University of Waterloo

Faculty of Engineering
Department of Computer and Electrical Engineering

Final Report

Voice - Device to Enable Real-time Sign Language to Speech Translation

Group 037

Akshay Budhkar (20457593)
Eliot Chan (20462016)
Biraj Kapadia (20467909)
Amish Patel (20460828)

Consultant: Prof. Dana Kulić
February 10, 2017

Abstract

There are 375,000 culturally deaf people in Canada, and in-person communication barriers between the deaf or mute and the general populace still remain high. Many people do not understand sign language, and communicating through a text medium is arduous and impersonal. The goal of this project is to design a device to translate sign language to speech in real-time, allowing the deaf or mute to communicate naturally even with those who do not know sign language. Voice uses custom-made ergonomic gloves equipped with accelerometers, flex sensors, and an onboard microcontroller to track and stream data on a user's hand positions and motions to a Bluetooth-connected mobile application. The application uses an ensemble of machine learning models to recognize a given sign and speak the word, mimicking true speech. Voice is portable and unobtrusive compared to current computer vision variants, requiring the minimum of hardware, giving a deaf or mute person the voice they never had.

Acknowledgements

The group would like to acknowledge its advisor Dr. Dana Kulić, whose experience in machine learning and human-computer interaction was vital to this project's success. The group also recognizes David Bell's help in guiding us through the safety inspections of the glove and Laura Winger for providing us a near-endless supply of wires. Finally, the group thanks the University of Waterloo work term report guidelines for providing a well-formatted structure of presenting this report.

Table of Contents

List of Tables

List of Figures

1 High Level Description

1.1 Motivation

There are 375,000 culturally deaf people in Canada [1], and an estimated 1,250,000 deaf in the United States [2], and in-person communication barriers between the deaf or mute and the general populace still remain high. As technology and society have advanced, accommodations for the culturally deaf have slowly improved, but members of the culturally deaf community can still only interact with others naturally in online or other text-based environments. Communication between a native sign language speaker and a person who does not know sign language still requires a translator, specialized hardware [3], or a reduction to a commonly accepted, but less personal medium - such as through text on a phone. There is no efficient and natural way for native sign language speakers to communicate with non-speakers.

1.2 Project Objective

The objective of this project is to break down communication barriers between the culturally deaf and the general populace by designing and building a system to allow sign-language speakers to communicate with non-speakers in an unobtrusive and natural way. A user's sign language symbols should be translated in real-time and converted to audio output to mimic speech.

The system should be composed of a small number of hardware components, and ideally utilize readily available (i.e. a smartphone). Current sign language translation systems are bulky and obtrusive, either covering the majority of a user's arms or require additional hardware such as cameras or laptops. This is obviously cumbersome, and it is of utmost importance that a user has minimal interaction with the system itself.

The solution aims to support a subset of American Sign Language. The translation functionality should be moderated by an accompanying smartphone application, allowing the user to control when they want the system to be actively translating.

1.3 Block Diagram

The proposed solution involves building custom-made gloves fitted with flex sensors, an accelerometer and a gyroscope to gather data during a user's signing motions. A microcontroller converts the analog signals from the flex sensors to digital signals, and aggregates that data with the digital data from the accelerometer and gyroscope. The digital data is then fed into the mobile subsystem via a Bluetooth connection. The data from the external subsystem is transformed into the format compatible for the Machine Learning subsystem which will analyze the given data and detect what word(s) the sign corresponds to. A high-level overview of the product is given in Figure 1. It consists of two main subsystems: the External Subsystem, the Mobile Subsystem and the Processing Subsystem.

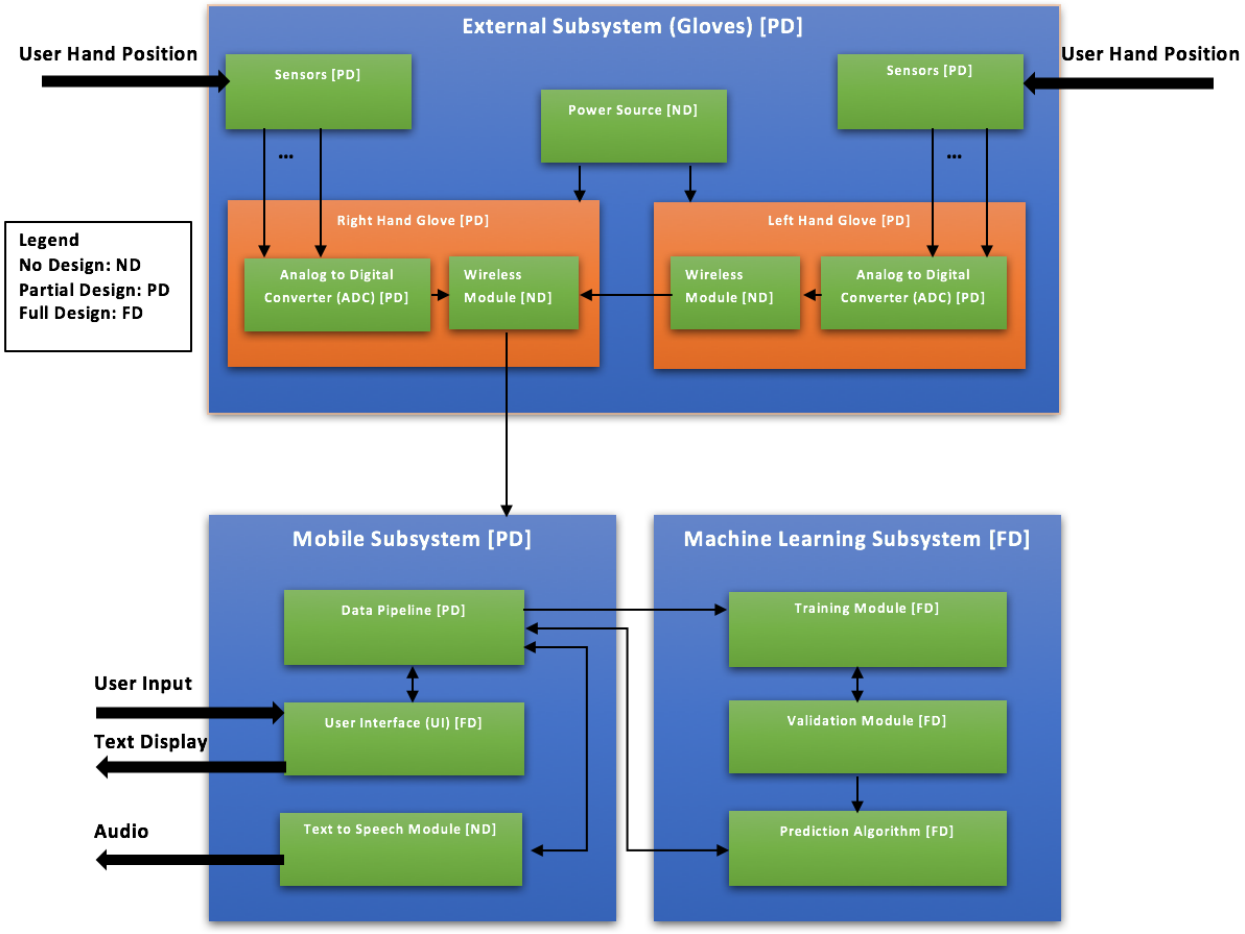

Figure 1: The block diagram of the system and the connections between each elements.

1.3.1 The External Subsystem

Flex sensors, accelerometers, gyroscopes, and magnetometers are used in the analog sensor array. Flex sensors are fitted on the fingers of the glove to determine the curvature and movement of any given finger - a total of ten are required, one for each finger. An accelerometer on each glove measures the 3-dimensional motion of the hand. A gyroscope on each hand allows for the tracking of the orientation of the hands. The magnetometers are used for calibration of the other two sensors and are used to prevent drift of the data. Inputs from these sensors are fed into a microcontroller, converting the signals to digital format. This data is then streamed to the Processing Subsystem for interpretation. The external subsystem also requires a power source, a dual-port battery pack which is used to power both the gloves simultaneously via micro-USB connections.

1.3.2 The Mobile Subsystem

The Data Pipeline Module is the receiver of the External Subsystem's outputs, and transforms the data to prepare it for the Machine Learning Subsystem (further explained in 1.3.3.). Predicted results given as output from the Machine Learning Subsystem are then communicated back to the Data Pipeline Module, which passes this information to the User Interface, and Text to Speech Module. The Data Pipeline Module also receives input from the User Interface to determine whether or not the entire system is currently active and translating. The UI can also display the transcript of the signs communicated by the user in real-time. The Text to Speech Module simply reads out the given text using the smartphone's speaker system to mimic speech.

1.3.3 Machine Learning Subsystem

The Machine Learning Subsystem consists of three distinct portions – the Training Module, the Validation Module, and the Prediction Algorithm. During the development of this prototype, the team manually created a set of training and validation data (two halves of the same set of data, randomly selected) once the External Subsystem and Data Pipeline Module were verified as working. The training data will be given to the Training Module to teach the model what particular signs look like, and the validation data will be used in the Validation Module to ensure that the model works. This process will be repeated multiple times to maximize the usage of the training set and to ensure there is little bias in the generated model.

Once the model is determined to be correct and accurate, the generated model will become the Prediction Algorithm, a completely separate entity that will be integrated into the accompanying smartphone application, acting as a real-time translation model using inputs from the Data Pipeline Module, and generating text outputs as a response.

2 Project Specifications

2.1 Functional Specifications

Table 1: Functional Specifications.

