Compute-efficient Real-time Voice Cloning

An ECE 44x Project Document

Connor Saltmarsh
Matthew Raffel
Micah Janzen
Grant Everson

1 - Overview:

Section 1 Overview details overall project information and summaries of project progression. This section will be focused on project management and communication between team members regarding individual roles, team protocols, timeline management, and specific milestones. The project definition is contained in this section and shall be continually revised for more clarity on the project scope as our research develops.

1.1 - Executive Summary

A voice cloning machine learning model receives a speech and text input and creates a new speech output reading the text input in the voice of the speaker input. Such a voice cloning procedure used to need nearly an hour of audio to create a realistic cloned voice; however, recent advances in machine learning research using an SV2TTS model has lowered the necessary audio to around 5 seconds [1]. The SV2TTS model is composed of three components consisting of a speaker encoder, a synthesizer, and a vocoder [2,3,4]. Since the SV2TTS model is composed of three submodels it requires a large number of parameters, is computationally expensive, and is slow to both train and run, preventing its use for low-end systems.

Our project is called “Compute Efficient Real-Time Voice Cloning.” Our project aims to both speed up and reduce the computational resources necessary to provide a voice-cloning model that will be uploaded to a low-end system. To achieve this objective we have divided the project into three separate stages:

- Implement to the “ESPnet” repository[6] a modified version of VITS [7] called YourTTS [8], a new state-of-the-art voice cloning model .
- Implement memory and processing optimizations such as quantization.
- Upload the final model to a low-end system and create additional peripherals for users to interface with the system.

The ESPnet repository is a machine learning toolkit to speed up the productivity of machine learning engineers [6]. By implementing a modified YourTTS model into ESPnet we gain access to an improved training and evaluation environment that is more accessible to a broader audience. The low-end system we plan to use in our system is a Raspberry Pi 4, a versatile microcontroller used in many electronic system applications. The peripherals consist of a miniature button keyboard with an attachable display and a microphone for user inputs and a speaker for a cloned voice output. Each of these peripherals are designed and implemented on a PCB.
Since the project's inception, our team has learned the basics of deep learning and has familiarized ourselves with the YourTTS model. Additionally, we have studied the

structure of the ESPnet repository as we begin our plan to implement the YourTTS model into it. In regards to hardware, we have designed schematics for the system peripherals and have started testing the microcontroller's ability to run machine-learning models.

1.2 - Team Contacts & Protocols

1.2.1 - Team Contacts

Table 1.2.1: Team Contacts

Name:	Contact:	Primary Role:	Contributions:
Connor Saltmarsh	saltmaco@oregonstate.edu	Treasurer	Output Hardware Integration
Micah Janzen	janzenm@oregonstate.edu	Scribe	Input Hardware Integration
Grant Everson	eversong@oregonstate.edu	Project Support	System Software Handling
Matthew Raffel	raffelm@oregonstate.edu	Technical Leader	Quantization Implementation

1.2.2 - Protocols

Table 1.2.2: Team Protocols

Topic	Protocol	Standard
Communication with Project Partner	Matthew communicates with Project Partner Dr. Chen.	Bi-Weekly emails to Dr. Chen outlining progress on the project and any lingering questions. Monthly meeting with Dr. Chen to discuss the project.
On-time Deliverables	If task(s) are not completed by the due date, explain to the team why, next steps, and possible re-assessment of task(s).	A one-paragraph message submitted to discord explaining reasoning for delay and a proposed plan for completing the task.
Meeting Duration and Frequency	Meet a minimum of 2x per week for 1.5 hours combined.	Meet every week at LINC on Tuesday and Thursday 5:15-6:00pm.

Task Management	Fill out a team weekly progress report under individual sections and any team progress made.	Weekly progress report must be half a page in length.
Team Communication	Weekly Discord Messages and fill in jobs in team Notion page	Answer questions and provide personal questions or suggestions about the project.

1.3 - Gap Analysis:

The SV2TTS model was a major advancement for voice cloning as it reduced the amount of source audio required to clone a voice; however it possesses three major drawbacks to make it widely adaptable. These include the computational complexity for it to clone a voice, the slow speech generation speed of its vocoder submodel [2, 5], and being implemented into an unofficial GitHub repository. As such our project exists to solve each of these issues, while also uploading our model into a low-end handheld system that allows the model to be easily usable for its target end user.

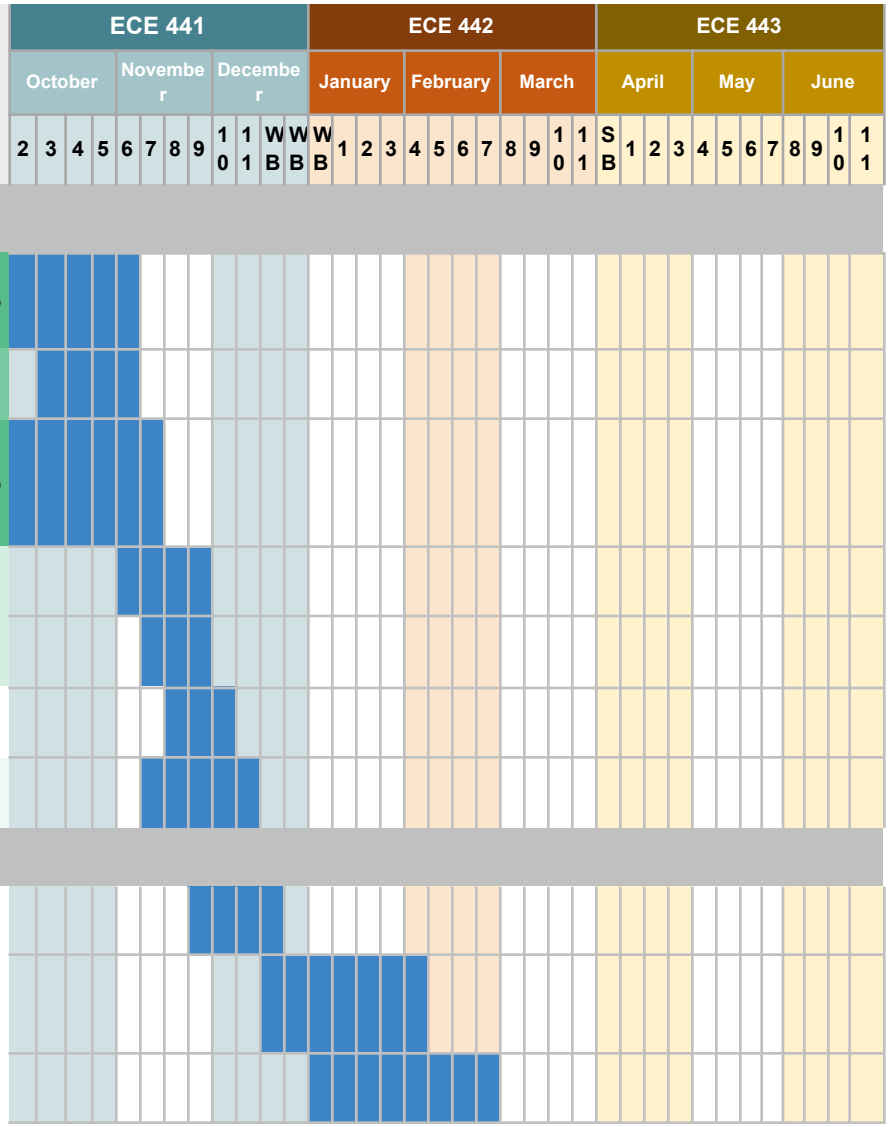
One target end user of our YourTTS model implemented on a low-end handheld system is those who have lost their voices [7]. With our system, this demographic will be capable of recapturing their voices with a low latency carriable system, which the SV2TTS model could not perform. A secondary end user is a simultaneous speech interpreter [1]. Typically, interpreters speak a translation simultaneously using their own voice of a source person's speech. However, with our faster, more compact model these interpreters could type the words of the individual they are translating and have the model speak in the source person's own voice. Furthermore, implementing the model into ESPnet[6] is more accessible to future ML engineers who can provide further improvements. Some key end-product stakeholders could be larger companies that want to improve their voice assistants, organizations that aim to help provide realistic speech to those unable to speak, or audiobook companies looking to create realistic cloned voices to read text.

We are assuming that the target end users and each of these stakeholders will value a low-latency system that requires a lower computational cost than previous methods when cloning voices. Additionally, we assume that there is merit to placing such a voice-cloning model on a low-end handheld system and that our model will not be

overshadowed by other voice-cloning models allowing ML engineers to continue adapting it for the foreseeable future on ESPnet.

1.4 - Timeline

TASK NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Research and Planning**					
1.1	Read Chapters 5-10 of Deep Learning Textbook	Everybody	10/3/2022	11/11/2022	39	100.00%
1.2	Read SV2TTS Research Papers	Everybody	10/10/2022	11/11/2022	32	80.00%
1.3	Verify SV2TTS Github Repository Implementation is Correct	Matthew	10/3/2022	10/14/2022	11	100.00%
1.4	Hardware Schematic Design	Connor	11/10/2022	11/25/2022	15	25.00%
1.5	Familiarize Yourself With ESPnet Code	Everybody	11/14/2022	11/25/2022	11	25.00%
1.6	Verify Jetson Nano runs SV2TTS model	Matthew	11/21/2022	12/2/2022	11	0.00%
1.7	Create Comprehensive Research Document	Everyone	11/14/2022	12/9/2022	25	10.00%
2	**Implement and Test**					
2.1	Implement SV2TTS model into ESPnet	Matthew	11/20/2022	12/20/2022	30	0.00%
2.2	Optimize SV2TTS model with updated submodels	Matthew	12/20/2022	2/1/2023	43	0.00%
2.3	Implement Quantization	Matthew & Micah	1/9/2023	2/28/2023	50	0.00%



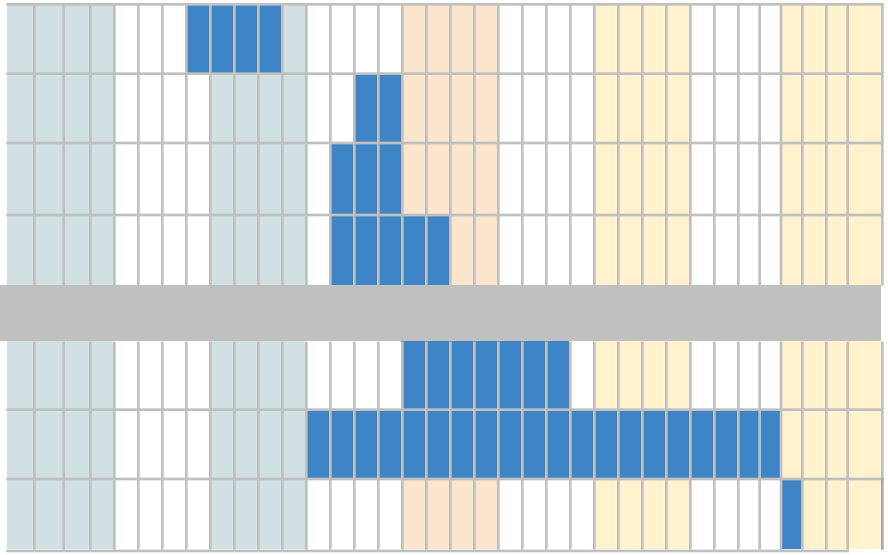
2.4	Design Interface PCBs	Connor,Micah, Grant	11/20/2022	1/20/2023	61	0.00%	
2.5	Order Interface PCBs	Connor	1/20/2023	1/27/2023	7	0.00%	
2.6	Create an Enclosure	Grant	1/9/2023	1/31/2023	22	0.00%	
2.7	Power and Memory Analysis	Micah and Grant	1/9/2023	2/10/2023	32	0.00%	
3	**Refine and Finalize**						
3.1	Verification and Testing	Everyone	2/1/2023	3/20/2023	47	0.00%	
3.2	Documentation of Progress	Everyone	1/9/2023	5/20/2023	131	0.00%	
3.3	Present Work	Everyone	6/1/2023	6/1/2023	0	0.00%	

Figure 1.4: Project Timeline

1.5 - References and File Links

1.5.1 - References

[1] Jia Y., “Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis”, arXiv e-prints, 2018.

[2] Kalchbrenner N., “Efficient Neural Audio Synthesis”, arXiv e-prints, 2018.

[3] Wang Y., “Tacotron: Towards End-to-End Speech Synthesis”, arXiv e-prints, 2017.

[4] Wan L., Wang Q., Papir A., and Lopez Moreno I., “Generalized End-to-End Loss for Speaker Verification”, arXiv e-prints, 2017.

[5] Jemine C., “Real-Time Voice Cloning,” M.S. thesis, Université de Liège, Liège, Belgique, 2019.

[6] Watanabe S., “ESPnet: End-to-End Speech Processing Toolkit”, arXiv e-prints, 2018.

[7] Casanova, E., Weber, J., Shulby, C., Junior, A. C., Gölge, E., and Antonelli Ponti, M., “YourTTS: Towards Zero-Shot Multi-Speaker TTS and Zero-Shot Voice Conversion for everyone”, arXiv e-prints, 2021.

[8] Kim, J., Kong, J., and Son, J., “Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech”, arXiv e-prints, 2021.

1.5.2 - File Links

Project Learning Materials:
https://docs.google.com/document/d/10PZG9tsSCM68SfZKgvimSlv4Om6vntkV_J9l6_o2hrY/edit?usp=sharing

1.6 - Revision Table

Table 1.6: Revision Table

Author:	Date:	Change:	Reasoning:
Grant	10-14-2022	Created Document	Draft Project assignment due

Connor	10-21-2022	Overview Content	Revision from Professor comments
Connor	11-4-2022	Executive Summary Addition	Update summary to include current design choices/approaches.
Matthew	11-14-2022	Revised Executive Summary, Added Table for Protocols Section, Added new timeline to document	Project Document Submission
Matthew	11-16-2022	Revised Gap Analysis	Project Document Submission
Matthew	11-17-2022	Added additional citations and filled out Table for Protocols Section	Project Document Submission
Micah	11-18-2022	Fixed simple spelling and grammar errors, revised document	Project Document Submission
Connor	3-12-2023	Updated project summary	Provide up to date information on relative hardware and microcontroller
Matthew	5-13-23	Updated Gap Analysis and Executive Summary with up-to-date information	Provide up to date information on voice cloning machine learning model

2 - Impact and Risks

The section 2 impacts and risks section outlines the potential risks and solutions to risks the team may face during the project. We provide both a design impact statement and a risk assessment table. Additionally, the section provides a references and links section and a revision table tracking changes.

2.1 - Design Impact Statement

Our project consists of creating a real-time compute-efficient voice cloning machine learning system. From designing and using such a system creates the possibility of negative

consequences. We explore these consequences under the guidance of four broad categories consisting of the following:

- Public Health, Safety, and Welfare
- Cultural and Social
- Environmental
- Economic

For each potential impact, we will explain the necessary background information and the causes leading to the implication. Additionally, we will introduce solutions to counter the effect. Even though we aim to solve each negative impact, some will be outside the scope of our abilities, and we must consider tradeoffs. Providing a comprehensive examination of our project's potential negative impacts enables us to avoid or diminish their effects. Furthermore, it provides our project partner and stakeholders assurance of our ability to foresee negative outcomes, create plans to avoid them and keep them informed on project details.

Voice cloning models can easily have negative impacts on public safety and welfare if it is not safely integrated into society. In today's society, where scam phone calls are repetitive and relentless, citizens have become better at deciphering them, however, some scamming organizations have begun utilizing advanced technology similar to voice cloning machine learning to record the voices of the consumer and clone their voice. Using a model such as this, scammers could get valuable information such as bank account access over the phone by sounding like the consumer, or vice versa, having a convincing or recognizable voice on the "banking" end and get people to give up banking information. One study conducted by Ovadya and Whittlestone explains that voice cloning models will only be easily implemented if there are "already established processes for phone scams into which new voice generation capabilities can be slotted easily, without any need to change goals or strategy" [1]. An important takeaway with this type of technology is that before it can be fully integrated to better society, there needs to be existing practices that can decipher a cloned voice and avoid malicious behavior. In regards to the specific project, the group's implementation of such a model will be designed as a simple, easy to use handheld device with recording and text input capabilities. One negative impact to this is obviously accessibility to scammers and people looking to use a voice cloning device for malicious intent. One solution toward limiting the malicious intent, albeit not very practical, would be to have a device registration requirement. Since the intent of the device is to assist the disabled/speech impaired (not as a consumer facing product) this process may be more feasible with a focused demographic.

When creating a system that can clone a human's voice, machine learning requires tons of data. With the data collection, it is often seen in engineers' eyes and the public eyes as being absolute and unbiased. However, data collection can have downfalls when the way it is collected is inherently biased or too focused on getting specific data. An example of this is seen in the Los Angeles police department trying to implement AI with predictive guesses as to where crime will occur. " Since the program uses skewered data collected by police over the years, it is likely to be biased against minorities, with critics suggesting the algorithm has been disproportionately targeting minority neighborhoods"[2]. This bias goes beyond how data is collected and can apply to have most white people collect data for white people. For our project,

the datasets used to train the model are from VCTK and LibriTTS, both of which are populated by native English speakers' recordings with differing accents[3 & 4]. Native English-speaking datasets are biased towards exactly what the dataset was created from native English speakers. This creates inaccuracies with cloning voices of minorities, or anyone that is not a native English speaker as the model will struggle to understand different cadences of non-English speakers. To mitigate bias within datasets, including non-native speakers would allow models trained on datasets to be utilized by non-native speakers. A secondary social impact of voice cloning is impersonating other people, not our voices. Since only a very short clip of anyone's voice is needed, anyone could clone someone else's voice with high accuracy. Kitti Palmai, a writer for BBC, writes about how voice cloning has been growing for voice actors and cybercriminals. Palmai quotes someone to interview on the topic, saying, "'For example, if a boss phones an employee asking for sensitive information, and the employee recognizes the voice, the immediate response is to do as asked. It's a path for a lot of cybercrimes.'"[5]. This idea can be easily expanded to any other area, where asking for sensitive information or impersonating another person can create trust from the victim of the cybercrime, as voice is still seen as a difficult human component to replicate. It's important to ensure measures are taken to limit the accuracy of voice cloning. As models get more accurate, there should still be inaccuracies in the voice that let people know the voice is produced from a computer.

All supervised machine learning tasks are composed of training and inference. During training, a model will learn parameters to perform a specific task, and during inference, the model will use the learned parameters to perform the trained task. In our project, our model will learn parameters to improve its ability to clone a voice using a dataset and a learning algorithm, allowing it to clone any provided voice. One issue with machine learning algorithms is that each requires many parameters to perform tasks like voice cloning well. These parameters coincide with the amount of energy consumed by the model while training and inferring. From requiring lots of energy to both train and evaluate a model, CO_2 emissions are released into the atmosphere. In the case of training, a standard machine learning Transformer model designed for natural language processing tasks produces 192 lbs of CO_2 while using a GPU [6]. For reference, the average American produces 99 lbs of CO_2 a day [6]. Since voice cloning is a more complex task than standard natural language processing tasks like automatic speech recognition or translation, it requires a combination of multiple different machine learning submodels. As such, the process of training a single voice cloning model will produce an even greater amount of CO_2 than that forecasted for the standard transformer model. As expected, training a model requires more energy than inference. However, this is only the case if the inference only occurs once[7]. But, in most cases, a model will undergo inference multiple times, which causes inference actually to consume more energy than training. For our voice-cloning system to be useful for end users such as those who have lost their voices, it will need to perform inference hundreds of times a day, effectively overshadowing the training costs in the long run regarding carbon emissions. In order to combat CO_2 emissions and other negative effects resulting from excessive energy consumption in the training and inference of a model, we plan to implement a combination of techniques, such as quantization, into our models [8]. By doing so, we will reduce the model's computational costs, in turn reducing the energy consumption of the model.

One quite prevalent occupation when dealing with foreign relations and business is a translator. According to the United States Bureau of Labor Statistics, there are an estimated 69,000 jobs in the interpreter and translator field of work. With an average pay of almost $50,000 per year, this occupation accounts for nearly 3.5 billion dollars of yearly pay. Even more so, the projected market for this occupation is set to grow by 20% over the next ten years [9]. Despite this, there may be a concern in the future that machine learning could potentially slow the growth of this profession or even start to replace humans when translating, especially as text-to-speech and voice cloning starts to sound more and more human. Utilizing machine learning in positions where translations are needed is a great way for companies to lower costs when dealing with foreign and even inter-company communications. As the world does not all speak and read the same language, there will always be a need for some type of translator. According to the article "Will artificial intelligence replace human translation?" there is no possibility that artificial intelligence, or machine learning, will replace human translators [10]. Citing that each language has different nuances, semantics, cultural impacts, and syntax, there will be no way to translate one phrase to mean the exact same thing in each language. Humans can generally provide a better translation as they understand each language's culture and can better reflect phrases between two distinct languages. Though this may not be a concern now, machine learning models continually evolve, so in the future, when models become more proficient, they may become more reliable with translations than humans. Even so, we will most likely not fully replace translators, as they are often needed for legal documents and other higher-stakes communication. Though for other situations, they may be good enough to serve as interpersonal communication translators. A last potential economic impact is voice clonings' ability to be used for fraud. While on much less of a grand economic scale, this type of machine learning could be used for small-scale thievery. Spam calls have become more common in recent years, and unfortunately, many tend to target the elderly. With a bit more targeting, this could be used to get a voice sample from a relative and, using that, clone the voice and target family members, tricking them into sending money.

2.2 - Risks

Table 2.2: Risk Breakdown

Risk ID	Risk Description	Risk Category	Risk Probability	Risk Impact	Performance Indicator	Action Plan
1	Falling Behind in Reading/Research	Timeline	Med	Med	Lack of Understanding	Reduce depth and increase the breadth of reading
2	Unable to understand Espnet or Voice Cloning Code	Technical	High	High	Lack of Understanding	Use debugger to step through code

3	Unable to gain access to GPUs to train models	Organizational	Med	High	No way to train models	Contact Project Partner
4	Model does not run well on Raspberry Pi	Technical	High	High	Unable to evaluate models	Use Jetson Nano
5	Lack of progress in updates to Project Partner	Organizational	Med	Med	Feedback from Project Partner	Come to understanding on expectations with Project Partner
6	Parameter reduction techniques negatively affect performance	Technical	High	High	Voice cloned does not sound realistic	Increase the bit representation of parameters
7	Losing edits made to code	Organizational	Low	High	Code is not longer available	Revert changes in Github repository
8	Insufficient funds to order PCBs and additional materials for project	Cost	Low	Med	Run out of allotted $300 from course	Use personal money

2.3 - References and File Links

2.3.1 - References

[1] Ovadya and J. Whittlestone, “Machine learning classification of malicious resident space objects,” *Computers and Society Cornell*, 2022.

[2] Palmai, Kitti. “Voice Cloning of Growing Interest to Actors and Cybercriminals.” *BBC News*, 11 July 2021, www.bbc.com/news/business-57761873.

[3] “VCTK Dataset.” *Machine Learning Datasets*, datasets.activeloop.ai/docs/ml/datasets/vctk-dataset/. Accessed 1 Dec. 2022.

[4] “LibriTTS.” *Google Research*, research.google/tools/datasets/libri-tts/. Accessed 1 Dec. 2022.

[5] Narayanan, Aswin. “AI Bias: How Technology Negatively Impacts on Minorities.” *ScreenRant*, 14 June 2020, screenrant.com/ai-bias-technology-negative-impact-minorities/. Accessed 4 Nov. 2022.

[6] Strubell, E., Ganesh, A., and McCallum, A., “Energy and Policy Considerations for Deep Learning in NLP”, arXiv e-prints, 2019.

