EchoSign

Authors: Ria Balli, Ricky Gao, Somya Pathak
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A pair of sensor-based gloves capable of identifying double handed BSL letters in real time and translating them to speech. This product is designed portable, accurate, and easy to use.**

***Index Terms*—Flex sensor, Gesture Detection, IMU, Machine Learning**

1 INTRODUCTION

The goal of our product is to devise a pair of gloves that have the ability to translate double-handed BSL letters to speech output. Its purpose is to help facilitate communication between the deaf and non-deaf population. The deaf/hard of hearing community primarily communicates through sign language, and our device will rely on training an machine learning model with sensor information of the gestures that make up sign language. Our hope is that our product is lightweight, easy to use, and has a sign-to-identification accuracy and speed of classification that is as high as possible.

2 USE-CASE REQUIREMENTS

Based on our goal of providing seamless communication from a deaf speaker to a non-deaf speaker, we focused our use-case requirements on both the performance of the system itself as well as its usability.

EchoSign should have an accuracy of 85% on the targeted vocabulary. This means that for every time a specific sign is performed, the predicted letter/word should match the sign at least 85% of the time. This aligns with our original goal to provide an avenue of communication between deaf to non-deaf speakers and accuracy is a cornerstone of that conversation. We settled on 85% because it was a nice middle-ground between professional research group performance (98.63% by UCLA [3]) and past capstone groups (75.86% by Gesture Glove [2]). This value is motivated by specific considerations into public health and safety. If this system becomes widely used, it needs to be as accurate as possible so accurate information can be conveyed which is especially important in emergency settings.

EchoSign should be able to output a prediction 500ms after the user signed. This means that after the user has signed, the non-deaf speaker should expect to hear the corresponding letter at latest 500ms from the signing. If nothing is announced after the 500ms, the recipient is expected to interpret that as a no signing gesture. The major concern that motivated this requirement is a social one. We wish to create a product that allows for a *seamless* conversation between deaf and non-deaf speakers. The rate at which conversation can occur is important for this and it will inherently be limited by our product. With this in mind, we seek to have only 500ms of delay because this reflects past research that identified that the average speed of signing is 2 signs per second [1]. With this goal, we can try to match the seamlessness of a signed conversation.

EchoSign should be able to classify all 26 letters from the British Sign Language alphabet. The British Sign Language alphabet follows the same characters presented in the American Sign Language alphabet, but uses different gestures to symbolize them all. Notably, the BSL alphabet uses both hands to sign each letter. Our use-case requirements establish that we should be able to reach our accuracy goals on a test set that involves all these letters. We believe that the alphabet provides a standard baseline in terms of vocabulary because it allows for all words to be potentially spelled out. Then, we chose the BSL alphabet because of its use of double handed signs, which will be the main focus and innovation point of our project. Figure 1 has a visual depiction of the BSL alphabet.

EchoSign should be a durable product that can last multiple hours on a single battery and withstand repeated signings without need for repairs or replacements. Again, our reasoning surround social and safety factors. We want to create a product that people can depend on with confidence for a significant period of time. With this in mind we want to ensure that the gloves can be powered by the battery we select for 2 hours at a time. We also want to ensure that our physical design of the glove allows it to be used for multiple charged sessions. The product shouldn't need to be fixed more often than it needs to be recharged / replaced battery.

EchoSign should be a comfortable product to use and light (100g). This is primarily motivated from a health perspective. As this is a product that is mean to be worn, we want to ensure that it is minimally impactful and easy to use. Comfort will be evaluated by user testing as we have people rate how easy it is to use. Weight will be measured by the different components that are added to the glove so we can evaluate their cumulative impact.

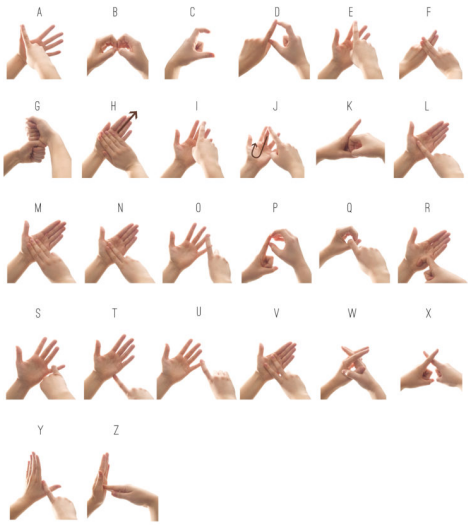


Figure 1: The BSL Alphabet

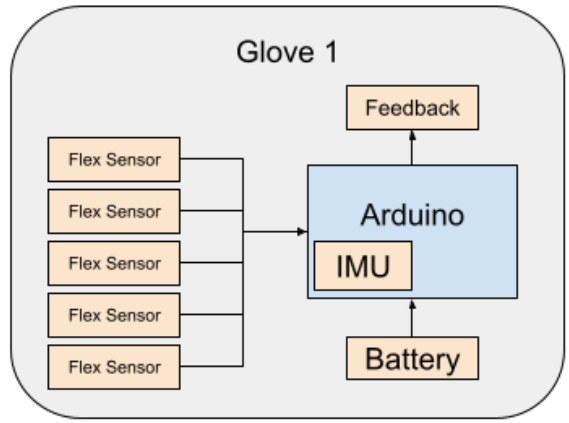
3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

EchoSign's design can be partitioned into its software and hardware components. There are three main components in EchoSign's MVP structure: Glove 1, Glove 2, and the Main Compute Unit. The figure below outlines the various components that make up EchoSign's glove architecture.