Specification	Type	Location	Description
Language Support	Essential	Processing Subsystem	Able to interpret 14 symbols in American Sign Language.
Latency	Essential	Entire System	Maximum latency of <800 ms from the end of any given sign.
Accuracy	Essential	Machine Learning Subsystem	Sign should be recognized with an accuracy of > 85%.
Mobile Application	Essential	Processing Subsystem	Controls the variant of sign language translated, can turn on/off the translation itself, and has the ability to display the text and output audio for the corresponding signs.
Wireless Support	Non-Essential	External Subsystem	Connects to processing device using Bluetooth or other wireless mechanism
Transcript Exporting	Non-Essential	Processing Subsystem	Capable of exporting the transcription of a session into one or more file formats.

Support for More Complex Signs	Non-Essential	External Subsystem	Capable of detecting 3 signs/symbols that require rotation of the thumb with respect to the palm.

Language support was reduced from 150 signs to 14 signs. The original specification was overzealous - one research team was only able to achieve an 85% accuracy with a 10-sign vocabulary [25].

2.2 Non-Functional Specifications

Table 2: Non-Functional Specifications.

Specification	Type	Location	Description
Hand Flexibility	Essential	External Subsystem	The gloves should allow a range of motion such that the user can complete any supported sign.
Portability	Essential	External Subsystem	90% of the surface area of the glove-mounted components should be on the glove itself. The glove-mounted components should weigh less than 100 grams.
External Subsystem Enclosed in Case	Essential	External Subsystem	Install a case to protect the microcontroller from environmental damage.
Battery Life	Essential	External Subsystem	Last one hour without exchanging/charging battery
Cost	Non-Essential	Overall System	The price of the entire system should not exceed $500.

3 Detailed Design

3.1 External Subsystem

3.1.1 Components

The external subsystem consists of five key components for each glove. The microcontroller, flex sensors, a combined gyroscope/accelerometer, a Bluetooth module, and an external power source.

3.1.1.1 Microcontroller

Voice considered a number of hobby microcontrollers for this iteration of the prototype. The microcontrollers considered had at least five analog inputs (for the flex sensors), one each of analog serial data (SDA) and serial clock (SCL) inputs (for a combined accelerometer/ gyroscope), a USB connection that allows for serial data transmission, and smaller than 70 cm^2, an approximate surface area of a palm [33].

The team used a decision matrix to determine the most appropriate microcontroller for the first iteration of the prototype. Other parameters were considered with the following weights (Table 3):

Table 3: Additional considered parameters for microcontroller for decision matrix.

Parameter	Weight
Surface Area	0.35
Clock Speed	0.30
SRAM (Static Random-Access Memory)	0.20
Cost per Unit	0.15

An ideal surface area would be low, allowing for a prototype with less intrusive electronic components. Clock speed and amount of SRAM would ideally be high, meaning we can poll and stream the data quickly. Cost is ideally low. Scores out of 10 will be assigned to each parameter per microcontroller. The following microcontrollers were considered: Arduino Nano, Adafruit

Metro Mini 328, and the Arduino Pro. These three options were considered as they were available in the ECE store or members of the team already owned them, allowing the team to begin prototyping earlier. Once aspects of the design's correctness has been validated, the team will re-evaluate microcontroller choices based on other parameters such as the size of the Arduino Sketch, the need for a USB input, the number of inputs, and power usage. Table 4 below shows the considered parameters per board.

Table 4: Considered parameters for each microcontroller. [4][5][6][7][8]

Microcontroller	Surface Area	Clock Speed	SRAM	Cost per Unit
Arduino Nano	8.1 cm^2	16 MHz	2 KB	$4.40
Adafruit Metro Mini 328	7.92 cm^2	16 MHz	2 KB	$14.95
Arduino Pro	27.78 cm^2	16 MHz	2 KB	$3.80

Table 5 below assigns scores to each parameter per microcontroller, and modifies the scores using the given weightings to determine the microcontroller that best fits the first prototype.

Table 5: Weighted decision matrix for microcontroller selection.

Microcontroller	Weighted Surface Area Score	Weighted Clock Speed Score	Weighted SRAM Score	Weighted Cost Score	Total Score
Arduino Nano	9 x 0.35 = 3.15	10 x 0.30 = 3.00	10 x 0.2 = 2.00	8 x 0.15 = 1.2	9.35
Adafruit Metro Mini 328	10 x 0.35 = 3.50	10 x 0.30 = 3.00	10 x 0.2 = 2.00	2 x 0.15 = 0.30	8.80
Arduino Pro	3 x 0.35 = 1.05	10 x 0.30 = 3.00	10 x 0.2 = 2.00	10 x 0.15 = 1.50	7.55

Given these results, the Arduino Nano was chosen as the microcontroller for this iteration.

3.1.1.2 Flex Sensors

Only one company produces commercially-available resistive flex sensors that can be bought in small numbers – SpectraSymbol. As there is no other competition, no true comparison can be made between possible choices in hardware. As such, Voice uses five SpectraSymbol FS-L-0095-103-ST flex sensors per hand to measure the user's finger curvature.

Since only five flex sensors are used per hand to measure the user's finger curvature there is the issue of potentially not supporting specific signs. One sensor is enough to measure the single degree of freedom in the 4 non-thumb fingers. However, one sensor is not enough to measure the two degrees of freedom available to the thumb. Since the thumb can bend and oppose, two sensors are required to capture all the possible signs that are possible to do with the thumb. However, the benefit of having the extra sensor may be limited - in the 100 most common ASL signs only 7% of the signs require a second degree of freedom in the thumb [9]. Care will be taken to ensure the supported signs do not rely heavily on the second degree of freedom in the thumb, and as such, signs that require an extra sensor fall under non-essential specifications.

3.1.1.3 Accelerometers/Gyroscopes/Magnetometer

As Voice needs to correctly determine Tait-Bryan angles of each hand, at the very least, it requires the usage of a combined accelerometer/gyroscope. Tait-Bryan angles represent rotations about three distinct axes and is vastly used in aerospace: yaw which obtains the bearing, pitch which gives the elevation, and roll which gives the bank angle. However, since drift is present in all yaw/pitch/roll calculations based on angular velocity measurements from gyroscopes due to the need to integrate the gyroscope readings, a magnetometer is helpful in allowing reorientation using Magnetic North and minimizes yaw drift [11]. As such, Voice uses a combined 9 DoF (degrees of freedom) sensor to create a relatively accurate estimate for orientation.

The microcontroller uses Kris Winer's implementation [13] of the Madgwick sensor fusion filter [12] for the LMS9DS0 (a particular inertial momentum sensor package that contains all three needed sensors), which was chosen as it has the highest update rates on a 16 MHz processor compared to the MPU-6050, the GY-80, the MPU-9150, and the MPU-9250 (other inertial momentum sensors [11]. The LSM9DS0 sensor has a built-in factory calibration system where the trimming values are stored inside non-volatile storage and inserted into the registers every time the device starts up [10], which allows for compensation in the sensors due to differences in

manufacturing of the components [14]. Winer's implementation also allows for additional calibration based on the minimum and maximum values of the sensors.

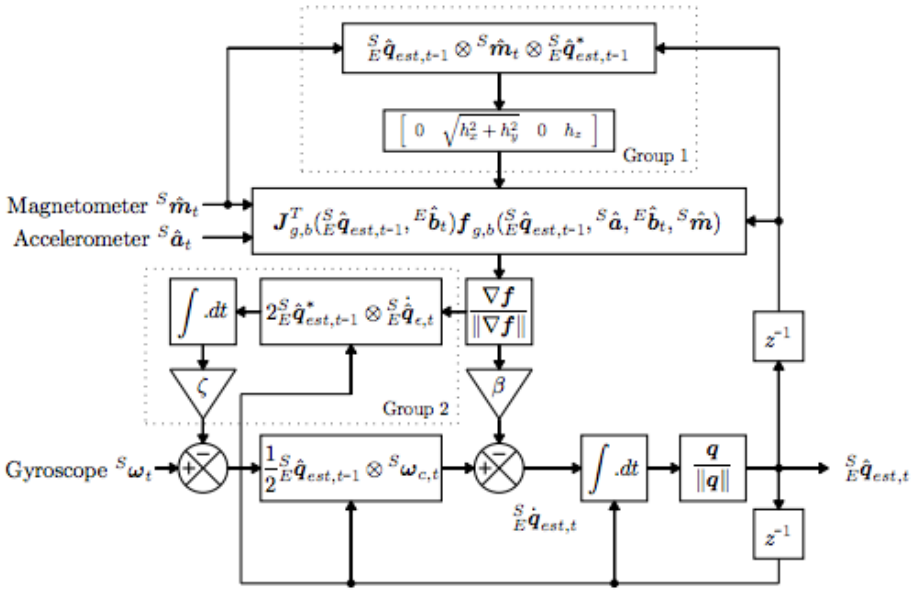

Figure 2: Block diagram representation for the Madgwick sensor fusion filter [12].

Figure 2 describes the Madgwick filter, which allow it to construct a quaternion (a four-component vector) representation of the estimate of current orientation using three inputs from the three available sensors, and compensates for both magnetic distortion and gyroscope drift. This quaternion is then transformed into yaw/pitch/roll angles with a relative accuracy of less than one degree [11].