[7] Desislavov, R., Martínez-Plumed, F., and Hernández-Orallo, J., “Compute and Energy Consumption Trends in Deep Learning Inference”, arXiv e-prints, 2021.

[8] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K., “A Survey of Quantization Methods for Efficient Neural Network Inference”, arXiv e-prints, 2021.

[9] “Interpreters and Translators : Occupational Outlook Handbook.” *U.S. Bureau of Labor Statistics*, U.S. Bureau of Labor Statistics, https://www.bls.gov/ooh/media-and-communication/interpreters-and-translators.htm. (accessed Sept. 8 2022)

[10] Limited, Crystal Hues. “Will Artificial Intelligence Replace Human Translation?” *LinkedIn*, https://www.linkedin.com/pulse/artificial-intelligence-replace-human-translation-crystal-hues-ltd. (accessed Apr. 23 2022)

2.3.2 - File Links

None.

2.4 - Revision Table

Table 2.4: Revision Table

Author:	Date:	Change:	Reasoning:
Matthew	11-2-22	Created Section 2	Draft Project Section 2 assignment due
Matthew	11-16-22	Added risk 8 to risk table	Project Document Submission
Matthew	4-25-23	Created Design Impact Statement	Design Impact Statement Submission

3 - Top-Level Architecture

3.1 - Block Diagram

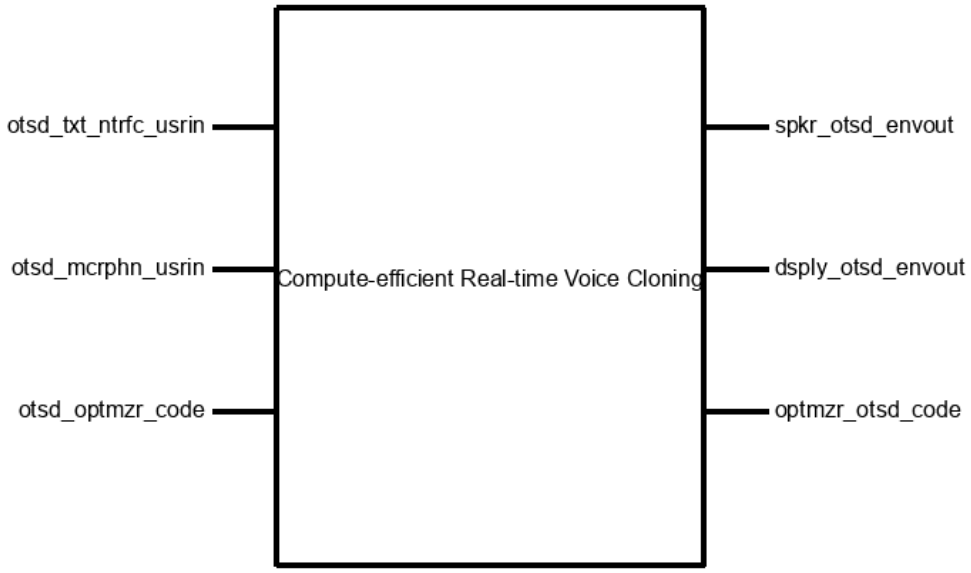


Figure 3.1.1: Top Level Black Box Diagram

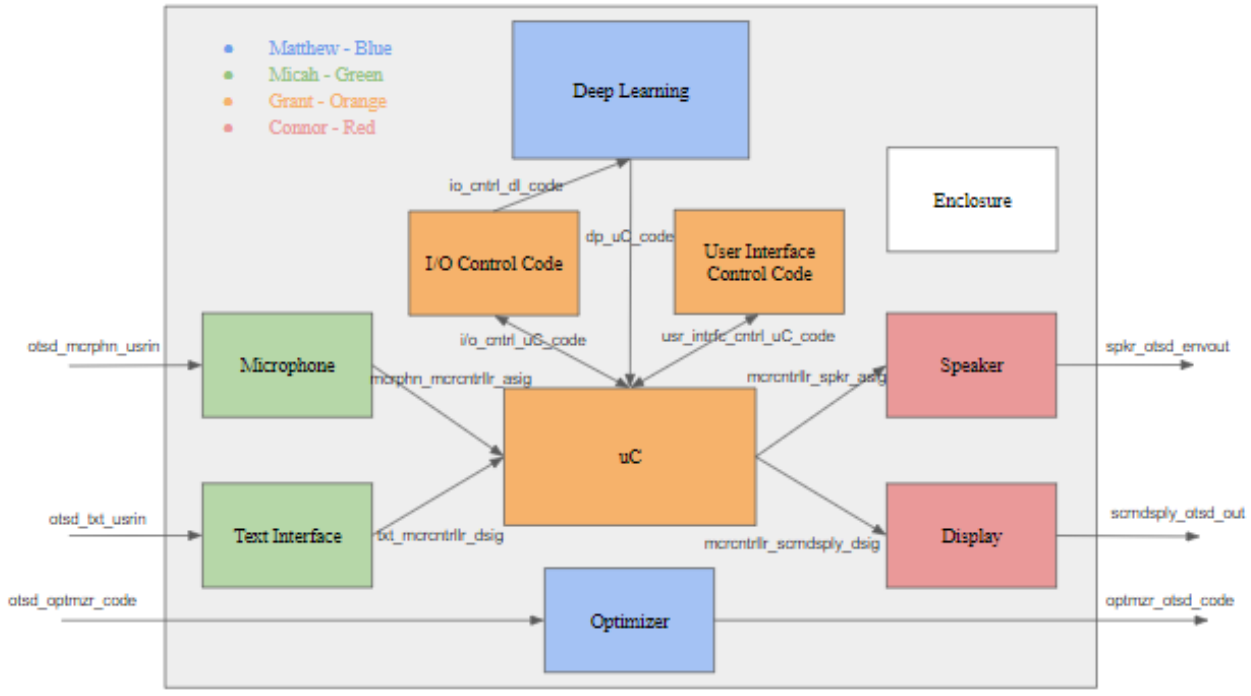


Figure 3.1.2: System Block Diagram

3.2 - Block Descriptions

Table 3.2: Block Descriptions

Name	Description
Speaker Champion: Connor Saltmarsh	The Speaker block will consist of an audio amplification circuit used to produce the cloned voice for the user to hear. Contained within this block will be an audio amplifier paired with a potentiometer for user volume control. The audio will then be output from a speaker built into the system. There will only consist of one input being the cloned voice audio, and the output will be the same audio just amplified and volume adjusted. Thus input will be an analog signal generated by the microcontroller and passed to the amplifier circuit to be output as an analog signal through the speaker to the environment. This block does not specifically interact with the final audio wave, but merely amplifies and outputs it for a user to easily understand and loud enough to be heard in a room with moderate ambient noise. The blocks circuit will also be integrated with other Printed Circuit Boards (PCBs) containing the input and output peripherals for each hardware block in the entire system.
Display Champion: Connor Saltmarsh	The Display block will be an additional peripheral for the user to receive system messages as well as view the text input they type into the system. The screen will be primarily used to display messages and text, so the screen shall be small enough to be handheld to fit our enclosure, but also large enough to be easily read from 3 feet away. The screen will communicate with the internal microcontroller to display a GUI for the user to interact with.
Text Interface Champion: Micah Janzen	The text interface block is a sub-system that will allow the user to input text using a 10 button letter interface to type text along with 1 button to delete a letter and another button to confirm the text string. The string of letters will send the information to the microcontroller where it will be used to tell the cloned voice what to say. The block will be built using a custom PCB using hardware to debounce buttons, communicating with the microcontroller via the Serial Peripheral interface, and having extra buttons for submitting and deleting text. The text will be displayed on the screen so the user can view what they are inputting using the text input interface. The 10-button letter interface allows the user to input any letter in the English alphabet by having each button represent 3 letters, where the letters are differentiated by the timing of the button pressed. A single press enters the first letter correlated to that button, two quick button presses enter the second letter and the same follows for the third letter. The debouncing of the buttons is completed using a resistor and capacitor circuit for each button. The circuit then utilizes an inverter and shift register to store and synchronize the data to send to the microcontroller via SPI.
Microphone Champion: Micah	This block would be researching and purchasing a microphone that will work with our microcontroller. This includes researching a microphone that is relatively inexpensive but also a quality microphone to ensure that the analog signal going

Janzen	into the microcontroller doesn't include large amounts of noise. Also, make sure the connection to the microcontroller is possible with the purchased microphone.
Microcontroller Champion: Grant Everson	This block consists of the research and decision of acquiring and setting up a micro controller capable of running our model within our system restraints, as well as interface with needed hardware and software modules within the system. The goal of this is to be our main computation unit for the project. The microcontroller will be responsible for not only interfacing, but also powering all hardware peripherals including a microphone, text interface, screen, and speaker amplifier. The microcontroller itself will be supplied power through the suggested power supply, supplied by the manufacturer for the specific microcontroller. This will act as the heart of the system, and consist of multiple coding blocks interacting with different parts of the system to deliver a proper user experience. This block includes the microcontroller processing code as well as the power distribution for the system.
Optimizer Champion: Matthew Raffel	The Optimizer block will adapt the Deep Learning Model block to be more computationally efficient. Such a block will first convert the deep learning model to an Open Neural Network Exchange (onnx) format and will then quantize it. By converting the model to an onnx format, it will be able to run on a wider variety of devices and optimize the model's architecture by combining neural network layers together. The voice cloning model implemented in the deep learning model is composed of millions of parameters that allow the model to make predictions. Each of these parameters is a number with a 32-bit representation. By applying quantization to the model, the precision of the parameter representation will reduce. In turn, both of these optimizations will allow the model to become faster and more computationally efficient, with a minimal impact on the model's voice cloning capabilities.
Deep Learning Champion: Matthew Raffel	The Deep Learning block will contain our machine learning model that clones the user's voice and a speaker encoder. The voice cloning model we will use is called YourTTS. The voice cloning model will be uploaded to the microcontroller, where it will receive a python dictionary. The python dictionary will contain two strings: a user text input and a file path to a .wav file containing a user's speech input. The .wav file provided at the file path will have at least 5 seconds of audio and possess minimal noise, according to 9/10 people. The sampling rate of the .wav file will be at least 22,050 Hz. From the .wav file, a speaker embedding will be generated using ECAPA-TDNN, which is a speaker encoder that takes a speech waveform and generates a 128-dimensional vector representation. By using the text input and the speaker embedding, the voice cloning model will produce an output speech waveform as a .wav file reading the text input in the user's voice. The produced .wav file will also have a sampling rate of 22,050 Hz, and 9/10 people will agree that the produced voice in the .wav file is reading the text input. For reproducibility purposes, the voice-cloning model will have an implementation in the ESPnet toolkit.

I/O Control Code Champion: Grant Everson	The I/O Control Code block will consist of the code necessary for the host computer to be able to interact with our other blocks. This host computer is planned to be a Jetson Nano, which on top of possessing USB inputs, has a bank of 40 GPIO pins that will allow us to interact with our other components. The blocks to send and receive data from will include our text input interface, a microphone, a user interface, the voice cloning model, and a speaker. This will allow data to be able to flow freely within the overall system pipeline. With that in mind the first few things the user is going to interact with is either the text interface or the microphone to input a sample voice to clone. This block will convert the 16 bit inputs of our text interface, which will consist of a 12 button layout, to controls such as record and playback along with english text input. The text will be shared with the user interface in order to display what the user is typing. It will also be converting a microphone input to a usable file format for the voice cloning model. Once this voice sample is recorded it will provide it to the voice cloning model to start its processing. Once the user has specified their input text and hit process this code block will pass off a file path to a text file to the voice cloning model and allow it to compute and return a .wav file of the user input. This code will then be able to playback the file out through to a speaker. Overall this block consists of a few smaller functionalities that link together our entire process.
Enclosure Champion: Connor Saltmarsh	This block contains the work done for assembling the final system into one cohesive unit. The enclosure shall reasonably protect and hide all wiring and microcontroller. The enclosure is also intended to be compact enough to remain a handheld system. In terms of functionality, the enclosure's main purpose is to condense and contain all electrical systems within the final system. The enclosure is not required to have a high level of ergonomics, other than meeting the criteria of being handheld.
User Interface Control Code Champion: Grant Everson	This block will handle displaying information to the user of the system. This will host buttons or other user interact-able objects to allow the user to record their voice to be cloned, enter text for the cloning model to read from, and playback the output audio from the model. By interacting with the I/O code and deep learning model it will be able to retrieve data to display.

3.3 - Interface Definitions

Table 3.3: Interface Definitions

Name	Properties
otsd_txt_ntrfc_usrin	Timing: button press signal reaches uC within 30ms Type: Debouncing RC filter delays button push by no more than 20ms

	between pushing the button and hex inverter output signal Type: Mechanical Cherry MX Switches used for 12 buttons
otsd_mcrphn_usrin	Other: Microphone produces waveform that is no greater than 20 dB as displayed from ffplay on terminal with microphone somewhat close to mouth Timing: Microphone can record audio consistently for at least 5 seconds Type: Microphone can produce audio that is above -45 dB using ffplay on .mp3 file of recording when 1 foot from microphone.
otsd_optmzr_code	Other: A .scp containing speaker ids and the path to an x-vector associated with each speaker. Other: A .zip file containing: a .pth file of the deep learning model checkpoint, a .yaml file of the zip file structure, and a .yaml file of the deep learning model configuration. Other: A text file containing speaker ids and their associated spoken text.
spkr_otsd_envout	Other: Distance: audible range measured at 2 feet away Other: dB max: < 35 dBA increase from ambient dBA (40dBA) Other: dB min: > 5 dBA increase from ambient dBA (40 dBA)
dsply_otsd_envout	Light: Produces less than 400 lux from 6 inches away with all white screen Other: Resolution: 800 by 480 pixels Other: Visibility: visible by 9/10 users during indoor midday lighting
txt_ntrfc_mcrcntrllr_dsig	Fall Time: RC debouncing fall time no greater than 10ms Logic-Level: Active High: 3.3V, represented as logic 1 in code Logic-Level: Low: 0V, represented as logic 0 in code
mcrphn_mcrcntrllr_asig	Other: Resistance between ground and channels of microphone should be a value around 1.2 with 20K mode on DMM Other: Microphone connects to device via 3.5mm audio jack Other: microphone is able to create a .wav file when using audacity and exporting audio.
mcrcntrllr_spkr_asig	Other: Incoming uC internal volume level: 20% of maximum volume output Vmax: supply: GPIO Power Pin 5V Vmax: audio signal: 0.9Vpp (peak to peak)
mcrcntrllr_dsply_dsig	Other: Touch screen feature disabled Other: Display through HDMI connector Other: Powered through USB connector
mcrcntrllr_i_cntrl_cd_data	Messages: Audio recording read for playback: Must be able to save audio in a .wav file format.

	Messages: Audio recorded utilizing a microphone: Must be able to record audio from a microphone input. Messages: Text input from a 12-keypad: encoded with 2 shift registers for a total of 16 bits, must be able to decode these values
mcrcntrllr_usr_ntrfc_cntrl_cd_data	Data Rate: Must update text display within 0.1 seconds of file changing Messages: Must be able to read text from a .txt file Other: Must be able to check for an input and output .wav file
optmzr_otsd_code	Other: The quantized .onnx file is at least 5% faster than the original .onnx file. Other: A .onnx file of the quantized deep learning model. Other: The quantized .onnx file is at most half the size of the original .onnx file and is under 100 MB.
dp_lrnng_mcrcntrllr_data	Messages: A .wav file containing the user's cloned speech. Other: According to 9/10 people the cloned voice in the .wav file reads the text input. Other: The .wav file sampling rate is at least 22,050 Hz
i_cntrl_cd_mcrcntrllr_data	Messages: Text output for user display: Must output text in a format readable by a graphical user interface. Messages: Audio playback to speaker: Must be able to playback audio from a .wav file. Protocol: Data output will be saved to a corresponding file type: .wav for audio and .txt for text
i_cntrl_cd_dp_lrnng_data	Messages: A python dictionary containing two python strings: a user text input and a file path to a .wav file of the users speech. Other: The .wav file at the provided file path has a sampling rate of at least 22,050 Hz. Other: The .wav file is at least 5 seconds in duration.
usr_ntrfc_cntrl_cd_mcrcntrllr_data	Other: User interface must display the existence of the input and output .wav files Other: User interface must fit within 800 x 480 pixels (size of our display) Other: Text interface must be legible from a minimum of 3 feet away.

3.4 - References and File Links

2.3.1 - References

None.

2.3.2 - File Links

None.

3.5 - Revision Table

Table 3.5: Revision Table

Author:	Date:	Change:	Reasoning:
Connor	3-9-2023	Added block descriptions and interface table from portal	System Verification due
Connor	3-13-23	Updated tweaks to microcontroller and enclosure descriptions	Portal updated because of section 4 details

4 - Block Validations

4.1 - Deep Learning Block

4.1.1 - Description

The Deep Learning block will contain our machine learning model that clones the user's voice and a speaker encoder. The voice cloning model we will use is called YourTTS [1]. The voice cloning model will be uploaded to the microcontroller, where it will receive a python dictionary. The python dictionary will contain two strings: a user text input and a file path to a .wav file containing a user's speech input. The .wav file provided at the file path will have at least 5 seconds of audio and possess minimal noise, according to 9/10 people. The sampling rate of the .wav file will be at least 22,050 Hz. From the .wav file, a speaker embedding will be generated using ECAPA-TDNN, which is a speaker encoder that takes a speech waveform and generates a 192-dimensional vector representation [2]. By using the text input and the speaker embedding, the voice cloning model will produce an output speech waveform stored as a .wav file reading the text input in the user's voice. The produced .wav file will also have a sampling rate of 22,050 Hz, and 9/10 people will agree that the produced voice in the .wav file is reading the text input. For reproducibility purposes, the voice-cloning model will have an implementation in the ESPnet toolkit [3].

4.1.2 - Design

The Deep Learning block is required to clone a voice. The black box diagram of the model is shown in figure 3.

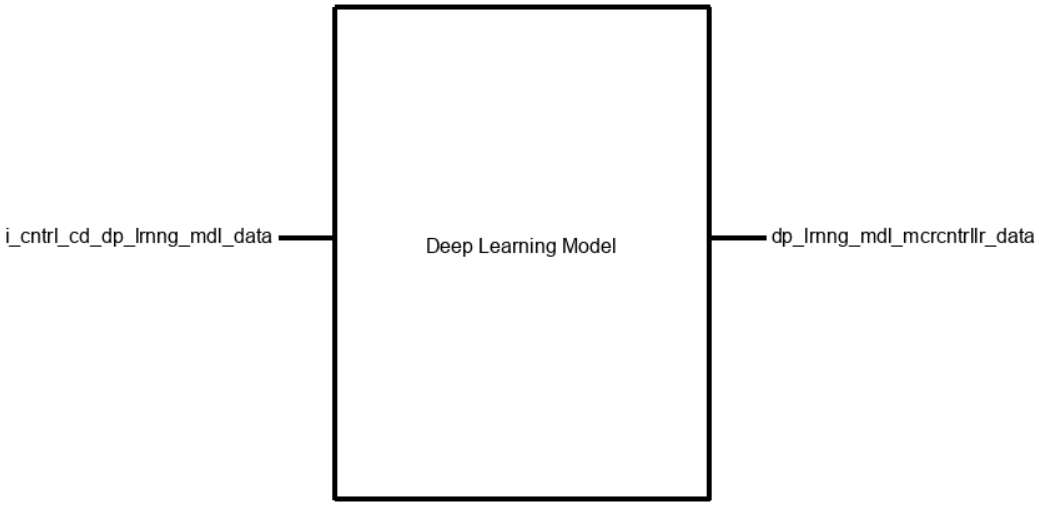


Figure 4.1.2.1: Black Box for Deep Learning Model.

It takes one input from the i/o control code (i_cntrl_cd_dp_lrnng_mdl_data). As previously stated, the input will be a python dictionary containing the user text input and a file path to a .wav file of the user's speech sample. The .wav file at the location of the file path must be at least 5 seconds long and free of distortion to allow for the generation of an accurate cloned voice. The output from the model is a new .wav file reading the provided text input using the voice of the provided speech sample and is saved to the microcontroller (dp_lrnng_mcrcntrllr_data). The process of voice cloning is centered around the deep learning model, YourTTS [1], based on VITS [4], shown in Figure 4.1.2.4. However, a secondary deep learning model called ECAPA-TDNN is used to generate a speaker embedding for YourTTS that captures the voice's characteristics in the provided .wav file input [2].

The inference for the block begins by retrieving the information in the provided .wav file containing the user's speech sample. This .wav file will undergo preprocessing to convert the speech waveform to a tensor format. Once in a tensor format, it will be processed by the ECAPA-TDNN to produce a 192-dimensional x-vector. Each x-vector is a speaker embedding, which is a tensor representation that captures the features in the speaker's voice. ECAPA-TDNN, shown in Figure 4.1.2.2, is composed of multiple neural network layers. The path of the input speech representation consists of a Conv1d+ReLU+BN block, a SE-Res2Block block, a Conv1D+ReLU block, an Attentive Stat Pooling+BN block, a FC+BN block, and finally, an AAM-Softmax block. Residual connections are added between layers in the neural network to help train the neural network. The term residual connection refers to the connections bypassing neural network blocks.

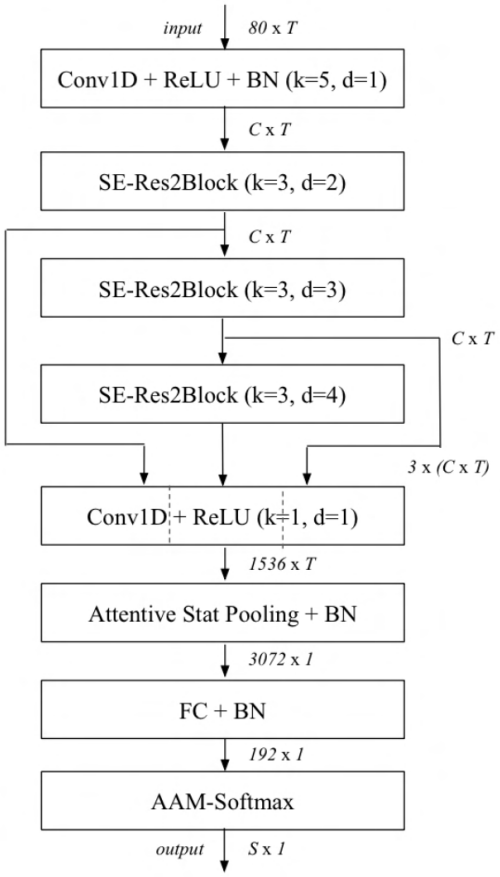


Figure 4.1.2.2: ECAPA-TDNN speaker encoder architecture [2].

The composition of each SE-Res2Block block is shown in Figure 4.1.2.3. Its architecture comprises a Conv1D+ReLU+BN block, a Res2 Dilated Conv1d+ReLU+BN block, and a SE-Block block. A residual connection bypasses all of these blocks.

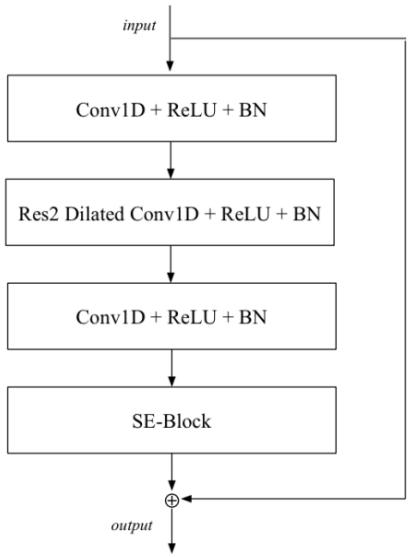


Figure 4.1.2.3: SE-Res2Block used in the ECAPA_TDNN speaker encoder architecture [2].