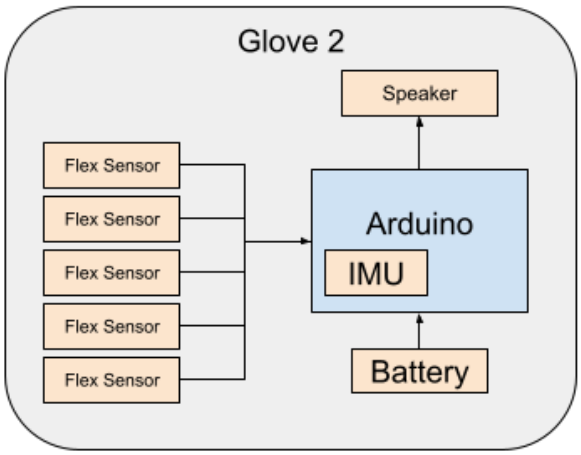
Glove 1 comprises the 5 flex sensors from which we will read voltage output from using our on-hand compute board. This computer on the glove is labeled "Arduino" in the diagram- trade studies on our board specifications can be seen in section 5. For the MVP, we are using the Arduino Nano 33 BLE Sense Rev2 for its compact design, on board IMU, and BLE capabilities. Feedback is also controlled with this compute unit to the haptic vibration motor mounted on the hand. Our battery, a 4.8V 150 mAh battery, will power this system.

Glove 2 differs from Glove 1 solely by the fact that a speaker will be attached to it in place of the haptic feedback component. We wanted to split these two components across the two gloves to distribute weight evenly especially since both items have a negligible difference in weight.

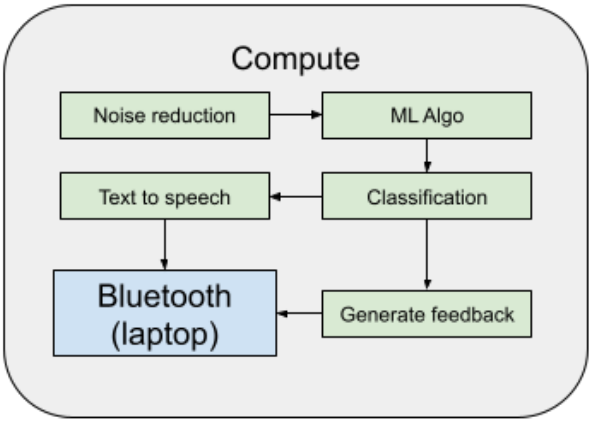
The last component of our system contains primarily software implementation. Once data is sent to the Main Compute Unit, it will be screened for noisy data and outliers, sent to our trained machine learning model, classified as a specific gesture based on classification heuristics we have defined, and then converted into speaker and haptic feedback. More information about our classification heuristics can be found in section 7.



(a)



(b)



(c)

Figure 2: System block diagram.

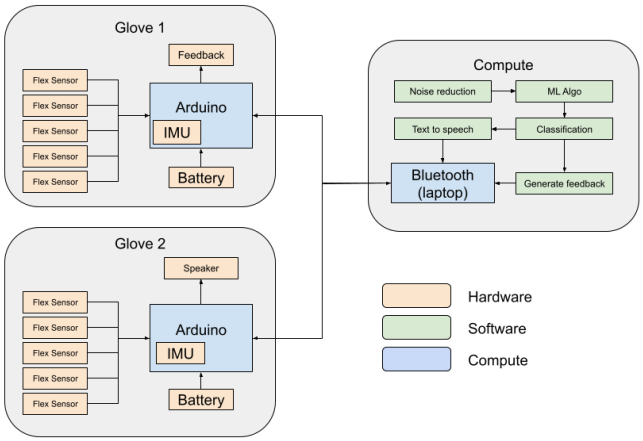


Figure 3: Full block diagram.

4 DESIGN REQUIREMENTS

We initially thought about what kind of sensor readings would help us classify gloves and an IMU was included in this list due to its ability to provide positional measurements. Most 6 axis and 9 axis IMUs include a triaxial accelerometer which will be integral to our data collection. **Since our design requires positional and gyroscopic data regarding hand placement, we found it optimal for space and weight considerations to use a board with an on board IMU.** There is a constant gravitational force pointed downward, and the x, y, and z components of acceleration of the glove at rest should add up to this download gravitational force. Given each component, we are presented with a unique orientation of the hand at some angle.

Further expanding on our sensor reading requirements, **we need at least 5 analog input pins on our board to connect five flex sensors to.** Typical flex sensors (and all of the ones we looked at) need a power source, a connection to ground, and a connection to an analog input pin on the board. This means that the board we choose must have at least 5 analog input pins. **We also need at least one digital out pin for our haptic motor**. In addition, we would like the range of voltage values outputted by the flex sensors to be as wide as possible. This is to ensure that the feature vector corresponding to one gesture is as distinct as possible from another. To accomplish this, we need to appropriately choose the pull-down resistor value that each flex sensor will be connected in series with.

The next feature we wanted to be built into our board was Bluetooth capabilities, preferable at least 4.x LE for its improved latency compared to older models. If we were to use a board without a built in Bluetooth controller, we would have to attach one to it and they are typically the length and width of an Arduino Nano but much thicker than them. This would likely cause us to exceed our weight requirements and size use case requirements so we prioritized having bluetooth built into the board.