The Madgwick filter corrects error by combining its estimate of orientation with the current values of its three sensors. The gyroscope output W (rads/s) can be thought of as the rates of change of the generated quaternion Q, the estimation of the sensor orientation. The error correction is performed by comparing the estimation of Q's acceleration component (A) to the actual acceleration measured by the accelerometer (A_{meas}). We also compare Q's magnetic field estimation (M) with the measured magnetometer value (M_{meas}).

$$Q_t = Q_t - 1 \times \left(1 + \frac{1}{2} W dt\right) + ((AQ - A_{meas}) + (MQ - M_{meas})) \times \beta \times dt \text{ [11]}$$

β is an adjustable parameter used to control the rate of convergence to a stable answer.

3.1.1.4 Bluetooth Module

There are two variants of Arduino-compatible Bluetooth adapters readily available on the commercial market – the HC-05, and the HC-06. These are manufactured by a number of differing manufacturers, though the specifications are essentially identical between the two models and many manufacturers. The major difference between the two is that the HC-05 is capable of performing both master and slave interactions, whereas the HC-06 is only capable of being a slave [39][40]. As such, the HC-06's command set can be seen as a subset of that of the HC-05's.

Voice uses a combination of HC-06 and HC-05 to create an array of communication interfaces between the gloves and the companion application on the user's phone. Since HC-06 and HC-05 only allow one-to-one communication between master and slave devices, Voice uses two HC-06 and one HC-05 module to make the communication between all the components possible. The first HC-06 module sends the data from one glove to the HC-05 module on the other glove. This data is then appended the data from the glove with the HC-05 module which is then transmitted to the user's phone via another HC-06 module which is mounted on the glove with the HC-05 module. To minimize cost only modules that are required master interactions in the serial communication are HC-05 modules, otherwise, they are HC-06 modules.

The Arduino sends data received from its sensor array using the Bluetooth modules over serial, and the companion application receives the data by establishing a Bluetooth socket connection with the respective slave, and then parses input by reading a particular window of bytes from the established Java InputStream.

3.1.1.5 External Power Supply

In order to power the glove without connecting it to a phone (which usually only has one external USB port), an external power supply becomes an essential component of the glove.

When the Arduino is powered using an external power supply it draws an absolute maximum current of 500 mA at 5 V [15]. Since all the components on the glove are powered using the regulator on the Arduino it is safe to assume that the glove will also draw the same current.

Therefore, if it is assumed that the Arduino is operating at its maximum capacity at all times, in order to satisfy the operating time specification (1 hour), the external power supply must have a capacity of at least 500 mAh at 5 V.

Voice uses an Aukey PB-018, a battery pack with a capacity of 12000 mAh and two power ports [16]. Its capacity is sufficient to power both the gloves simultaneously for a minimum of 12 hours, which exceeds the minimum specification.

3.2 Machine Learning Subsystem

3.2.1 Feature Extraction and Selection from Sensor Values

Feature extraction is a crucial data processing step required to filter out non-essential information for classification (predicting a sign).

Every sign generated by a person requires a certain amount of time. It is nearly impossible that the sensor data for any two instances of the same sign is the same at a given point in time. As such, the sensor values are measured over a time range rather than at a specific instance. For the purposes of building a machine learning model, the time window is dynamic. A one second pause between action is used to separate distinct signs.

Acceleration values tend to fluctuate significantly during movement. This is evident in Figure 3 and Figure 4- plots of acceleration sensor values for slight up-and-down and left-and-right movements, respectively. The values oscillate along a stable point, and the magnitude of the deviation from this stable point is indicative of the magnitude of the acceleration.

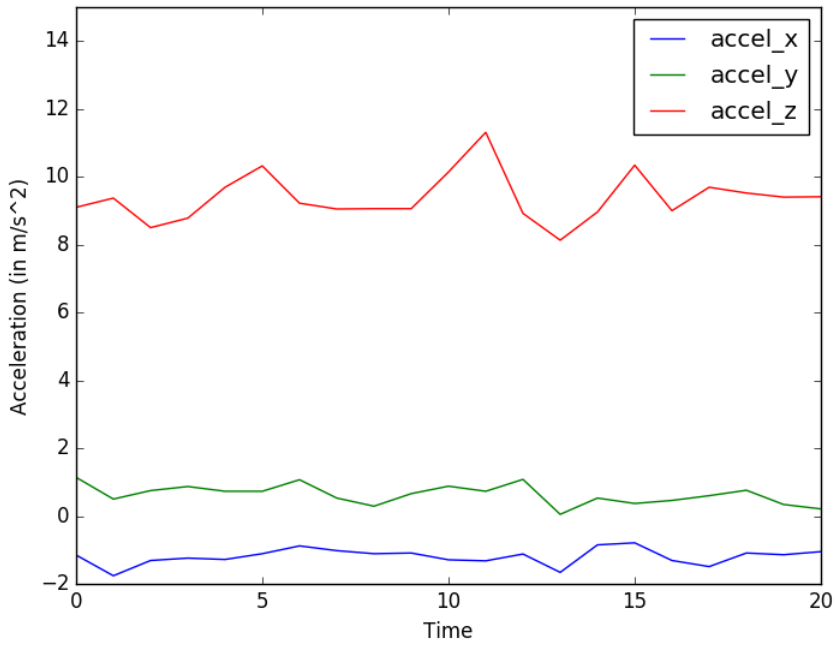

Figure 3: A graph showing acceleration values over time when moving the accelerometer up and down along the z-axis.

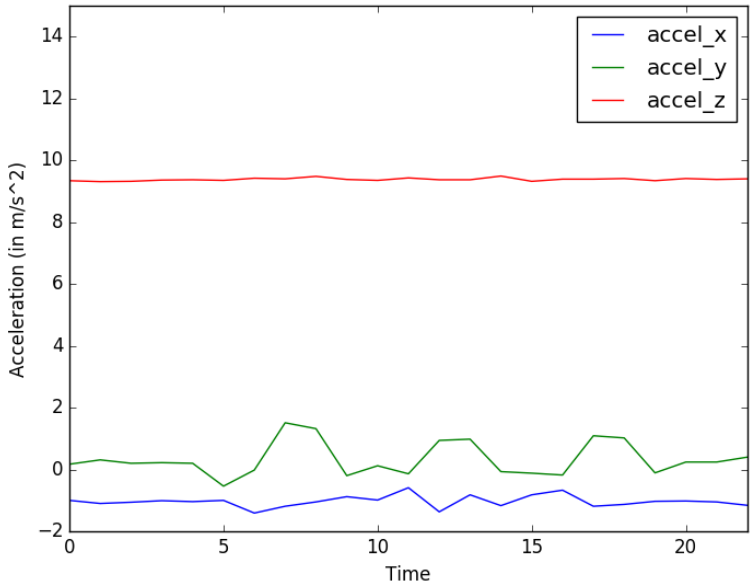

Figure 4: A graph showing acceleration values over time when moving the accelerometer left and right in the x-y plane.

It has been shown that data from a tri-axis accelerometer can be used to accurately predict motions in human activity recognition when they are extracted using the Discrete Cosine Transform (DCT) [17]. Thus, DCT will be used on each of the three sets of acceleration values obtained in a given time frame. For a given time window containing n sensor values denoted by $X = \{x_1, x_2, \dots, x_n\}$ the DCT is defined as follows:

$$DCT(X(k)) = \sum_{i=0}^{n-1} x_i \cos\frac{\pi}{n}\left(i + \frac{1}{2}\right)k \qquad k = 0, 1, \dots, n-1$$

Figure 5 shows the analog values received from a flex sensor attached to a finger. The finger starts in a straight stretched position, flexes all the way till the finger can't bend anymore and then returns to its original position. The resistance values clearly indicate the flex of the fingers by increasing as they flex, and decreasing as they extend.

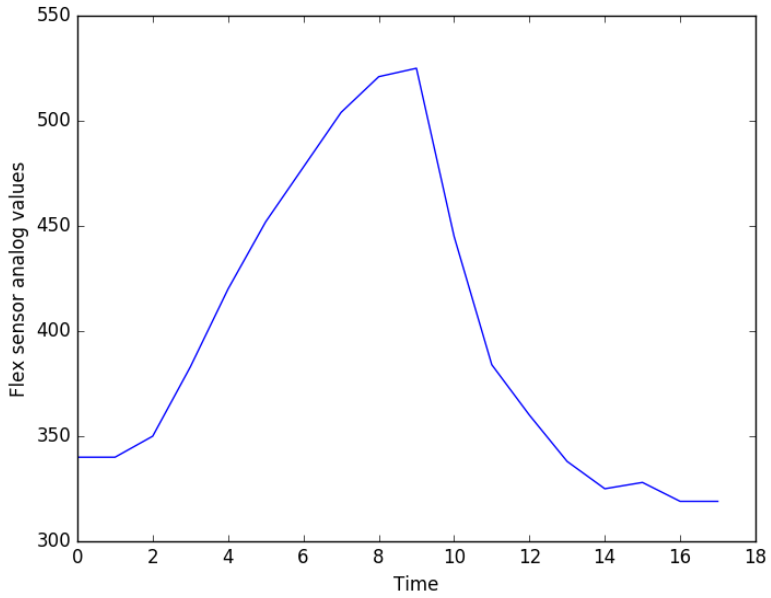

Figure 5: A graph showing flex sensor analog values over time when bending the finger from a stretched position to a fully flexed position and back.

Due to the large range of resistance values possible in the flexing of a finger, this information is incorporated as a feature using the standard deviation (SD, the average and the max values in a given time range) rather than using the raw values. The usage of standard deviation is a

recommended practice for machine learning models like neural networks [18]. For n sensor values denoted by $X = \{x_1, x_2, \ldots, x_n\}$ we can calculate the average and SD as follows:

$$AVG(X) = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$SD(X) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

The gyroscope provides measurements of angular velocity (degrees/second) along all three axes, and shows a trend similar to the accelerometer. Figure 6 shows the angular velocity values from the gyroscope when it is rotated about the z-axis.