Once both the text input and the x-vector of the speaker's voice are available, they are fed into YourTTS, shown in Figure 4.1.2.4, to generate the cloned voice reading the text input. The input text is provided to the char embedding block, which converts the text representation into a tensor representation. Following this conversion, the two inputs follow separate yet converging paths in the model. The main components of the model are the transformer-based encoder, a linear projection, the alignment generation, a flow-based decoder, a stochastic duration predictor, a speaker embedding layer, and the HiFi-GAN generator [5]. The input to the HiFi-GAN generator is a MEL spectrogram, which in this case, is an intermediate representation of the cloned speech output. HiFi-GAN uses this MEL spectrogram to create the final waveform output.

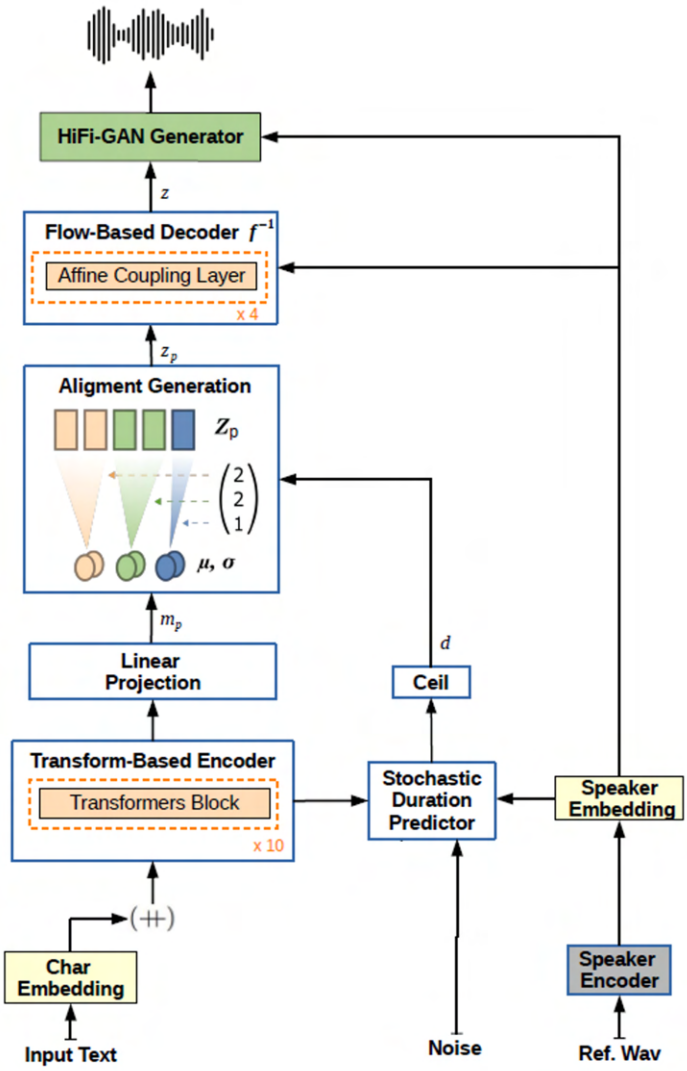


Figure 4.1.2.4: YourTTS deep learning model architecture [1].

4.1.3 - General Validation

As a recap, the deep learning model block needs to be capable of cloning a provided voice. It receives a text and speech input and generates a new speech output reading the provided text in the voice of the speech input. As such, the model must be capable of receiving these inputs from the user and using them to generate a realistic cloned voice. In the context of our project, we will be using the model YourTTS [1] to clone the provided voice. However, YourTTS must use ECAPA-TDNN to convert the provided speech sample into an x-vector before it begins the voice cloning process [2]. Since ECAPA-TDNN is state-of-the-art in generating speaker embeddings, we are confident it will be able to provide YourTTS with the necessary information to clone the speaker's voice. In a normal setting, we would need to train our own version of ECAPA-TDNN; however, to save time, we will use a pre-trained model published by huggingface. We are confident this pre-trained model will work as intended due to the published scores attached to the model. Similarly, YourTTS is also a state-of-the-art model regarding voice cloning in our extensive literature review; therefore, we are confident that it will generate an accurate cloned voice that will read any provided text input.

From our project partner, we have been instructed to have an implementation of our voice cloning model in the espnet repository [3]. Fortunately, an implementation for YourTTS is provided in espnet already. We were able to verify the implementation was correct by testing a pre-trained model on the LibriTTS dataset [6]. We found the voice cloning capabilities of this model to be at a good standard.

Examining the training script for the pre-trained YourTTS model indicates that the training time for the model is two weeks using four V100 GPUs. Fortunately, we have access to V100 GPUs on the OSU high-performance computing cluster, and given our testing with the pre-trained model, we are confident the time expenditure will not be wasteful. However, if we do not have time or find the trained model unsuitable, other voice cloning models requiring less training time are available on espnet. We are not using these at the moment as, according to their papers, they provide worse quality voice generation than YourTTS. Furthermore, one of the benefits of YourTTS is it is an end-to-end model which differs from the cascade architectures of the other models. These cascade models may have slightly faster training times, but they also introduce other issues, such as error propagation, and fine-tuning difficulty. This is due to a cascade architecture requiring the sub-models making up the architecture to be trained in a series. As such, if the first model is not properly trained to a sufficient level, the second model will be trained on useless data.

In order to perform inference, we must be capable of saving the weights and architecture of our trained model. Fortunately, espnet possesses training scripts that perform such a task. Once saved, we must be capable of reading the model and processing new data provided to it. This new data will be the user-provided text input and the speech waveforms. We know such processes will work from our inference tests using a pre-trained model. A more general inference objective of the whole system is for the generated audio to be produced in real-time. In other words, it should take no more than one second to generate one second of audio by the model. Such a requirement depends on the hardware and further optimizations to the model architecture that will be carried out by the optimizer block. Some of the optimizations the optimizer will carry out are converting the model to an onnx format and quantizing it. By quantizing the model, it will change the weights to a lower bit representation making operations quicker and less memory intensive. The onnx format fuses modules together to optimize them for hardware. If the optimizer block is insufficient, we can make a further edit to YourTTS by

using a cut-down version of HiFi-GAN, which will cut inference times in half with a minimal impact on its voice cloning capabilities [5].

4.1.4 - Interface Validation

Table 4.1.4.1: i_cntrl_cd_dp_lrnng_data Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Messages: A python dictionary containing two python strings: a user text input and a file path to a .wav file of the user's speech.	The DL model is programmed in python, and to clone a voice, it needs a source speech sample and a text input.	The DL model architecture is designed to receive a text and a speech input.
Other: The .wav file at the provided file path has a sampling rate of at least 22,050 Hz.	The sampling rate is at least 22,050 Hz to provide the model with enough fine-grain information.	The model was trained on speech samples with a sampling rate of 22,050 Hz.
Other: The .wav file is at least 5 seconds in duration.	Without an adequate length audio sample, the DL model cannot extrapolate the features present in the voice.	The DL model was trained using audio segments ranging between 3 and 10 seconds.

Table 4.1.4.1: dp_lrnng_mcrcntrllr_data Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Messages: A .wav file containing the user's cloned speech.	A .wav file allows for the storage of a generated cloned voice.	The final stage of the DL model, HiFi-GAN [5], is designed to generate an audio waveform.
Other: According to 9/10 people the cloned voice in the .wav file reads the text input.	Reading a text input in the voice of a cloned speaker is the objective of our project.	The DL model is designed using YourTTS [1], which is a state-of-the-art model for voice cloning.
Other: The .wav file sampling rate is at least 22,050 Hz	The sampling rate is 22,050 Hz to provide smooth audio.	The DL model was trained to generate a speech waveform with a sampling rate of 22,050Hz.

4.1.5 - Verification Process

1. Acquire an audio sample from the MUST-C dataset [7], which is a professionally created dataset containing high-quality speech free of noise from Ted Talks.
2. Create a test script that clones a voice using the deep learning model.
    a. The script will prompt the user for the file path to a .wav file containing a spoken voice and a text input on the command line. Both of the inputs from the user will be saved into a dictionary (mcrcntrllr_dp_lrnng_data).
    b. The script will ensure the duration of the .wav file is at least 5 seconds and has a sampling rate of at least 22,050 Hz.
    c. The script will use the inputs and the deep learning model to generate an output speech waveform in a .wav file (dp_lrnng_mdl_mcrcntrllr_data).
    d. The script will check that the output .wav file has a sampling rate of at least 22,050 Hz.
3. Run the Test Script
    a. The script will generate a .wav file containing the cloned voice reading the provided text (dp_lrnng_mdl_mcrcntrllr_data).
    b. The script will provide information on whether the deep learning block passed all interface properties except for determining if the output speech reads the text input.
4. Provide the cloned voice in the .wav file to 10 different people who will determine whether the output speech waveform reads the text input.

4.1.6 - References and File Links

[1] Casanova, E., Weber, J., Shulby, C., Junior, A. C., Gölge, E., and Antonelli Ponti, M., “YourTTS: Towards Zero-Shot Multi-Speaker TTS and Zero-Shot Voice Conversion for everyone”, arXiv e-prints, 2021.

[2] Desplanques, B., Thienpondt, J., and Demuynck, K., “ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification”, arXiv e-prints, 2020.

[3] Watanabe, S., “ESPnet: End-to-End Speech Processing Toolkit”, arXiv e-prints, 2018.

[4] Kim, J., Kong, J., and Son, J., “Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech”, arXiv e-prints, 2021.

[5] Kong, J., Kim, J., and Bae, J., “HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis”, arXiv e-prints, 2020.

[6] Zen, H., “LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech”, arXiv e-prints, 2019.

[7] Cattoni, Di Gangi, M. A., Bentivogli, L., Negri, M., & Turchi, M., “MuST-C: A multilingual corpus for end-to-end speech translation”, *Computer Speech & Language*, 2021.

4.1.7 - Revision Table

Table 4.1.7: Revision Table

Author:	Date:	Change:	Reasoning:
Matthew	1/15/23	Created and Drafted the Block Validation Assignment	Block Validation Assignment due
Matthew	2/9/23	Edited introduction to include specific requirements	Comment from Rachel
Matthew	2/10/23	Expanded the design section while also reformatting and repositioning figures	Comment from Rachel
Matthew	2/10/23	Added/Altered interfaces in interface validation section.	Came up with new interface properties.
Matthew	2/10/23	Added more steps to the verification plan and added additional sources to the references section	Comment from Rachel
Matthew	2/11/23	Made the general validation section more concise and added more explicit mentions of the specific interface requirements.	Comment from Rachel

4.2 - Optimizer Block

4.2.1 - Description

The Optimizer block will adapt the Deep Learning Model block to be more computationally efficient. Such a block will first convert the deep learning model to an Open Neural Network Exchange (onnx) format and will then quantize it. By converting the model to an onnx format, it will be able to run on a wider variety of devices and optimize the model's architecture by combining neural network layers together. The voice cloning model implemented in the deep learning model is composed of millions of parameters that allow the model to make predictions. Each of these parameters is a number with a 32-bit representation. By applying quantization to the model, the precision of the parameter representation will reduce. In turn, both of these optimizations will allow the model to become faster and more computationally efficient, with a minimal impact on the model's voice cloning capabilities.

4.2.2 - Design

The optimizer block will convert the deep learning model to a quantized onnx model representation. Such a change will allow the model to become both faster and more memory efficient. The black box for the optimizer block is shown in figure 7. The optimizer has a single

input interface (otsd_optmzr_code). This interface consists of a .zip file with a DL model to convert and calibrate data for the quantization phase of the optimizer.

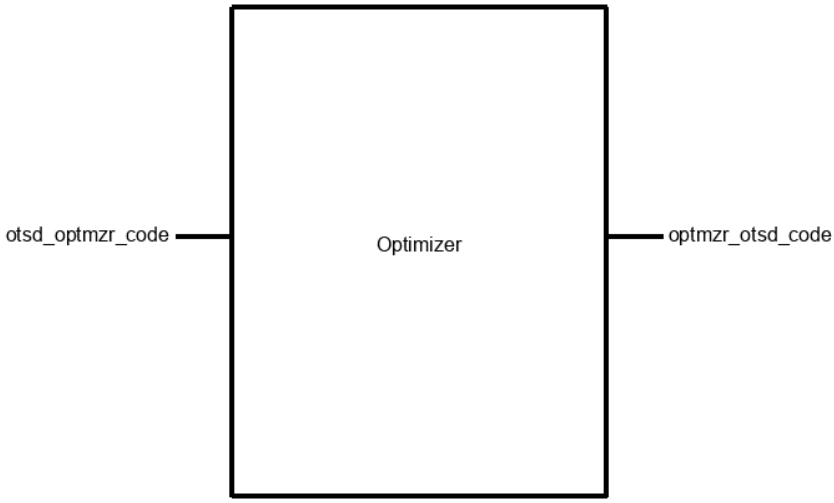


Figure 4.2.2.1: Black Box for Optimizer.

The diagram shown in figure 8 demonstrates the phases for converting the DL model to a quantized onnx format. First the DL learning model contained in the .zip file is converted to an onnx format. Following this step there is a pre-processing stage which optimizes the onnx graph and prepares the model to be quantized. Finally in the quantization stage the optimized onnx model is quantized using calibration data to produce the final output, a quantized onnx model of the original DL model.

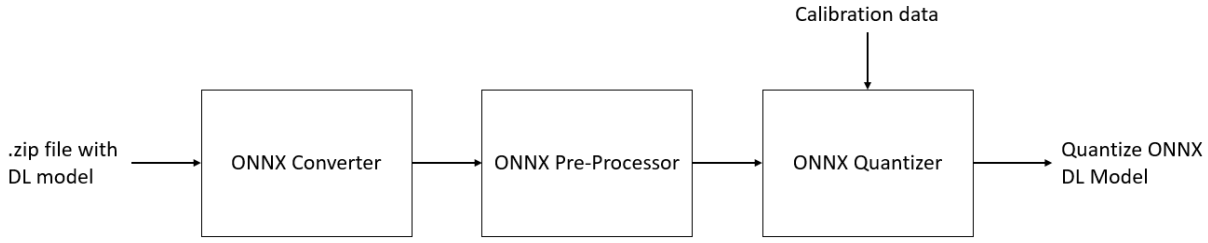


Figure 4.2.2.2: Internal subcomponents of Optimizer

4.2.3 - General Validation

Our system requires a deep learning voice cloning model to run on an embedded system with at most 4GB of memory. The model running on this embedded system will need a real-time factor of 1. A real-time factor refers to the amount of time required to generate one second of audio. The optimizer block contributes towards this goal by cutting the model size in half while also providing a 5 percent speed up without noticeably impacting the model's generated audio quality. This is done by converting the DL model to a quantized onnx model. As an onnx model our DL model will be more versatile and faster in a wider range of hardware architectures.

4.2.4 - Interface Validation

Table 4.2.4.1: otsd_optmzr_code Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: A .scp containing speaker ids and the path to an x-vector associated with each speaker.	The DL model is calibrated during quantization with calibration x-vector data	The DL model was trained with data in the format of a .scp containing speaker ids and the path to an x-vector associated with each speaker
Other: A .zip file containing: a .pth file of the deep learning model checkpoint, a .yaml file of the zip file structure, and a .yaml file of the deep learning model configuration.	After training the DL model on espnet the training script produces a packed .zip file with a .pth file of the deep learning model checkpoint, a .yaml file of the zip file structure, and a .yaml file of the deep learning model configuration [1].	The espnet training script produces a .zip file with the deep learning model checkpoint, a .yaml file of the zip file structure, and a .yaml file of the deep learning model configuration [1].
Other: A text file containing speaker ids and their associated spoken text.	The DL model is calibrated during quantization with calibration x-vector data	The DL model was trained with data in the format of a text file containing speaker ids and their associated spoken text.

Table 4.2.4.2: optmzr_otsd_code Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: The quantized .onnx file is at least 5% faster than the original .onnx file.	The optimization should speed up the inference speed of the model	An onnx model is more optimized for hardware than a pytorch model and 8-bit int quantized operations are cheaper computationally than 32-bit fp operations.
Other: A .onnx file of the quantized deep learning model.	The inference script requires a .onnx model file	The optimizer creates a .onnx file
Other: The quantized .onnx	The embedded system has a	A completely 8-bit quantized

file is at most half the size of the original .onnx file and is under 100 MB.	maximum of 4GB of memory and therefore requires a model with a low memory footprint.	model is theoretically a quarter of the size of a 32-bit fp model.

4.2.5 - Verification Process

1. From the LibriTTS dataset [2],aquire a .scp containing speaker ids, the path to an x-vector associated with each speaker and a text file containing speaker ids and their associated spoken text (`otsd_optmzr_code`).
2. From the espnet training [1] process acquire a .zip file containing: a .pth file of the deep learning model checkpoint, a .yaml file of the zip file structure, and a .yaml file of the deep learning model configuration(`otsd_optmzr_code`).
3. Run the code for the optimizer block after specifying the file paths to the required components in steps (1) and (2). The code will produce a quantized onnx model (`optmzr_otsd_code`).
4. Compare the sizes of the quantized .onnx DL model and the original DL model to determine if the quantization reduced the model size by half.
5. Create a test script that uses the quantized onnx model and original model to clone a provided voice.
    a. The script will prompt the user for the file path to a .wav file containing a spoken voice and a text input on the command line.
    b. The script will use the inputs and the deep learning model to generate an output speech waveform in a .wav file.
    c. The script will output the real-time factor from both the quantized .onnx model and the original model.
6. Run the Test Script
    a. Determine if the real-time factor is 5 percent lower for the quantized .onnx model than the original model.

4.2.6 - References and File Links

[1] Watanabe, S., “ESPnet: End-to-End Speech Processing Toolkit”, arXiv e-prints, 2018.

[2] Zen, H., “LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech”, arXiv e-prints, 2019.

4.2.7 - Revision Table

Table 4.2.7: Revision Table

Author:	Date:	Change:	Reasoning:

Matthew	3/12/23	Created and Drafted the Block Validation for the Optimizer block	Project Document Assignment due

4.3 - Speaker Block

4.3.1 - Description

The Speaker block will consist of an audio amplification circuit used to produce the cloned voice for the user to hear. Contained within this block will be an audio amplifier paired with a potentiometer for user volume control. The audio will then be output from a speaker built into the system. There will only consist of one input being the cloned voice audio, and the output will be the same audio just amplified and volume adjusted. Thus input will be an analog signal generated by the microcontroller and passed to the amplifier circuit to be output as an analog signal through the speaker to the environment. This block does not specifically interact with the final audio wave, but merely amplifies and outputs it for a user to easily understand and loud enough to be heard in a room with moderate ambient noise. The blocks circuit will also be integrated with other Printed Circuit Boards (PCBs) containing the input and output peripherals for each hardware block in the entire system.

4.3.2 - Design

For the design of this speaker block, the main focus is to be able to properly amplify the audio signal being produced by the microcontroller. The heart of this system is built around the LM386 audio op amp chip. This chip can be customized to different gain levels of our audio signal, which is what will be done in order to produce an audible signal the user can hear. The block will also feature a potentiometer for user controlled volume. Since we are amplifying an audio signal of only a voice, we do not need large amplification or focus on high quality acoustics and bass. Therefore, the circuit will be designed using the standard internal gain of the LM386 chip, coupled with a simple DC blocking capacitor to minimize noise through the speaker. The circuit will also be designed with a potentiometer to provide further volume control over the standard internal amplification for the user depending on their hearing needs and surrounding audio levels.

Figure 9 below is the block box, containing only two interfaces. The LM386 chip will receive an audio signal from the microcontroller as input, and output an amplified version of the same signal to the environment through a speaker.

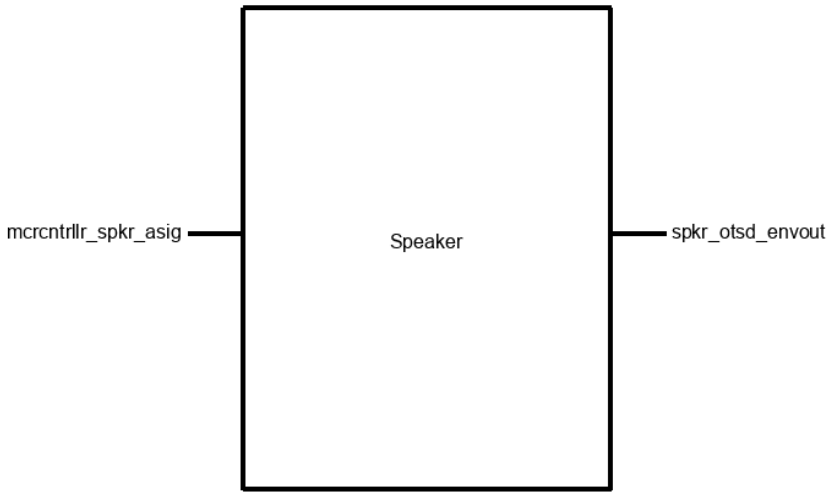


Figure 4.3.2.1: Black Box Diagram for Speaker Block

Because the input of this block is a digitized audio signal coming from a microcontroller, a DAC converter (coupled to a USB adapter) will be used to convert the microcontroller digital signal into an analog signal to play through the speaker. Along with a DAC converter, an audio jack will be used to complete this interface to effectively pass the input into the speaker's amplifier. The reason for this design was to ensure that the audio signal would be clear, as a noisy signal being amplified would also amplify noise and have an undesirable output. This design choice also allows our input interface to be USB, which is much more compatible with many types of microcontroller blocks.

Figure 10 represents the hardware schematic of the entire circuit for the Speaker Block. At the input of the LM386 op amp, there is a 10k resistance potentiometer. This potentiometer adjusts the resistance of the input, in turn reducing the voltage drop coming from the audio input, which translates to how much the signal is amplified (volume). The output of the LM386 chip has a simple filter to block unwanted noise and DC signal to the speaker. Since the LM386 op amp only needs minimal amplification, the circuit is currently designed for an internal gain of 20. Using pins 1 and 8 can allow for an increasing gain factor if needed, however this application can be satisfied with the LM386 internal gain alone. This gain translates to approximately 26dB, which paired with the input potentiometer shall amplify the signal to a desired maximum noise level (and minimum noise level).

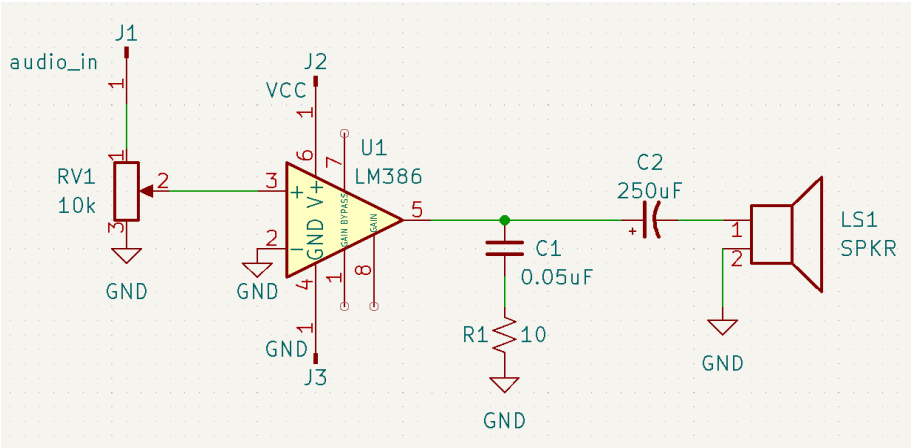


Figure 4.3.2.2: Hardware Schematic

Figure 4.3.2.3 depicts the PCB footprint for the given hardware schematic. The footprints selected were all standard sizing for the capacitors, resistor, and connectors, erroring on the side of larger footprints to ensure there is room to place each component on the PCB. Another design selection was creating a two sided PCB. As seen below, the potentiometer is mirrored and highlighted a different color. This is representing that it is being reflected on the other side of the board. The purpose for this design is that the user peripherals can be facing upward, and the rest of the circuit and components can be hidden from the user. This avoids obstructing the users ability to access the potentiometer, as well as protect the components. Using male pin connectors for the interfaces was another design choice as this system is interfacing with a microcontroller, it is the easiest method of connection for the given design. The Vcc and GND pins will be used to power the LM386 chip at a nominal voltage of 5 volts.

