# University of Waterloo

Faculty of Engineering

Department of Computer and Electrical Engineering

# Detailed Design and Project Timeline

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

June 30, 2016

# Table of Contents

# 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 populous 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. In-person communications between a native sign language speaker and person who does not know sign language still requires a translator to be present, 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 populous 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 an audio output to mimic speech.

The system should be composed of the fewest hardware components possible, ideally utilizing hardware that a person would normally have with them (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 two sign language dialects, namely American Sign Language and Japanese Sign Language, a novel feature that has not been adopted by any other publicly-revealed prototype. The language selection (and overall translation functionality) should be moderated by an accompanying smartphone application, allowing the user to control when and what language they want to speak in.

## 1.3 Block Diagram

The proposed solution involves building custom-made gloves fitted with flex sensors, an accelerometer, a gyroscope to gather data during a user's signing motions. A microcontroller converts the analog signals from the flex sensors to digital, and aggregates that data with the digital data from the accelerometer and gyroscope. The digital data is then fed into a processing 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 comprises of two main subsystems: the External Subsystem, and the Processing Subsystem.

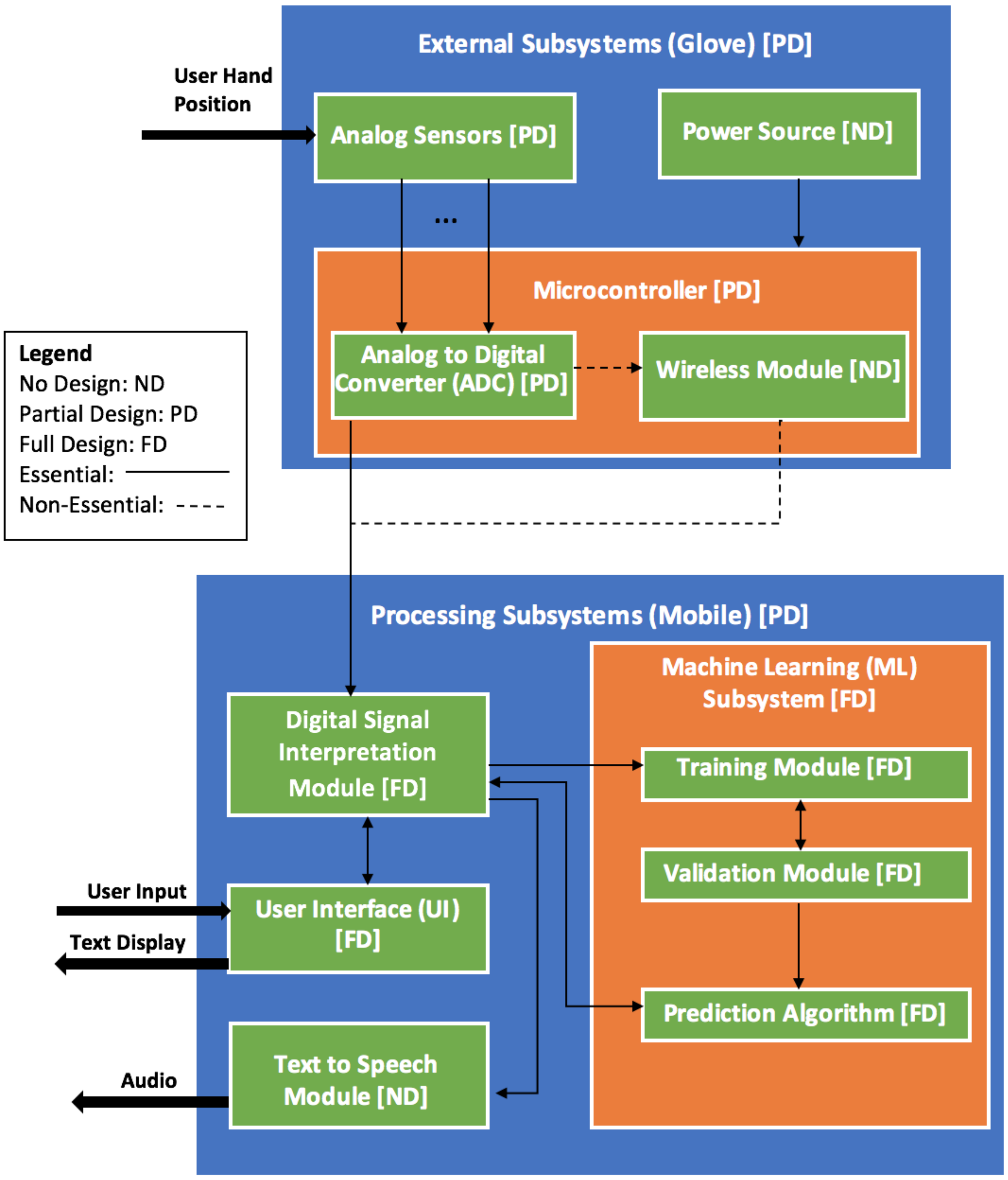


*Figure 1: The block diagram of the system and the connections between each elements.*

### *1.3.1 The External Subsystem*

Flex sensors, accelerometers, and gyroscopes are used in the analog sensor array. Flex sensors would be fitted on the fingers of the glove to determine the curvature and movement of any given finger - a total of ten would be required, one for each finger. An accelerometer on each glove would measure the 3-dimensional motion of the hand, an integral part of "speaking" many signs. A gyroscope on each hand would also allow for the tracking of the orientation of the hands. 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, from either a battery or USB connection to the user's smartphone.

### *1.3.2 The Processing Subsystem (excluding the Machine Learning Subsystem)*

The Digital Signal Interpretation 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 to the Digital Signal Interpretation Module, which passes this information to the User Interface, and Text to Speech Module. The Digital Signal Interpretation Module also receives input from the User Interface to determine which variant of sign language to interpret the signals with, and whether 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 Model simply reads out the given text using the smartphone's speaker system to mimic speech.