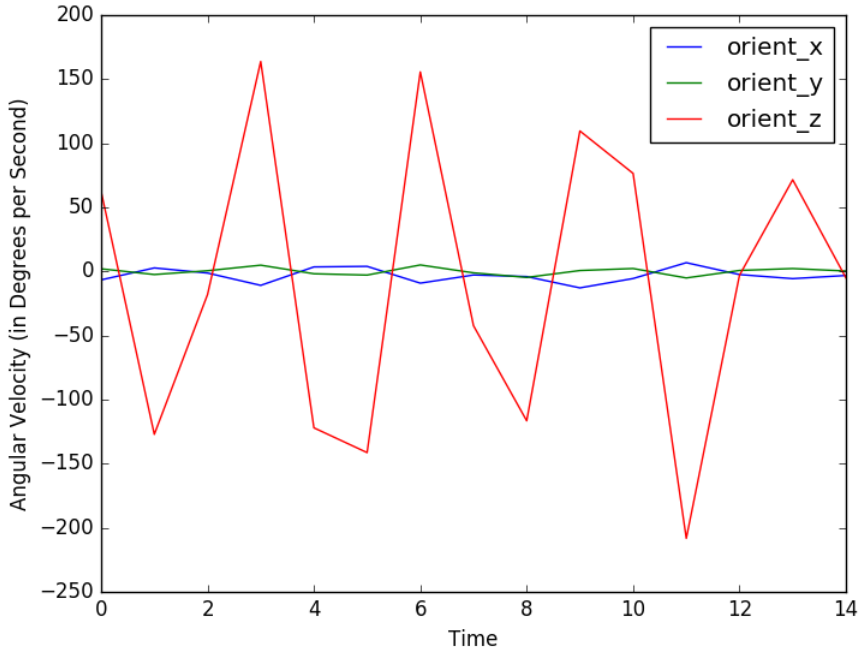

Figure 6: A graph showing angular velocity values over time when rotating a gyroscope along the z-axis.

Again, due to the large range of scalar values, we turn it into features using the standard deviation from zero movement (where the angular velocity is zero) per axis. In addition, we also consider the average value, and the mean absolute deviation (MAD) from zero. SD, MAD, and

the average value have previously been used as accurate feature representations to detect gyroscope swing for human activity recognition [37].

For n sensor values denoted by $X = \{x_1, x_2, \dots, x_n\}$ we can calculate the MAD as follows:

$$MAD(X) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}|(x_i - \bar{x})|}$$

It is possible that the extracted features carry redundant information. In order to ensure that minimum number of features are used in the prediction models, the model uses Correlation Feature Selection (CFS). CFS works under the assumption that features should be highly correlated with the given class (in this case, a given sign) but uncorrelated with each other [19]. When gathering data for human activity recognition using accelerometers and gyroscopes, CFS is commonly used as the method of determining the most important features [37][20].

The merit of a feature subset (a measure of that subset's accuracy) is calculated using the below equation. If the equation yields a higher merit value than the original feature set, the unessential features are dropped. For a subset S, consisting of k features, the merit is calculated as follows:

$$Merit(S_k) = \frac{k\overline{r_{cf}}}{\sqrt{k + (k-1)k\overline{r_{ff}}}}$$

where $\overline{r_{cf}}$ and $\overline{r_{ff}}$ are average values of the classification-feature and feature-feature correlations [19].

3.2.2 Machine Learning Models

After feature extraction from the analog signals are completed, each generated feature vector is fed to various machine learning models along with its corresponding sign. These vectors and their signs are the training data set.

Three supervised machine learning models are trained using the same dataset, and an ensemble of these models is used to predict a sign given a feature vector as an input. Python's scikit-learn, a machine learning library, is used to train the models [21].

3.2.2.1 Naïve Bayes

The first model, Naïve Bayes, is a variant of the Bayesian classifier type. It calculates the conditional probability of that a feature vector belongs to a certain class, given a specific label. This model works under the assumption that most of the features are independent [22]. Even though acceleration values are highly correlated with each other (as there are physical limits to the movements of hands), these values are independent of the gyroscope values and the flex sensor values. These conditions imply Naïve Bayes can be used to predict symbols [23].

Scikit-learn's Multinomial Naïve Bayes model is trained to predict signs given a feature vector. For a given feature vector $X = \{x_1, x_2, \dots, x_n\}$ the probability that the vector corresponds to a given label Y is:

$$P(X|Y) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

where p_i is the probability that an event i occurs.

The classifier will predict a label Y for a given feature vector X as follows:

$$Y = \max_{k \in [1,K]} p(Y) \prod_{i=1}^{n} p(x_i|Y)$$

Naïve Bayes is time efficient in both the training and the recognition phases due to its near-linear time complexity, and detects sign language using haptic sensors with a very high accuracy compared to the other models [25]. It is known to work well when the dataset is incomplete and will have additions over time [36], which is true for this case as this prototype only supports 14 signs to begin with.

3.2.2.2 Decision Trees Learning

The second model, Decision Trees Learning, involves constructing of a decision tree from a set of training feature vectors.

Each branch of the decision tree represents the outcome of a test and each node of the tree is a test with one of the features in the training tuple. Decision trees are helpful for visualizing the data, and work well regardless of the independence of the input features. This helps mitigate the

discrepancies created by the Naïve Bayes model, which assumes all the features are independent of each other.

In order to how to create branches in the tree, the model uses a metric called information gain. Information gain realizes the most relevant features that cause maximum split. It is calculated using the entropy of the content (the unpredictability of the given values). Mathematically, the entropy is defined as:

$$I_E(p) = H(T) = -\sum_{i=1}^{n} f_i \log_2 f_i$$

where f_i are fractions that add up to 1, representing the percentage of each class present in the child node that results from a split in the tree [35].

The expected information gain is defined as the change in the entropy from a parent to its children, and is given by:

$$Info.Gain = Entropy(parent) - Weighted\ Sum\ of\ Entropy\ (children)$$
$$= H(T) - H(T|a)$$

The C4.5 algorithm variant of this model is trained and used to predict the symbols due to its success in multi-label classification [35]. Decision trees have also been used specifically for sign language recognition using haptic, orientation and acceleration sensors with a high accuracy [25][26]. Once the decision tree has been constructed, a given feature vector is evaluated by it to get the corresponding symbol.

3.2.2.3 Support Vector Machines

The third model, Support Vector Machines (SVMs), are a type of supervised machine learning model which involves constructing hyperplanes from input data sets to get a linear separation between the data.

SVMs use the kernel method to find patterns in the input training tuples, and use the kernel trick to calculate the relative inner product of all the pairs of the data, rather than the actual distance between coordinates of the data in the hyperplane [27].

The default radial basis function (RBF) used in the C-Support Vector Classification (SVC) model in scikit-learn's SVM is used as the kernel function for symbol classification. The multi-class classification is handled according to a one-to-one scheme [28], where every class is compared to another until there is only one left. The RBF kernel function for any two inputs x and y represented as vectors in an input space is given by:

$$K(x, y) = \exp\left(-\frac{|x - y|^2}{2\sigma^2}\right)$$

where $|x - y|^2$ is the squared Euclidean distance between the two features and σ is a free parameter defined by the SVM while training the SVC model.

The SVM model has been used in the past to detect human motions with an accuracy of near 97% [17].

3.2.2.4 Ensemble Model

All three supervised classification models used are not perfect in predicting the correct labels for a given feature vector.

Therefore, the final model used involves the results from all three models and chooses the symbol predicted with a majority. As seen in Section 4.3.3, the ensemble model performs better than the individual models when tested on four signs with data collected by two users. Even though training the models separately involves a more computational power and/or time, the prediction itself still completes in under 800 ms once training has completed.

The three models used are fairly different from each other when they approach the given dataset, and therefore provide a variety in symbol distinction enabling a stronger model overall. However, it is possible that all three models predict different symbols and there is no majority. In that case, the decision from the Support Vector Machines will be chosen over the other two models, because it is more commonly used to find trends in Human Activity Recognition [17] [29]. This anomalous feature vector will also be flagged so that it can be used to re-train the models in the future.

3.2.3 Evaluation

Even though a Confusion Matrix is originally defined for binary classifications, it can be extended to the current case of multi-class classification [30]. In the case of symbol prediction, for a given label, true positives (TP) and true negatives (TN) are defined as the number of positive instances of that label classified correctly and incorrectly, respectively. For a given symbol label, false positives (FP) and false negatives (FN) defined as the number of negative instances of the label classified correctly and incorrectly respectively.

For every symbol, the accuracy is calculated as:

$$Accuracy_{symbol} = \frac{TN + TP}{TN + TP + FN + FP}$$

As the training dataset has the same number of instances for every label, the overall accuracy of the model is calculated by taking average of the accuracy for all the symbols predicted using that model. For models with similar accuracy, precision is used to determine the model providing a better result. Precision is the ratio of the positive instances predicted correctly. For every label, precision is defined as:

$$Precision_{symbol} = \frac{TP}{TP + FP}$$

Similar to accuracy, the overall precision of the model is calculated by taking the average of the precision for all the symbols predicted using that model.

3.2.4 Cross Validation

The data for a total of 14 ASL signs were used to train the machine learning model. Four people worked to generate a training data set with 100 iterations for each sign. Cross validation is a statistical approach to measure how the final model performs when it is generalized to a larger testing audience [38]. A 10-fold cross validation model and a leave-one-out cross validation model are used to measure the performance of the ensemble model when used by new users who did not participate in generating in training dataset.