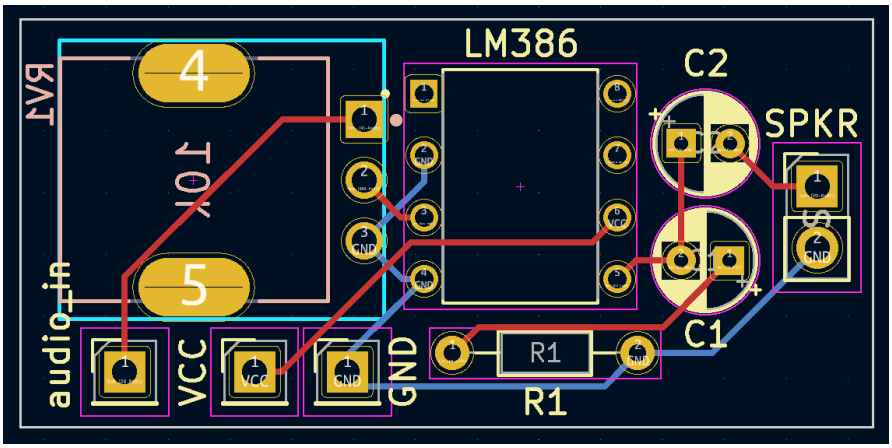


Figure 4.3.2.3: PCB Footprint

4.3.3 - General Validation

The objective of this individual block is simple, thus the design implemented to achieve such a task achieves this objective effectively. The input of this block receives a fully constructed audio signal already, and only needs amplification through a device capable of producing the cloned

voice audibly. For a simple task the design remained simple, only adding complexity where desired to improve the user experience.

One major driving factor for this block was availability of parts and engineering time. All components for this block were already in possession or easily accessible. Having access to every component early on takes away the extra steps of seeking out parts online, ordering parts, and waiting for parts. It also reduces the cost of the block to $0. Furthermore, the engineering time is also significantly reduced. This quick access allows for immediate assembly, testing and verification of the system. Since this system can be tested independently and the parts are accessible, there can be confidence when ordering a PCB with a longer lead time without fear of losing that time because of a missing verification process.

Another important driving factor of the schematic design itself is the reputation of the circuit and components themselves. The LM386 audio op amplifier chip is a very reputable op amp, with an extensive datasheet providing useful resources and specifications to create a simple, robust, and efficient circuit for the needs of this speaker block. The circuit was designed with guidance from the LM386 datasheet, including consideration of the specifications and ratings of various components. This provides further support for the design choices made for this block.

Even though this block is focused on the design and validation of predominantly amplifying the desired audio signal, the audio input interface into this block is a critical interface to design effectively. Various microcontrollers have Analog Digital Converter (ADC) and Digital Analog Converter (DAC) support, however not all do. In order for this interface to be properly validated, one of the design choices was to turn this audio input interface into a USB interface, so it can be compatible with almost any input device. This design choice helps ensure the system will effectively produce a clean audio output, mimicking the deep learning code’s reproduced cloned voice.

Another piece of validation in our design is having a quick alternate solution towards fixing volume and amplification issues. If the designed circuit gain is too low or too high, there are solutions that extend off the existing design to boost/reduce our gain if need be. This alternate solution gives the design room for improvement and fine tuning to truly meet our expected decibel range requirement. Along with this, this alternate solution takes advantage of the pre existing circuit, saving valuable time and resources by avoiding a complete redesign of the system.

When looking at the validation for the PCB, one important driving factor was again the LM386 datasheet for common footprint layouts. For best performance, keeping the analog traces away from any digital traces mitigates any possible distortion or noise altering our analog signal. Another important consideration was to keep the output trace to our speaker small (again to reduce as much noise as possible from our analog signal). Since the goal of this block is to amplify the audio signal being received, it is important to mitigate any interference and noise as it could be amplified and distort the final cloned voice output signal.

4.3.4 - Interface Validation

Table 4.3.4.1: mcrcntrllr_spkr_asig Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Vmax (Supply): - GPIO power pin 5V	This is the designed supply voltage for the system	For the LM386N-1 recommended operating cond: - MAX V is 12V. - MIN V is 4 V.
Vmax (audio signal): - 0.9V peak to peak	This is the expected maximum voltage for an audio signal line level [2].	LM386 is designed for low voltage amplification, and the maximum voltage remains as a low voltage.
Other: uC internal volume level: - 10% of maximum volume output	Properly control volume amplification and potentiometer control	Speaker block will only read the audio jack as input.

Table 4.3.4.2: spkr_otsd_envout Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
dB_min: <30 dBA @ 3 feet away	At lowest volume there should be quite noise produced, no quieter than 20dB	For the LM386N-1: - Internal Gain = 20 (approx. 26 dB)
dB_max: >70 dB @ 3 feet away	At max volume its output should be capped at a certain level (60dB is regular conversation at 3 feet away)	For the LM386N-1: - Max Gain = 200 (approx. 46 dB)
Audible: 9/10 users can make out the audio output at min and max volume in a room with moderate ambient noise	Provide a sufficient audio signal for users to easily get from the environment.	For the LM386N-1: - Using an internal gain of 20 will be a minimum of 26dB (typical for quiet

		conversation).

4.3.5 - Verification Process

1. Assemble the given circuit in Figure 2 on a breadboard/protoboard
2. Connect the audio_in input (mcrcntrllr_spkr_asig: audio signal) audio jack to an audio player of your choice
    a. This can include a computer, a script that outputs an audio signal, etc.
    b. Set the audio players internal volume to 10% its max volume (mcrcntrllr_spkr_asig)
3. Connect the power supply (mcrcntrllr_spkr_asig: supply) to the circuit starting with 5V
    a. Connect a 5V power pin and GND from a uC to supply power to the circuit
4. Play the audio signal from the audio player chosen in step 2
5. Using an oscilloscope, measure the voltage across the audio input
    a. Observe the waves peak to peak voltage as audio plays and record its maximum peak to peak (mcrcntrllr_spkr_asig: audio signal)
6. From 3 feet away measure the decibel level the speaker outputs while audio is playing (spkr_otsd_envout)
    a. (Spectrum Analyzer app on a phone works well for measuring decibels)
    b. Measure the decibel level with the potentiometer at minimum volume
        i. Minimum here is the lowest potentiometer value that produces the audio wave, setting it to lowest setting blocks the audio signal
    c. Measure the decibel level with the potentiometer at maximum volume
        i. Maximum here is the highest the potentiometer can be turned
7. Observe the decibel level from each recording to ensure the minimum and maximum levels are as desired
8. Repeat step 6 for 10 users and observe how many users can hear the audio at each volume level
9. Disconnect the power supply, audio input signal, and any other peripherals used.

4.3.6 - References and File Links

[1] Texas Instruments, “LM386 low voltage audio power amplifier datasheet (rev. C).” [Online]. Available: https://www.ti.com/lit/ds/symlink/lm386.pdf?ts=1612350151545.

[2] “Line level,” *Wikipedia*, 25-Dec-2022. [Online]. Available: https://en.wikipedia.org/wiki/Line_level.

4.3.7 - Revision Table

Table 4.3.7: Revision Table

Author:	Date:	Change:	Reasoning:
Connor	1/16/23	Created Block Validation Draft	Block Validation Assignment due
Connor	1/17/23	Updated Design Section and Validation Sections	Assignment due
Connor	1/18/23	Finalized document Word count: 1519	Assignment due
Connor	2/09/23	Updated Verification Plan. Added further details. Word count: 1798	Assignment due
Connor	2/10/23	Updated Gen. Val. section to include input interface info Word Count: 2141	Assignment due
Connor	2/16/23	Update Interfaces and Verification process	Final block demo preparation

4.4 - Display Block

4.4.1 - Description

The Display block will be an additional peripheral for the user to receive system messages as well as view the text input they type into the system. The screen will be primarily used to display messages and text, so the screen shall be small enough to be handheld to fit our enclosure, but also large enough to be easily read from 3 feet away. The screen will communicate with the internal microcontroller to display a GUI for the user to interact with.

4.4.2 - Design

Due to the complexity of designing a viable screen disable from scratch, this block was implemented into our system using a pre-purchased module. This design choice allows for a reliable screen to display the interactive GUI the user will interact with. Even though the module itself was purchased, there are still design constraints related to integrating the screen into our given system as the display module. A few of these design constraints consist of the size of screen, compatibility with the systems given microcontroller, and the user experience visibility.

Figure 4.4.2 below shows the black box diagram of the display block. The screen will interface to the systems microcontroller and will output the desired GUI display to the environment for the user.

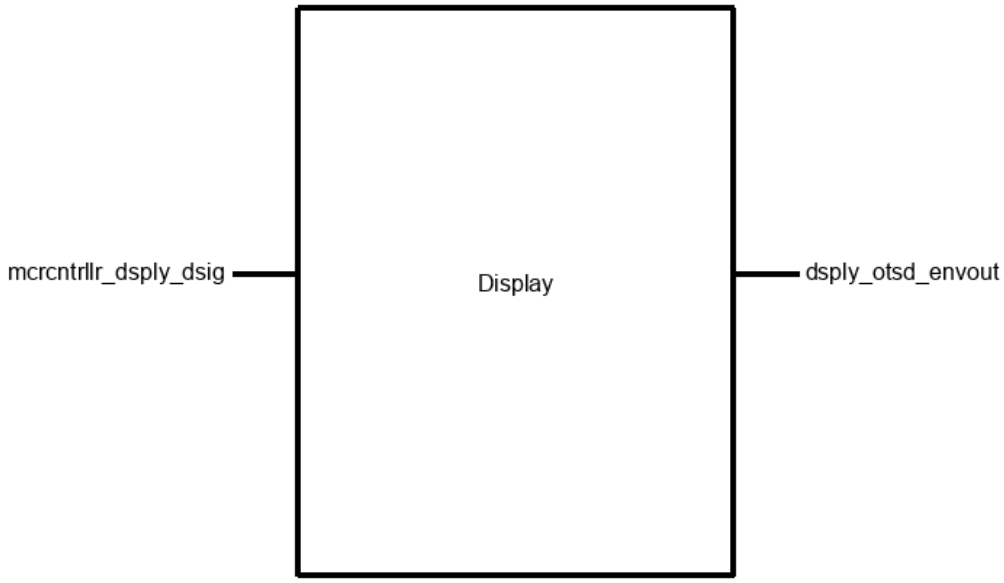


Figure 4.4.2: Black Box Diagram of Display Module

As mentioned above, the design constraints for this system are related to the integration of the screen to the rest of the system. For the input to the display, it will require HDMI communication coming from the microcontroller to interpret and display the desired interface. This input also contains the responsibility of powering the display module from the microcontroller as well. This will be done through USB, since it is the most versatile connection regardless of microcontroller choice.

The output of this module focuses on the user experience and is designed within the constraints of the entire system to give the user the best possible experience. The most important property for a screen's output is user visibility. To ensure quality visibility on a smaller system, the resolution for the screen will be set to an optimal 800 by 480 pixel display. This resolution fits the physical dimensions of the screen the best. Since the system is meant to be handheld, the screen will be relatively small, but with this desirable resolution the size of the screen shall not hinder the ability to view, read, and interact with the screen. Another factor to consider with visibility on a small screen is brightness. It is important the brightness is properly set for the user to have an easy experience in lighter or darker rooms without extra assistance or interaction. To achieve this, the brightness is set to an optimal lux level of no more than 400 lux given the size and distance a user would be interacting with the module.

4.4.3 - General Validation

Due to the display being a purchased module, the design constraints were fairly restrictive and thus resulted in the design choices to be made closely in relation to the interfacing block. The

design choices made above fit the needs of the system overall and satisfy all the requirements needed for interacting with a microcontroller.

Because the type of microcontroller may differ, this block was designed to be easily adaptable to any microcontroller. The screen module purchased communicates over HDMI, and has various adapters in the event the interfacing microcontroller does not have a direct HDMI connection. The screen also is powered via USB, which allows for versatile adaptation if needed, and uses one of the more common interface connectors.

For this display block, the role of the module is straightforward and was thus designed to meet these needs of the system. The design choices made for this block keep the system within a handheld size, and provide the user an easy to interact interface. The pre-purchased module even has designed mounting holes to connect to the Raspberry Pi 4 microcontroller (which is the top choice for microcontroller in the system). This ensures the interfaces between the microcontroller and screen are limited in size, and maximize the user experience.

4.4.4 - Interface Validation

Table 4.4.4.1: mcrcntrllr_dsply_dsig Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: - Display through HDMI connector	Required interface for screen display	Purchased module contain an HDMI to mini HDMI adapter connecting uC to screen
Other: - Powered through USB connector	Required interface for screen display	Purchased module contains a USB to micro USB adapter connecting uC to screen power
Other: - Touch screen feature enabled	Include extra user interactive buttons through GUI	Purchased module enabled touch screen through power interface

Table 4.4.4.2: dsply_otsd_envout Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**

Light: - Produces less than 400 lux from 6 inches away with all white screen	Provide optimal lighting for visibility and strain	Screen module can preset a controlled brightness to illuminate screen
Other: - Resolution: 480 by 800 pixels	Optimal resolution given physical size of screen	Per datasheet, hardware resolution is set to 480 by 800 pixels
Other: Visibility - Visible by 9/10 users during indoor midday lighting	Further verify the size and visibility of the screen is optimal for interacting users	Based on judgment and datasheet recommendations for optimal display.

4.4.5 - Verification Process

1. Connect Screen module to Raspberry Pi 4 via HDMI to mini HDMI
2. Connect Screen module to Raspberry Pi 4 for power via micro USB to USB
    a. Connect power to the ‘Touch’ female port on screen module
3. Power on Raspberry Pi 4
4. Raspian OS desktop should appear on screen
5. Verify resolution by going to settings -> display preferences -> resolution
    a. Resolution should be set to 800 by 480 pixels
6. Using a lux meter (various phone apps will work) measure the lux output from the screen from 6 inches away
    a. This will verify the optimal brightness of the screen
7. Finally, provide a small script of text on the screen for users to read back.
    a. 9/10 users shall be able to clearly read the text on the screen from an appropriate distance when holding the module in their hands

4.4.6 - References and File Links

[1] “Waveshare 4.3inch HDMI LCD (b),” *4.3inch HDMI LCD (B) - Waveshare Wiki*. [Online]. Available: https://www.waveshare.com/wiki/4.3inch_HDMI_LCD_(B)

[2] L. Han, H. Zhang, Z. Xiang, J. Shang, S. Anjani, Y. Song, and P. Vink, “Desktop lighting for comfortable use of a computer screen,” *Work (Reading, Mass.)*, 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7902945/#:~:text=According%20to%20EN%2012464%20%5B10,75%20lux%20and%20150%20lux.

4.4.7 - Revision Table

Table 4.4.7: Revision Table

Author:	Date:	Change:	Reasoning:
Connor	3/12/23	Created and Drafted the Block Validation for the Display block	Project Document Assignment due

4.5 - Text Interface Block

4.5.1 - Description

The text interface block is a sub-system that will allow the user to input text using a 10 button letter interface to type text along with 1 button to delete a letter and another button to confirm the text string. The string of letters will send the information to the microcontroller where it will be used to tell the cloned voice what to say. The block will be built using a custom PCB using hardware to debounce buttons, communicating with the microcontroller via the Serial Peripheral interface, and having extra buttons for submitting and deleting text. The text will be displayed on the screen so the user can view what they are inputting using the text input interface.

The 10-button letter interface allows the user to input any letter in the English alphabet by having each button represent 3 letters, where the letters are differentiated by the timing of the button pressed. A single press enters the first letter correlated to that button, two quick button presses enter the second letter and the same follows for the third letter. The debouncing of the buttons is completed using a resistor and capacitor circuit for each button. The circuit then utilizes an inverter and shift register to store and synchronize the data to send to the microcontroller via SPI.

4.5.2 -Design

The design for the text input interface focuses on the usability of this interface with users. This means that the interface should be as quick as possible in transferring data from the buttons to the microcontroller. This is especially important with pressing the button several times in quick succession, where each press changes the letter. The goal of the design is to have it where as long as the button is pressed 1-3 times within a short period of time around .4-1 second, the user can change the letter input. Beyond that range of time, the button press(es) acts as a character input and moves to the next character. Within the circuit design, hardware debouncing is implemented to reduce processing time for button mechanical button bouncing. It is using a simple circuit that uses a voltage divider with a capacitor to smooth out the bouncing voltage signal from the mechanics of a push button. This means very quick inputs can be registered to improve the usability of the text input, along with timing for the 10-button keyboard also allowing for a smaller and more compact interface.

Figure 4.5.2.1 shows the block box diagram along with the interfaces for the text interface. The block diagram encompasses the hardware of all the buttons along with the resistor and capacitor debouncing filter, inverter, and shift register to communicate the data over SPI. The left interface of the block simply represents user input, where a user would press the buttons and specifies the rough timings for button presses to be registered within the microcontroller. The right “txt_ntfc_mcrcntrllr_dsig” interface describes the digital signal and communication between the shift register and the microcontroller. For this communication, voltage is defined as being 3.3 volts as the supply voltage as well as the operating voltage representing a *high* value across the circuit.

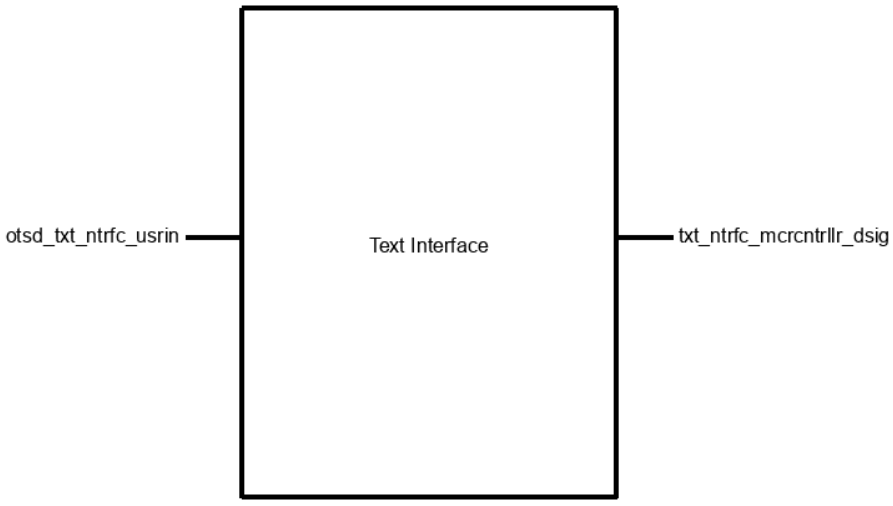


Figure 4.5.2.1: Block box Image with interfaces

Figure 4.5.2.2, shown below, displays the schematic, with the supply voltage passing through the resistor-capacitor debouncing network to then be inverted and transferred via SPI using a shift register. The 4HC04 is used as the inverter to have an active high with five volts when the button is pressed, and then low, or zero volts, when the button is depressed. This information is then loaded into the shift register using the 74HC165 and communicated to the microcontroller with SPI protocol.

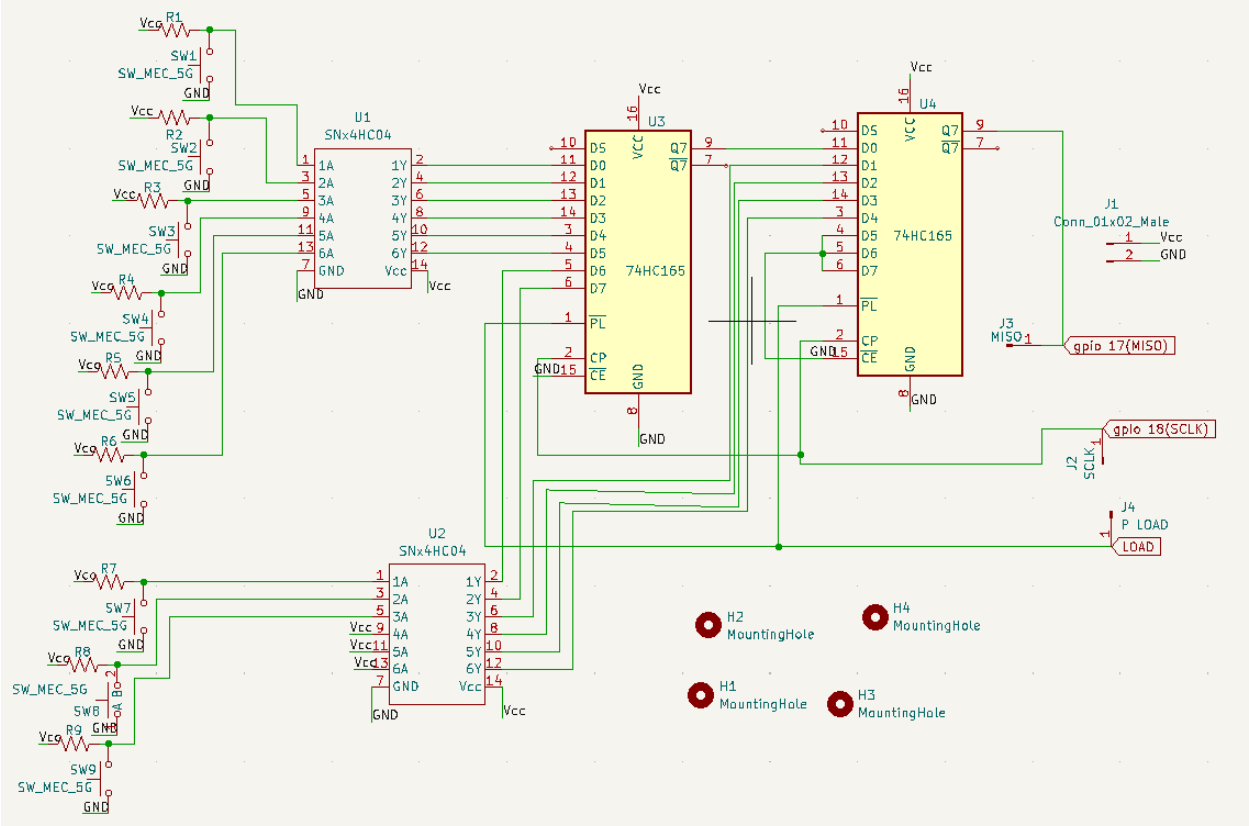


Figure 4.5.2.2: Schematic design of text interface

The footprint for the buttons will be using already established footprint files for the Cherry MX low-profile switches. This ensures the connection fits between the pushbuttons and the switches. The Cherry MX mechanical switches were chosen to provide a satisfying typing experience while also allowing it to be placed on a custom PCB with either the pins soldered directly to the board or using wires.

