary Text-to-Speech,” Saarland Univ.: Language Technology Lab and Institute of Phonetics, 2018. [Online]. [Accessed Oct. 31, 2019].
<http://mary.dfki.de/documentation/overview.html>
- [62] “eSpeak: Speech Synthesizer,” SourceForge, 2017. [Online]. [Accessed Oct. 31, 2019]. <http://espeak.sourceforge.net>
- [63] P.A. Taylor, A. Black, and R. Caley, “The architecture of the festival speech synthesis system,” in the *Third ESCA Workshop in Speech Synthesis, SSW3 1998, Blue Mountains, Australia, November 26-29, 1998*. pp. 147-151. [Online]. [Accessed Oct. 31, 2019]. <http://www.cstr.ed.ac.uk/projects/festival/>
- [64] “Cloud Text-to-Speech,” Google Cloud, 2019. [Online]. [Accessed Oct. 31, 2019].
<https://cloud.google.com/text-to-speech/>
- [65] “Text to Speech API,” Microsoft Azure, 2019. [Online]. [Accessed Oct. 31, 2019].
<https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech/>
- [66] “Amazon Polly - Turn text into lifelike speech using deep learning,” Amazon Web Services, 2019. [Online]. [Accessed Oct. 31, 2019].
<https://aws.amazon.com/polly/>
- [67] “Speech to Text Conversion,” Vocapia Research, 2019. [Online]. [Accessed Nov. 15, 2019]. <https://www.vocapia.com/speech-to-text.html>
- [68] Z. Feng and H. Nian, “NiuTrans: A Statistical Machine Translation System,” NiuTrans, April 16, 2014. [Online]. [Accessed Nov. 15, 2019].
<http://www.niutrans.com/niutrans/NiuTrans.html>
- [69] “Watson Language Translator,” IBM, 2019. [Online]. [Accessed Nov. 15, 2019].
<https://www.ibm.com/watson/services/language-translator/>
- [70] B. Scott and A. Barreiro, “OpenLogos MT and the SAL Representation Language,” Repositorio Institucional de la Universidad de Alicante, Nov. 2009. [Online]. [Accessed Nov. 15, 2019] <http://hdl.handle.net/10045/12023>
- [71] “Translation API Documentation,” Google Cloud, 2019. [Online]. [Accessed Nov. 15, 2019]. <https://cloud.google.com/translate/docs/>
- [72] “Microsoft Translator Text API,” Microsoft, 2019. [Online]. [Accessed Nov. 15, 2019]. <https://www.microsoft.com/en-us/translator/business/translator-api/>
- [73] “Machine Translation,” Yandex LLC, 2019. [Online]. [Accessed Nov. 15, 2019].
<https://yandex.com/company/technologies/translation/>

- [74] “Welcome to SYSTRAN.io,” SYSTRAN, 2015. [Online]. [Accessed Nov. 15, 2019]. <https://platform.systran.net/index>
- [75] “GramTrans,” GrammarSoft ApS, 2019. <https://gramtrans.com>
- [76] “PROMT Neural Translation Server 19 is a multi - functional translation system,” PROMT LLC, 2019. [Online]. [Accessed Nov. 15, 2019]. https://www.promt.com/translation_software/corporate/promt-translation-server-neural/
- [77] “Babylon Translation Software and Dictionary Tool,” Babylon Software Ltd. [Online]. [Accessed Nov. 15, 2019]. https://www.babylon-software.com/translation_software
- [78] “IdiomaX Review,” business.com, May 20, 2019. [Online]. [Accessed Nov. 15, 2019]. <https://www.business.com/reviews/idiomax/>
- [79] F. Shaikh, “Why are GPUs necessary for training Deep Learning models?,” Analytics Vidhya, May 28, 2017. [Online]. [Accessed Oct. 10, 2019]. <https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/>
- [80] F. Fallahlalehzari, “FPGA vs GPU for Machine Learning Applications: Which one is better?,” Aldec Inc. [Online]. [Accessed Sept. 30, 2019]. <https://www.aldec.com/en/company/blog/167--fpgas-vs-gpus-for-machine-learning-applications-which-one-is-better>
- [81] S. Deoras, “GPU vs. FPGA: The Battle For AI Hardware Rages On,” Analytics India Magazine, Dec. 31, 2018. [Online]. [Accessed Sept. 30, 2019]. <https://analyticsindiamag.com/gpu-vs-fpga-the-battle-for-ai-hardware-rages-on/amp/>
- [82] D. Steinkraus, I. Buck, and P.Y. Simard, “Using GPUs for Machine Learning Algorithms,” in *Proceedings of the 2005 Eighth International Conference on Document Analysis and Recognition, ICDAR 2005, Seoul, South Korea, Aug. 31-Sept. 1*. pp. 1115-1120. IEEE Xplore, Jan. 16, 2006. [Online]. [Accessed Oct. 10, 2019]. <https://ieeexplore.ieee.org/document/1575717>
- [83] “Understanding How a Voltage Regulator Works,” Analog Devices, 2019. [Online]. [Accessed Dec. 4, 2019]. <https://www.analog.com/en/technical-articles/how-voltage-regulator-works.html>
- [84] “Linear and Switching Voltage Regulators – An Introduction,” Predictable Designs, 2019. [Online]. [Accessed Dec. 3, 2019]. <https://predictabledesigns.com/linear-and-switching-voltage-regulators-introduction/>
- [85] EEE4309 Lab Manual.