A 10-fold cross validation model is used to evaluate the accuracy of the machine learning models. The data available for training a supervised model is shuffled and divided into ten equal parts. Out of these ten subsamples, nine are used to train the models while one is used to validate

the model. This process is repeated ten times, to use every subsample at least once to evaluate model accuracy. The overall accuracy of the model is obtained by averaging the ten accuracy values obtained by using every subsample as a validation set.

In addition to the 10-fold cross validation model, leave-one-out cross validation is applied while training the input data set. In this case, the features extracted from the data of one of the four subjects selected at random is used to validate the model. This is done for each of the 14 signs and the overall accuracy is obtained by averaging the accuracy values received for each sign. This approach allows the study of the impact specific signs have when used by new users who have not trained the data, and thus helps to evaluate the risks associated with the chosen model [24].

3.3 Mobile Subsystem

3.3.1 Data Pipeline

The Data Pipeline is responsible for piping data received from the microcontroller to the cloud-hosted prediction ensemble through web sockets in real-time.

This prototype uses socket.io, a web sockets framework that enables bi-directional event-based communication [41]. This uses a publisher-subscriber design pattern in which the client (the Android application) can publish events to the server (notably, an event communicating that a set of data is ready to be sent).

The server then responds with the text representation of the predicted symbol based on the received digital data. This text is then piped to the user interface and the text-to-speech module.

3.3.2 User Interface (UI)

The user interface for this system is an Android application. Figure 7 below shows the wireframe design of the application. The user has the ability to turn off the active interpretation at any given point in time. It also allows the user to download a .txt file of a transcript of the signs interpreted. This transcript is displayed in a text box and is updated in real-time as the user performs signs with the gloves. The user can also reset the transcript at any point after being prompted with a confirmation dialog.

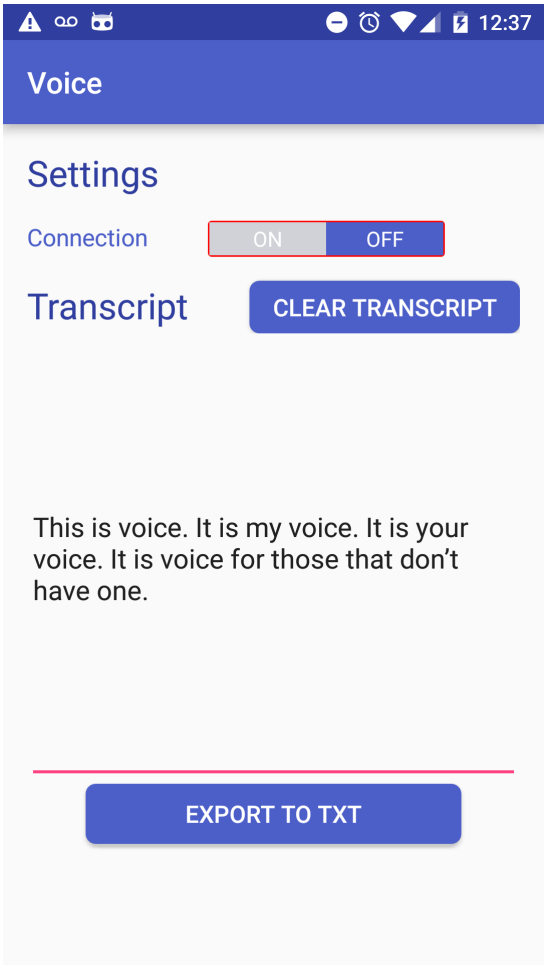

Figure 7: UI mock-up of the android application for Voice.

3.3.3 Text-to-Speech

Android has a built-in Text-To-Speech (TTS) module which is used to convert text to speech during application development. Figure 8 below shows a snippet of code for initializing the instance with required parameters such as language choice and an example of the code necessary to read a particular string aloud. Android comes with a default TTS engine (Google's TTS system) but the user has the power to modify their own default engine.

```
TextToSpeech tts = new TextToSpeech(this, this);
tts.setLanguage(Locale.US);
tts.speak("Text to say aloud", TextToSpeech.QUEUE_ADD, null);
```

Figure 8: Code snippet for carrying out the TextToSpeech translation in Android.

4 Prototype

4.1 Prototype Design

The design of the hardware connected to the glove consists of a pair of Bluetooth (HC-06 and HC-05) modules, five flex sensors, a Flora Accelerometer/Gyroscope/Magnetometer 9-DOF chip (LSM9DS0) and a microcontroller. The overall schematic of the design is shown in Figure 9.

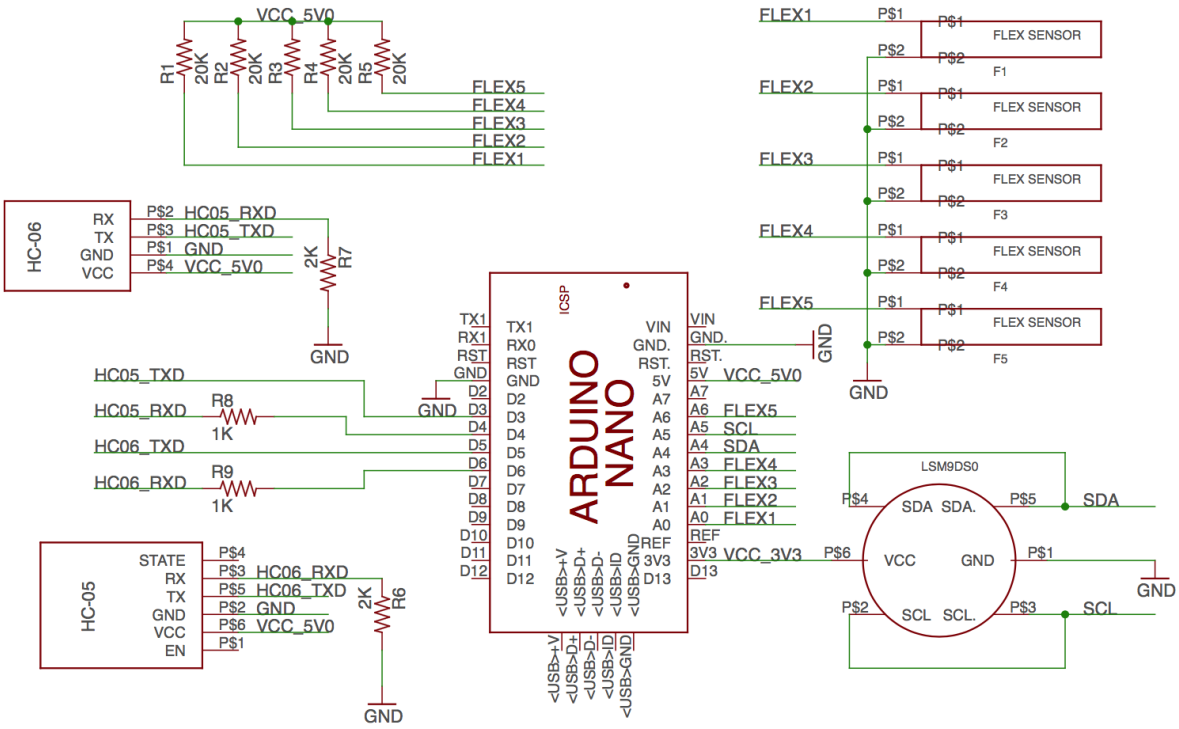

Figure 9: Schematic of External Subsystem components.

All the non-sensor components (i.e. microcontroller, Bluetooth module and other passive circuitry) are attached to the main PCB (layout and schematic are shown in Figure 10 and Figure 11, respectively). The main goal of the PCB is to serve as a shield for the microcontroller and provide a simple interface for the sensors that are attached to the glove.

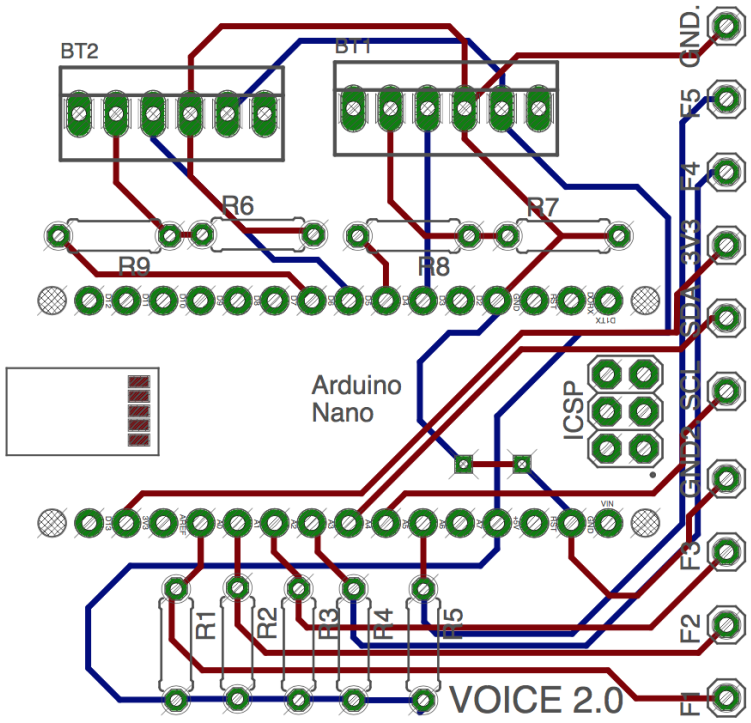

Figure 10: The layout of the main PCB board on the glove.

