SIGN LANGUAGE TEACHING GLOVE

By

Daniel Fong
Mayapati Tiwari
Reebbhaa Mehta

Final Report for ECE 445, Senior Design, Spring 2013
TA: Igor Federov

01/May 2013
Project No. 29

Abstract

Our group designed and built a glove that can help a person learn gestures to alphabets in sign language. The device consists of a sensing unit, which detects the gestures and a classification algorithm to check the each gesture. The device also provides feedback to the user after each gesture. The device is connected to the computer via Bluetooth to avoid any wires thereby making the device user friendly. Our device can help a person learn sign language quicker.

Table of Contents

1.0 Introduction

1.1. Statement of Purpose

Presently there are no gloves available in the market that can teach sign language, but research has been done to make sign language gloves that can interpret gestures and convert them to speech. The aim of this project was to make a glove device that can be useful to teach sign language. The goal of was to create a glove device that detects sign language gestures for letters used in American Sign Language, and inputs them into a computer, where a computer program checks the gesture. Thereafter a feedback unit, consisting of LED's, informs the users if the gesture was correct. It can be a good learning tool for sign language and the idea can be built on in the future. Our main focus while undertaking this project was to make the device easy to handle, and user friendly.

1.2. Objectives

1.2.1. Goals

- Build a glove device to detect sign language.
- Develop a database of the gestures of sign language alphabets for the computer program to use as reference while checking input gestures.
- Develop a computer program that checks the input gestures.
- Develop a feedback unit to make the user aware of any mistake
- Bluetooth communication between computer and glove device.
- Maximum utilization of space available on the glove so that all the hardware components can fit on it.

1.2.2. Functions

- Flex Sensors, accelerometers and gyroscopes to detect sign language gestures.
- Bluetooth communication between device and computer.
- Perceptron program checks each gesture.
- LED's to help the user correct any errors.
- Glove device powered by batteries to avoid any risk of shocking users.

1.2.3. Benefits

- Vibration feedback and LED alerts help to correct any errors with gestures.
- Convenient to use as the device is wirelessly connected to the computer.
- Help users to become more adept with sign language.

1.2.4. Features

- Tri-axis accelerometer and Tri-axis gyroscope in one glove.
- Bluetooth transmission between computer and glove.
- LED response for error correction.
- One-directional flex sensors on each finger for reference.
- Mathematical Kalman filtering
- Perceptron Algorithm

2.0 Design

2.1. Design Procedure

The top-level block diagram can be seen in Appendix A, Fig.1. As can be seen from the diagram the top level diagram, is further divided into three main separate components – the main board, sensing unit and feedback unit. The main board consists of the battery, microcontroller and Bluetooth module and the feedback unit as can be seen in Appendix A, Fig 2. The sensing unit, Appendix A Fig.3, consists of accelerometers, gyroscopes and flex sensors that are responsible for detecting gestures. The feedback unit, Appendix A, Fig. 4, only consists of LED's. Initially, the feedback unit consisted of LED's and vibration motors to implement haptic feedback. However, the haptic feedback was removed from the project as the scope of the project was getting out of hand.

The data from the sensing unit is sent to the computer via Bluetooth where it is filtered using Kalman Filter. The filtered data is then checked with the help of a machine-learning algorithm, Perceptron Algorithm, to check if the users gesture was correct. After the check is performed, the valid feedback is sent to the feedback unit to notify the user of the final result of the gesture. The overall data flow for the whole device is shown Appendix A Fig. 5.

2.1.1. Hardware

2.1.1.1. Main Board

The main board consists of the power supply, Bluetooth module, microcontroller and feedback unit. The main board schematic is shown in Appendix A Fig. 6.

Power Supply:

The power supply is used to power all the hardware components. The power supply needed in our design was one that could provide a 5V to the microcontroller and 3.3V to all other hardware components. Moreover, we needed a power supply that would occupy very less space on the glove since a major factor in our design was to mount all the hardware components on the glove. We did not want to power the components using an USB, because we wanted to reduce the number of wires going from the glove to the computer. For this purpose, Lithium Backpack, supplying 5V, made especially for the Arduino Uno microcontroller, was our first choice. Its PCB matched the size of the Arduino Uno and could be mounted on the back of the microcontroller helping us to utilize the space available on the glove.