4.5.3 - General Validation

To accomplish a user interface for entering text within a small form factor, and fast and effective usability, the 10-button letter keyboard is utilized to allow for user text input to the microcontroller. A layout of three buttons per row and four rows, allowing for 10 buttons to correspond to 3 letters covering the English alphabet and 2 buttons to confirm text and delete a letter. The first button can be pressed one time for the letter ‘A’, two times in quick succession for the letter ‘B’, and three times in quick succession for the letter ‘C’. This is the process for each of the buttons. The design of the circuit and software controlling the communication should be fast so that there is a distinction in times for pressing a button twice to move to the next letter and wanting two of the same letter. The specific timing and ranges for switching letters and typing the next letter will be determined in the software, with the design of the hardware being as fast as possible and eliminating unnecessary delays such as debouncing in software. For debouncing in hardware, the values of the resistor and capacitor can be determined by

capacitor discharging and charging equations[1]. With testing, the timing can be determined with an oscilloscope, and adjust the values if need be. Once the values for the pushbutton hardware debouncing have been determined, the inverter and shift registers must be tested in series with the debouncing, along with a microcontroller to ensure the correct data is being communicated from the shift register. The inverter being used is the SNx4HX04[2] simply due to the availability of the IC, and it is a hex inverter so two chips cover all inputs. The shift register being used is a 74HC165[3], more specifically it is an 8-bit parallel in, serial out shift register. This register is something the group has worked with and allows for extremely quick asynchronous input capture, and output over SPI allows for simple but fast communication. The buttons themselves have not been determined as well as the PCB hasn't been completed as of yet, but this section would describe what they are, the reason, and so on. With designing the PCB, one design consideration and concern is the size of the packages for the ICs. For ease of assembly and replacement of parts, the team has decided to use through-hole components. This will take up a significant amount of space compared to surface mount components, however, the availability of parts and using what the team already has is the reason for using through-hole components. One concern is with the availability of the shift register, as this component is something that each team member has from a previous class, however, if these ICs are broken or lost, the availability online is sparse. An alternate solution for the 10-button interface if the timing is too difficult, would be to have the user type one character at a time, then press enter button when the user has gotten to the desired character. This would be simpler but less enjoyable to use and frustrating if a longer amount of text is desired from the user.

4.5.4 - Interface Validation

Table 4.5.4.1: otsd_txt_ntrfc_usrin Interface

Interface property: otsd_txt_ntrfc_usrin	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
A Button press should reach the microcontroller within 30ms	Mechanical Switches will often rattle between open and short circuits over small periods of time. 30ms allows for some rattling and the use of RC debouncing hardware to be accommodated and achievable. The propagation and transition times of both of the chips are within 1uS so adding 10ms to the total time allows ample space for timing.	Using hardware debouncing delays from the mechanical switch can range, but conservatively is 20ms. The Hex inverter has a propagation delay of 95ns at 2V and a transition time of 75ns, both according to Section 6.7: Switching Characteristics[3]. The 74HC165 shift register datasheet provides that the transition time is 75ns maximum at 2 volts Vcc, with propagation delay times being no more than 200ns for shifting data from registers to

		output, found in section 11 table 7: Dynamic Characteristics.
Debouncing RC filter delays button push by no more than 20ms between pressing the button and the hex inverter receiving the signal	A maximum debouncing time is needed to complete calculations for determining values of the RC filter, so assuming a conservative push button debouncing time enables achievable values without impeding the signal propagation time.	From reference [1] the equations used for debouncing hardware with the resistor and capacitor circuitry begins with the assumption of debouncing time for the mechanical switch used, the rest of the values follow based on the timing assumption to resolve the assumed value.
The Cherry MX Mechanical Switches are used for 12 buttons	This property value is necessary as it defines what needs to be measured for typical debouncing time, and the switches especially were chosen to provide a cleaner user interface and satisfactory interaction with the pushbuttons.	The amount of time for the signal for the rattling of the mechanical switch was measured using an oscilloscope as well as the purchase of the switches has already been made and received.

Table 4.5.4.2: txt_ntrfc_mcrcntrllr_dsig Interface

Interface property: txt_ntrfc_mcrcntrllr_dsig	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
RC debouncing fall time is no greater than 15ms	The RC circuitry fall time is an assumed value that is then used to calculate the rest of the values for both the resistor and the capacitor. The time is estimated to be around the worst-case scenario for a mechanical switch rattling from high to low after being pressed.	Based on the calculations from the equation found in the design details, which was originally sourced from *The Ganssle Group-debouncing pt2*[1]. This calculation and theory have been confirmed with an oscilloscope.
Active high: 3.3V, represented as logic 1 in code	The microcontroller used for the project provides 3.3 volts from its GPIO pins. This is the voltage then used to power the circuit, and what is required to send over SPI	The voltage of 3.3 volts is supported by each chip used: ● SNx4HC04 inverter has a maximum rating for input and output voltage as Vcc which

	from the shift register so the microcontroller can read the data. The high is assigned to 3.3 volts for ease of understanding of other teammates.	is 7 volts, section 6.1: Absolute Maximum Ratings[2] ● 74HC165 has a maximum input and output voltage of 7 volts as well found in section 8 table 4: Limiting Values[3]. 3.3 volts has also been measured with a multimeter as input and output of each junction between chips and mechanical switch
Low: 0V, represented as logic 0 in code	With high being defined as 3.3 volts, low is defined as the complement of that being 0 volts. Logic 0 is then defined as 0 volts as the complement of the already defined high value of 3.3 volts being logic 1 in software.	The voltage of 0 volts is supported by each chip used: ● SNx4HC04 inverter has a minimum rating for input and output voltage of 0 volts, section 6.3: Recommended Operating Conditions[2] ● 74HC165 has a minimum input and output voltage of 0 volts found in section 9 table 5:Recommended operating conditions[3]. 3.3 volts have also been measured with a multimeter as input and output of each junction between chips and mechanical switch

4.5.5 - Verification Plan

a. Create a circuit on a breadboard to test the functionality of the RC filter and test for the timing to determine resistor and capacitor values.
b. Measure voltages at each stage of the circuit to ensure correct voltages.
c. Measure data communication and timings on the Oscilloscope at the output of shift register
d. Connect schematic to Atmega128 microcontroller to test SPI communication

e. Verify correct data is being received from the circuit by printing the binary value or equivalent form(such as decimal, hex, etc.) of buttons pressed to the LCD screen on Atmel atmega128.

4.5.6 - References and File Link

[1]“8-bit parallel-in/serial out shift register - nexperia,” *8-bit parallel-in/serial out shift register*, Sep1-2021. [Online]. Available: https://assets.nexperia.com/documents/data-sheet/74HC_HCT165.pdf. [Accessed: 12-Feb-2023].

[2]The Ganssle Group, “A guide to debouncing - part 2, or, how to debounce a contact in two easy pages, by Jack Ganssle,” *Debouncing, hardware and software, part 2*. [Online]. Available: http://www.ganssle.com/debouncing-pt2.htm. [Accessed: 11-Feb-2023].

[3]“SN74HC04-Q1 automotive hex inverters - texas instruments,” *SNx4HC04 Hex Inverters datasheet (Rev. H)*, Dec-1982. [Online]. Available: https://www.ti.com/lit/ds/symlink/sn74hc04-q1.pdf. [Accessed: 12-Feb-2023].

4.5.7 - Revision Table

Table 4.5.7: Revision Table

Author:	Date:	Change:	Reasoning:
Micah Janzen	1/19/23	Created document and outline. Started block description	Block validation assignment completion
Micah Janzen	1/20/23	Completed each section of the outline.	Assignment is due
Micah Janzen	1/27/23	Updated schematic image	Feedback from student review
Micah Janzen	2/4/23	Updated Design Description	Slight changes to the design of block
Micah Janzen	2/11/23	Updated Design Description, interface Validation, general validation and reference files and links section	Matched interface validation to the website and then design description to interface. Updated information for the general validation. Changed references to IEEE format

4.6 - Microphone Block

4.6.1 - Description

The microphone block is a simple peripheral needed for recording the user's voice. The block is a purchased module, with the important factors of the microphone being that it is small and somewhat good quality. The microphone purchased was the ESTIQ Professional Lavalier Lapel Microphone, which provided a cheap, small and decent quality option for the microphone. The microphone connects via a 3.5mm audio jack and allows for a plug and play interface. The microphone is omnidirectional, allowing for the user to speak from any orientation about the system, however it only picks up the user's voice when relatively close, around 1-3 feet from the microphone. This is a benefit as it can reduce and eliminate noise from the room, such as when the device is being showcased with a large amount of noise in the room.

4.6.2 - Design

While designing a microphone with an amplifier circuit is very possible, to ensure the smallest possible form factor for our project, we went with a pre-purchased module since we had enough blocks that were built. For the design of the microphone, the size had to be small, so that it could be placed inside an enclosure that was smaller than a typical laptop. The only other design requirements were that the quality of the microphone or the audio from the microphone be average quality. The ESTIQ microphone purchased fits those needs as it provides great quality when in an isolated environment, and decent quality with background noise, and the size of the lavalier microphone is very small.

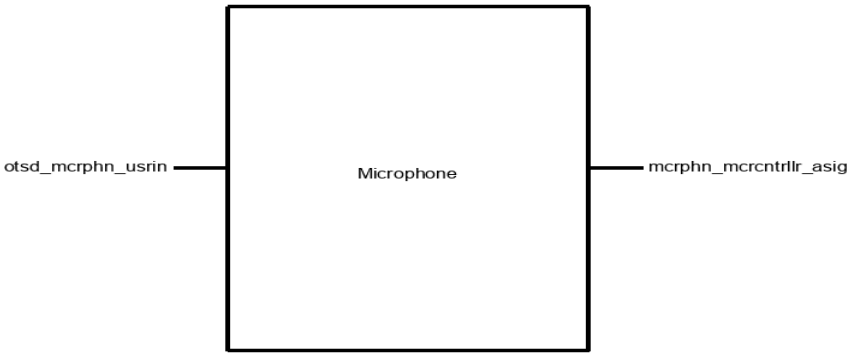


Figure 4.6.2: Black Box Diagram of Microphone Module

The input of the microphone comes from the outside sound waves, ideally the microphone will only pick up on close waveforms and the user's voice, but realistically there will be noise from other people in the room or just random background noise. The design for the input doesn't require a perfect input signal as the models for voice cloning are able to deal with noise and isolate the human voice for the most part. This allows for less strict design requirements for the microphone and not needing an expensive microphone. The output of the microphone could've been either USB or an audio jack. For the output speaker module, the team had purchased an audio adaptor that had an input and output audio jack port, so the decision was made to stick

with a simple 3.5mm audio jack for the output interface of the block. This reduced size and utilized space are already allocated for the speaker output adaptor port.

4.6.3 - General Validation

The microphone purchased fits all the needs of the system and block while being cheap. The audio input is able to be captured from a close distance, cutting out most background noise. Testing the microphone showed that the audio when close to the user's mouth, or under 1 foot from the user's, was able to produce around -10 to 5 dB picked up. This when played back on the computer used for testing was able to be clearly heard and understood. The size of the microphone is very small providing ample possibilities for placing the microphone within the enclosure no matter what design we choose for the enclosure. These two restraints were the largest factors to consider and the microphone is able to pass those requirements while only being around $15 on Amazon, which provides the ability for more to be purchased with a quick turnaround time if need be.
The interface between the microphone and the system reading the audio is extremely quick and easy to set up. The connection from the microphone is a common 3.5mm audio jack, though there is an extension that changes the audio jack from a 4-band audio jack to a 3-band that is needed to work with certain systems which may be a problem for plugging in with the enclosure. A similar but slightly different microphone may be needed to purchase to ensure that the audio jack doesn't stick out too much from the embedded system's audio jack port to make sure the connection to the system isn't too large for the ideal enclosure we have planned.

4.6.4 - Interface Validation

Table 4.6.4.1: otsd_mcrphn_usrin Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: - The microphone produces a waveform that is no greater than 20 dB as displayed from ffplay on the terminal with microphone, somewhat close to mouth	Ensure that the audio from the microphone is too loud when close to the microphone and potentially peaking when replayed.	Tested using Audacity, recording audio with the microphone right next to the mouth when talking and then using ffplay within Linux to read the dB levels from the audio recording. Over several trials audio never went over 20dB
Timing: - Microphone can record audio	Required to make sure audio was consistent when recording to ensure voice	Testing over several trials provided consistent results for recording for periods of

consistently for at least 5 seconds	cloning model gets provided quality recording	time greater than 5 seconds.
Type: - Microphone can produce audio that is above -45 dB using ffplay on .mp3 file of recording when 1 foot from microphone.	-45 dB is too quiet of an audio level to hear to measure for being over this value ensures hearable audio from the microphone.	Tested using Audacity, recording audio with the microphone right next to the mouth when talking and then using ffplay within Linux to read the dB levels from the audio recording. Over several trials, audio never went under -45dB when speaking into a microphone from 1 foot away.

Table 4.6.4.2: mcrphn_mcrcntrllr_asig Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: - Resistance between the ground and channels of the microphone should be a value around 1.2 with 20K mode on DMM	Because it was measured with a DMM	It does because it was measured to be that value
Other: - The microphone connects to the device via 3.5mm audio jack	Specified in the description of the microphone from both the parent company who produces the microphone as well as Amazon	This audio jack is able to be plugged into any computer or device that has a standard audio jack that is 3.5mm
Other: - the microphone is able to create a .wav file when using audacity and exporting audio.	The microphone is specified on Amazon to be plug-and-play with most devices, meaning that audio should be recordable once plugged in.	Was able to record audio and play back audio using Audacity on my home computer.

4.6.5 - Verification Process

1. Connect a microphone to the computer via an audio jack, and record audio from the microphone using Audacity, make sure to record for longer than 5 seconds
2. Export audio to .mp3 file format and run ‘ffplay -f lavfi "amovie=name.mp3, asplit [a][out1]; [a] showvolume=f=.95:b=4:w=720:h=68:c=VOLUME [out0]"’ to produce window displaying volume
    a. Add r=.95 for slowing down if needed
3. Ensure values are within range
4. Measure resistance by connecting to the first and 3rd band of the audio jack with DMM

4.6.6 - References and File Links

A. R. T. H. U. R. D. E. K. K. E. R. SAVAGE, “DP,” *Amazon*, 2016. [Online]. Available: https://www.amazon.com/dp/B08C7CZQ6J?psc=1&ref=ppx_yo2ov_dt_b_product_details. [Accessed: 13-Mar-2023].

4.6.7 - Revision Table

Table 4.6.7: Revision Table

Author:	Date:	Change:	Reasoning:
Micah	3/12/23	Created and Drafted the Block Validation for the Display block	Project Document Assignment due

4.7 - I/O Control Code Block

4.7.1 - Description

The I/O Control Code block will consist of the code necessary for the host computer to be able to interact with our other blocks. This host computer is planned to be a Jetson Nano, which on top of possessing USB inputs, has a bank of 40 GPIO pins that will allow us to interact with our other components. The blocks to send and receive data from will include our text input interface, a microphone, a user interface, the voice cloning model, and a speaker. This will allow data to be able to flow freely within the overall system pipeline. With that in mind the first few things the user is going to interact with is either the text interface or the microphone to input a sample voice to clone. This block will convert the 16 bit inputs of our text interface, which will consist of a 12 button layout, to controls such as record and playback along with english text input. The text will be shared with the user interface in order to display what the user is typing. It will also be converting a microphone input to a usable file format for the voice cloning model. Once this voice sample is recorded it will provide it to the voice cloning model to start its processing. Once the user has specified their input text and hit process this code block will pass off a file path to a text file to the voice cloning model and allow it to compute and return a .wav file of the user

input. This code will then be able to playback the file out through to a speaker. Overall this block consists of a few smaller functionalities that link together our entire process.

4.7.2 - Design

As stated at the end of the description section this block will consist of more of a patchwork of sections that will hold the overall process of our project together. Separating it logically into smaller components it will include parts to communicate with an audio input, a user interface, the voice cloning model, and a speaker. As sheer speed of our data handoffs is not a large concern and more integrity and compatibility this block will be built off of python. This gives it a strong, and easy, interaction with our voice cloning model, the user interface, and the GPIO pins as our host computer, the Jetson Nano, was built to be close in operation to a raspberry pi, which has a large amount of support for python.

There are many inputs and outputs required for this block as it will tie together the internal code with the hardware inputs and outputs. A black box view showing all interfaces with a naming convention of source_destination_type is shown below in Figure 16.

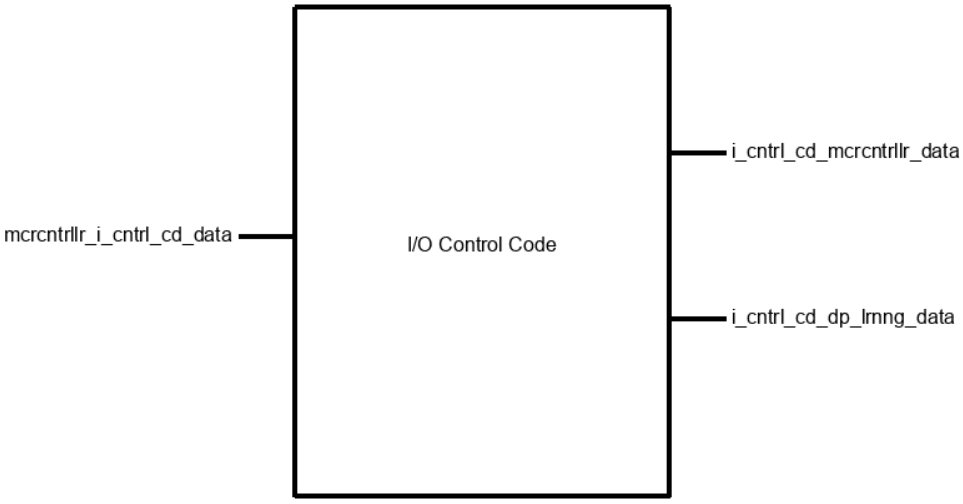


Figure 4.7.2.1: Black Box Diagram for I/O Block

An advantage of how our system is set up is many of these interfaces will not be used concurrently. This means that this block will mainly need to focus on one function at a time, freeing up computational power to just focus mainly on its current process. A typical user will either enter their text input or voice sample first. Typing will consist of a constant monitoring of our text interface as that will be the live input that the user can start at any time, along with acting as controls for the other capabilities of our system. Once the user hits the input to start recording the process for capturing voice input will be started and the only button that will be monitored on the text interface will be to stop recording. After this the program can hand off the captured voice to our voice cloning model for background processing while the user inputs their text. Similarly when the user hits the button to produce an output this block will then hand off the text to the voice cloning model to then process and produce the output .wav file. In order to do these handoffs this block will utilize multiprocessing in order to keep a consistent reading of the

user inputs while handling other processes. Once completed this file will be retrieved by this block and when the user denotes that they want to playback the recording it will do so. Below in Figure 16 is a state diagram of this very process.

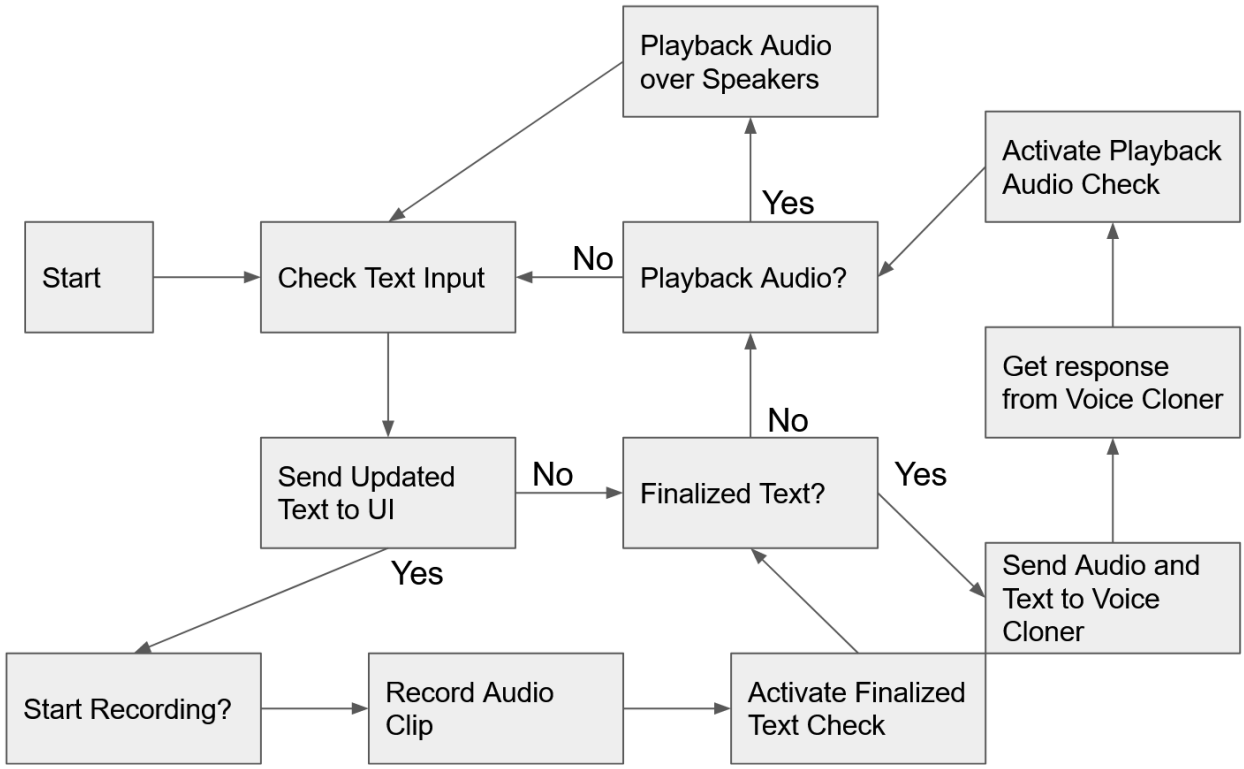


Figure 4.7.2.2: I/O State Diagram

4.7.3 - General Validation

The overall objective of this block is to tie together our entire system from the software components to the hardware components. In order for our design to function together it needs to have a way to reach outside of the Jetson Nano and to external inputs and outputs. With this block consisting entirely of code it is able to be removed from many concerns typical to designs. Mainly the cost, availability of parts, and physical size.

The main motive for this block is the engineering time and understanding necessary for its implementation. As it interacts with many other blocks, but is a core component of them working together, it needs to be designed almost side by side with them. By producing sections focusing on each interface this block is able to be a bit more flexible when it comes to requirements. While hardware components need a specified range to operate properly this block will be modifiable to mold to the requirements of the hardware components. By doing this it is able to ensure functionality of individual components with each other, given that there are no major issues with the individual hardware components themselves.

Since our project partner wants our system to be designed to fit in a small form factor and act similar to a kiosk that could be set up, a lower end and small computer is necessary. This is the

main reason the Jetson Nano was chosen as a fitting host computer. The Jetson Nano is created by Nvidia to be specifically used for AI projects and to behave similar to a more powerful Raspberry Pi, especially with its discrete GPU. By being built with AI in mind it needs to have a large amount of support for python. With many drivers and libraries designed to make the Jetson Nano as compatible as possible with python it makes it a perfect choice to build our core program with python. With an included GPIO library for python it will allow us to not need an external programmable microcontroller to interact with our custom built parts through IO pins.

A previous concern with this block was caused by limitations of the hardware or chosen software language. This, however, is no longer a concern. Our most recent Jetson Nano we possess has 4GB of memory, only a quarter of which is taken by the operating system. Our model is projected to use more than 1GB of memory, putting us with a comfortable amount of memory left over to handle other processes.