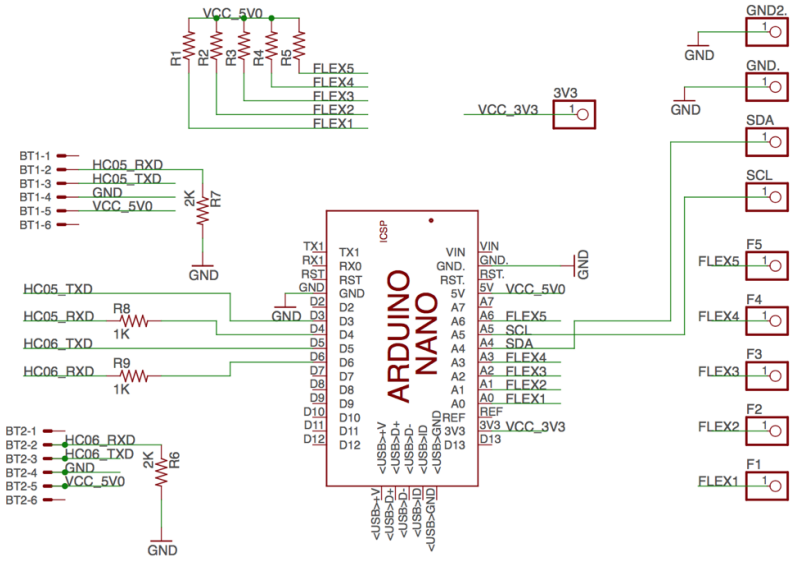

Figure 11: Schematic of the main PCB board on the glove.

The dimensions for the enclosure in Figure 12, below, is 58 mm x 58 mm with a height of 35.7 mm. The opening at the top is designed to allow a square piece of transparent acrylic glass to slide in. This is to allow the user to view the LEDs of the microcontroller, which indicate when a user can perform the sign. The plexiglass is 57.5 mm x 57 mm with a height of 1.7 mm.

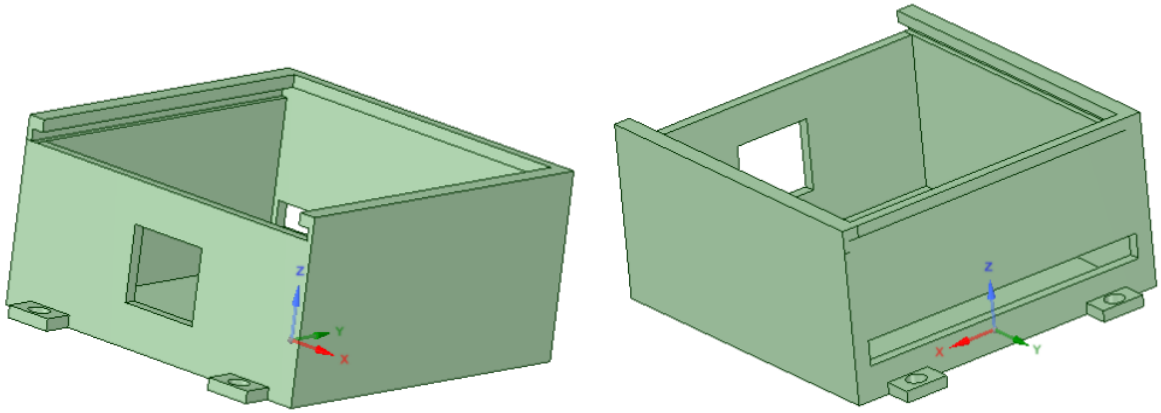

Figure 12: Design of the enclosure mounted on the glove.

4.2 Structure and Placement of Components

Figure 13 and Figure 14 below show the component placement of the External Subsystem.

The flex sensors are inserted into pockets sewn onto the tops of each of the fingers. The pockets prevent the flex sensors from shifting positions. The bottoms of the sensors are sewed in place so that the flex sensors replicate the same movement as the fingers when the user bends their fingers. As such, the sensor will have the greatest resistance in its bent state and least resistance when fingers are extended; the difference between the two is maximized.

The accelerometer/gyroscope is sewed in place using conductive thread near the middle of the back of the hand. This position was chosen to minimize the effects of finger and wrist movement on the sensor, and retain normal hand usage with the gloves on.

Figure 13: Component placement for External Subsystem using conductive threads.

The PCB has exposed circuitry and hence, it must be placed into the enclosure discussed in the previous subsection. The Arduino Nano and its respective HC-06 Bluetooth module are contained within the enclosure. Wires are used to connect the PCB and its components to the sensors (Figure 14) through the small rectangular opening on the side of the enclosure as seen in Figure 12. The opening on the other end of the enclosure is for the connection to the battery pack.

Figure 14: Component placement for External Subsystem using enclosure and wires.

4.2 Safety and Consideration

Several iterations of the prototype were made to address safety concerns. The accelerometers are weather-proofed using a conformal coating to prevent corrosion. Flex sensors were covered with heat-shrink tubing to provide abrasion resistance and environmental protection.

An enclosure was designed, with dimensions as discussed in Section 4.1, to enclose the PCB and its connected components as shown in Figure 14. This ensures that the PCB survives any external damage due to physical movements or environmental changes, and prevents direct contact between the circuitry and the glove itself, which may be a fire hazard.

The initial prototype used uninsulated conductive thread for flexibility and a more professional look. During safety tests, however, it was evident that 1.5 A of current through the conductive thread would cause the wire to burn. Fabric glue, which is commonly used in e-textile projects

with conductive threads to prevent cases of short circuits, was considered, but wires were a better guarantee. The cost of this, however, is that wires are less flexible than conductive thread, which make the usage of the gloves more difficult.

4.3 Prototype Data

At the time of writing, two members of the group had collected data for the first four signs in the specifications. Every sign, thereby, had 200 data points, 100 per user. The transition of the data as a user makes the sign, hello, to it been translated into speech is discussed below.

4.3.1 External Subsystem and Mobile Data

A user has two seconds to make a sign. During this time, the data from the seven sensors is polled approximately every 16.6 milliseconds. The end of that iteration is marked with the "END" keyword, making it easier to process the data on the back-end.

```
-8.09, 1.83, 5.24, -0.01, 0.03, -0.05, 334, 357, 347, 328, 349|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-8.47, 2.46, 5.08, -0.01, 0.05, -0.00, 334, 357, 345, 328, 349|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-7.57, 2.18, 4.07, 0.68, -0.07, -0.85, 335, 356, 345, 328, 348|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-9.38, -2.98, 4.62, 0.31, -0.27, -0.54, 335, 359, 345, 328, 347|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-9.08, -2.32, 5.23, -0.10, 0.03, 0.06, 334, 358, 346, 327, 347|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-9.42, -1.46, 4.15, -0.42, 0.31, 0.43, 334, 357, 345, 328, 347|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-9.06, 2.33, 2.23, -0.57, 0.26, 0.49, 334, 358, 345, 328, 347|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-8.22, 7.14, 2.40, -0.11, -0.04, 0.08, 333, 360, 346, 327, 346|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-8.74, 3.54, 3.27, -0.03, -0.06, -0.18, 335, 359, 346, 327, 346|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-9.25, 2.40, 3.97, -0.17, 0.01, -0.04, 334, 360, 346, 328, 348|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
-9.02, 2.37, 3.61, -0.05, 0.00, -0.06, 335, 358, 346, 328, 346|0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
END
```

Figure 15: Arduino Serial Output Data.

Figure 15 shows an iteration of the Arduino serial output, when Biraj Kapadia (a team member), signed "hello". As it is a one-handed sign, the data from the left-hand glove is switched off – setting all 11 values to zero for every left-handed data point. In this scenario, this is done by manual configuration in the back-end server. This is makes pre-processing faster and more efficient than if the server processed the empty data from the left hand. When both one-handed and two-handed signs are expected, the intermediary Android application will determine if the sensor values have a large enough deviation such that it needs to forward data from both gloves or just one.

These data points are then relayed in batches by the Android application to the back-end for pre-processing, for either training or prediction.

4.3.2 Machine Learning Subsystem Data

After receiving data from the Android application, the back-end performs the pre-processing discussed in Section 3.2.1. Each two second iteration generates 20 processed features per glove. If the data is used during the training stage, a label is associated with every set of features, otherwise a label will be predicted using the existing ML models. The name of the user is also stored for purposes of validation of the accuracies over different users.

```
{'features': [-16.493, 4.545, 8.832, 0.0, -0.0, 0.0, -0.0, 0.0, -0.0, 1.0,
              332.0, 1.0, 359.0, 1.0, 349.0, 1.0, 327.0, 1.0, 342.0],
'user': 'biraj',
'label': 'hello'}
```

Figure 16: Machine Learning processed data.

Figure 16 shows an example snippet of a dictionary used to store the processed features of the "hello" sign. The feature array has done by Biraj in Figure 15. Again, as hello is a one-handed sign, only 20 features, and has 40 if the sign involves both the hands. As this data was used to train the ML models, a label associated with it is also included in the dictionary.

200 such dictionaries per sign (800 in total at the time of writing) are shuffled and used to train the three models - Naïve Bayes, Decision Trees Learning and Support Vector Machines. The data is split 80-20 between training and validation at this stage to ensure these models produce accuracies close to our specifications. If this validation check has a very low accuracy, other machine learning models will be considered for the purpose of sign prediction.

4.3.3 Accuracy

640 data points are used to train the three models and the remaining 160 used to verify the results by comparing the associated and the predicted labels. These accuracies are tabulated in Table 6 shown below.