Finally, we preferred to purchase a board with a sufficient processor that would be capable of transmitting data as fast as possible to our remote MCU and, in a later prototyping stage, running our ML model on board. We determined that it is approximately 32MHz. With 25 readings per second, this would leave us with approximately 5 million clock ticks in between readings. This leaves a lot of room for other computation that we may want to move onto the board later on. The board we decided on is the Arduino Nano 33 BLE Sense Rev2. The range of its IMU is $\pm$16g. We decided to base our IMU selection on the board we choose (instead of first deciding which IMU meets our needs) because this project does not require a state of the art accelerometer and we felt that the one mounted on the Arduino Nano 33 BLE Sense Rev2 satisfied our sensitivity needs.

We then needed to determine latency targets for our data flow and transmission to get a better idea of how we will reach our target use case requirement of .5 seconds from sign to speech. For general device communication like sensor readings and based on BLE specifications, we can expect latencies of about 10 ms to 100 ms. Based on this data, **we will strive to achieve a sign to ML input latency of 100 ms**. We will achieve this by making the data we send as compact as possible when formatting sensor readings. **We also determined that we will allow 25 ms for our ML model to output a prediction** based on a past capstone project's results, and we will wait for 8 consecutive gestures that are the same to classify the predictions as a gesture with confidence. This gives us a total of 200 ms to classify a sign. **Then we will then strive for 200 ms for data to go back to the gloves for haptic feedback and speaker output.** Bluetooth latency for fully wireless headphones can range from 100 to 300 ms so we will set our requirement to 200 ms since we are not transmitting high quality audio data but want a conservative estimate. Adding these values together gives us 500 ms, our use case latency requirement. We will attempt to minimize latency in all cases and identify bottlenecks that can help us go below this number if possible. If we find that Bluetooth is a bottleneck, we will consider moving to a WiFi communication protocol with the Arduino Nano 33 IoT. See 5 for more information.

The ML model should be able to perform inference at a rate of less than 25ms per prediction. This metric is directly tied to the speed use-case requirement. We predicted that the bulk of the time delay will stem from the latency caused by the wireless transmission. With this in mind, we hoped that the classification section will only take up less than half of the allotted 500ms requirement from sign to prediction. Then, since our final classification module might require repeat ML model predictions, we divided that time by 8 to arrive at our design

goal of 25ms.

The ML should be able to achieve a testing accuracy of 90% on excluded test data. This is directly tied with the accuracy use-case requirement. We plan on adding additional checks on the output of the ML model to make the entire system more robust to noise and interference. With that in mind, we at least wanted to start out with a solid baseline with a model that performs relatively well as a standalone. We arrived at 90% because we believe that we can correctly fix the small case of errors with our additional final classification paradigms we will present later.

Our software should be ready to collect around 2000 datapoints per letter with a variety of different signers. This design requirement is directly tied to our accuracy requirements and indirectly tied to our desired vocabulary size. The main hope with this requirement is to establish enough data to train all our ML models sufficiently while leaving enough wiggle room for additional testing or cross-validation. By seeking a variety of signers to collect our data, we hope to make the model more robust to achieve our use-case requirement in accuracy.

5 DESIGN TRADE STUDIES

Glove Sensors vs Computer Vision

When we came up with our initial goals to enable communication from deaf to non-deaf speakers, we were immediately faced with a decision to make: whether to use a glove-based sensor design or a camera with a computer vision backend.

A computer vision based approach would've involved having a camera take images of someone else while they are signing. The resulting images would be passed into a CV/ML backend for classification. The pros of this method is that the high dimensionality of the input data allows for more minute movements to be detected. Even slight flexions in a particular finger might be detected due to the nuances of CV techniques. Additionally, we would most likely be able to make a less user-impactful product, meaning that a camera alone would most likely be easier to carry. However, the cons are that the CV setup requires a lot of training and a consistent background environment. Due to the high dimensionality of images, CV methods often require collecting a huge dataset to train the proper CNN architectures for performance. Collecting data can be very labor-intensive due to the time collecting the data as well as the additional effort to label each image. Additionally, there is a requirement to match the background and framing of the training data. If this requirement isn't match, performance is very likely to suffer. This makes this product extremely hard to use in the wild.

A glove-based approach involves using flex sensors and IMUs to detect the positioning of the fingers and hand to predict the corresponding signed letter. The pros involve a quicker training process and a system that is more robust to varying environmental factors. The quicker training process stems from the smaller dimensionality of the data, namely only the flex sensors and the IMU. And since the readings will be relatively consistent regardless of varying conditions, we can expect a similar performance at all times. The cons are the sensitivity of the sensors. Being less information dense than an image, we are at the mercy of the sesnors being able to convey enough information to make accurate predictions.

At the end of the day, based on previous groups' success, and the more robust design, we decided to choose our glove sensor to tackle the problem of deaf communication to non-deaf speakers.

New User Calibration / Adaptability

One of the initial considerations for the project was to develop an additional calibration mode for new users to adjust the ML model backend to fit the current user more closely. This was inspired by similar designs like Apple's face recognition or fingerprint identification. The premise is that the way that people vary from person to person and the classification might suffer if the variations differ from the training set too much.

Ultimately, we decided to put adaptability as a stretch goal of ours. This was due to our belief that the variability will be minimal enough to still maintain a relatively high performance even with outside testing input. If we have enough slack time, this would be a feature we would definitely implement to maximize accuracy across different users.