- [86] “What is Unreal Engine 4,” Epic Games, Inc, 2019. [Online]. [Accessed Oct. 28, 2019]. <https://www.unrealengine.com/en-US/>
- [87] “Unreal Engine | Features,” Epic Games, Inc, 2019. [Online]. [Accessed Oct. 28, 2019]. <https://www.unrealengine.com/en-US/features>
- [88] “Unreal Engine | EULA,” Epic Games, Inc, 2019. [Online]. [Accessed Oct. 28, 2019]. <https://www.unrealengine.com/en-US/eula>
- [89] D. Takahashi, “Unity Technologies CTO declares the company isn’t up for sale,” VentureBeat, Oct 16, 2014. [Online]. [Accessed Oct. 29, 2019].

- <https://venturebeat.com/2014/10/16/unity-cto-declares-the-company-isnt-up-for-sale/>
- [90] “Unity 2019: Performance by default, high-fidelity real-time graphics, and artist tools,” Unity Technologies, 2019. [Online]. [Accessed Oct. 28, 2019]. https://unity3d.com/unity?_ga=2.189971691.1333607966.1572374500-54223329.1572153467#editor
- [91] “Unity Software Additional Terms - Unity,” Unity Technologies, 2019. [Online]. [Accessed Oct. 29, 2019]. <https://unity3d.com/legal/terms-of-service/software>
- [92] “FAQ New Subscription - Unity,” Unity Technologies, 2019. [Online]. [Accessed Oct. 28, 2019]. <https://unity3d.com/unity/faq>
- [93] Daniru17, “HiguchiM,” MikuMikuDance Wiki, June 23, 2011. [Online]. [Accessed Oct. 28, 2019]. <https://mikumikudance.fandom.com/wiki/HiguchiM>
- [94] “Learn MikuMikuDance - The MMD Instructions you always wanted!,” LearnMMD.com. [Online]. [Accessed Oct. 29, 2019]. <https://learnmmd.com/>
- [95] “About - blender.org,” Blender, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://www.blender.org/about/>
- [96] “Requirements - blender.org,” Blender, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://www.blender.org/download/requirements/>
- [97] “License - blender.org,” Blender, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://www.blender.org/about/license>
- [98] “Godot Engine - Free and open source 2D and 3D game engine,” Godot, 2019. [Online]. [Accessed Nov. 14, 2019]. <https://godotengine.org/>
- [99] “What specs are good for a computer that you are planning to use godot with?,” Godot Engine Q&A, Feb. 4, 2018. [Online]. [Accessed Nov. 14, 2019]. <https://godotengine.org/qa/23206/what-specs-are-good-for-computer-that-planning-use-godot-with>
- [100] Adafruit Industries, “Mini USB Microphone” 3367 datasheet, Feb. 2017. [Online]. [Accessed Nov. 13, 2019]. https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3367_Web.pdf
- [101] Adafruit Industries, “I2S Output Digital Microphone,” SPH0645LM4H-B datasheet, 2015. [Online]. [Accessed Nov. 13, 2019]. <https://cdn-shop.adafruit.com/product-files/3421/i2S+Datasheet.PDF>
- [102] Knowles, “Top Port SiSonic Microphone,” SPW2430HR5H-B datasheet, Feb. 2014. [Online]. [Accessed Nov. 14, 2019]. <https://www.knowles.com/docs/default-source/model-downloads/spw2430hr5h-b.pdf>
- [103] Analog Devices, “Omnidirectional Microphone with Bottom Port and Analog Output,” ADMP401 datasheet, 2012. [Online]. [Accessed Nov. 14, 2019]. <https://www.analog.com/media/en/technical-documentation/obsolete-datasheets/ADMP401.pdf>
- [104] I.R. Titze, “Principles of Voice Production,” Prentice Hall, Mar. 1994.