Table 6: Accuracy of ML models for four signs made by two users.

Model	**Accuracy**
Naïve Bayes	77%
Decision Trees Learning	78%
Support Vector Machines	78.6%
Ensemble	81.2%

Table 6 shows the accuracy of the models for four signs with data generated by two users. It is clear that SVM has a slight edge over other models the individual case. As expected, the ensemble model has a higher accuracy when compared with the individual performances of each of the three models used. This accuracy also approaches the 85% specification. With two more users collecting data for these four signs, and all the users collecting data for the remaining 10 signs, there should be enough data to fine-tune the model to reach 85% by the conclusion of the project.

4.3.4 Latency

The latency of Voice was measured by randomly choosing one of the four signs with data available to train the ML models and timing the sign translation using Java's System.nanoTime() function call, from when it receives all the necessary data from the gloves to when the server responds with a prediction. Figure 17 plots the latencies for 10 such trials.

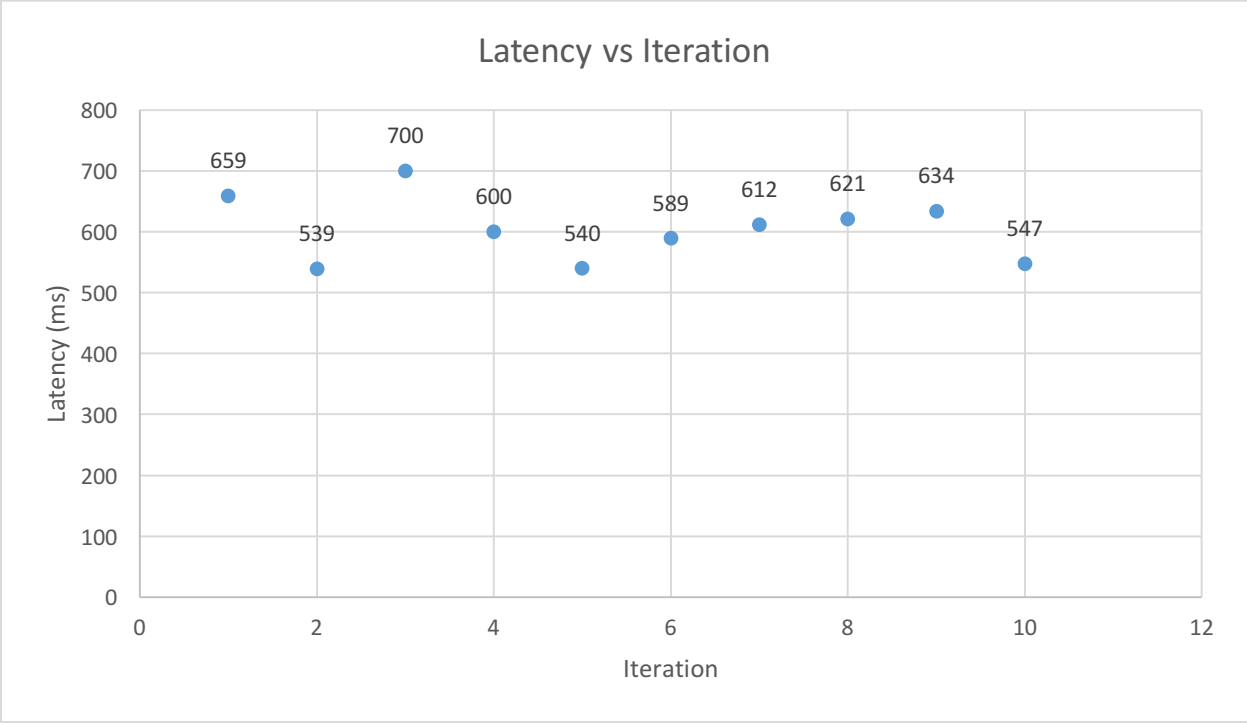

Figure 17: Plot of the latency for 10 iterations of signs.

It is evident that most of the sign translations happen within 700 ms, with the average time being around 604 ms, which meets our specification of 800 ms.

5 Discussion and Conclusions

5.1 Evaluation of Final Design

The Android application satisfies the requirement for having a mobile application, and enables the user to turn on/off the translation itself. It can also display and export the transcript. The trained models that perform the translation satisfy the language support specification.

The HC-06 Bluetooth modules satisfy the requirement for wireless support, and ensuring the external battery pack's wires are longer than the total finger-to-finger length of a person's arms means the full range of motion for a user's hands and arms is possible. The battery pack itself satisfies the operating time specification.

As seen in Figure 14, all of the components mounted on the glove are contained to the surface area of the glove itself. The total mass of the external subsystem is 53 g, excluding connecting

wires and battery pack, which is not mounted on the glove. Therefore, this is within the weight specification of under 100 g for a single modified glove.

With this design, the current cost of the entire system (excluding an Android phone) is $309.60 (Table 7). This is well within the cost specification, which states the entire system should be under $500.

Table 7: Total cost to build a pair of gloves for the prototype.

Materials	**Amount ($)**
Gloves	22
Flex Sensor x 10	150
Arduino Nano x 2	8.8
Bluetooth Chips (HC-05/HC-06) x 3	30
PCB x 2	8.8
Flora Accel/Gyro/Magn 9-DOF x 2	56
3D-Printed Enclosure x 2	28
Other	6
Total Cost	**309.6**

The accuracy of the ensemble model is approaching our required specification at 81%, and with further iteration it can be improved to the required 85%. The latency averages 604 ms, which is under the required latency.

5.2 Use of Advanced Knowledge

The software portion of the system incorporates software architecture principles taught in ECE 452 - Software Architecture and Design Patterns. In particular, the architecture pattern used in the system is the layered architecture style where the mobile application makes a request to its machine learning model subsystem, and receives a response based on the input parameters

provided. These responses are further propagated upwards to the user interface and to the TTS engine.

ECE 316 – Probability is also required as a basis to understand many machine learning concepts. Since machine learning predictions are a result of probabilistic calculations, and as such, in order to understand the inner workings of a model and its behaviour, it is pertinent that the concepts such as sets, expectation and variance are well understood, despite the lack of direct application.

The entire machine learning subsystem of the project is built on the concepts learned in CZ 4041 – Machine Learning (taken on exchange at Nanyang Technological University). Specifically, principles like feature extraction, feature selection, specific knowledge about different machine learning models, model evaluation and cross validation are all used in the implementation of the model trained to detect signs from the sensor data.

5.3 Creativity, Novelty, and Elegance

One of the biggest draws of Voice is the elegance of its implementation in comparison to other existing products whose main objective is to bridge gaps between native sign language speakers and non-native or non-speakers. Existing products are either glorified text terminals, require the usage of a connected personal computer, or require additional hardware such as cameras or other clunky peripherals. With Bluetooth-enabled gloves designed to only have the minimum amount of electronics and a companion mobile application, the system can be easily integrated into a user's life without drastically changing their daily routine.

In addition, no existing sign language recognition system performs the translation in a mobile environment, with an extremely small amount of prior setup. This feature is completely unique to Voice. This allows the user the freedom to speak when they want to – not only when they have the ability to setup the necessary environment for another system.

5.4 Quality of Risk Assessment

The group met the timeline estimated in 498A through the co-op term. However, at the start of the 4B term, the group ran into safety issues with the prototype as discussed in Section 4.2. This affected the remaining tasks - namely, collecting data for training the machine learning models and evaluating the models. After a series of safety inspections determining many of the

conductive thread stitches were unsafe, the group had to remove these stitches from the glove. This set back much of the progress made during the co-op term.

As the machine learning portion of the project was bottlenecked by the hardware's completion, the group had to reduce the number of signs in the specifications to make progress in the remaining time. It was a mistake on the group's part to not carefully evaluate safety considerations before attempting to create the supposed final hardware iteration. The group should have consulted the advisor or a safety inspector before going from insulated wires in our prototype demonstration in 4A to conductive thread through the co-op term.

5.5 Student Workload

Table 8 below shows break down of the work distribution between the team members of the group. There are no significant discrepancies of workload between the four members.

Table 8: Percentage of workload performed by each team member.

Student Name	**Percentage of Workload**
Akshay Budhkar	25%
Eliot Chan	23%
Biraj Kapadia	26%
Amish Patel	26%

References

[1] "Statistics on deaf Canadians," Canadian Association of the Deaf, 2015. [Online]. Available: http://cad.ca/issues-positions/statistics-on-deaf-canadians/. Accessed: May 30, 2016.

[2] T. Harrington, "Deaf population of the U.S. - Local and regional deaf populations," Gallaudet University Library Library, 2010. [Online]. Available: http://libguides.gallaudet.edu/content.php?pid=119476&sid=1029190. Accessed: May 30, 2016.

[3] sComm, "UbiDuo 2 wired," 2015. [Online]. Available: https://www.scomm.com/product/ubiduo-wired/. Accessed: May 30, 2016.

[4] "Arduino - ArduinoBoardNano", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardNano. [Accessed: 26- Jun- 2016].

[5] "XCSOURCE 5pcs Mini USB Nano V3.0 ATmega328P 5V 16M Micro Controller Board F Arduino TE359: Amazon.ca: Electronics", *Amazon.ca*, 2016. [Online]. Available: https://www.amazon.ca/XCSOURCE-ATmega328P-Controller-Arduino-TE359/dp/B015MGHH6Q/ref=sr_1_1?ie=UTF8&qid=1466886073&sr=8-1&keywords=arduino+nano. [Accessed: 26- Jun- 2016].