Choice of Sensors

We have decided to use Spectra Symbol's 'Original Flex Sensors' for a variety of reasons, the first being precedent—this type of sensor was most commonly used by similar projects. The second primary reason was cost. Since our design has two gloves, we need 12 sensors (ten for each finger, two for slack), so cost was important to keep in mind, especially if the goal of this project in the distant future is to be as accessible as possible. Spectra Symbol has since come out with the more lightweight and sensitive 'SpectraFlex Flex Sensors', but these were $12.5 more expensive than the sensors we went with. We were also looking into Bend Lab's flexible soft sensors, which were appealing due to their silicone-based material that seemed to offer higher sensitivity, but at $49.00 a sensor this was unfeasible in terms of cost. In addition, we considered other sensor options, like Hall effect sensors and conductive thread based sensors but opted against these they were not as common and used by research groups with significantly more resources.

BLE Sense Rev2 Board vs IoT Board

As described in the 4, we are using the Arduino Nano 33 BLE Sense Rev2 board. This board has a 64MHz clock fre-

quency, BMI270 and BMM150 IMU's on board (totalling to 9 degrees of freedom) giving us accelerometer, gyroscope, and magnetometer readings. It was stated that we are considering using the Arduino Nano 33 IoT in the near future due to its WiFi communication capability. WiFi is faster than BLE, which is why we will utilize it if we fail to meet latency use case requirements with Bluetooth. The reason we are not using WiFi right off the bat is because Bluetooth is simpler to use and the BLE Sense board has more computing power. Since we plan to move some computation to the board in our final prototype as described in 7, it was clear that our priorities meant that the BLE Sense board was more appropriate. However, the 48MHz clock speed on the IoT board is not trivial and we may be able to take some advantage of its compute power if we BLE does not allow us to meet latency requirements.

One other factor we considered is that the IoT board's IMU is the LSM6DS3 which gives us 6 degrees of freedom whereas the BMI270 and BMM150 on the Sense Rev2 gives us 9 (additional magnetometer). While we do not foresee using a magnetometer currently, it may be a datapoint of interest if we decide to change the kind of data we gather at any point. This may become useful if IMU and flex readings for two gloves do not give us the accuracy we are looking for causing us to want data about how far the hands are from each other using a magnet. This additional benefit of the BLE Sense further confirmed that we should start with that board.

Presence of a Remote MCU

Part of our MVP design is that we have a laptop running the classification model as well as noise reduction. Past projects have used a smartphone or laptop to show proof of concept speech output, however we believe the product will be more versatile and useful if there is no dependency on these devices. However, we are aware of the complexity that such a change could bring to our system. Therefore, our system implementation discusses how we will achieve wireless, independent functionality only after our MVP requirements have been met.

6 RAPID PROTOTYPES

We have adapted a rapid prototype approach to completing our product within the given class time.

Prototype 1

Our first prototype will be a one glove design with flex sensors, battery, and Arduino mounted on the glove. One the team members' laptops will be used to do computation and speaker output will be from the laptop. The "haptic feedback" will initially be an LED lighting up on the Arduino to visually inform the user that a sign was successfully converted to speech. Communication will be done via USB for this prototype. Our design will be able to interpret 10 ASL letters. We chose this number because it is small and does not require an ample amount of data to be collected but it is large enough to give us some idea of our system accuracy. We chose ASL for this prototype due to its single handed alphabet as opposed to BSL's double handed.

Prototype 2

Our second prototype will have wireless communication (initially Bluetooth but optionally WiFi if we do not meet latency requirements). We will duplicate our working glove with one having a haptic vibration motor attached and the other with a speaker. This design will be trained against data representing the 26 letter double-handed British Alphabet as described in section 2.

Prototype 3

This final prototype will eliminate the need for a remote MCU and all computation will be moved to the glove after the model is trained. A laptop will be needed to train the model but glove functionality beyond this will be completely independent and wireless. We will have each user calibrate the glove with their own style of signing with a laptop to achieve maximum accuracy for each person given that everyone signs slightly differently.

7 SYSTEM IMPLEMENTATION

Physical Design

A significant step in our architecture design was deciding how to mount everything on the user's hands. Below is a sketch of where each component would be placed. The Main Compute Unit is not pictured.

As described, the speaker and vibration motor are split between the two hands. Each glove will have the battery attached to the wrist, the board will be vertically attached such that the USB cable can be easily accessible for code upload, and the flex sensors reach beyond the knuckles. Not pictured are the solder boards on each hand, resistors that will be connected to the flex sensors to modulate voltage readings, wires to connect components together, and the gloves that all of these pieces will be mounted on.

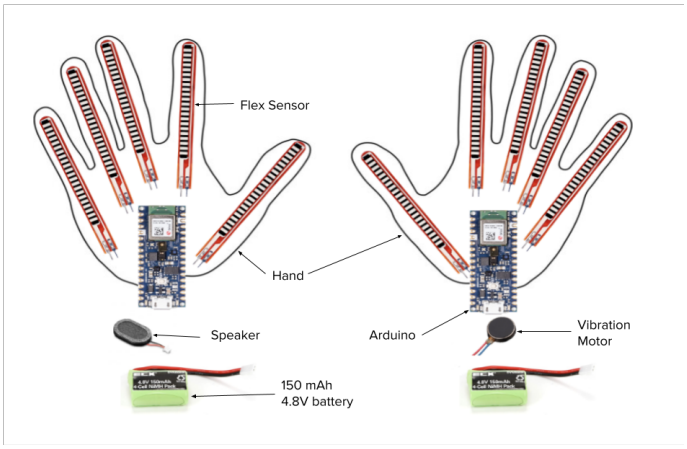


Figure 4: Component mounting design.