### *1.3.3 The Processing Subsystem (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 Digital Signal Interpretation 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 Digital Signal Interpretation Module, and generating text outputs as a response.

## 2 Project Specifications

### 2.1 Functional Specifications

*Table 1: Functional Specifications.*

| Specification | Type | Location | Description |
|---|---|---|---|
| Multi-Language Support | Essential | Processing Subsystem | Able to interpret 100 symbols in American Sign Language and all 49 symbols in Japanese Sign Language syllabary. |
| 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 20 signs/symbols that require rotation of the thumb with respect to the palm. |

## *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.<br><br>A singular modified glove should weigh less than 100 grams. |
| External Subsystem Enclosed in Case | Non-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 six key components for each glove. The microcontroller, flex sensors, a combined gyroscope/accelerometer, a Bluetooth module, the glove itself, and an external power supply.

#### *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 was smaller than 40 cm$^2$, an approximate surface area of a palm.

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*

It seems that 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. 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. 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 using the Madgwick filter compared to the MPU-6050, the GY-80, the MPU-9150, and the MPU-9250, all of which are 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.

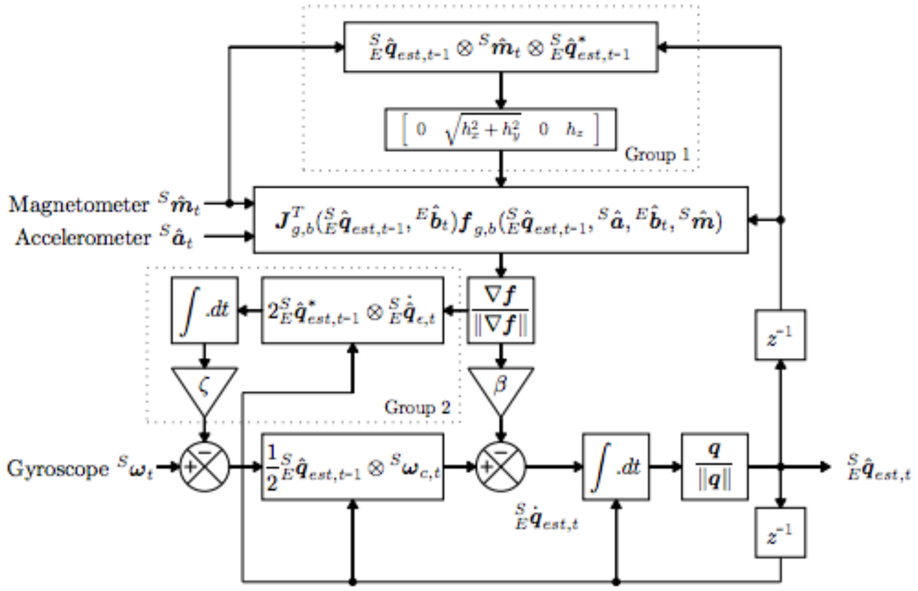


*Figure 2. Block diagram representation for the Madgwick sensor fusion filter [12].*

Figure 2 shows the inner workings of the Madgwick filter, which allows it to construct a quaternion 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].

### *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. As such, the HC-06's command set can be seen as a subset of that of the HC-05's.

Voice uses the HC-06 module partially for simplicity's sake, partially to assist in the fulfillment of the non-essential cost specification (as the HC-06s are generally cheaper than HC-05s), and to fulfill the non-essential wireless support specification. The user's phone acts as the master, and the two gloves the slaves.

The Arduino sends data received from its sensor array using the Bluetooth module 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 InputStream.

#### *3.1.1.5 Glove*

In order to limit the noise added to the sensor data due to static electricity between the hands and the gloves, Electro-Static Discharge (ESD) gloves are used in the prototype. There are many forms of ESD gloves such as nitrile, vinyl, latex and fabric (carbon and nylon) but the only glove that offered reusability was a fabric variant. Other types would be permanently damaged from sewing or other modifications, and as such, the TenActivTM ESD Carbon Glove [34] was chosen to be used in the prototype.

To ensure that the flex sensors remain in place while it is being used, each flex sensor was individually stitched onto the fingers of the glove to ensure that they track the movement of the fingers as closely as possible. Also the flex sensors were enclosed in a layer of ESD material to minimize exposure to the external environment. The glove also has the microcontroller and the gyroscope/accelerometer mounted on it.

#### *3.1.1.6 Battery*

In order to power the glove without connecting it to the phone 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 when using its internal voltage regulator [15]. 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.

The Nano itself has a recommended input voltage of 7V to 12V [4]. 9V batteries were considered as an external power source, however these batteries usually have capacities of around 500 mAh, meaning that it may not be able to power the external subsystem for the entire hour due to variations between individual battery capacities. In addition, 9V batteries are generally pricey and not rechargeable, making them cost-inefficient in addition to perhaps not fulfilling the capacity requirements.

As such, the power supply consists instead of six rechargeable AA batteries in series (producing 9V in total) with capacities of 2400 mAh. This pack is able to power one glove for a minimum of 4 hours, and both gloves for a minimum of 2 hours, and is not mounted on the glove itself for convenience's sake.

To ensure that the weight of the six AA batteries (approximately 31 g each and 186 g in total) doesn’t hinder our specifications for the maximum weight on the glove, a 6 x AA battery holder will act as the "Battery Pack" for the glove. The battery holders come in many shapes and sizes but a double sided 6 x AA battery holder with the dimensions of 5.6 x 4.5 x 3cm [16] is small enough to fit in any average sized pocket, or attached on the user's belt line using a belt clip.

### *3.1.2 Structure and Placement of Components*

Figures 3 and 4 below show the component placement and schematic of the External Subsystem.