Microcontroller:

The microcontroller used was the Arduino UNO. The microcontroller will collect all the information from the sensing unit. After which, it will relay the data to the computer via Bluetooth or serial interface. Reasons we chose the Arduino:

It is easily programmable, since it has numerous communication libraries available, for I^2C protocol as well as communicating with the MPU6050 (accelerometers and gyroscopes) and it has on board voltage regulators to provide 3.3V and 5V as needed by the components.

Bluetooth:

The Bluetooth transmitter and receiver was the SeeedStudio Bluetooth Shield for the Arduino UNO. It was chosen due to low cost, low power and device ergonomics.

Feedback Unit:

In our design we planned to include a feedback unit to notify the user of the result of the gesture for each alphabet. This is an important feature, as it helps the users learn better by letting them know of what was wrong for each gesture. Providing feedback helps a user to better grasp the gestures for each sign, thus helping him to learn faster. We decided to use LED's for feedback.

2.1.1.2. Sensing Unit

The purpose of the sensing unit is to detect each gesture as it is performed. The unit was designed as whole as the focus was laid on the components needed to detect and differentiate each gesture for each alphabet in the American Sign Language.

For sign language, it is important to differentiate between the fingers, which are bent from the ones, which are not. We opted to use one directional flex sensors, FLX-03 on each finger to serve this purpose.

For each gesture, it is also important to know the orientation of each finger along with its position and movement. For this purpose, we decided to use accelerometers and gyroscopes placed at the fingertips of each finger. Accelerometers were used for tilt sensing, which helped us to know the orientation of the glove and help us to differentiate hand gestures. Gyroscopes were used to detect the angular velocity in three-axis, which will help us in calculating the angle of the glove in each direction that will help us to know the orientation of the glove. Different hand gestures can be differentiated this way. Gyroscopes will be very useful for two particular sign language gestures, i.e, 'J' and 'Z' as they required finger movements in air.

Thus we decided to use five gyroscopes and accelerometers on the fingertips. Additionally we used one gyroscope and accelerometer on the palm of the hand to know the relative position of the hand along with each finger. It will be helpful to make the glove perform in real time. Due to lack of space available near the fingertips, MPU-6050 made by Invensense, which has an integrated 3-axis gyroscope and 3-axis accelerometer, was used. It helped us to utilize the space available on the glove more efficiently and avoid crowding of devices at the fingertips.

2.1.2. Software

While going over the design, we realized that we would need a filter to reduce the noise from the data provided by sensing unit. A machine-learning algorithm was needed to check each input gesture.

2.1.2.1. Kalman Filter

A filter was needed in our design to filter out the noise from the data received from the sensing unit to provide more stable and suitable values. Accelerometer and gyroscope data has a high level of noise due to high-speed movements and drift respectively. Thus, an algorithm was needed to filter out the noise to provide more accurate data. We decided to implement a Kalman Filter in our design because it can provide a high accuracy level from the data.

2.1.2.2. Perceptron Algorithm

A machine-learning algorithm was required to check and differentiate between gestures. We decided to use the perceptron-learning algorithm as our classification software because it is perfect for an n-dimensional linear functions and using this we can map any n-dimensional linear separator.

2.2. Design Detail

2.2.1. Hardware

2.2.1.1. Main Board

Power Supply:

The 5V power supply from Lithium Backpack was used to power the Arduino Uno and thereafter voltage regulators onboard of the microcontroller were used to provide 3.3V to power all other components, as per the design requirements. Moreover, it was rechargeable via USB. However the Backpack got shorted while working on a metal table, and an alternate method was used.

Alternative: With no time to replace the Lithium Backpack, an Energizer 9V battery was used. The voltage regulators on the Arduino Uno were used to provide 5V and 3.3V to power the microcontroller and all the other hardware components respectively. The schematic for power supply can be seen in Appendix A Fig. 7.

I^2C Switch [1]:

Since each of the MPU6050 devices have the same address on the I^2C bus, we needed to use the 8-channel I^2C switch by TI the TCA9548A. It has eight bidirectional switches that can be controlled via the I^2C bus. Any individual channel can be selected which is determined by the contents of the Programmable control register. The system master can thus communicate with each slave device on the bus individually.
This allows us to ensure the integrity of the data we are gathering from each of the sensors.