Flex Sensor Integration

Flex sensors are typically used in conjunction with a pull-down resistor to form a voltage divider circuit. This is because a flex sensor is a variable resistor that changes its resistance based on the degree of bending. As such, a voltage divider configuration allows for the conversion of the resistance variations into a voltage signal that can be easily read by the Arduino's analog-to-digital converter. In addition, the pull-down resistor ensures that there is a defined voltage range across the flex sensor—we want as large of a range as possible as this implies distinctiveness across the data readings for each sign. In terms of unit testing, we will test a series of resistors within a predetermined range with a few of the letters in our MVP letter set to see which produces the largest range of V_{out}. The flex sensors we chose have a resistance range from $\approx$ 10,000 Ohms (flat resistance value) to 20,000 Ohms (2x the flat resistance), and our power supply is 4.8V.

Below is how we determined what this range will be, given that we ideally want a voltage output range of 1V to 4V:

$$1 \leq 4.8V \cdot \frac{R_{pull}}{10,000 + R_{pull}} \leq 4 \tag{1}$$

Solving for R_{pull}:

$$10,000 + R_{pull} \leq \frac{4.8R_{pull}}{1} \leq 4(10,000 + R_{pull}) \tag{2}$$

Lower Bound: $10,000 + R_{pull} \leq 4.8R_{pull}$
Subtract R_{pull} from both sides:

$$10,000 \leq 3.8R_{pull} \tag{3}$$

Divide by 3.8:

$$R_{pull} \geq \frac{10,000}{3.8} \tag{4}$$

Upper Bound: $4.8R_{pull} \leq 4(10,000 + R_{pull})$
Distribute 4 on the right side:

$$4.8R_{pull} \leq 40,000 + 4R_{pull} \tag{5}$$

Subtract $4R_{pull}$ from both sides:

$$0.8R_{pull}le40,000 \tag{6}$$

Divide by 0.8:

$$R_{pull} \leq \frac{40,000}{0.8} \tag{7}$$

Thus, by simplifying the inequality, the pull-down resistor range becomes $2631.51\Omega \leq R_{pull} \leq 50000\Omega$. The most readily available resistors within this range are $2.7k\Omega, 3.3k\Omega, 4.7k\Omega, 10k\Omega$ and $47k\Omega$, so we will test all of these. Below is a circuit schematic of our flex sensors' wiring.

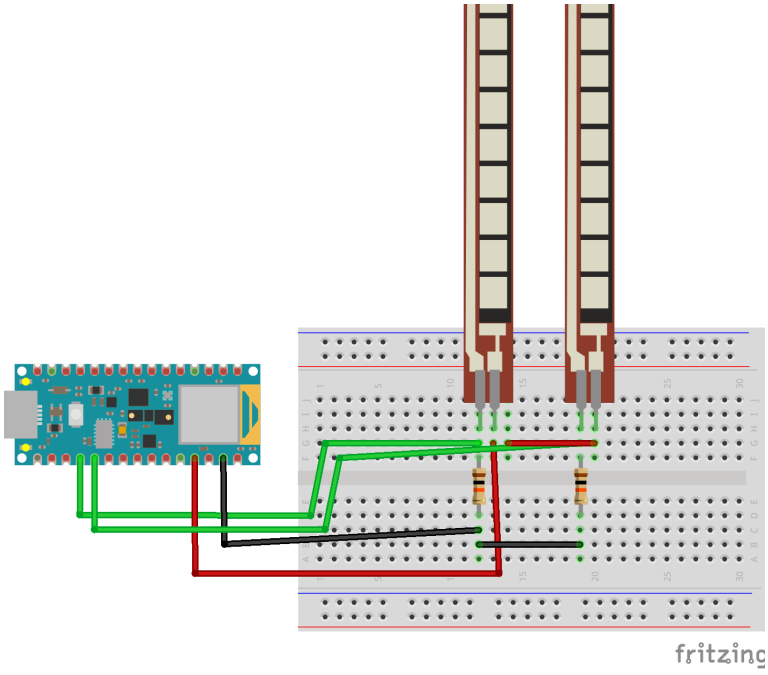


Figure 5: Flex sensor circuit diagram.

IMU Sensor Integration

When deciding how to format IMU sensor data, we considered a few factors related to our ML model's performance. Firstly, this step is beneficial because, of course, it minimizes noise, regularizes our data preventing potential model overfitting, gets rid of outliers, and reduces the size of the message we send to the MCU. However, we lose some details about our data and may require a more complex model to recognize the trends and patterns in the data.

We felt that using .5 increments for our digitization would provide enough precision to prevent the drawbacks from occurring as much as possible while also benefiting from the positive aspects of this noise reduction step. Furthermore, we felt that removing outliers in data was a step that would not negatively impact us too much.

Data Communication

Our primary mode of communication between gloves will be initially wired USB for our prototyping phase and eventually wireless for our MVP. We will perform extensive experiments to meet latency requirements with Bluetooth communication and compare its performance against WiFi. More information on this can be found in 5. In Prototype 2, we will still have the presence of a remote MCU doing computation. However, in Prototype 3, one of the gloves will be assigned this responsibility and data from Glove 1 will be sent via Bluetooth to Glove 2. In both cases, data will be timestamped rounded to the nearest 1/25th of a second (25 readings are sent per second) and only data that is matched by timestamp will be processed and sent through the ML model.