Flex sensors are sewed in place near the tips of the fingers, and additional ESD material are sewed overtop to prevent the flex sensors from shifting position. The bottoms of the sensors are not mounted in place to allow for movement forward and backward as the user bends his/her fingers.

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 in order to accurately determine the current orientation of the user's hand, and any changes in orientation.

The Arduino Nano and its respective HC-06 Bluetooth module are placed near the back of the hand to allow the hand the greatest amount of freedom and minimize obstruction of regular activities.

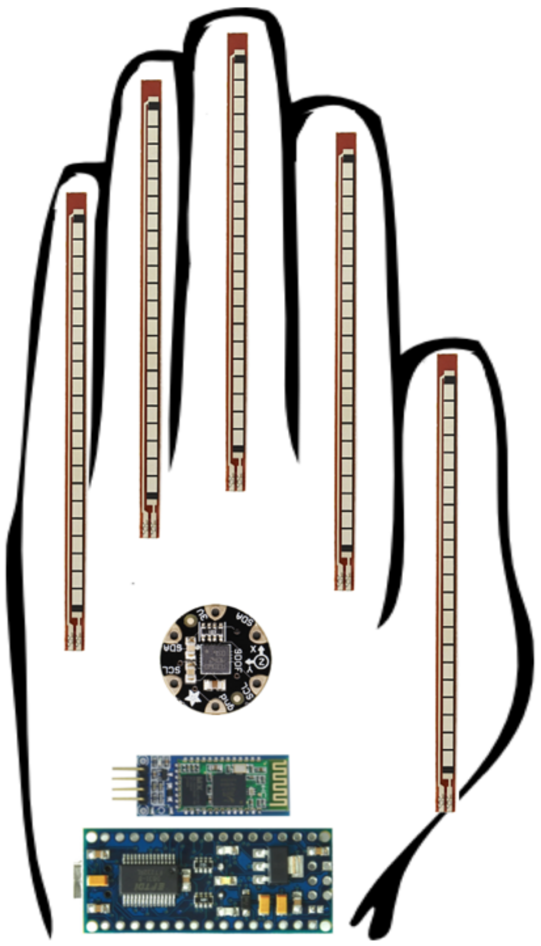


*Figure 3: Diagram showing component placement on External Subsystem.*

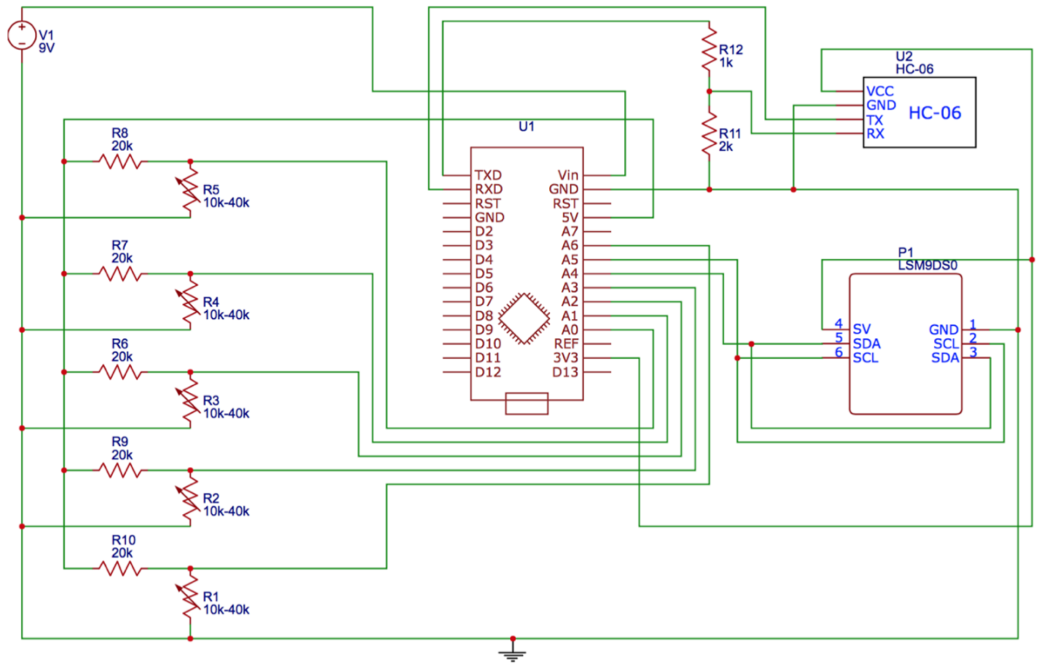


*Figure 4: Schematic of External Subsystem components.*

## 3.2 Processing Subsystem

### *3.2.1 Machine Learning Subsystem*

#### *3.2.1.1 Feature Extraction and Selection*

Every sign generated by a person takes certain time and is not instant. 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. Therefore, 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 separates two actions from each other and these pauses are used to separate signs from one another. Feature extraction is a crucial step while pre-processing the data in order to filter out the relevant information that is used to classify these values into symbols.