[6] A. Industries, "Adafruit Pro Trinket - 5V 16MHz ID: 2000 - $9.95 : Adafruit Industries, Unique & fun DIY electronics and kits", *Adafruit.com*, 2016. [Online]. Available: https://www.adafruit.com/products/2000. [Accessed: 26- Jun- 2016].

[7]"Arduino - ArduinoBoardPro", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardPro. [Accessed: 26- Jun- 2016].

[8]"XCSOURCE 5pc Pro Mini Enhancement ATMEGA328P 16MHz 5V Compatible Arduino PRO Module TE362: Amazon.ca: Electronics", *Amazon.ca*, 2016. [Online]. Available: https://www.amazon.ca/XCSOURCE-Enhancement-ATMEGA328P-Compatible-TE362/dp/B015MGHLNA/ref=sr_1_2?ie=UTF8&qid=1466886135&sr=8-2&keywords=arduino+pro. [Accessed: 26- Jun- 2016].

[9] J. Lapiak, "Most used ASL vocabulary for know", Handspeak.com, 2016. [Online]. Available: http://www.handspeak.com/word/most-used/index.php?dict=100&signID=1207. [Accessed: 29- Jun- 2016].

[10] "LSM9DS0 Datasheet", 2016. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/LSM9DS0.pdf. [Accessed: 29- Jun- 2016].

[11] "kriswiner/MPU-6050", *GitHub*, 2016. [Online]. Available: https://github.com/kriswiner/MPU-6050/wiki/Affordable-9-DoF-Sensor-Fusion. [Accessed: 29- Jun- 2016].

[12] S. Madgwick, "An efficient orientation filer for inertial and inertial/magnetic sensor arrays", 2010.

[13] K. Winer, *LSM9DS0 9DOF sensor AHRS sketch*. https://github.com/kriswiner/LSM9DS0, 2014.

[14] "Analog Trimming, Calibration: Automotive, Medical Industry: sensors, implanted devices, LCD display and touch panel controllers, PMICs - Sidense Corp. OTP NVM", *Sidense.com*, 2016. [Online]. Available: https://www.sidense.com/applications/analog-trimming-and-calibration. [Accessed: 29- Jun- 2016].

[15] Texas Instruments, "μA78Mxx Positive-Voltage Regulators". [Online]. Available: http://www.ti.com/lit/ds/symlink/ua78m05.pdf. Accessed: Jun. 27, 2016.

[16] [2]"Amazon.com: Aukey 12000mAh Portable Charger Power Bank External Battery Pack - Retail Packaging - Black: Cell Phones & Accessories", *Amazon.com*, 2017. [Online]. Available: https://www.amazon.com/Aukey-12000mAh-Portable-Charger-External/dp/B00M835WGE?tag=trust925-20. [Accessed: 09- Feb- 2017].

[17] Z. He and L. Jin, “Activity recognition from acceleration data based on discrete consine transform and svm,” in IEEE International Conference on Systems, Man and Cybernetics, pp. 5041–5044, 2009.

[18] I. King, *Neural information processing*, 1st ed. Berlin: Springer, 2006.

[19] M. A. Hall, "Correlation-based Feature Selection for Machine Learning". 1999. Available: http://www.cs.waikato.ac.nz/~mhall/thesis.pdf. Accessed: June 24, 2016.

[20] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, “Activity recognition and monitoring using multiple sensors on different body positions,” in Proc. International Workshop on Wearable and Implantable Body Sensor Networks, pp. 113–116, 2006.

[21] "scikit-learn: machine learning in Python", *Scikit-learn.org*, 2016. [Online]. Available: http://scikit-learn.org/stable. [Accessed: 26- Jun- 2016].

[22]"1.9. Naive Bayes", *Scikit-learn.org*, 2016. [Online]. Available: http://scikit-learn.org/stable/modules/naive_bayes.html. [Accessed: 26- Jun- 2016].

[23] M. Kurosu, *Human-computer interaction*. Cham [u.a.]: Springer, 2014.

[24] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection", *Statistics Surveys*, vol. 4, no. 0, pp. 40-79, 2010.

[25] C. Shahabi, A. Ghoting, L. Kaghazian, M. McLaughlin, G. Shanbhag, "Analysis of haptic data for sign language recognition", *Proc. First International Conference on Universal Access in Human-Computer Interaction (UAHCI)*, pp. 5-10, August 2001.

[26] G. Fang, W. Gao and D. Zhao, "Large Vocabulary Sign Language Recognition Based on Fuzzy Decision Trees", *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 34, no. 3, pp. 305-314, 2004.

[27] J. Breneman, "Kernel Methods for Pattern Analysis", *Technometrics*, vol. 47, no. 2, pp. 237-237, 2005.

[28] "sklearn.svm.SVC", *Scikit-learn.org*, 2016. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed: 27- Jun- 2016].

[29] H. Banaee, M. Ahmed and A. Loutfi, "Data Mining for Wearable Sensors in Health Monitoring Systems: A Review of Recent Trends and Challenges", *Sensors*, vol. 13, no. 12, pp. 17472-17500, 2013.

[30] Mohan Patro and M. Ranjan Patra, "A Novel Approach to Compute Confusion Matrix for Classification of n-Class Attributes with Feature Selection", *TMLAI*, 2015.

[31] "NumPy — Numpy", *Numpy.org*, 2016. [Online]. Available: http://www.numpy.org/. [Accessed: 26- Jun- 2016].

[32] "Python Data Analysis Library", *Pandas.pydata.org*, 2016. [Online]. Available: http://pandas.pydata.org/. [Accessed: 26- Jun- 2016].

[33] "Average Hand Size - Your resource for Hand Size, Palm Size and Finger Length Averages", *Theaveragebody.com*, 2017. [Online]. Available: http://www.theaveragebody.com/average_hand_size.php. [Accessed: 09- Feb- 2017].

[34] "S13TAPUCF0 TenActivTM ESD Carbon Glove with PU Palms, Size 0, Safety Work Gloves - Amazon Canada", *Amazon.ca*, 2016. [Online]. Available: https://www.amazon.ca/S13TAPUCF0-TenActivTM-Carbon-Glove-Palms/dp/B01B8J8U2Q. [Accessed: 27- Jun- 2016].

[35] S. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993", *Mach Learn*, vol. 16, no. 3, pp. 235-240, 1994.

[36] C. Elkan, "Boosting And Naive Bayesian Learning", University of California, San Diego, 1997.

[37] N. Capela, E. Lemaire and N. Baddour, "Feature Selection for Wearable Smartphone-Based Human Activity Recognition with Able bodied, Elderly, and Stroke Patients", *PLOS ONE*, vol. 10, no. 4, p. e0124414, 2015.

[38] P. Burman, "A Comparative Study of Ordinary Cross-Validation, v-Fold Cross-Validation and the Repeated Learning-Testing Methods", *Biometrika*, vol. 76, no. 3, p. 503, 1989.

[39] S. Lee, "Electronic Brick of HC-06 Serial Port Bluetooth", 2013. [Online]. Available: http://ftp://imall.iteadstudio.com/Electronic_Brick/IM120710006/DS_IM120710006.pdf. [Accessed: 09- Feb- 2017].

[40] "HC-05 -Bluetooth to Serial Port Module", 2010. [Online]. Available: http://www.electronicaestudio.com/docs/istd016A.pdf. [Accessed: 09- Feb- 2017].

[41] "FEATURING THE FASTEST AND MOST RELIABLE REAL-TIME ENGINE," *socket.io*. [Online]. Available: http://socket.io/. [Accessed: Jan. 9, 2017].

Appendix

Appendix A: Code to Read Data from one Flex Sensor

```
int sensorPin = A3;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);

  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
  Serial.println(sensorValue);
  delay(5);
}
```

Appendix B: Code to read data from the LSM9DS0 using the Adafruit Library

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_LSM9DS0.h>
#include <Adafruit_Sensor.h>

Adafruit_LSM9DS0 lsm = Adafruit_LSM9DS0();
void setupSensor()
{
  // 1.) Set the accelerometer range
  lsm.setupAccel(lsm.LSM9DS0_ACCELRANGE_16G);

  // 2.) Set the magnetometer sensitivity
  lsm.setupMag(lsm.LSM9DS0_MAGGAIN_2GAUSS);

  // 3.) Setup the gyroscope
  lsm.setupGyro(lsm.LSM9DS0_GYROSCALE_245DPS);
}

void setup()
{
  Serial.begin(9600);
  delay(1000);
  if (!lsm.begin())
  {
    Serial.println("LSM9DS0 not found!");
    while (1);
  }
  Serial.println("Found LSM9DS0 9DOF");
  Serial.println("");
  Serial.println("");
}

void loop()
{
  lsm.read();
  Serial.print("Accel X: "); Serial.print((int)lsm.accelData.x); Serial.print(" ");
  Serial.print("Y: "); Serial.print((int)lsm.accelData.y);       Serial.print(" ");
  Serial.print("Z: "); Serial.println((int)lsm.accelData.z);     Serial.print(" ");
  Serial.print("Gyro X: "); Serial.print((int)lsm.gyroData.x * 0.00875);   Serial.print(" ");
  Serial.print("Y: "); Serial.print((int)lsm.gyroData.y * 0.00875);        Serial.print(" ");
  Serial.print("Z: "); Serial.println((int)lsm.gyroData.z * 0.00875);      Serial.println(" ");
  Serial.print("Temp: "); Serial.print((int)lsm.temperature);    Serial.println(" ");
  delay(200);
}
```