4.7.4 - Interface Validation

Table 4.7.4.1: mcrcntrllr_i_cntrl_cd_data Interface

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Messages: Text input from a 12-keypad: encoded with 2 shift registers for a total of 16 bits, must be able to decode these values	Our hardware will consist of two shift registers to encode 12 buttons. This will be to ensure no ghosting of buttons will occur (such as pressing 2 buttons is read as the same as pressing a third button alone). In order to read this we need to be able to decode a 16 bit value to match our 12 button interface.	Utilizing the I2C pins among the GPIO pins on the Jetson Nano we are able to get an input from our shift registers. This will be decoded by this I/O code with one bit for each button. The first 12 bits of the 16 bits of the shift registers will each correspond to a key on our text interface.
Audio recording read for playback: Must be able to save audio from a microphone in a .wav file format.	The audio required by our voice cloning software must be in a .wav file format. The I/O control code must be able to record audio from a microphone and save it in this format.	Utilizing PyAudio and SciPy this code will be able to record and save audio to a .wav file format [1][2].
Messages: Audio recorded utilizing a microphone: must be sampled at a rate of at	Mostly a design limitation set by our voice replication software, to meet the minimum quality of audio it needs to be sampled at a rate of	The python library we will utilize, PyAudio which is built off of PortAudio, has a sampling rate that is only limited by the sampling rate of the hardware. The I/O code will

least 22,050 Hz	22,050 Hz or more.	be built to utilize the highest quality available [2].

Table 4.7.4.2: i_cntrl_cd_mcrcntrllr_data Interface

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Messages: Audio playback to speaker: Must be able to playback audio from a .wav file.	The output of our voice replication software will be in a .wav format for audio. In order to play this to the speaker we design the I/O code will need to be able to read and playback a .wav file.	Utilizing PyAudio and SciPy this code will be able to playback an audio file to the speaker [1][2].
Messages: Text output for user display: Must output text in a format readable by a graphical user interface.	In order for the user interface to display what the user is currently inputting it is necessary for these to be able to host text in a format readable by another part of the program.	The user interface will likely be running within the same memory space as the I/O code (this will be a library utilized by our main process) and therefore will be able to share a string with the user interface.
Protocol: Data output will be saved to a corresponding file type: .wav for audio and .txt for text	This requirement is somewhat duplicated as many different parts of our project need a similar functionality.	Utilizing PyAudio and SciPy this code will be able to record and save audio to a .wav file format [1][2].

Table 4.7.4.3: i_cntrl_cd_dp_lrnng_data Interface

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Messages: A python dictionary containing two	In order for our model to train it needs to be pointed to the proper files. These two files will consist of	This is functionality built into python itself as a child process will be created with arguments that point it

python strings: a user text input and a file path to a .wav file of the users speech.	a text input and audio input, one for the sample audio and one for the sentence to be produced.	to the correct files.
Other: According to 9/10 people the .wav file contains minimal noise.	In order for our model to produce a good output it needs a good input. This means our created audio file needs to be of a high quality.	This will mostly be reliant on the microphone quality rather than the software. The software will be sampling audio at a high rate.
Other: The .wav file at the provided file path has a sampling rate of at least 22,050 Hz.	The model itself samples the audio input at a rate of 22,050 Hz. This means we need a minimum sampling speed of 22,050 Hz to ensure data quality.	Utilizing PyAudio and SciPy this code will be able to record and save audio to a .wav file format [1][2]. The only limit to the sampling speed is the hardware itself.
Other: The .wav file is at least 5 seconds in duration.	In order for our model to train well enough to sound similar to the user at least 5 seconds of audio is needed.	As the only limit to how long we can record is the amount of memory on our hardware this requirement will be fulfilled with the basic functionality of PyAudio and SciPy.

4.7.5 - Verification Process

A brief explanation before the verification plan steps:
This will be split into 4 numbered sections, one for each input and output, and have steps for each of those. This is an effective layout for these tests because this program consists mainly of 4 separate functions all working together.

Our text input consists of a very basic encoding scheme with each button consisting of a single bit within a combined 16 bit sequence. This will be tested with the following steps:

1. Text input
    a. Create a file that will be read that follows the encoding scheme of our keys (1000 0000 0000 0000 = button 1).
    b. Feed these sequences to the I/O code via reading directly from the file or utilizing the GPIO pins to read from another device.
    c. When the encoded data is read it will be converted to the corresponding button and displayed via the terminal.

The audio input will consist of a microphone connected via a USB driven ADC. This will need to be read and save the audio to a .wav file.

2. Audio input
    a. Run the record audio function.
    b. This will start a 5 second timer during which it will record all inputs from the microphone.
    c. The recorded audio will be saved to a .wav file in the same location as the program.

In order for the user to hear what our program produces there needs to be a way to playback the audio. This test will prove the I/O code can do such.

3. Audio output
    a. By running the playback function and pointing it to the audio recorded from the previous test audio playback should begin.

Partially accomplished with the text input and audio input tests there still is a need to test if this program can create a process and point it to proper input files.

4. Data handoff
    a. Via a command line argument this program will create a child process.
    b. This child process will simply consist of a function to print to the command prompt that it has been created.
    c. Afterwards the I/O code will print to the screen that the child process has successfully finished.

4.7.6 - References and File Links

[1] Hubert Pham, PyAudio Documentation, Available:
https://docs.scipy.org/doc/scipy/tutorial/io.html

[2] The SciPy community, File IO (scipy.io), Available:
https://people.csail.mit.edu/hubert/pyaudio/docs/

[3] nVidia Jetson Nano DataSheet, Available:
https://developer.download.nvidia.com/assets/embedded/secure/jetson/Nano/docs/JetsonNano_DataSheet_DS09366001v1.1.pdf

4.7.7 - Revision Table

Table 4.7.7: Revision Table

Author:	Date:	Change:	Reasoning:
Grant	1/15/23	Created and laid out block validation draft	Block Validation Assignment due
Grant	1/19/23	Finished writing block validation draft	Assignment due
Grant	2/8/23	Updated interface to match new ones online	Mismatch between online portal and documentation
Grant	2/11/23	Finalized document	Assignment due

4.8 - UI Block

4.8.1 - Description

This block will handle displaying information to the user of the system. This will host buttons or other user interact-able objects to allow the user to record their voice to be cloned, enter text for the cloning model to read from, and playback the output audio from the model. By interacting with the I/O code and deep learning model it will be able to retrieve data to display.

4.8.2 - Design

As stated in the description of this block the goal is to display information to the user. This will consist of the text that they are inputting, tappable buttons for recording, playback, submit for processing, and help. By utilizing Tkinter as the framework for the user interface one is able to produce such results. Below is a black box diagram of what inputs and outputs are coming from the user interface.

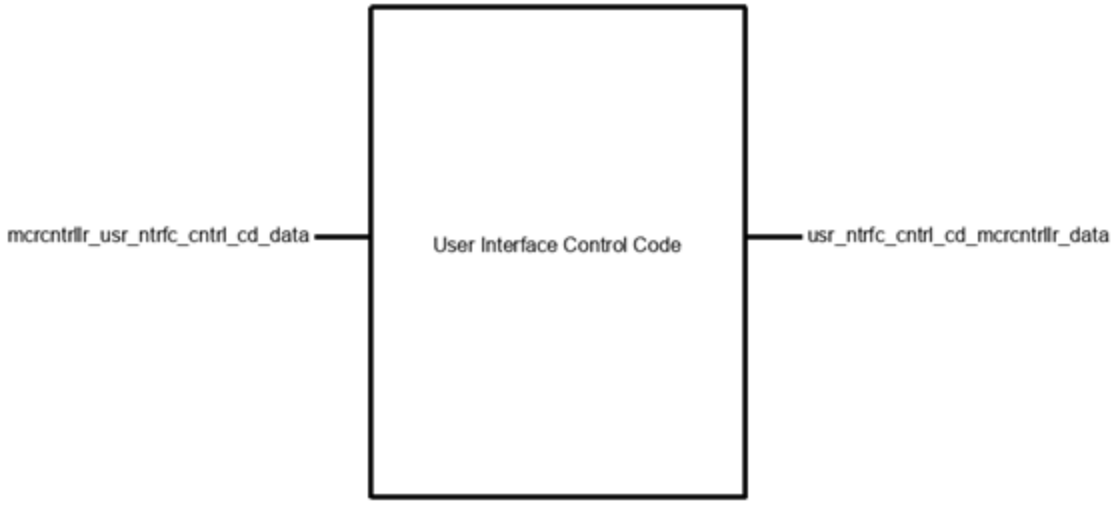


Figure 4.8.2.1: Black Box Diagram for User Interface

Mentioned before this block will be created using Tkinter. A user interface needs to be intuitive and clear, while fitting all required information within the screen space available. This will be accomplished with the design below in Figure 4.8.2.2.

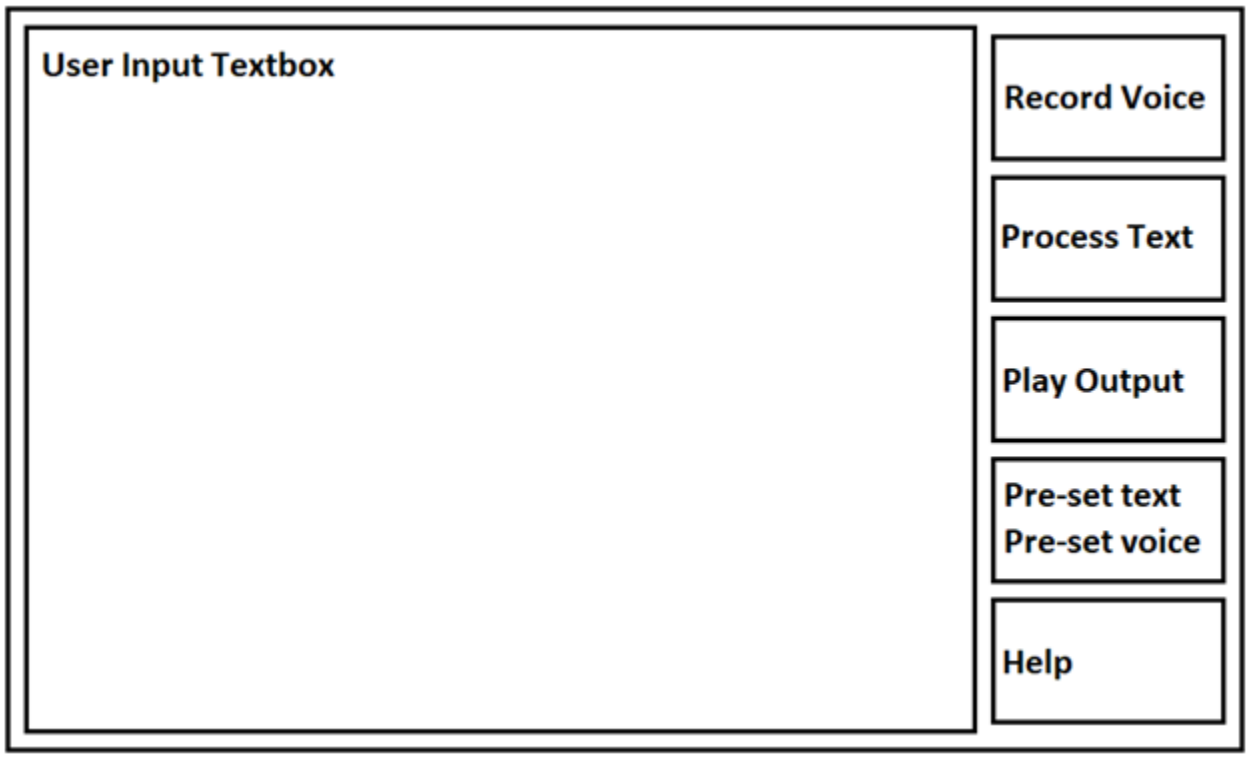


Figure 4.8.2.2: User Interface Design

This design seen above should meet the requirements of our system. By giving a clear button layout and a large enough textbox to comfortably fit a large amount of text this interface will

provide a good user experience. The record voice and play output buttons can have bordering colors, or even just be recolored, to signify the existence and readiness of the input and output files.

4.8.3 - General Validation

The interfaces defined for this block are mainly software inputs. To validate that this user interface meets requirements is simply designing it to meet said requirements. Needing to be quick and responsive, as having no more than 0.1 seconds of input lag from when the user presses a button and the action occurring is entirely based on how the user interface was made. Tkinter can accomplish this with a lightweight and quick interface. The requirement of reading a text file and looking for the existence of a .wav is built into the base functionality of Python, the language this interface will be built with.

For outputs the window must fit within 800 by 480 pixels, a limit set by the resolution of our screen, displaying the existence of input and output .wav files, and be legible from a minimum of 3 feet away. The easiest of these to accomplish is the 800x480 pixel requirement, as this is simply the defined size of the created user interface. Displaying the existence of the input and output .wav files will be done simply with green or red colored light on the record voice and play output buttons. Last is the requirement of being legible from 3 feet away will be accomplished by making text large enough to read.

4.8.4 - Interface Validation

Table 4.8.4.1: mcrcntrllr_i_cntrl_cd_data Interface

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Data Rate: Must update text display within 0.1 seconds of file changing	In order for the user to feel that the interface, and system overall, is responsive a quick UI is necessary. A delay of 1/10th of a second should work fine for this situation.	This design will be met simply by having a less-intensive user interface. Since the design above is simple and will not need a lot of processing power it will accomplish this requirement.
Messages: Must be able to read text from a .txt file	The system needs some way to display information to the user, which will be accomplished by the UI. This requirement will be needed for the UI to be able to show what is currently available for the system.	This functionality is built into python and will be accomplished simply by the inclusion of it in the design.

Other: Must be able to check for an input and output .wav file	Similar to above, the system needs to display the availability of inputs to the user, which the UI needs to be able to read.	Again, this functionality is built into python and will be accomplished simply by using that language.

Table 4.8.4.2: i_cntrl_cd_mcrcntrllr_data Interface

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: User interface must display the existence of the input and output .wav files	What is the point of checking for the existence of files without displaying said information to the user.	This will be accomplished using the input for the .wav files and displaying its existence via a red or green light.
Other: User interface must fit within 800 x 480 pixels (size of our display)	This requirement is set by our display, being 800x480 pixels.	Tkinter has options to define the screen size and make the interface fullscreen. This will be utilized to make the UI an exact fit.
Other: Text interface must be legible from a minimum of 3 feet away.	This device is intended to be a handheld system, or at least small enough to be one. Typical users of handheld devices do not hold it at a distance of 3 feet, so this requirement is set above that to ensure visibility.	By making text large enough (hence the design having a large area textbox) we are able to ensure that anyone with reasonable sight quality can read the text in the interface.

4.8.5 - Verification Process

Confirming the requirements above are mostly done simply visually. Starting from the top the list below will explain how each requirement will be confirmed:

1. Must update text display within 0.1 seconds.
    a. This will be accomplished by typing something into the user interface and ensuring that it updates with little to no input lag.
    b. As 0.1 seconds is something very difficult to measure by hand it will be done purely visually.
2. Must be able to read text from a .txt file

    a. By creating and filling out a text file that is saved where the user interface checks for it we are able to show that it is able to read from it.
3. Must be able to check for the existence of an input and output .wav file and user interface must display the existence of the input and output .wav files
    a. Like the requirement above this will be done by creating .wav files and visually showing that the interface displays the existence of these files
4. User interface must fit within 800x480 pixels.
    a. This is accomplished by showing either on the display we will use in the system or showing that it takes up no more than a quarter of a typical 1920x1080 display.
5. Text interface must be legible from a minimum of 3 feet away.
    a. Another visual requirement, the person accomplishing the test will visually inspect to make sure that they can read an input text from 3 feet away.

4.8.6 - References and File Links

N/A

4.8.7 - Revision Table

Table 4.8.7: Revision Table

Author:	Date:	Change:	Reasoning:
Grant	1/15/23	Created and laid out block validation draft	Block Validation Assignment due
Grant	1/19/23	Finished writing block validation draft	Assignment due
Grant	2/8/23	Updated interface to match new ones online	Mismatch between online portal and documentation
Grant	2/11/23	Finalized document	Assignment due

4.9 - Microcontroller Block

4.9.1 - Description

This block consists of the research and decision of acquiring and setting up a micro controller capable of running our model within our system restraints, as well as interface with needed hardware and software modules within the system. The goal of this is to be our main

computation unit for the project. The microcontroller will be responsible for not only interfacing, but also powering all hardware peripherals including a microphone, text interface, screen, and speaker amplifier. The microcontroller itself will be supplied power through the suggested power supply, supplied by the manufacturer for the specific microcontroller. This will act as the heart of the system, and consist of multiple coding blocks interacting with different parts of the system to deliver a proper user experience. This block includes the microcontroller processing code as well as the power distribution for the system.

4.9.2 - Design

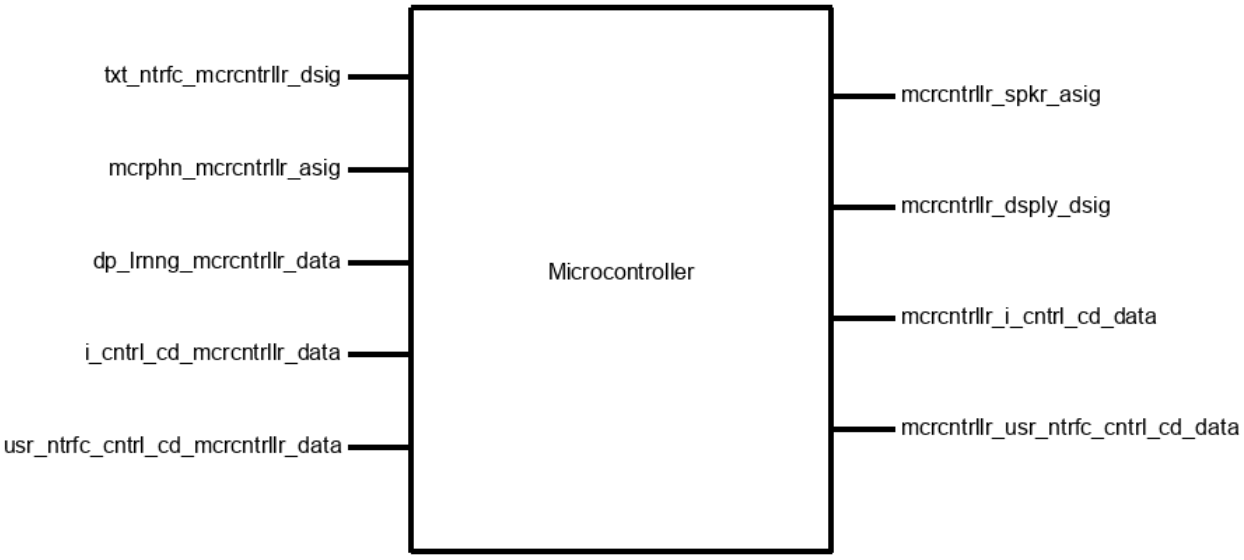


Figure 4.9.2: Black Box Diagram of Microcontroller

The design of the microcontroller underwent several iterations and changes as more information became available. At first, our design for the microcontroller was an embedded system that would be able to handle a deep neural network for the voice cloning model to run off of. This remained our goal, however as the model was optimized and made more compact, the team was able to try out different embedded systems. Initially, the Jetson Nano was chosen as the microcontroller as it had a GPU that the team thought would be able to easily handle the model. At first, we purchased this module and set up the model to run on only the CPU, but once we tried the GPU we found that it was not actually faster than the CPU on the Jetson Nano or even the Raspberry Pi. With this, our design for the microcontroller changed and shifted to work with the Raspberry Pi moving forward. The size of the design was a major component of the design choice, ensuring that the device was capable of running the model but also significantly smaller than a typical computer to allow the possibility of a handheld device as the final product. Dealing with all the inputs and outputs was another design decision that had to be considered, the raspberry pi was able to meet the requirements by having more than enough USB ports, a mini HDMI port for a screen, easy to use GPIO pins that could communicate with SPI, and the ability to use WIFI for setting up everything. The Text input interface communicates over SPI, sending in a total of 16 bits of information to the raspberry pi using the GPIO pins. The microphone signal is connected to an adaptor that connects via USB and handles the conversion from AC to

DC, which is then able to be recorded as a .wav file for the model to use. The microcontroller also deals with user input via a screen as a purchased module, which has a touch screen. On the output side, there is the output of the model which is played as an audio signal to an amplification board connected via a 3.5mm audio jack. The display signals for the screen utilizes an HDMI port as well as a power cable that uses usb type C.

4.9.3 - General Validation

The microcontroller chosen for the system to fit all the needs changed from originally being the Jetson Nano to the Raspberry Pi. This microcontroller fits all the needs of the system as the main requirement of our microcontroller is the size and ability to run the deep neural network for voice cloning. The raspberry pi is extremely small when compared to the size of a normal laptop, and after testing, it was found that the CPU was comparable to the Jetson Nano's CPU which was the only competitor in terms of processing within our budget. The Jetson Nano also has a GPU which was originally thought to have better processing speed than the CPU, but in terms of the model's ability in the speed of transferring speech input to output, the speed was actually worse. The speed is measured in terms of the real-time factor, which is for x amount of seconds of recording, how fast can the model produce y seconds of audio. So if the model gets 1 second of input, how long does that take to process. With the raspberry pi, the model was able to be optimized on the raspberry pi to be under 1 RTF. This means that the Raspberry Pi would fit our design requirements perfectly. The Raspberry Pi also has more power delivery abilities with their GPIO when compared to the Jetson Nano, the GPIO of the raspberry pi is generally easier to work with as well as raspberry pi is more up-to-date with software and libraries used for languages such as python. The raspberry pi overall is significantly easier to work with, equal in computation ability and size, and has no significant downfalls for what is needed within the system for the block of the microcontroller.

4.9.4 - Interface Validation

Table 4.9.4.1: mcrcntrllr_spkr_asig Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Vmax (Supply): - GPIO power pin 5V	This is the designed supply voltage for the system	For the LM386N-1 recommended operating cond: - MAX V is 12V. - MIN V is 4 V.
Vmax (audio signal): - 0.9V peak to peak	This is the expected maximum voltage for an audio signal line level [2].	LM386 is designed for low voltage amplification, and the maximum voltage remains as

		a low voltage.
Other: uC internal volume level: - 10% of maximum volume output	Properly control volume amplification and potentiometer control	Speaker block will only read the audio jack as input.

Table 4.9.4.2: mcrcntrllr_dsply_dsig Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: - Display through HDMI connector	Required interface for screen display	Purchased module contain an HDMI to mini HDMI adapter connecting uC to screen
Other: - Powered through USB connector	Required interface for screen display	Purchased module contains a USB to micro USB adapter connecting uC to screen power
Other: - Touch screen feature enabled	Include extra user interactive buttons through GUI	Purchased module enabled touch screen through power interface

Table 4.9.4.3: mcrcntrllr_i_cntrl_cd_data Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Messages: Text input from a 12-keypad: encoded with 2 shift registers for a total of 16 bits, must be able to decode these values	Our hardware will consist of two shift registers to encode 12 buttons. This will be to ensure no ghosting of buttons will occur (such as pressing 2 buttons is read as the same as pressing a third button alone). In order to read this we need to be able to decode a 16 bit value to match our 12 button interface.	Utilizing the I2C pins among the GPIO pins on the Jetson Nano we are able to get an input from our shift registers. This will be decoded by this I/O code with one bit for each button. The first 12 bits of the 16 bits of the shift registers will each correspond to a key on our text interface.

Audio recording read for playback: Must be able to save audio from a microphone in a .wav file format.	The audio required by our voice cloning software must be in a .wav file format. The I/O control code must be able to record audio from a microphone and save it in this format.	Utilizing PyAudio and SciPy this code will be able to record and save audio to a .wav file format [1][2].
Messages: Audio recorded utilizing a microphone: must be sampled at a rate of at least 22,050 Hz	Mostly a design limitation set by our voice replication software, to meet the minimum quality of audio it needs to be sampled at a rate of 22,050 Hz or more.	The python library we will utilize, PyAudio which is built off of PortAudio, has a sampling rate that is only limited by the sampling rate of the hardware. The I/O code will be built to utilize the highest quality available [2].