Acceleration values tend to fluctuate significantly when there is some movement. This is evident in Figure 5 and Figure 6 which plot the acceleration values for slight up-down and left-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. 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:

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

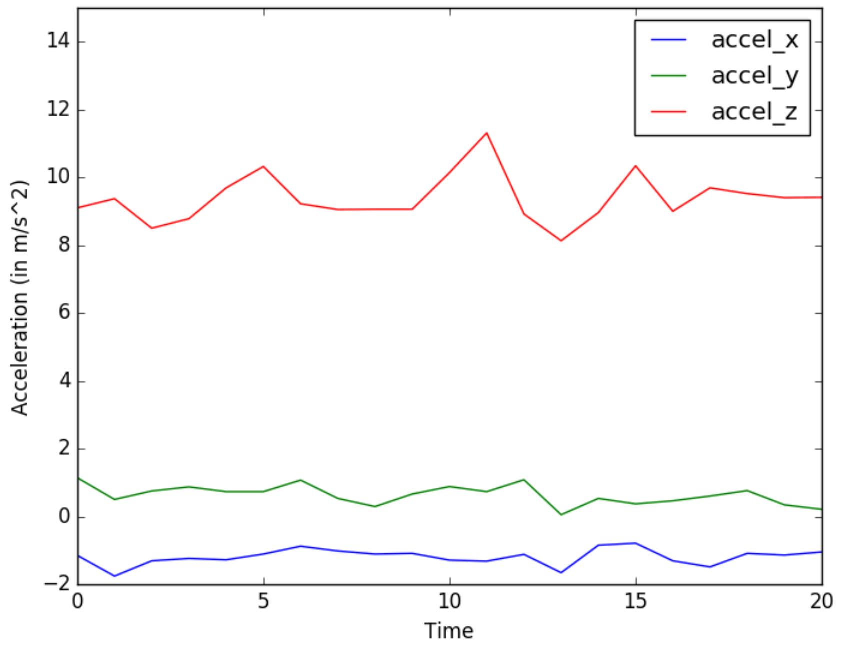


*Figure 5: A graph showing acceleration values over time when moving the accelerometer up and down along the z-axis.*

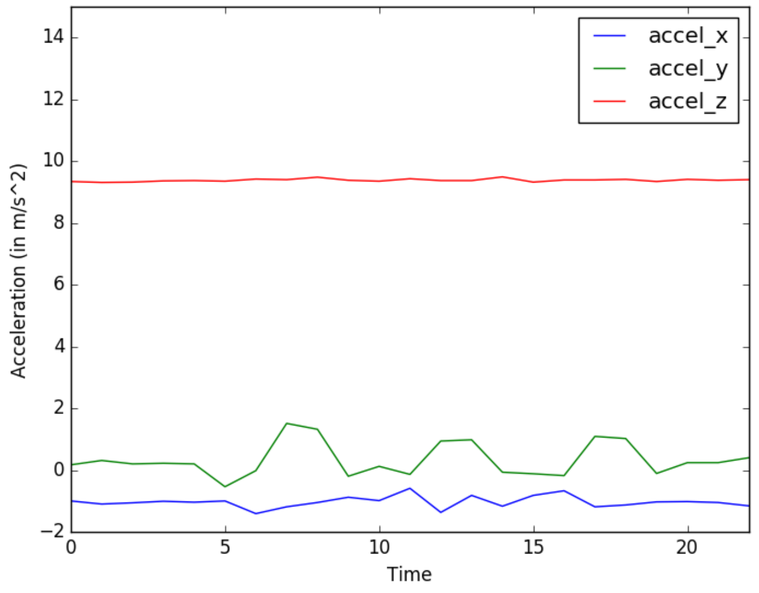


*Figure 6: A graph showing acceleration values over time when moving the accelerometer left and right in the x-y plane*

Flex sensor resistances show a clear trend in the finger movement when making signs. Figure 7 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 be bend anymore and then returns to its original position.

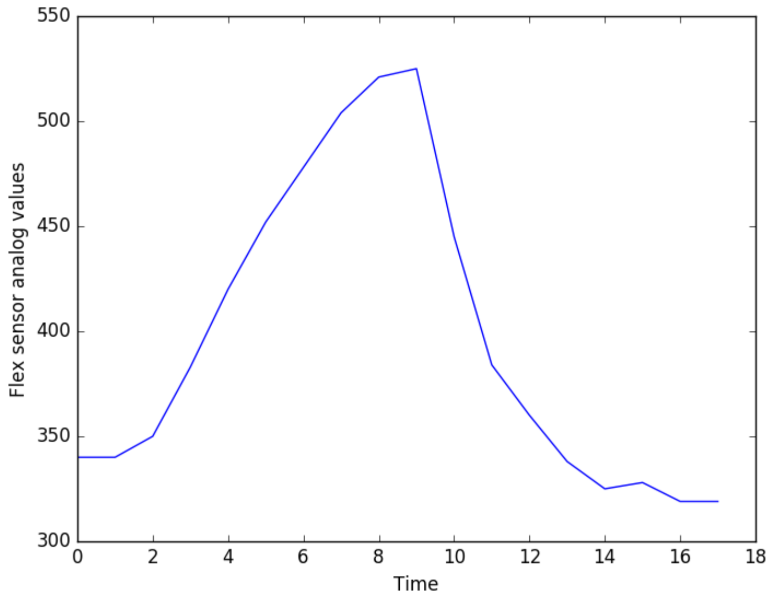


*Figure 7: A graph showing flex sensor analog values over time when bending the finger from a stretched position to a fully flexed position and back.*

Instead of using all the resistance values in defining a certain sign, this information can be incorporated as features using the standard deviation (SD), the average and the max values in a given time range. This has been successfully done before to extract relevant features from the flex sensor for machine learning models like neural networks [18]. For n sensor values denoted by $X = \{x_1, x_2, \dots, x_n\}$ we can calculate the average and the 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 angular velocity along all three axes, and shows a trend similar to the accelerometer, albeit with more noise. Figure 8 shows the angular velocity values from a gyroscope when it is rotated about the z axis.