- [105] “Buy the Latest Jetson Products | NVIDIA Developer,” NVIDIA Corporation, 2019. [Online]. [Accessed Nov. 8, 2019]. <https://www.nvidia.com/en-us/autonomous-machines/jetson-store/>
- [106] “10.1" Display & Audio 1280x800 IPS - HDMI/VGA/NTSC/PAL,” Adafruit, 2019. [Online]. [Accessed Oct. 31, 2019]. <https://www.adafruit.com/product/1694>

- [107] “Sceptre 20” 75Hz LED Monitor HDMI VGA Build-in Speakers, Brushed Black 2019 (E205-16003S),” Amazon.com, Jan. 2019. [Online]. [Accessed Oct. 31, 2019]. <https://www.amazon.com/Sceptre-Monitor-Speakers-Brushed-E205W-16003S/dp/B07MLGGHTP/>
- [108] “UPERFECT 12.3-inch Touch Display Portable Monitor 10 Point Capacitive Touchscreen 1600×1200 Resolution PC Display 4:3 60HZ Speakers VESA for Security Camera Laptop Phone Mac Raspberry Pi PS4 Nintendo,” Amazon.com, Jan. 2019. [Online]. [Accessed Oct. 31, 2019]. <https://www.amazon.com/UPERFECT-12-3-inch-Touchscreen1600%C3%971200-Resolution/dp/B07N4LB7FT>
- [109] “Simple RF T4 Receiver - 315MHz Toggle Type,” Adafruit, 2019. [Online]. [Accessed Nov. 14, 2019]. <https://www.adafruit.com/product/1097>
- [110] “Keyfob 2-button RF Remote Control - 315MHz,” Adafruit, 2019. [Online]. [Accessed Nov. 14, 2019]. <https://www.adafruit.com/product/1391>
- [111] “Intel Dual Band Wireless-AC 8265 Product Brief,” Intel Corporation, 2016. [Online]. [Accessed Dec. 4, 2019]. <https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/dual-band-wireless-ac-8265-brief.pdf>
- [112] K. Shaw, “The OSI model explained: How to understand (and remember) the 7 layer network model,” Network World, Oct. 22, 2018. [Online]. [Accessed Nov. 1, 2019]. <https://www.networkworld.com/article/3239677/the-osi-model-explained-how-to-understand-and-remember-the-7-layer-network-model.html>
- [113] G. Phillips, “The Most Common Wi-Fi Standards and Types Explained,” MakeUseOf, Mar. 13, 2019. [Online]. [Accessed Nov. 1, 2019]. <https://www.makeuseof.com/tag/understanding-common-wifi-standards-technology-explained/>
- [114] J.D. Allen, Ed., *The Unicode Standard: The Unicode Consortium*. Boston, MA: Pearson Education Inc, 2007.
- [115] “CUDA C Programming Guide,” NVIDIA, Oct. 2012. [Online]. [Accessed Nov. 15, 2019]. https://www3.nd.edu/~zxu2/acms60212-40212/CUDA_C_Programming_Guide.pdf
- [116] N. Botros, “HDL With Digital Design: VHDL and Verilog,” Dulles, VA: Mercury Learning and Information, 2015.
- [117] IEEE Standard for Verilog Hardware Description Language, IEEE Standard 1364, 2005.
- [118] System Verilog for Quality of Results (QoR), IEEE Standard 1800, 2008.
- [119] IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language, IEEE Standard 1800, 2017.
- [120] K. Papineni, S. Roukos, T. Ward, and W.J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002, Philadelphia, PA, July 2002*. pp. 311-318. ACL Anthology. [Online]. [Accessed Oct. 18, 2019]. <https://www.aclweb.org/anthology/P02-1040/>
- [121] “R/pred.score.R,” predictionnet, May 6, 2019. [Online]. [Accessed Oct. 18, 2019]. <https://rdrr.io/bioc/predictionnet/src/R/pred.score.R>
- [122] V. Nguyen, G. Klein, and SeisQ, “Metrics (Bleu, ppl, gold ppl, pred),”

- OpenNMT Forum, March 2017. [Online]. [Accessed Oct. 18, 2019].