Data Collection

We will implement a rigorous number of scripts in Python to handle data collection from all possible avenues. Recall, though our system will be primarily implemented for bluetooth (wireless) performance, we will have access to a USB connection to the microcontroller as well. Therefore, we plan on designing a suite of scripts that can handle data input from both a wired connection and a wireless connection. Within the system, the data from the flex sensors and IMU will be formatted in a pre-definied, consistent way to make sure there are minimal issues with data compatibility and reformatting. We will develop multiple scripts to handle collecting data one class at a time or a consecutive script that iterates through all classes for a full dataset collection in one shot.

As of right now, we will be planning on using the raw data as the features to be passed to the various forms of training. We will evaluate issues with numerical stability and look into normalizing the data if we run into significant issues.

For Prototype 1, the data that we collect will be from a single glove, so there will be less features to collect. For Prototype 2, we will be collecting more features from two gloves so additional time synchronization will need to be considered during data collection.

Noise Reduction

We will implement a software module in Python that will be responsible for handling the initial input from the microcontroller to the laptop / main computing unit. As of right now, this module will primarily implement a simple filter for outliers. The bounds for the outliers will be determined by an EDA of the training dataset.

Additionally, we will perform a precursor step on the glove Arduinos to minimize noise. The IMU data and flex sensor readings will be analog floating point numbers. We will convert this data to digital data by rounding down to the nearest half integer value.

The last step we will take to reduce noise is eliminate incomplete input data when there are inconsistencies between the two gloves and what data they send. For example, if glove A is down for three seconds, glove B's input will be disregarded for those three seconds. Additionally, we will check all input data against a validator to ensure that its information was not garbled in transmission. This can be done with a simple conditional chunk of code before data is extracted from the input strings and sent to our ML model.

Machine Learning Classification

The primary backbone of the software component will be focused on the classification algorithm. Since this is one of the most important components of the system, we will be experimenting with four models and then select the one with the highest performance. The four models we will be evaluating include: neural networks, SVM, kNN, and decision trees/forests.

The training of models will be handled directly from the laptop's computing unit. We envisioned that our dataset will not be big enough to warrant the use of a GPU or learning accelerator. If it turns out the training / inference is taking too long, we will explore options including cloud computing platforms like AWS or Google Colab. Otherwise, the plan is to train four models with the same data that was collected from the collecting data routine. The trained models will be saved in a separate directory, and can be loaded for additional training or test-time inference. Saving of models will take leverage of the *joblib* and *pytorch* packages.

Within the training of models, there will be a heavy emphasis placed on cross-validation training. In each model, there are specific hyperparameters that need to be tuned. For NN, this includes the model's activation function and learning rate. For SVM, we can look at the kernel and the tradeoff coefficient. For kNN, we will look at the k value, or the number of neighbors to consider. For decision trees/forests, we will look at branch and leaf numbers.

For inference, we will have a final script that will load in a pre-selected model that we have fully trained. This model will continuously receive inputs from the microcontroller and make predictions on the letter that is being signed. Then, it will pass the predictions to another module responsible for final classification.

Final Classification Module

To prevent potential issues from noise or transition states between different signings, we plan on implementing an additional module that will take in the output from

the ML model and make a final prediction on the signing of a letter. This module will collect and store a number of the past ML model outputs and identify patterns to predict if a sign is actually being signed. We will primarily implement this as a repetition check. This means that we will like to see a specific number of repeats of the same classification prediction before we are confident in the ML model. The exact nuances of the strategy will be adapted based on their performance. This idea was inspired by past implementations from Gesture Glove [2] .

Feedback System

Once a classification is generated by our classification module, we will send a signal to each glove indicating the event that has occurred.

Haptic Feedback

When a sign is successfully classified, we will send a signal that indicates a success to the glove with the haptic vibration motor. This success signal will translate into two quick pulses with the vibration motor. In any other case, no signals or pulses will be sent. This design minimizes confusion because there is only one type of indication to the user of an event occurring. Our .5 second latency requirement ensures that the user will not be waiting for a long time just to "feel" back some indication of a successful signing.

Audio Output and Text to Speech

We will be using the pyttsx3 library to create audio signal data. For Prototype 1, the output will be coming from the laptop. For Prototypes 2 and 3, it will come through the speaker mounted on the hand.

8 TEST & VALIDATION

Again please use the guidance on Canvas and the Word template for what to include in this section.

Unit Tests

Unit Tests for Sensors

We want our sensors to have as high sensitivity as possible, as low sensor sensitivity is commonly listed as being a challenge past projects faced. As such, we will test a spectrum of pull-down resistors to be used in conjunction with the flex sensors that exist within a specific range (see System Implementation for the mathematical derivation of this range). We will test each resistor value on a few signs that are distinct in shape, and whichever resistor yields the widest range in voltage output we will select for our final glove circuitry. In addition, if we still encounter low sensitivity issues, it has been show that the use of op-amps can mitigate this, as they can amplify the small voltage variations produced by the flex sensor to a more measurable and usable level.

Tests for ML Model Speed

It is extremely important for us to evaluate how fast the ML model can make predictions at test time. Because we are planning on passing input into the model sequentially, we plan on testing by having one input passed into the model. Then, we will leverage the *time* package in python to calculate the time it took for the ML model to output a prediction. We will perform this for all the letters around 50 times each. Then, we will calculate the average time elapsed as our final metric to see if the ML timing requirement was met.