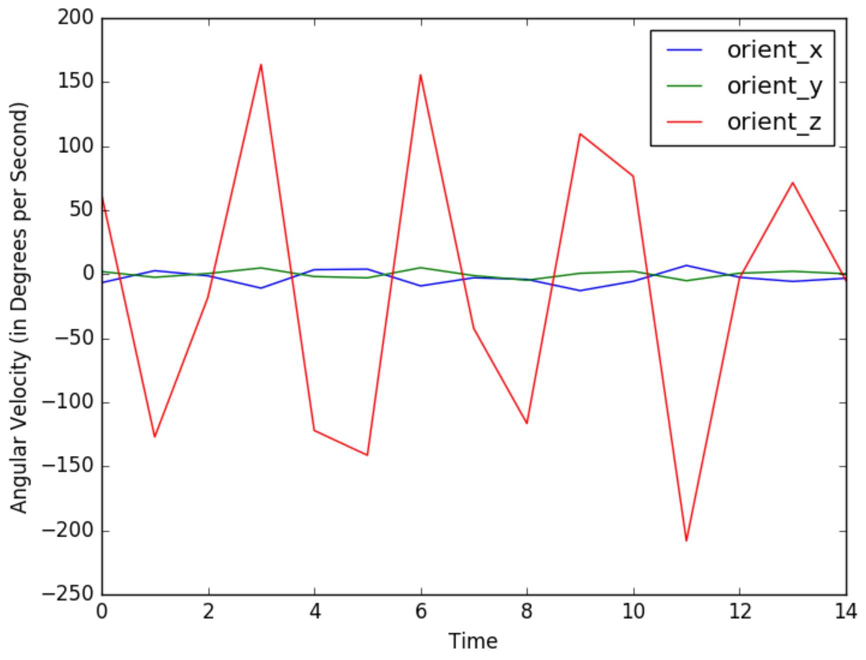


*Figure 8: A graph showing angular velocity values over time when rotating a gyroscope along the z-axis.*

The evidence of rotation along a certain axis can thus be realized by studying the trend in the deviation of the gyroscope readings from a certain stable value - generally zero. To integrate this information as features without using all the values in a time frame, the average value, the standard deviation and the mean absolute deviation (MAD) along all the axes can be considered. SAD and MAD have previously been used as accurate feature representations to detect gyroscope swing for human activity recognition (HAR) [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 Correlation Feature Selection (CFS) method is used. CFS works under the assumption that features should be highly correlated with the given class but uncorrelated with each other [19]. When gathering data for HAR using accelerometer and gyroscope, CFS is commonly used as the feature selector [37], [20].

The merit of a feature subset 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].

#### *3.2.1.2 Machine Learning Models*

After feature extraction from signals in a given time frame, each instance of a feature vector is fed to various machine learning models with its corresponding symbol as a part of 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 symbol given a feature vector as an input. Python's "scikit-learn" machine learning library is used to train the models [21].

Separate models will be trained for ASL and JSL using the same features, as the two languages have overlapping hand motions for various signs. Restricting the model to one language helps detect signs with greater accuracy and makes it easier to incorporate new signs into the model.

It is possible to have signs with multiple parts to it, where more than one action together constitutes a single word. For the purposes of this project, only one-to-one action-to-sign mapping cases are considered, in order to reduce the complexity in the machine learning model that would be added otherwise.

##### *3.2.1.2.1 Naïve Bayes*

The first model, Naïve Bayes, a variant of the Bayesian classifier type, calculates posterior probabilities of each class and estimates the conditional probability of a given feature vector to correspond to 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, they act fairly independent of the gyroscope values and the flex sensor values, making this a good model for predicting symbols [23].

Naïve Bayes is time efficient in both the training and the recognition phases due to its almost 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 will only support 149 signs to begin with.

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:

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

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

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

#### *3.2.1.2.2 Decision Trees Learning*

The second model, Decision Trees Learning, involves construction 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 of the features in the training tuple. Decision trees make it very easy to visualize 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. 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 this decision tree is made while training the model, a given feature vector is passed through it to get the corresponding symbol.

Information gain is used as the metric for splitting input labels at a certain node, in order to realize the most relevant features that cause maximum split. Information gain is calculated using the entropy of the content, which defines the unpredictability of the given values. Mathematically, the entropy is defined as:

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

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

$$\begin{aligned} Info.Gain &= Entropy(parent) - Weighted\ Sum\ of\ Entropy\ (children) \\ &= H(T) - H(T|a) \end{aligned}$$

#### *3.2.1.2.3 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 coordinates of the data in the hyperplane [27].

The SVM model has been used in the past to detect human motions with an accuracy of near 97% [17]. 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 till 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 $\sigma$ is a free parameter defined by the SVM while training the SVC model.

#### *3.2.1.2.4 Ensemble Model*

All three supervised classification models used are not perfect in predicting the right 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. Even though, training the models separately involves a lot of computational power, they are fairly fast in predicting a symbol given a vector once they are completely trained.

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 feature vector will also be flagged so that it can be used to re-train the models in the future.

### *3.2.1.3 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) is 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) is 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.

### *3.2.1.4 Cross Validation*

The data for a total of 149 signs for ASL and JSL was used to train the machine learning model. Four people worked to generate a training data set with 200 iterations for each sign.

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 149 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].

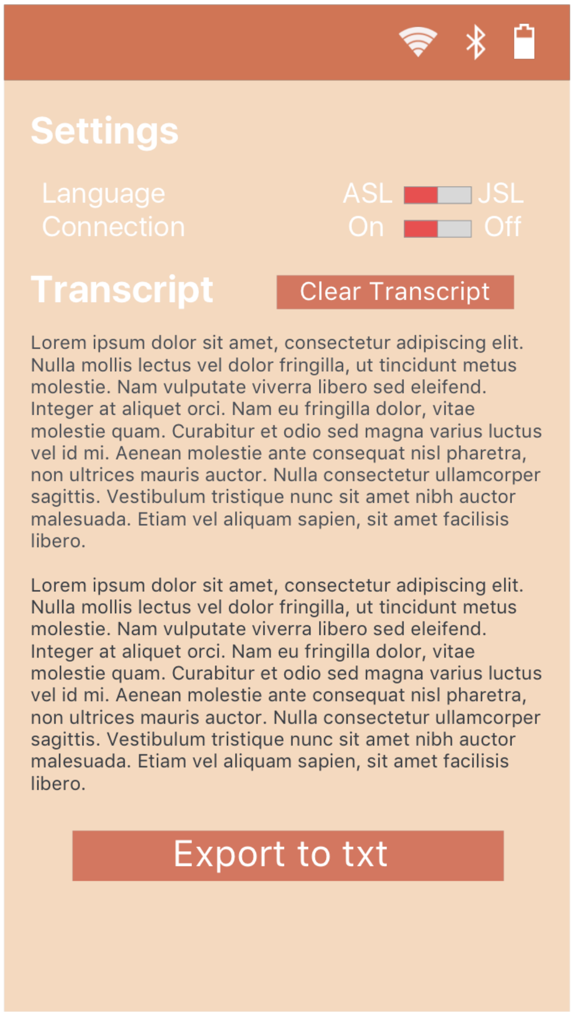
#### *3.2.1.5 Export to Mobile Application*