<http://forum.opennmt.net/t/metrics-bleu-ppl-gold-ppl-pred/249>
- [123] AntoViral, "Predictive R-squared according to Tom Hopper," RPubS, Aug. 16, 2015. [Online]. [Accessed Oct. 18, 2019]. <https://rpubs.com/RatherBit/102428>
- [124] M. Zambrano-Bigiarini, "nrmse function," RDocumentation, Aug. 8, 2017. [Online]. [Accessed Oct. 18, 2019].
<https://www.rdocumentation.org/packages/hydroGOF/versions/0.3-10/topics/nrmse>
- [125] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric," San Francisco, CA: PLOS ONE, June 2, 2017. [Online]. [Accessed Oct. 19, 2019].
<https://doi.org/10.1371/journal.pone.0177678>
- [126] C.Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Proceedings of the ACL-04 Workshop, W04-10 2004, Barcelona, Spain, July 25-26, 2004*. pp. 74-81. ACL Anthology, June 15, 2004. [Online]. [Accessed Oct. 18, 2019]. <https://www.aclweb.org/anthology/W04-1013/>
- [127] "Summary of the HIPAA Security Rule," Washington, D.C.: U.S. Dept. of Health and Human Services, Office for Civil Rights, July 26, 2013. [Online]. [Accessed Oct. 18, 2019]. <https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html>
- [128] "Summary of the HIPAA Privacy Rule," Washington, D.C.: U.S. Dept. of Health and Human Services, Office for Civil Rights, July 26, 2013. [Online]. [Accessed Oct. 18, 2019]. <https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html>
- [129] *I2C-Bus Specification and User Manual*, NXP Semiconductors, standard UM10204, 1986. Rev. 06, Apr. 4, 2014.
- [130] *I2S-Bus Specification*, NXP Semiconductors, 1986. Revised June 5, 1996.
- [131] *Universal Serial Bus Specification*, Compaq, Intel, Microsoft, NEC. Rev. 1.1, Sept. 23, 1998.
- [132] *Universal Serial Bus Specification*, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. Rev. 2.0, April 27, 2000.
- [133] *Universal Serial Bus 3.0 Specification*, Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, NEC Corporation, ST-NXP Wireless, Texas Instruments. Rev. 1.0, Nov. 12, 2008.
- [134] *Universal Serial Bus 3.1 Specification*, Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, Renesas Corporation, ST-Ericsson, Texas Instruments. Rev. 1.0, July 26, 2013.
- [135] *Universal Serial Bus 4 (USB4™) Specification*, Apple Inc., Hewlett-Packard Inc., Intel Corporation, Microsoft Corporation, Renesas Corporation, STMicroelectronics, Texas Instruments. Rev. 1.0, August 2019.
- [136] T. Fisher, "USB: Everything You Need to Know," Lifewire, Nov. 13, 2019. [Online]. [Accessed Nov. 15, 2019]. <https://www.lifewire.com/universal-serial-bus-usb-2626039>
- [137] T. Mayor, "Ethics and automation: What to do when workers are displaced," MIT

- Management, Sloan School, July 8, 2019. [Online]. [Accessed Nov. 14, 2019].
<https://mitsloan.mit.edu/ideas-made-to-matter/ethics-and-automation-what-to-do-when-workers-are-displaced>
- [138] P. Khargonekar and M. Sampath, “Socially Responsible Automation: A Framework for Shaping the Future,” EmTech Next, June 12, 2019. [Online Video]. [Accessed Nov. 14, 2019].
<https://events.technologyreview.com/video/watch/pramod-khargonekar-meera-sampath-socially-responsible-automation/>
- [139] Rien, Calvin, “SavWav.cs: Unity3D script to save an AudioClip as a .wav file,” *GitHub Gist*, 7 April 2012. [Online]. [Accessed 4 April 2020].
- [140] Brownlee, Jason. “A Gentle Introduction to Calculating the BLEU Score for Text in Python.” *Machine Learning Mastery*, 18 Dec. 2019, machinelearningmastery.com/calculate-bleu-score-for-text-python/
- [141] Papineni, Kishore, et al. “Bleu.” *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 2001, doi:10.3115/1073083.1073135.
- [142] Chen, Boxing, and Colin Cherry. “A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU.” *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2014, doi:10.3115/v1/w14-3346.