Table 4.9.4.4: mcrcntrllr_usr_ntrfc_cntrl_cd_data Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Data Rate: Must update text display within 0.1 seconds of file changing	In order for the user to feel that the interface, and system overall, is responsive a quick UI is necessary. A delay of 1/10th of a second should work fine for this situation.	This design will be met simply by having a less-intensive user interface. Since the design above is simple and will not need a lot of processing power it will accomplish this requirement.
Messages: Must be able to read text from a .txt file	The system needs some way to display information to the user, which will be accomplished by the UI. This requirement will be needed for the UI to be able to show what is currently available for the system.	This functionality is built into python and will be accomplished simply by the inclusion of it in the design.
Other: Must be able to check for an input and output .wav file	Similar to above, the system needs to display the availability of inputs to the user, which the UI needs to	Again, this functionality is built into python and will be accomplished simply by using that language.

	be able to read.	

Table 4.9.4.5: mcrphn_mcrcntrllr_asig Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: - Resistance between the ground and channels of the microphone should be a value around 1.2 with 20K mode on DMM	Because it was measured with a DMM	It does because it was measured to be that value
Other: - The microphone connects to the device via 3.5mm audio jack	Specified in the description of the microphone from both the parent company who produces the microphone as well as Amazon	This audio jack is able to be plugged into any computer or device that has a standard audio jack that is 3.5mm
Other: - the microphone is able to create a .wav file when using audacity and exporting audio.	The microphone is specified on Amazon to be plug-and-play with most devices, meaning that audio should be recordable once plugged in.	Was able to record audio and play back audio using Audacity on my home computer.

Table 4.9.4.6: txt_ntrfc_mcrcntrllr_dsig Interface

Interface property:	**Why is this interface property this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
RC debouncing fall time is no greater than 15ms	The RC circuitry fall time is an assumed value that is then used to calculate the rest of the values for both the resistor and the capacitor. The time is estimated to be around the worst-case scenario for a mechanical switch rattling from high to	Based on the calculations from the equation found in the design details, which was originally sourced from *The Ganssle Group-debouncing pt2*[1]. This calculation and theory have been confirmed with an oscilloscope.

	low after being pressed.	
Active high: 3.3V, represented as logic 1 in code	The microcontroller used for the project provides 3.3 volts from its GPIO pins. This is the voltage then used to power the circuit, and what is required to send over SPI from the shift register so the microcontroller can read the data. The high is assigned to 3.3 volts for ease of understanding of other teammates.	The voltage of 3.3 volts is supported by each chip used: ● SNx4HC04 inverter has a maximum rating for input and output voltage as Vcc which is 7 volts, section 6.1: Absolute Maximum Ratings[2] ● 74HC165 has a maximum input and output voltage of 7 volts as well found in section 8 table 4: Limiting Values[3]. 3.3 volts has also been measured with a multimeter as input and output of each junction between chips and mechanical switch
Low: 0V, represented as logic 0 in code	With high being defined as 3.3 volts, low is defined as the complement of that being 0 volts. Logic 0 is then defined as 0 volts as the complement of the already defined high value of 3.3 volts being logic 1 in software.	The voltage of 0 volts is supported by each chip used: ● SNx4HC04 inverter has a minimum rating for input and output voltage of 0 volts, section 6.3: Recommended Operating Conditions[2] ● 74HC165 has a minimum input and output voltage of 0 volts found in section 9 table 5:Recommended operating conditions[3]. 3.3 volts have also been measured with a multimeter as input and output of each junction between chips and mechanical switch

Table 4.9.4.7: dp_lrnng_mcrcntrllr_data Interface

Interface Property	**Why is this interface**	**Why do you know that your**

	property this value?	**design details <u>for this block</u> above meet or exceed each property?**
Messages: A .wav file containing the user's cloned speech.	A .wav file allows for the storage of a generated cloned voice.	The final stage of the DL model, HiFi-GAN [5], is designed to generate an audio waveform.
Other: According to 9/10 people the cloned voice in the .wav file reads the text input.	Reading a text input in the voice of a cloned speaker is the objective of our project.	The DL model is designed using YourTTS [1], which is a state-of-the-art model for voice cloning.
Other: The .wav file sampling rate is at least 22,050 Hz	The sampling rate is 22,050 Hz to provide smooth audio.	The DL model was trained to generate a speech waveform with a sampling rate of 22,050Hz.

Table 4.9.4.8: i_cntrl_cd_mcrcntrllr_data Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Messages: Audio playback to speaker: Must be able to playback audio from a .wav file.	The output of our voice replication software will be in a .wav format for audio. In order to play this to the speaker we design the I/O code will need to be able to read and playback a .wav file.	Utilizing PyAudio and SciPy this code will be able to playback an audio file to the speaker [1][2].
Messages: Text output for user display: Must output text in a format readable by a graphical user interface.	In order for the user interface to display what the user is currently inputting it is necessary for these to be able to host text in a format readable by another part of the program.	The user interface will likely be running within the same memory space as the I/O code (this will be a library utilized by our main process) and therefore will be able to share a string with the user interface.
Protocol: Data output will be saved to a corresponding file type: .wav for audio and .txt	This requirement is somewhat duplicated as many different parts of our	Utilizing PyAudio and SciPy this code will be able to record and save audio to a

for text	project need a similar functionality.	.wav file format [1][2].

Table 4.9.4.9: usr_ntrfc_cntrl_cd_mcrcntrllr_data Interface

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: User interface must display the existence of the input and output .wav files	What is the point of checking for the existence of files without displaying said information to the user.	This will be accomplished using the input for the .wav files and displaying its existence via a red or green light.
Other: User interface must fit within 800 x 480 pixels (size of our display)	This requirement is set by our display, being 800x480 pixels.	Tkinter has options to define the screen size and make the interface fullscreen. This will be utilized to make the UI an exact fit.
Other: Text interface must be legible from a minimum of 3 feet away.	This device is intended to be a handheld system, or at least small enough to be one. Typical users of handheld devices do not hold it at a distance of 3 feet, so this requirement is set above that to ensure visibility.	By making text large enough (hence the design having a large area textbox) we are able to ensure that anyone with reasonable sight quality can read the text in the interface.

4.9.5 - Verification Process

Since the microcontroller is the heart of the system, its verification process will mimic the final system verification process.

1. Plug the VCC and GND connectors of the Speaker PCB into the 5V VCC and GND pins on the Raspberry Pi respectively.
2. Plug the AUDIO IN and GND connectors of the 3.5mm audio jack for the speaker PCB into the DAC/ADC converter, and plug the DAC/ADC converter into the USB port of the Raspberry Pi.
3. Plug the respective VCC, GND, MISO, SCLK, PLOAD pins of the text interface PCB into the Raspberry Pi.

4. Plug the Screens HDMI to mini HDMI adapter into the HDMI of the screen and mini HDMI of the Raspberry PI. These components will be aligned on the backsides of each other.
5. Plug the micro USB to USB adapter into the screen's touchscreen power port and the Raspberry Pi USB port.
6. Plug the microphone into the Raspberry Pi.
    a. If the microphone is a 3.5mm audio jack, use the DAC/ADC adapter on the input port shared by the speaker.
    b. If the microphone is USB, plug it into the USB port of Raspberry Pi.
7. At this point, all hardware should be properly integrated. Power the Raspberry Pi using the USB-C wall power adapter.
8. The Raspberry Pi will automatically begin running all software needed. From the GUI, begin recording input, following directions displayed on the GUI.
    a. If startup is disabled, run python3 gui.py from the voiceclone directory to pop up the GUI to begin interacting with the system.
9. Click the "Record Input" button and Speak into the microphone providing viable speech audio while the recording is live.
10. Next, Provide any text to the textbox on the GUI, or select a preset text input and hit Process Text.
11. Finally select Play output.
12. Observe the user recording cloned voice and repeat the text fed in by the user through the speaker amplifier.

4.9.6 - References and File Links

[1] "DATASHEET - Raspberry Pi 4 Model B," 01-Jun-2019. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf.

4.9.7 - Revision Table

Table 4.9.7: Revision Table

Author:	Date:	Change:	Reasoning:
Connor	3/12/23	Created Block Validation section for the microcontroller block.	Project Document Assignment due
Connor	3/13/23	Populated Description, Interface validation tables	Project document due
Micah	3/14/2023	Populated Design and General Validation	Project document due
Connor	3/14/2023	Created Verification Process	Project document due

5 - System Verification Evidence

5.1 - Universal Constraints

5.1.1 - The system may not include a breadboard

Our system uses PCB's rather than breadboards.

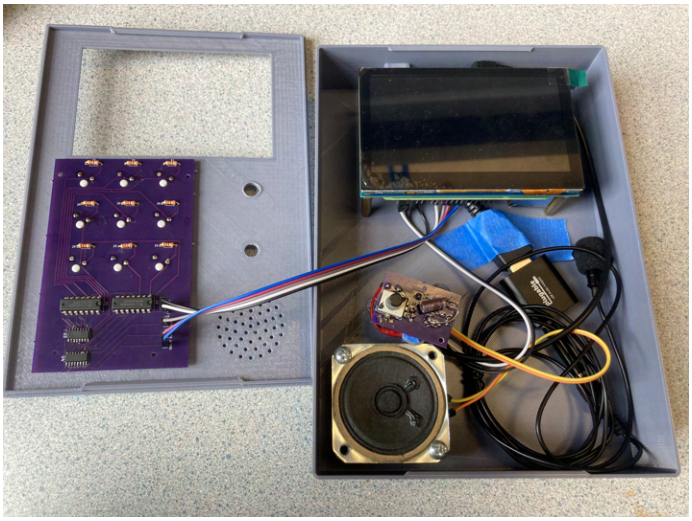


Figure 5.1.1: Internal System Hardware

5.1.2 - The final system must contain a student-designed PCB.

Our system has a PCB for the speaker output and for the keyboard input. These PCBs contain a total of 32 surface mount pads.

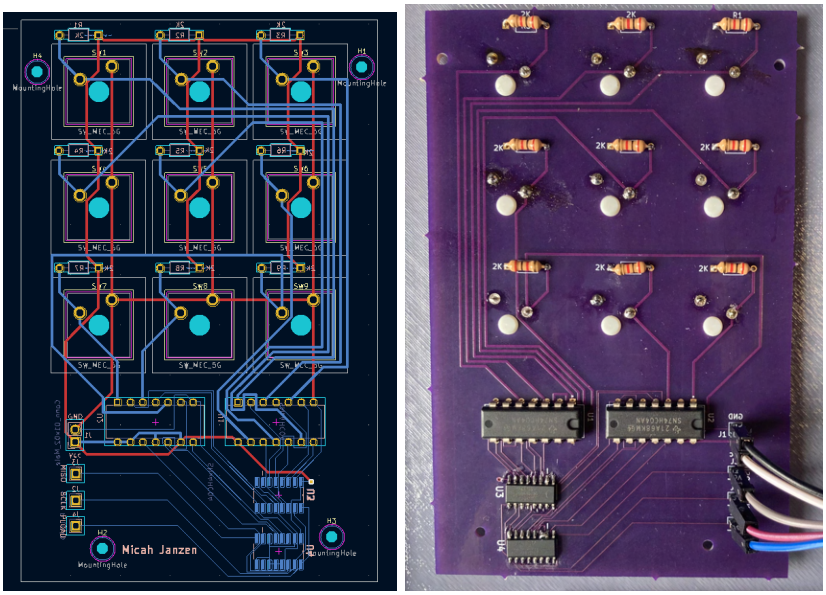


Figure 5.1.2: Student Designed PCB Schematic and Final PCB

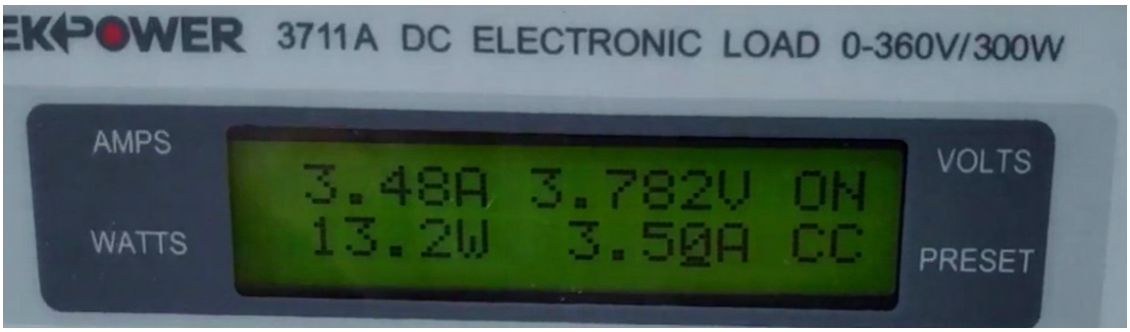
5.1.3 - All connections to PCBs must use connectors.

Our system uses jumper wires and header pins to connect to the PCBs.

Figure 5.1.3: System Hardware Connections

5.1.4 - All power supplies in the system must be at least 65% efficient.

Our system utilizes a power supply that came with our hardware that will be the core of the system. To measure the power efficiency we used a wall power meter, a multimeter, and a DC electronic load. The DC electronic load drew 3.48 amps of current. The voltage across the multimeter was 4.78 V. This means the output power was 16.63 Watts. The wall power meter read a max 22.5 Watts. As such the power efficiency was 0.74 which is above the required 0.65 to satisfy the universal constraint.



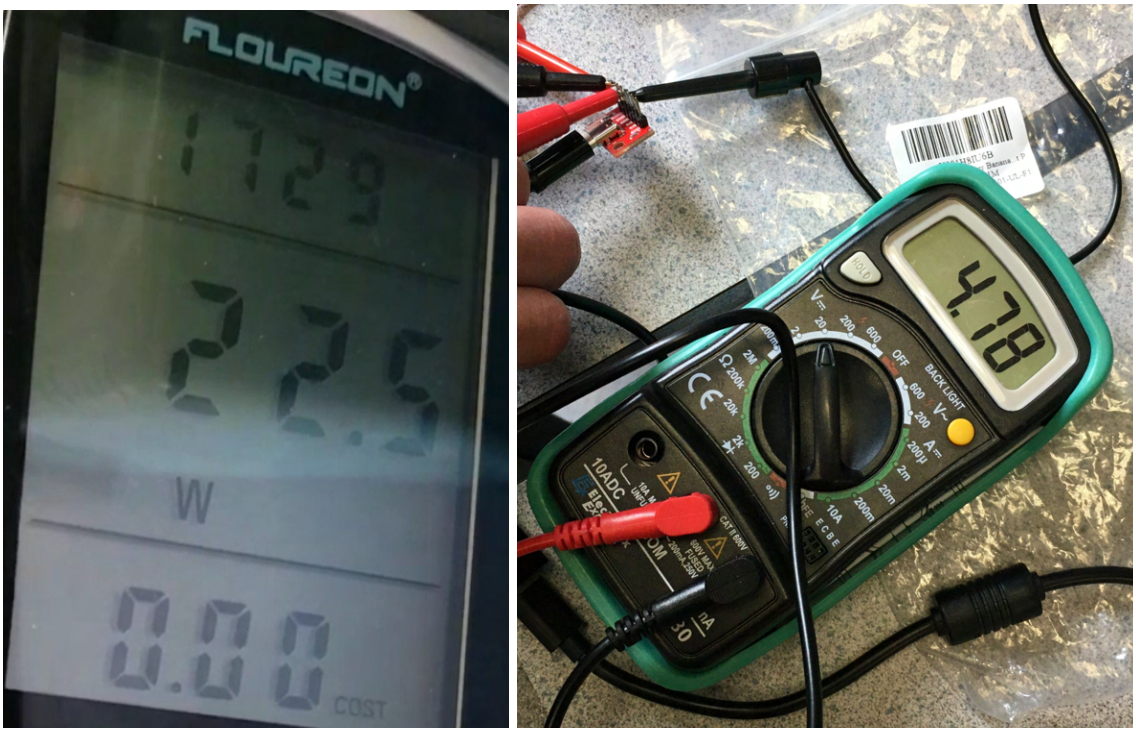


Figure 5.1.4: Power Supply Efficiency Measurements

5.1.5 - The system may be no more than 50% built from purchased 'modules.'

Percentage of built modules: 70%

Table 5.1.5: Module Listing

Blocks	Module Type
Deep Learning Model	Built
Optimizer	Built
Microphone	Bought
Text Interface	Built
I/O Control Code	Built
User Interface Control Code	Built
Microcontroller	Bought
Speaker/Audio Amplifier	Built
Display/Screen	Bought
Enclosure	Built

5.2 - Requirements

5.2.1 - Limited Computation Ability

5.2.1.1 - Project Partner Requirement:

The machine learning model must be capable of running on a low-end embedded system

5.2.1.2 - Engineering Requirement:

The system shall run using at most 4GB of RAM

5.2.1.3 - Verification Process:

This will be verified visually by watching our code running on the microcontroller take an input and generate an output. View the datasheet provided for our microcontroller - has a maximum specification of 4GB of RAM or less.

5.2.1.4 - Testing Evidence:

[1] “DATASHEET - Raspberry Pi 4 Model B,” 01-Jun-2019. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf.

Page 6, Section 2.1 contains the hardware specification for RAM on the Raspberry Pi 4 Model B.
Verified: 2/28/23

5.2.2 - Reproducibility

5.2.2.1 - Project Partner Requirement:

The machine learning model code must be implemented into a widely used machine learning toolkit and be readable

5.2.2.2 - Engineering Requirement:

The system code sub-system shall have its machine learning model implemented into the ESPnet toolkit with at least one comment for at least every 10 lines of code

5.2.2.3 - Verification Process:

View the given ESPnet repository code for comments describing the behavior and process of the voice cloning model

5.2.2.4 - Testing Evidence:

https://drive.google.com/file/d/1tGZWbrKvObpTB8psnlXzEK9AGn9KXUgN/view?usp=share_link

The link provided is a video viewing all the comments in the ESPnet repository. Comments are colored forest green and orange.
Verified: 3/8/23

5.2.3 - Size

5.2.3.1 - Project Partner Requirement:

The system must be smaller than a standard laptop

5.2.3.2 - Engineering Requirement:

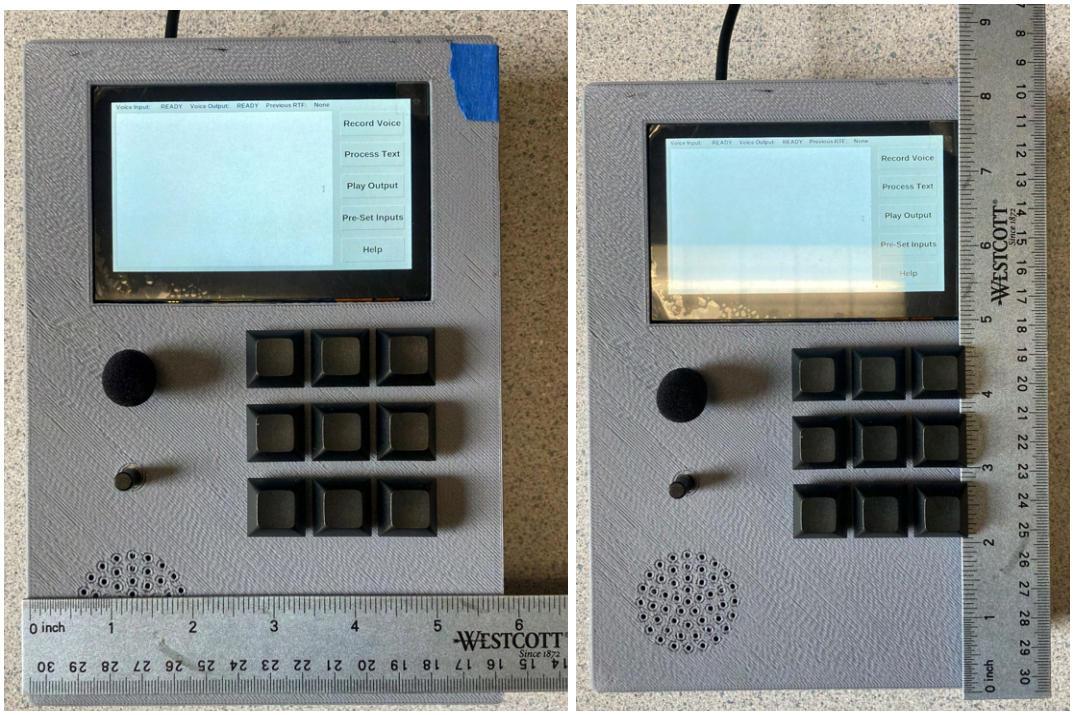
This system shall be contained in an enclosure no more than 11x7x4 inches

5.2.3.3 - Verification Process:

Measure the final dimensions of the enclosure containing all system components

5.2.3.4 - Testing Evidence:

Verified: 5/9/2023



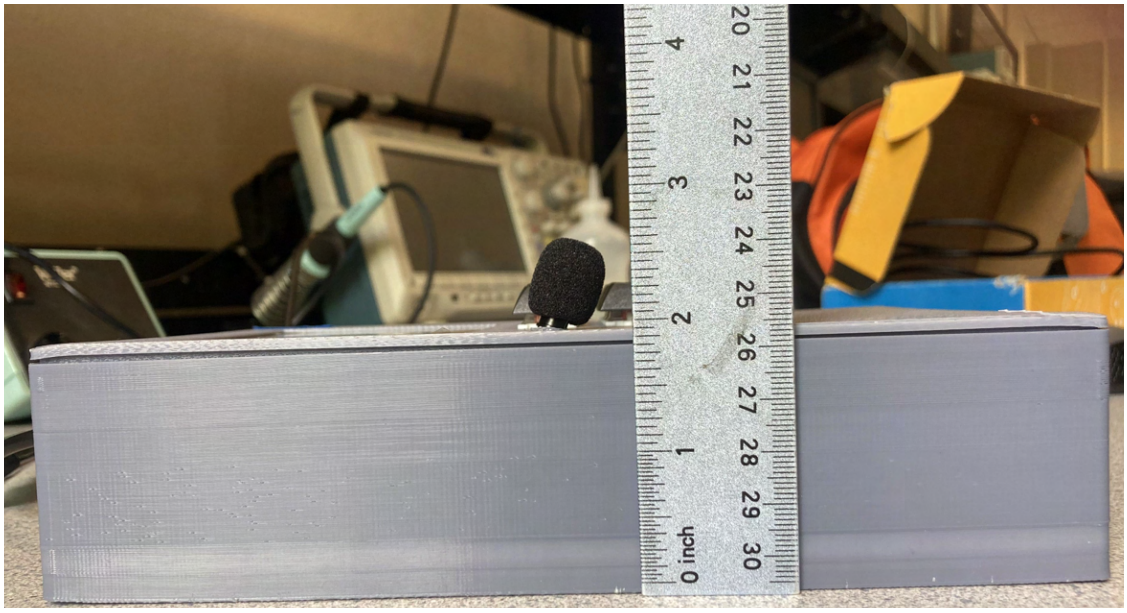


Figure 5.2.3: Physical Dimension Measurements of Enclosure

5.2.4 - Speech input

5.2.4.1 - Project Partner Requirement:

The user must be capable of providing the machine learning model with speech audio

5.2.4.2 - Engineering Requirement:

The system shall allow a speech input to be captured for a duration of at least 10 seconds and 9/10 users are satisfied with the playback quality of their recording.

5.2.4.3 - Verification Process:

Begin recording process with a visual indicator displayed on the GUI, indicating the start and stop. Verify the input was successfully recorded by outputting a recognizable cloned voice through the speaker

5.2.4.4 - Testing Evidence:

https://drive.google.com/file/d/1_2IaG1PIydOvqSZ2NmKhZAsZ6P4t6MD0/view?usp=sharing
The link provided shows a user recording their voice and playing it back.
Verified: 5/9/23

5.2.5 - Speech Output

5.2.5.1 - Project Partner Requirement:

The speech output from the system must be loud enough to hear and sound relatively similar to the end users voice

5.2.5.2 - Engineering Requirement:

The system shall produce a speech waveform that increases ambient dBA by a minimum of 15dBA at maximum volume, and 9/10 users shall be able to recognize their cloned voice.