Tests for ML Model Accuracy

To test accuracy, we will depend on a held out set of the original dataset that we collected that was not used for training. This held out set will hold an equivalent amount of each letter and we hope to make it about 20% of the original dataset. We will proceed to run classification on this held out test set, and report the accuracy on those datapoints. This metric will then be used to verify the accuracy metric of the ML model.

Integration Tests

Double Glove Integration Tests

A major step in our project will be duplicating our initially working glove to create the double glove design. This will introduce error and synchronization issues that may affect overall system latency. Since we are dropping glove sensor information that does not have corresponding sensor information from the other glove (matched by timestamp), we may see longer times for sign to speech latency. We will perform rigorous timing analysis once we integrate two gloves and optimize specific parts of our program if we run into major issues. Timestamps and latency will be measured in the same way that they were for our single glove design. Important things to consider are minimizing data loss or data sending failures which would cause us to drop information as well as being able to establish two reliable wireless and almost identical connections to both gloves in terms of transmission latency and bandwidth.

Our ML model and amount of data flowing through our system will also be more complex and bulky with this change. We will consider this when measuring latency during this integration step.

Use Case Tests

Battery Life

Our product is designed to run for two hours. We will test over a long duration of time in one shot to measure that

the time passed is two hours or very close to it. We will also measure time during work sessions until a fresh battery runs out and ensure that the time we worked before the battery ran out adds up to very close to two hours.

Weight

We also want to ensure that the weight of each glove is less than 100 g. Each component in our design added together is approximately 50 grams with some give or take. Adding wires and more slack gives us a 100g total that we will stay under at every prototype completion checkpoint.

Testing Overall System Accuracy

This will be the main test for our use-case requirement. There will be two tests performed. One test will involve performing an equal amount of each letter and recording the glove's response to each signing. This data will be used to create a confusion matrix and be used to determine the average accuracy by calculating the one's correctly predicted divided by the total amount of signing. The second test will involve taking a brief passage from a randomly selected text and signing each letter as appears until we hit 200 letters. Then, we will perform the same accuracy calculation. This method will allow us to both evaluate the performance of the glove where all the letters are weighted equally, and a situation that is more common to the average speaker in terms of letter frequency. Both these metrics will be compared to our use-case requirement.

Testing Overall System Speed

This will be the main test for our use-case requirement of speed of prediction. To do so, we will plan on having one of use sign each letter in the BSL alphabet 5 times. For each signing, we will only focus on the times where there was a speaker output, when the system has made a firm prediction. In these cases, we will look at the time-stamps for when the computing unit received the initial sensor input that led to the prediction. Since this is all done in the software, we will use python's time tracking packages to determine the difference in time from the input to the classification output. Then, we will average all the different time ranges measured to calculate our final value that we will compare to our use-case requirement.

9 PROJECT MANAGEMENT

This section describes your project schedule, team member responsibilities, bill of materials with budget and risk mitigation plans.

Schedule

The bill of materials is shown in Fig. 7.

Team Member Responsibilities

Ricky will primarily be working on the ML integration. Ria will be working on data communication and overall software design. Somya will be working on sensor interfacing. We will collect data, test, and perform integration altogether.

Bill of Materials and Budget

The schedule is shown in Table 1.

Risk Mitigation Plans

If the speaker output is not clear enough or loud enough for our MVP, we will move that to our Prototype 3 design and iron out those details in that phase. We also have a plan to use WiFi as opposed to Bluetooth if Bluetooth is too slow. While the IoT board's processor is strong, if it is far from satisfying our latency requirements, we will stick with our MVP design with the remote MCU (laptop).

10 RELATED WORK

Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays [3] This project was a collaboration between UCLA and Chongqing Normal University to develop a new wearable system that can detect a variety of signings. Their system consisted of a singular glove paired with additional sensors on the face to detect facial expressions. Their system primarily innovated on the development of their own yarn-based flex sensors to track the finger flexion. It also innovates on the use of additional facial sensors to convey more meaning than can be conveyed with solely the hands.

Although their research was a significant undertaking with many collaborators, we hope to innovate on their work by incorporating the use of the second glove. We believe that if we can demonstrate a system with two gloves, we can greatly improve the possibilities for communication between deaf and non-deaf speakers.

Gesture Glove [2] This was a past capstone group who also sought to develop a glove that could read in ASL sign language and convert it to speech. Their design uses flex sensors, IMU, and tactile sensors as input. These are processed using ML to predict letters at output. Notably, their design requires a wired connection to their laptop for performance.

We hope to use their insights and experience as a springboard for us to achieve higher accuracy and speed metrics. We also hope to innovate from their past design by implementing the bluetooth (wireless) component so users can sign more freely.

Table 1: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Flex Sensors	182	Adafruit	12	$12.95	$160.55
Arduino Nano 33 IoT	ABX00032	Arduino	1	$27.00	$187.55
Arduino Nano 33 BLE Sense Rev2	ABX00069	Arduino	1	$39.24	$226.79
Pair of Cotton Gloves	N/A	COYAHO	3	$6.99	$233.78
USB to USB-C Adapter	N/A	JSAUX	1	$8.09	$241.87
					$14.00

11 SUMMARY

To summarize, we have provided in this report a comprehensive overview of how our gloves will be designed from a hardware, signal processing, and software perspective. We have provided justification for each critical design choice, namely our selection of sensors/computing units, as well as the machine learning training scheme we plan on using to enable accurate gesture identification. We have made these decisions keeping the user's experience in mind, and look forward to creating a novel product that represents an improvement and is well-substantiated by the work of similar groups.