The model training for all the symbols in ASL and JSL is done on personal computer or cloud computing services as they have higher processing power and they provide more flexibility in the tools available to run the models. Python's "scikit-learn" machine learning library [21] is used on the computers to train the models. Python also provides data science libraries like numpy [31] and pandas [32] allowing the manipulation of features to fit the requirement of the training models.

These trained models are ported over to the Android application to predict the symbols based on the current sensor values in a time frame. This ensures that the product remains portable, as a phone with the prediction algorithm is the only device necessary to translate sign language to its corresponding text.

### *3.2.2 User Interface (UI)*

The user interface for this system is an Android application. Figure 9 below shows the wireframe design of the application. In this, the user has the ability to choose the language preference (ASL or JSL) and also the ability to turn off the active interpretation at any given point in time. Another feature it provides is for 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 anytime whereby a confirmation dialog will appear as a final verification step before clearing the transcript.



Settings

Language ASL JSL

Connection On Off

Transcript

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla mollis lectus vel dolor fringilla, ut tincidunt metus molestie. Nam vulputate viverra libero sed eleifend. Integer at aliquet orci. Nam eu fringilla dolor, vitae molestie quam. Curabitur et odio sed magna varius luctus vel id mi. Aenean molestie ante consequat nisl pharetra, non ultrices mauris auctor. Nulla consectetur ullamcorper sagittis. Vestibulum tristique nunc sit amet nibh auctor malesuada. Etiam vel aliquam sapien, sit amet facilisis libero.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla mollis lectus vel dolor fringilla, ut tincidunt metus molestie. Nam vulputate viverra libero sed eleifend. Integer at aliquet orci. Nam eu fringilla dolor, vitae molestie quam. Curabitur et odio sed magna varius luctus vel id mi. Aenean molestie ante consequat nisl pharetra, non ultrices mauris auctor. Nulla consectetur ullamcorper sagittis. Vestibulum tristique nunc sit amet nibh auctor malesuada. Etiam vel aliquam sapien, sit amet facilisis libero.

Figure 9. UI mock-up of the android application for Voice

### *3.2.3 Text to Speech*

Android has a built-in TextToSpeech (TTS) module which is used to convert text to speech during application development. Figure X 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. Two alternatives to this is Ivona and eSpeak.

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

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

The essential requirement for a TTS engine is that it must be able to read in English and Japanese. Unfortunately, Ivona does not have any support for Japanese and hence, cannot be considered as a feasible option for Voice [33]. Google's TTS and eSpeak support both English and Japanese. Some other factors to be considered when choosing an engine is reliability. Google's TTS rating on the application store is 4/5 whereas the ratings for eSpeak is 3.5/5. Majority of the 1 star reviews for eSpeak argued that the engine was crashing on their devices while others thought that the voice packs they had were too monotone whereas for Google, most of the complaints are for the fact that the engine does not come with a demo application. Hence, Google's TTS is the most feasible solution for Voice.

# 4 Discussion and Project Timeline

## 4.1 Evaluation of Final Design

The Android application satisfies the requirement for having a mobile application, and enables the user to transition between ASL and JSL. It also allows the user to turn on/off the translation itself, and has the ability to display and export the transcript. The trained models that perform the translation satisfies the multi-language support specification.

The HC-06 Bluetooth modules satisfies 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 that 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 3, 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 approximately 30 g, excluding connecting wires and battery pack. The battery pack will be in an enclosed casing and not on the glove itself. 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 (assuming that a physical phone is not part of it) is around $225. This is well within the cost specification, which states the entire system should be under $500. As of right now, the design does not include a case for particular components of the external subsystem, though this can be considered at a later time if deemed necessary.

It is difficult at the design stage to determine whether the accuracy and latency specifications will be fulfilled as this will depend on the machine learning models used and the amount and quality of training data the team can generate. It is hoped that these specifications will be met upon implementation of this design, though time has been allocated for further iteration if required.

## 4.2 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.

### 4.3 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. In addition, supporting multiple languages is a feature that does not exist in any other sign language recognition system.

### 4.4 Student Hours

Table 6 below shows the number of hours spent as of June 30, 2016. The team is slightly behind schedule but has plans to make up the lost hours in the co-op term (see Section 4.6).

*Table 6. Total hours per student from May 11, 2016 to June 30, 2016*

| Student Name | Hours |
|---|---|
| **Akshay Budhkar** | 57 |
| **Eliot Chan** | 59 |
| **Biraj Kapadia** | 56 |
| **Amish Patel** | 54 |

### 4.5 Potential Safety Hazards

As the gloves are worn by the person, the components need to be insulated properly by having conductive wires or threads in the circuit covered by ESD material. This method would also prevent sharp edges of components from making direct contact to the skin. In addition, the gloves need to be kept away from liquids in order to prevent damage and potential electric shock.

## 4.6 Project Timeline

Given that all four team members are working in downtown Toronto for the Fall 2016 term, the team can continue working on the project at a highly reduced intensity (approximately one to two hours per week, on average). As such, the project timeline will span the co-op term in addition to both 4A and 4B academic terms.