5.2.5.3 - Verification Process:

Have users walk through the voice cloning process from start to finish, and have users report back whether they can recognize their own voice. Measure the decibel output of the speaker while the user is listening to verify the output loudness

5.2.5.4 - Testing Evidence:

https://drive.google.com/file/d/1_hSQNZS33o7DWkfHDxWsoMRZx9hQrpfN/view?usp=share_link

The link provided shows a user playing back their voice and the decibel readings of the playback.

Verified: 5/9/23

5.2.6 - Speed

5.2.6.1 - Project Partner Requirement:

The machine learning model must be capable of processing an end users inputs quickly

5.2.6.2 - Engineering Requirement:

The system code sub-system shall produce a speech output with a real time factor of 1

5.2.6.3 - Verification Process:

While running the system, the GUI will display a continuous value of the RTF of each generated voice waveform. Observe this RTF value to verify it is less than or equal to 1.

5.2.6.4 - Testing Evidence:

https://drive.google.com/file/d/1s1gU6bs1CofMx_vSoNUb50AjTgxgBK82/view?usp=sharing
The link provided is a video showing the RTF value of some processed.
Verified: 5/9/2023

5.2.7 - Text Input

5.2.7.1 - Project Partner Requirement:

The user must be capable of providing a text input to the machine learning model

5.2.7.2 - Engineering Requirement:

After using the system, 9/10 users shall report the interface button presses are responsive.

5.2.7.3 - Verification Process

A user shall provide the system any text input they want containing the letters A-Z and the space character. The GUI shall display this text back to the user for confirmation, within a certain amount of time satisfiable to the user.

5.2.7.4 - Testing Evidence:

https://drive.google.com/file/d/1_Vh-Bwn7UvLY_zwVV7gcpRHrzoTUn-z0/view?usp=share_link
The link provided shows a user using the text input buttons and saying they feel responsive.
Verified: 5/9/23

5.2.8 - Usability

5.2.8.1 - Project Partner Requirement:

The system should be easily usable after proper guidance

5.2.8.2 - Engineering Requirement:

The system shall be operable by 9 out of 10 users after a 3 minute training process.

5.2.8.3 - Verification Process:

A goal to test this is to see if people are capable of using our product. We will get a group of 10 people and give them a brief explanation of how to use our product. Then we will tell them to demonstrate using it and if they are incapable of doing so within 5 minutes then they have failed, otherwise they have succeeded. These results will be recorded and compared to our goal

5.2.8.4 - Testing Evidence:

https://drive.google.com/file/d/12Ov_qO5t3QonwCIgd1I3LiUyua0GsEfV/view?usp=share_link
The link provided shows a user being trained on how to use the device. The video length is less than 3 minutes, showing that this process can be done within our required time.
Verified: 5/9/23

5.3 - References and File Links

For the 9 out of 10 users sign-offs check the link below:
https://docs.google.com/spreadsheets/d/15XvtHX-u2aL4ID0-mnV4ChoKr82Q4szVSfAlMrmTsIw/edit?usp=sharing

5.4 - Revision Table

Table 5.4: Revision Table

Author:	Date:	Change:	Reasoning:
Connor	3-10-2023	Created Section 5 outline	System Verification due
Connor	3-12-2023	Updated section 5.2.x.3	Drafted verification processes for each system requirement
Connor and Matthew	3-15-2023	Added testing evidence for first 3 system requirements	System verification checkoff

6 - Project Closing

6.1 - Future Recommendations

6.1.1 - Technical Recommendations

Our current model suffers from an inability to clone a user's voice in real time to a high-quality standard. The current model that we have implemented to act as the voice cloning system is YourTTS[1], a minor variation of VITS[2]. Although VITS is state-of-the-art for the task of voice cloning, it does not perform well when provided with a small voice sample from a user, which our system requires. The first solution to such a problem is to utilize an updated model with better real-time voice cloning capabilities once such a paper is published. A secondary solution is to fine-tune the hyperparameters of VITS to a greater extent. Finally, we suggest exploring additional datasets to include in the training data in addition to LibriTTS[3]. One notable dataset that could be included is the VCTK dataset[4]. By providing more data in the training dataset, the model will be capable of generalizing better to unseen data.

A secondary technical issue with our system is that the enclosure has a height of 38.5mm, a width of 150mm, and a length of 210mm. As such, our system is not suitable for use as a handheld device, even though it meets our engineering size requirement. Our plan for the enclosure came to fruition late in the design process at the end of the second term due to fluctuations in the choice of hardware and microcontroller. A future group that uses our hardware as a starting point would be capable of optimizing the enclosure to fit all the components while also being handheld.

A tertiary technical issue with our system is it does not utilize a built-in power supply. Currently, the system uses power from a wall outlet. As such, a future improvement would be to include a battery in the enclosure. Doing so would allow the system to be portable rather than stationary.

Our final technical issue with the design is the slow model speed. Such an issue results from our voice cloning model VITS being slow to generate speech waveforms [2]. However, it is also partially due to our raspberry pi model 4B being inadequate for machine learning inference. Some potential obvious solutions for the slowness issue include using an updated model which prioritizes speed or using more powerful hardware for the microcontroller. However, some alternative subtler approaches include applying structured pruning or knowledge distillation. Structured pruning refers to removing entire nodes rather than individual connections from a neural network, as is done in unstructured pruning[5]. Structured pruning is a preferable option over unstructured pruning as the hardware is able to take advantage of the network optimization, increasing performance. Knowledge distillation refers to using a larger, more complex model to aid in training a simplified model[6].

6.1.2 - Global Impact Recommendations

1. Ensure realism of cloned voice is still distinguishable between the human and machine.
    a. With real-time voice cloning, the dangers of easily recreating and mimicking a voice to social engineer people and manipulate others with a realistic voice pose a real threat at a global level. Within the SV2TTS paper[7], there is mention that the synthesis of the voice of a cloned voice is generated to be distinguishable from the real voice. This should be further researched and made sure to be implemented in future improvements or changes to the model used for voice cloning within embedded devices.
2. Ensure accessibility of device and interface to those with disabilities
    a. One key goal of the project is to allow those who are losing or have lost their voice to quickly gain it back and have the ability to talk again using a handheld device. As such, it is recommended that future improvements keep in mind the target audience for usability with a focus on easy usability and accessibility.

6.1.3 - Teamwork Recommendations

Our first teamwork recommendation is to create a meeting agenda whenever you meet as a team. Our group found that when we did not have a meeting agenda prior to the meeting, we would fail to use our time properly and would get distracted. By creating a meeting agenda, it will ultimately save everyone in your team time and ensure all important points are discussed. We found the outline provided by [8] to be effective at creating our own agendas. Each agenda entry includes an item, a desired outcome, a priority, a time, a person, and a methodology [8].

Our second teamwork recommendation is to create a project timeline early and stick to it. In our team, we were late to create a project timeline and did not start adhering to the timeline until the middle of the second term. As such, we were rushed at the end to complete our blocks. Additionally, since we were not in sync with our project timeline, our whole team was out of sync causing difficulties in collaborating on blocks that interacted with each other. We suggest once a project timeline is created, continuously update it as the project progresses. Not every task will be completed in the time expected, and as such, adjustments are expected[9].

6.2 - Project Artifact Summary with Links

Github Repository Zip file link:
https://drive.google.com/file/d/1C81jRG0r4Kq9vkeb_UQTGStHYampRNcA/view?usp=sharing

Schematic Diagrams:

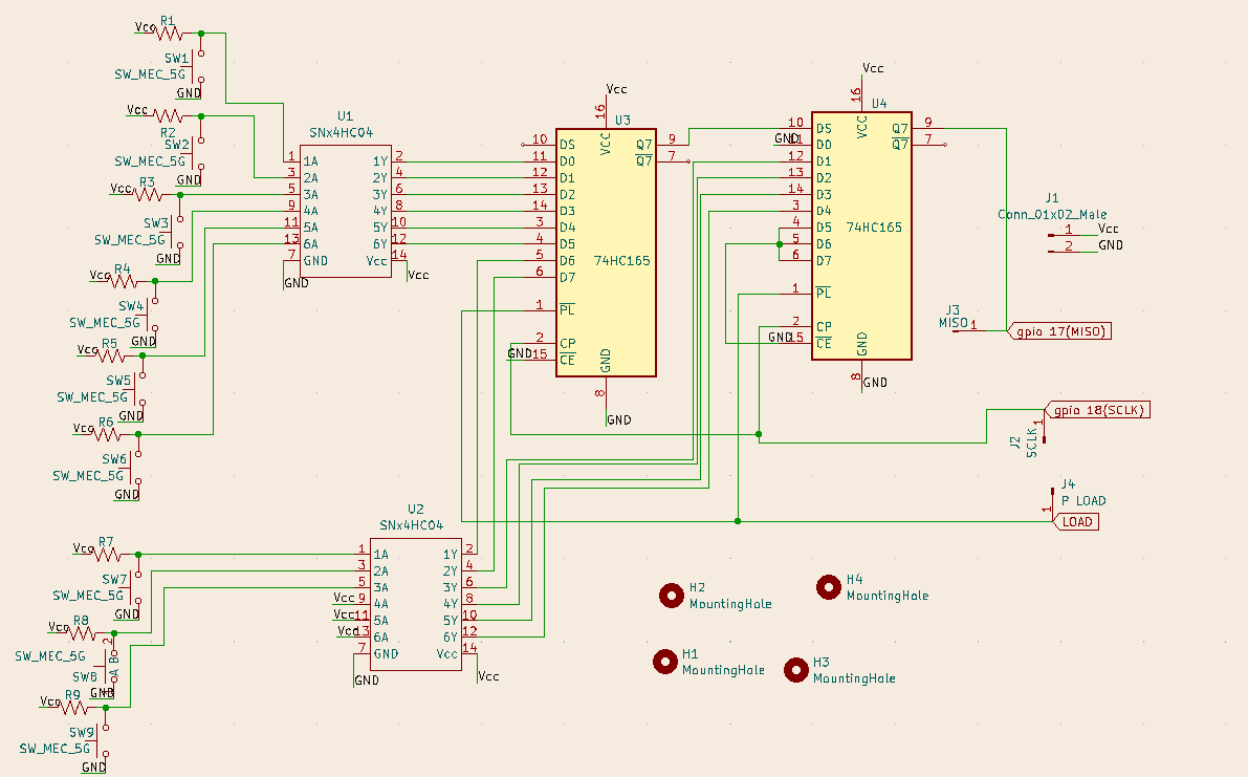


Figure 6.2.1: 10 Key Button Text Interface Schematic

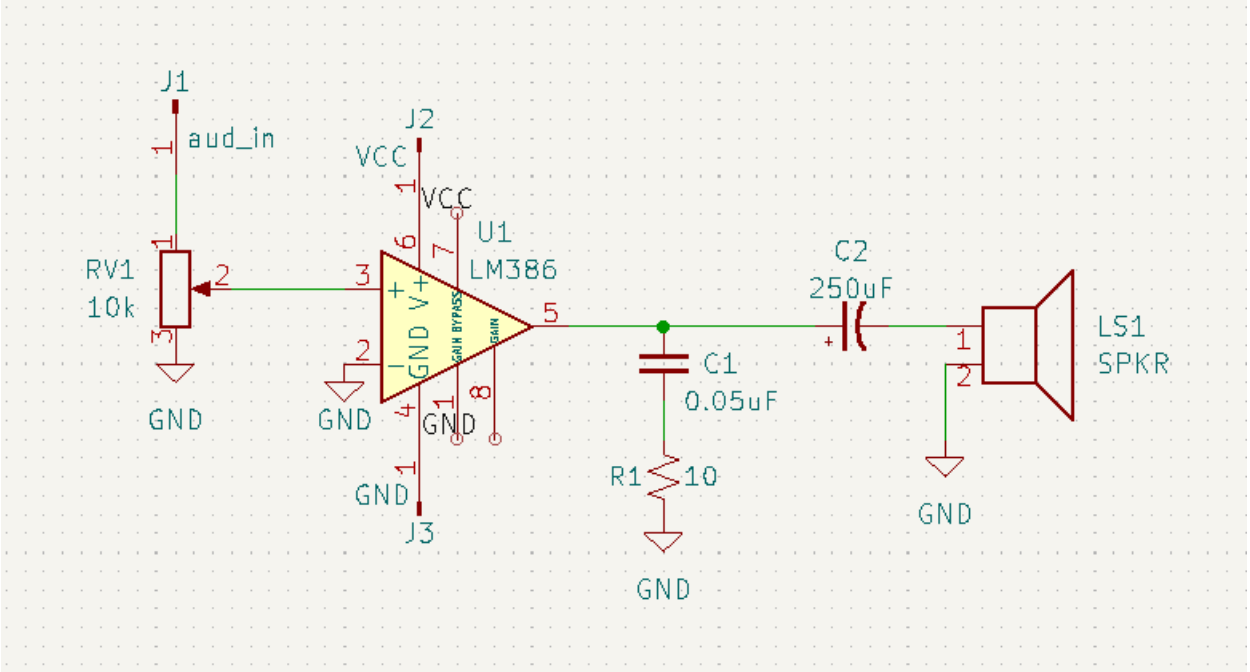


Figure 6.2.2: Audio Amplifier Schematic

Enclosure diagrams:

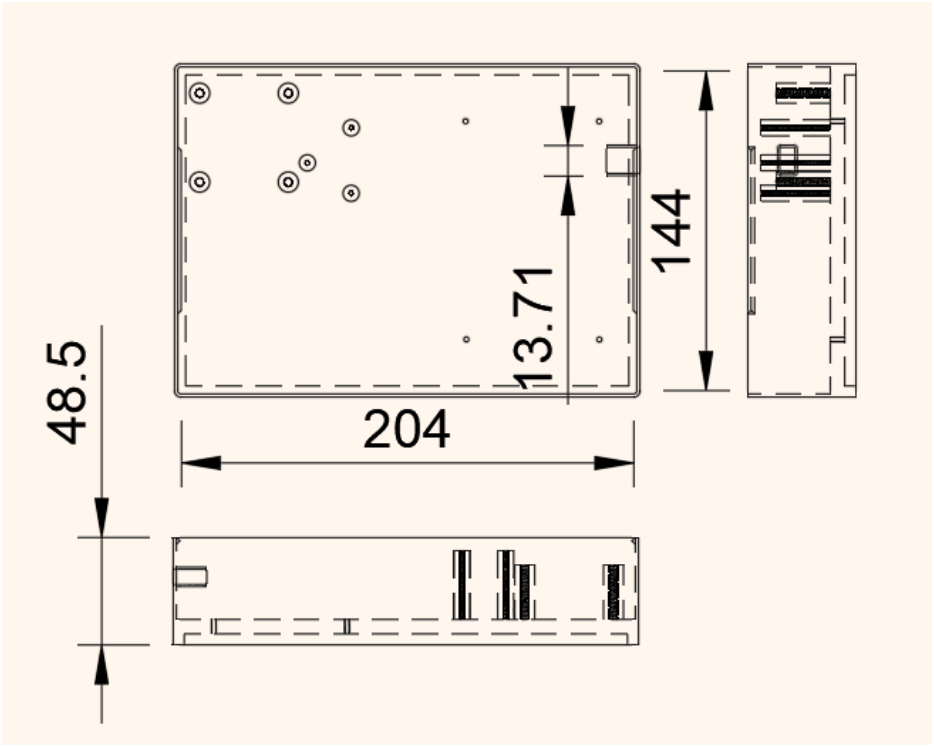


Figure 6.2.3: Enclosure dimensions in millimeters

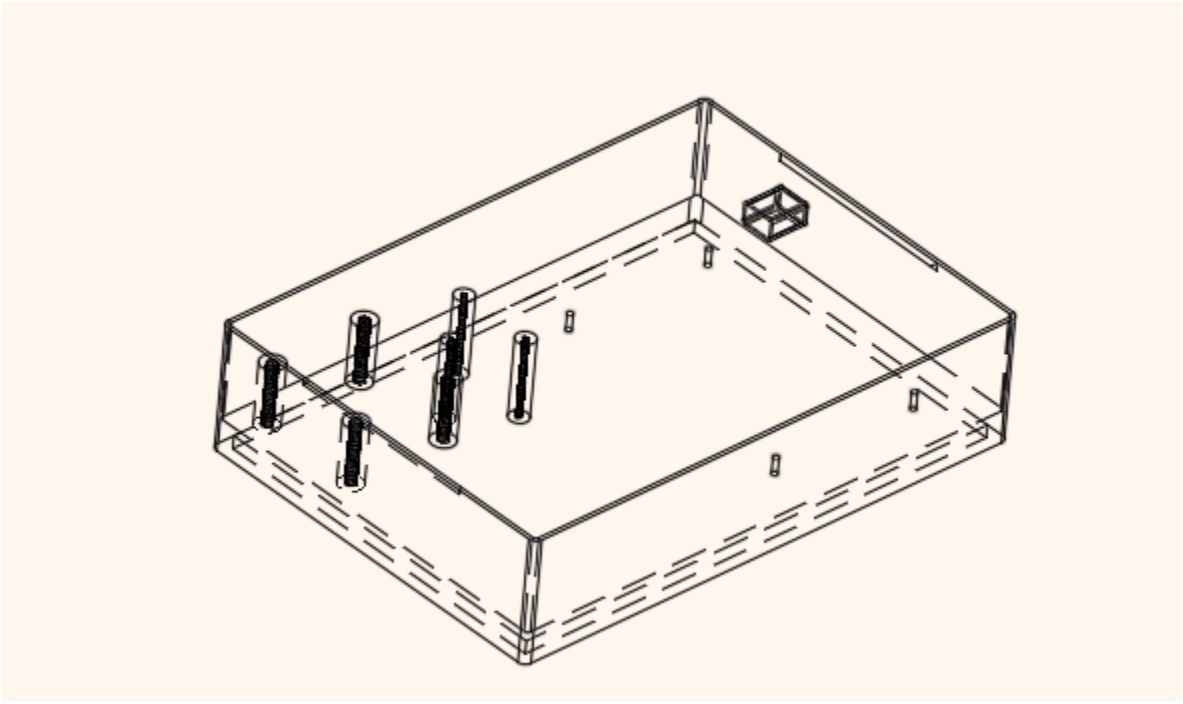


Figure 6.2.4:Transparent image of Enclosure. Screw holes for mounting parts visible above.

6.3 - Presentation Materials

Link to showcase website:
https://eecs.engineering.oregonstate.edu/project-showcase/projects/?id=DB9y1TtmH8BPSD3F

Project Poster:

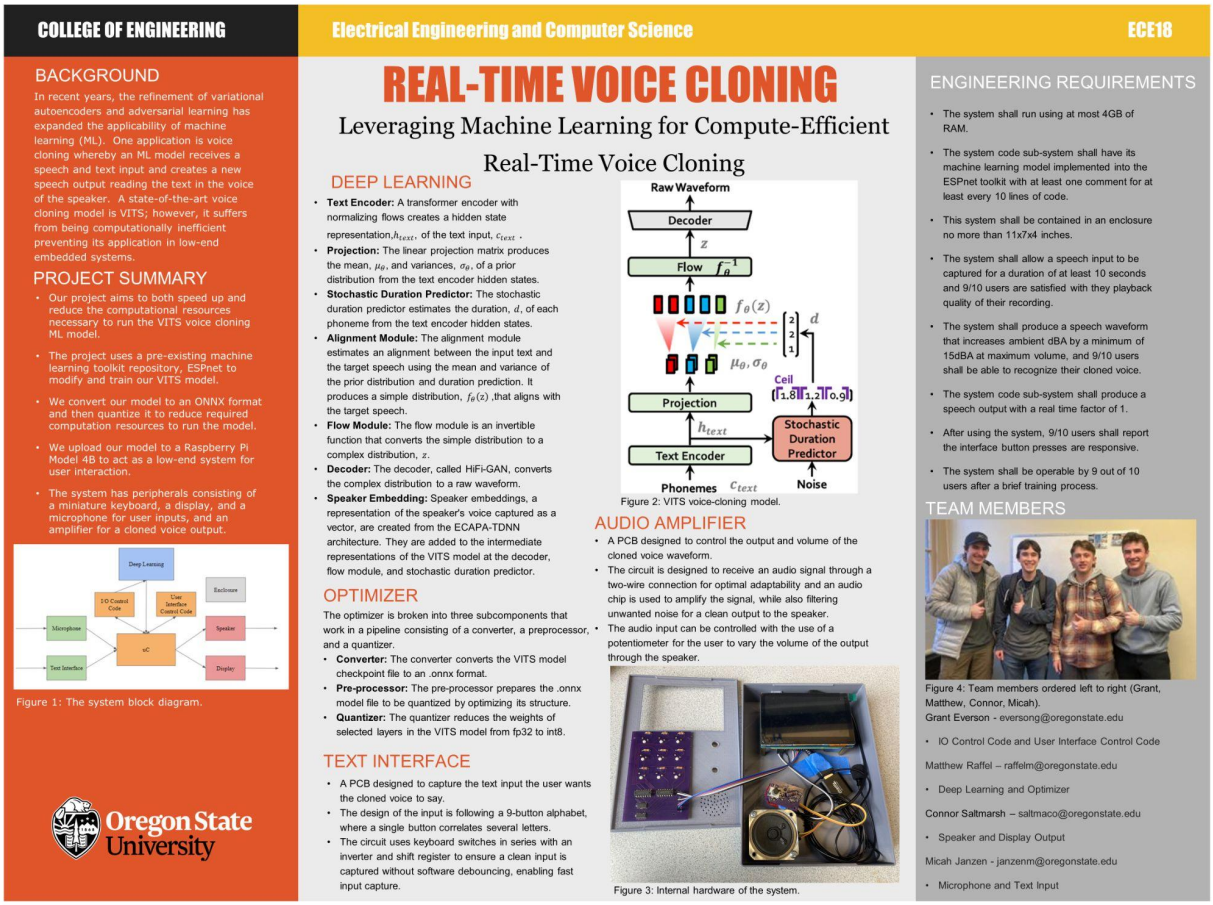


Figure 6.3: Project Poster

6.4 - References and File Links

[1] Casanova, E., Weber, J., Shulby, C., Junior, A. C., Gölge, E., and Antonelli Ponti, M., “YourTTS: Towards Zero-Shot Multi-Speaker TTS and Zero-Shot Voice Conversion for everyone”, arXiv e-prints, 2021.

[2] Kim, J., Kong, J., and Son, J., “Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech”, arXiv e-prints, 2021.

[3] Zen, H., “LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech”, arXiv e-prints, 2019.

[4] “VCTK Dataset.” *Machine Learning Datasets*, datasets.activeloop.ai/docs/ml/datasets/vctk-dataset/. Accessed 1 Dec. 2022.

[5] Anwar, S., Hwang, K., and Sung, W., “Structured Pruning of Deep Convolutional Neural Networks”, arXiv e-prints, 2015. doi:10.48550/arXiv.1512.08571.

[6] Hinton, G., Vinyals, O., and Dean, J., “Distilling the Knowledge in a Neural Network”, arXiv e-prints, 2015. doi:10.48550/arXiv.1503.02531.

[7] Jia Y., “Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis”, arXiv e-prints, 2018.

[8] “How and why to use a meeting agenda,” MIT Human Resources, https://hr.mit.edu/learning-topics/meetings/articles/agendas (accessed May 13, 2023).

[9] L. Hennigan, “How to create a simple, effective project timeline in six steps,” Forbes, https://www.forbes.com/advisor/business/software/create-a-project-timeline/ (accessed May 13, 2023).