Glossary of Acronyms

- ASL – American Serial Bus
- AWS – Amazon Web Services
- BLE – Bluetooth Low Energy
- BSL – British Sign Language
- CNN – Convolutional Neural Network
- CV – Computer Vision
- EDA – Exploratory Data Analysis
- GPU – Graphics Processing Unit
- IMU – Intertial Measurement Unit
- IoT – Internet of Things
- kNN – k Nearest Neighbors
- MCU – Main Computing Unit
- ML – Machine Learning
- MVP – Minimum Viable Product
- NN – Neural Network
- SVM – Support Vector Machine
- USB – Universal Serial Bus

References

[1] Ursula Bellugi and Susan Fischer. "A comparison of sign language and spoken language". In: *Cognition* 1 (1972), pp. 173–200. DOI: 10.1016/0010-0277(72)90018-2.

[2] Sophia Lau, Rachel Tang, and Stephanie Zhang. *18-500 Design Review Report - Oct 11, 2021*. Gesture Glove. 2021.

[3] Zhihao Zhou et al. "Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays". In: *Nature Electronics* 3.9 (2020), pp. 571–578. URL: https://www.nature.com/natureelectronics.

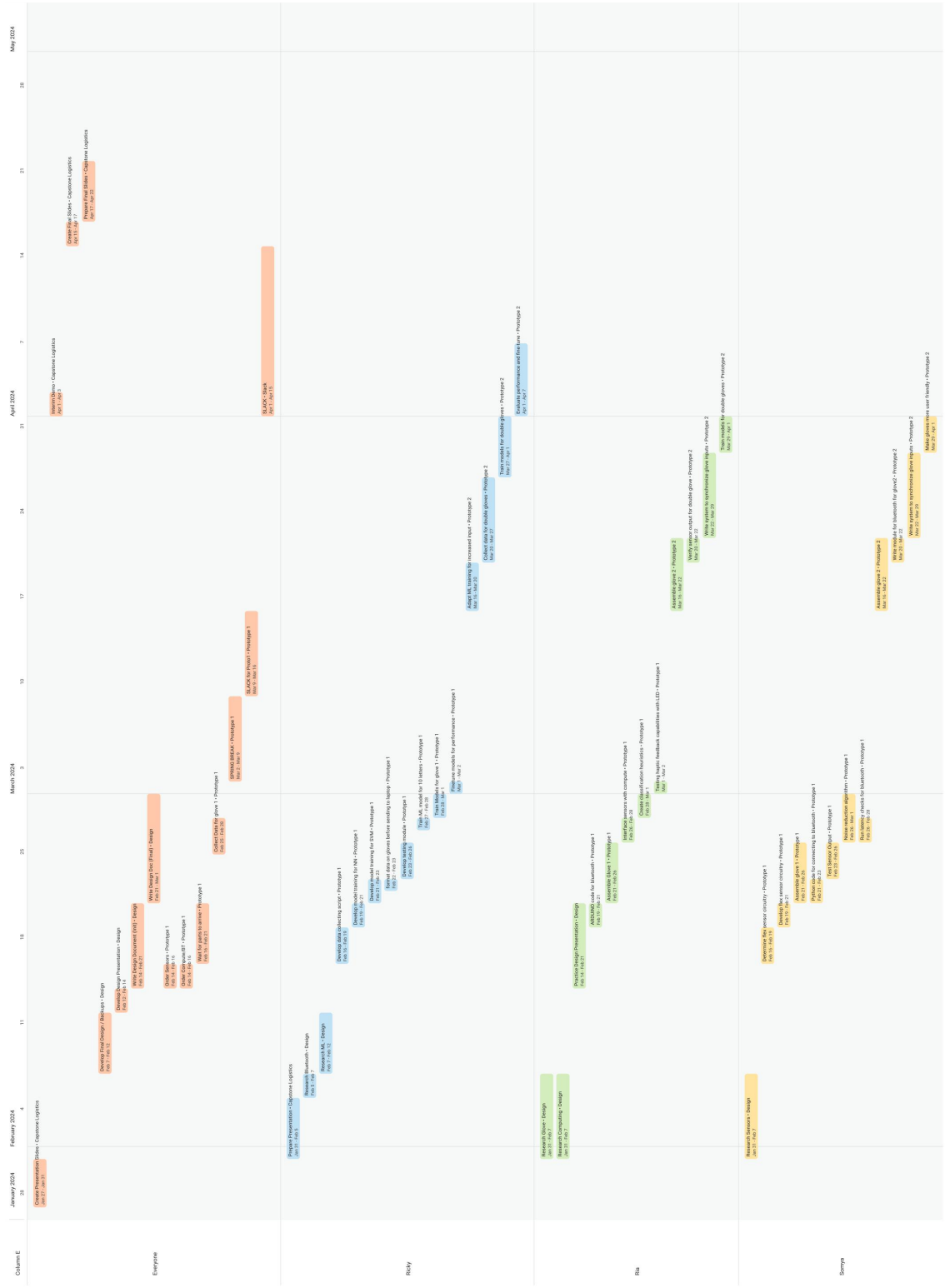


Figure 7: Gantt Chart