The project is broken down into two major phases and four major tasks. Phase one of the project will be done throughout the 4A term and phase two of the project will be carried out during the final coop term and 4B. Figure 10 and Figure 11 illustrate our timeline during the two phases along with the time allocated to each group of tasks. Time allocated to designing and building the external subsystem prototype is in red. Any time allocated towards machine learning tasks are in green, and any tasks related to the companion mobile application are in yellow. Finally, other miscellaneous tasks are shown in blue.

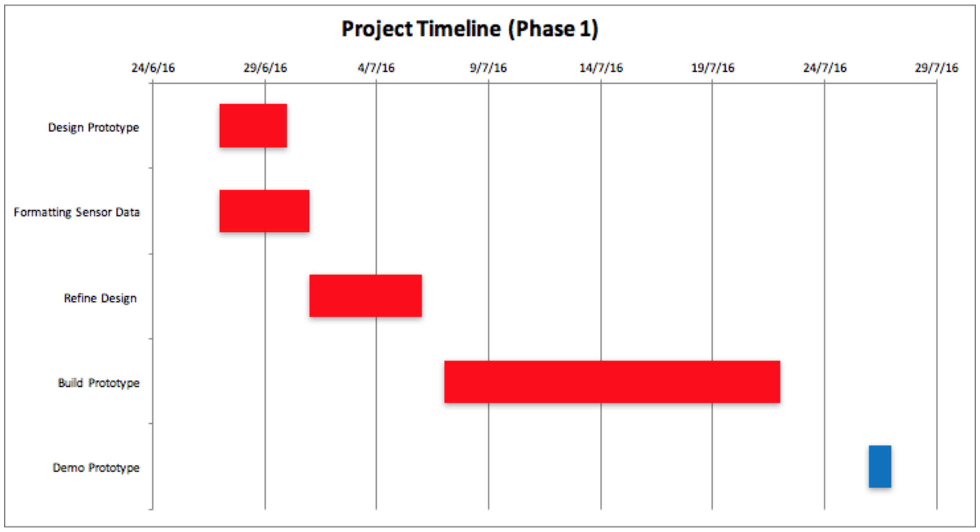


*Figure 10: Phase One Gantt Chart (June 27th to July 27th)*

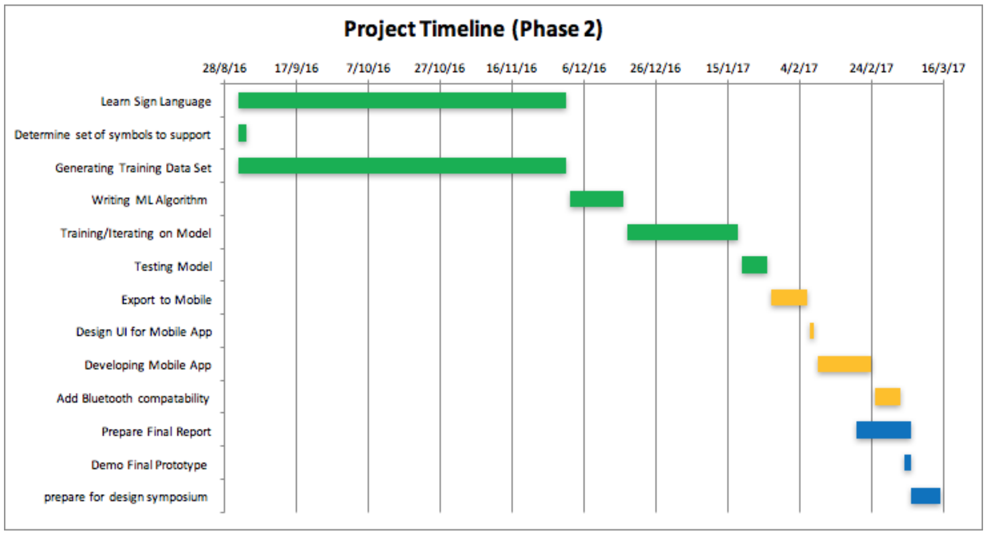


*Figure 11: Phase Two Gantt Chart (September 1st to March 15th)*

## 5 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] "Double Side Spring Loaded 6 x 1.5V AA Battery Holder Storage Case Box: Amazon.ca: Electronics", Amazon.ca, 2016. [Online]. Available: https://www.amazon.ca/Double-Spring-Loaded-Battery-Storage/dp/B00X741EHY/ref=sr_1_1?ie=UTF8&qid=1467229131&sr=8-1&keywords=6+x+AA+battery+holder. [Accessed: 29- Jun- 2016].

[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] *Neural information processing*. 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, "Analysis of Haptic Data for Sign Language Recognition", University of Southern California.

[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] IVONA Software, "Voices," 2016. [Online]. Available: https://www.ivona.com/us/about-us/voice-portfolio/. Accessed: Jun. 26, 2016.

[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].

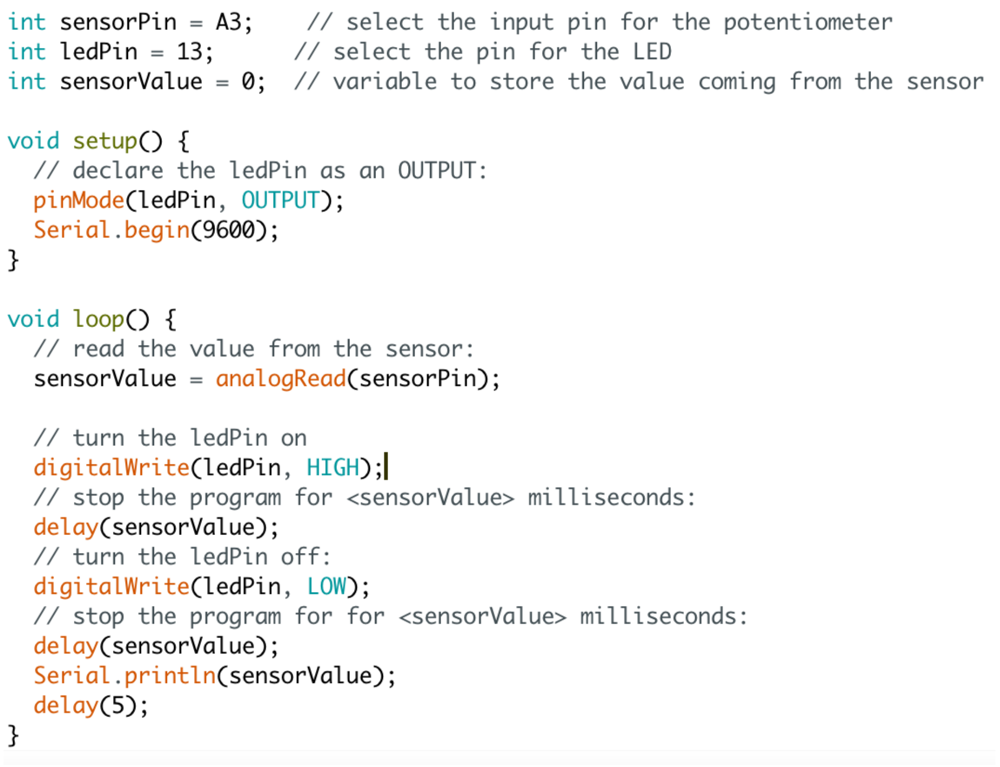
[35] S. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993", *Mach Learn*, vol. 16, no. 3, pp. 235-240, 1994.

[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.

# 6 Appendix

## Appendix A: Code to Read Data from one Flex Sensor



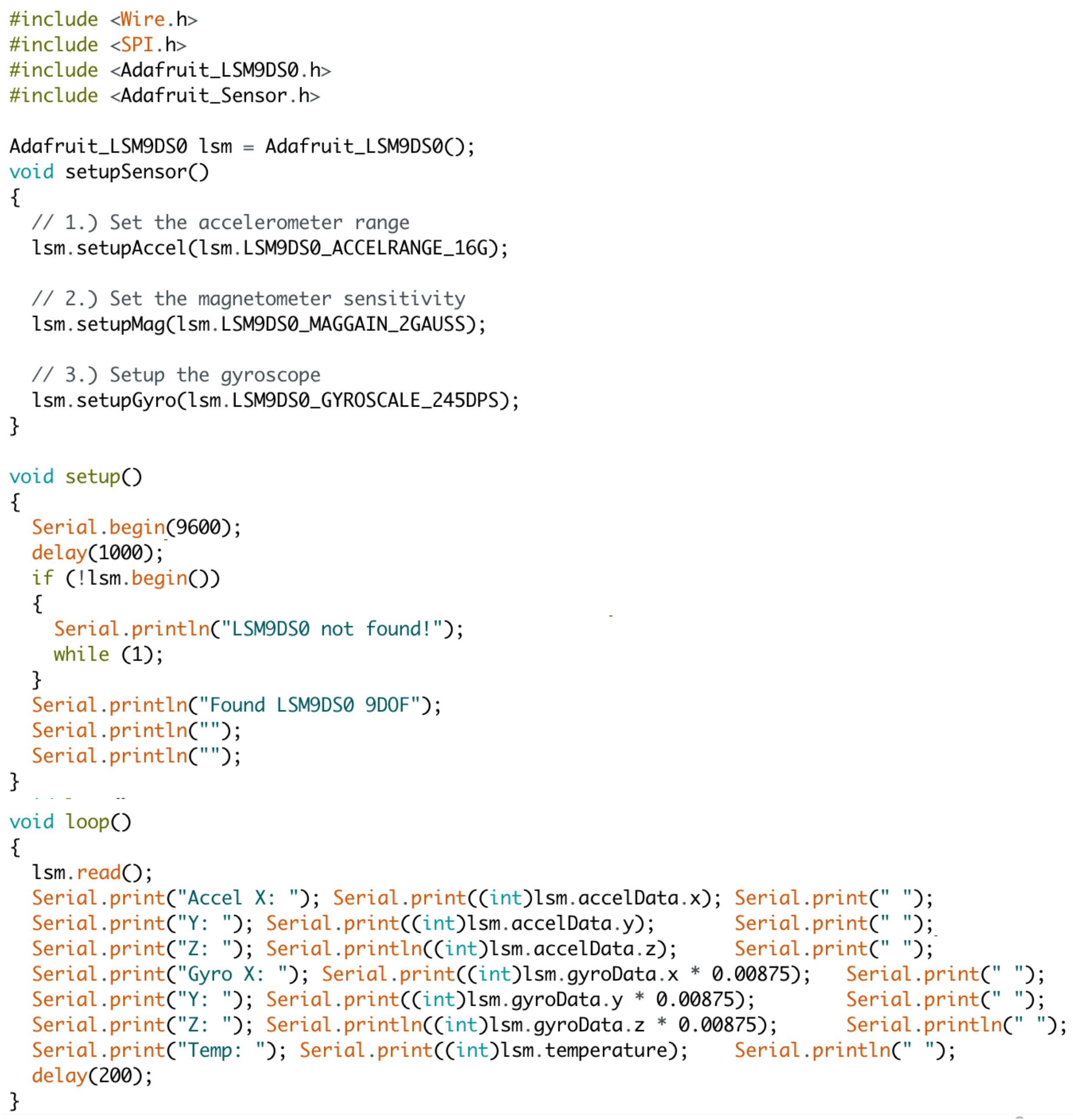
```
int sensorPin = A3;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

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);
}
```