Microcontroller:

The Arduino receives data from the sensing unit and receives the result of the analyzed data from the computer. It has two outputs: firstly, the microcontroller sends the users input gesture data to the Bluetooth module to be analyzed on the computer. Secondly, it sends a signal to the

Feedback unit to alert the user of the result it has received from the computer where the users inputs were compared using the machine-learning algorithm.

Bluetooth:

The data to the Bluetooth module comes from two sources. First, it gets data from the sensing unit via the microcontroller that it sends to the computer for comparison and analysis. Secondly, after the computer software has processed and compared the data, the Bluetooth receives the result of the data analysis of the users input gesture, which it relays to microcontroller for feedback.

Alternative: When the Bluetooth module stopped working in our project, we just used a USB cable for data transfer.

Feedback Unit:

The feedback unit consists ten LEDs (5 red and 5 green), placed on each finger. If the red LED turns on, it signifies that the corresponding finger was in the correct position for that gesture while a red LED shows that a finger was incorrectly placed.

Initially, the plan was to use haptic feedback using vibration motors along with LED's, but the idea was abandoned due to time constraint and the scope of the project. Finally, the feedback unit was implemented on the main board PCB. The feedback unit circuit on the main PCB is shown in Appendix A Fig. 6. The feedback circuit by itself is shown in Appendix A Fig. 8.

2.2.1.2. Sensing Unit

Flex Sensors:

FLX-03 flex sensors have resistance range of 10kΩ to 40kΩ. When completely unflexed, the resistance of the sensor is 10kΩ but as the resistance is flexed more the resistance starts to increase. When the sensors are completely flexed, the resistance can be a maximum of 40kΩ. All the data from the flex sensor is sent to the microcontroller.

To incorporate the flex sensors in the design, a basic flex sensor circuit (voltage divider circuit) was built using LM358 op-amps as impedance buffers because the low bias current of the op amp reduces error due to source impedance of the flex sensor as voltage divider. The flex sensor circuit is shown in Appendix A Fig. 9 [2]. From the figure Equation 1.1 is derived to find the output voltage of the circuit.

$$V_{out} = V_{in}\left(\frac{R_1}{R_1 + R_2}\right) \qquad \text{... (1.1)}$$

where R_1 is 10kΩ and R_2 is the flex sensor resistance.

Initially the flex sensors were tested using Fig. 9 in Appendix A. The resistance values were observed at different degrees of bends in the flex sensor. The results can be observed in Table 1 and Figure 10 in Appendix A for different degrees of bends. The output voltage was calculated using Equation 1.1. From the table and the corresponding graph it can be seen that resistance of the sensor was 10kΩ when the sensors were unflexed and the resistance increased as the sensors were bent. Thus, they met the initial requirements.

Tri-axis Gyroscopes and Tri-axis Accelerometers:

The data from the accelerometers were used to find the orientation of the fingers of the glove by identifying the direction of the gravity vector through the accelerometer. The scaled accelerometer data (ax, ay and az) is calculated using Equation 1.2, where 16384.0 is the sensitivity of the accelerometer. The sensitivity is set to this value.

$$ax, ay, az = \frac{AccelerometerData}{16384.0} \qquad \text{... (1.2)}$$

The orientation of MPU 6050 about the X-axis and Y-axis is known as roll and pitch respectively. Roll and pitch can be calculated using Equation 1.3 and Equation 1.4 respectively.

$$Roll = \tan^{-1}(\frac{ay}{az}) \qquad \text{... (1.3)}$$

$$Pitch = \tan^{-1}(\frac{ax}{\sqrt{(ay)^2 + (az)^2}}) \qquad \text{... (1.4)}$$

Yaw of the system (orientation about the Z axis) cannot be obtained without using a magnetometer, so this was not implemented in this system.

The gyroscopes were used to measure changing orientation of the system, and these would have been useful if the real time implementation of the sign language glove was achieved. The scaled gyroscope data (gx, gy and gz) is calculated using Equation 1.5, where 131.0 is the sensitivity of the gyroscope. The sensitivity is set to this value.

$$gx, gy, gz = \frac{GyroscopeData}{131.0} - GyroscopeZeroOffset \qquad \text{... (1.5)}$$

The layout for MPU-6050 or mini fingerboards along with the MUX (TCA9548A) can be seen in Appendix A Fig. 11. The schematic for MPU-6050 can be seen in Appendix A Fig. 12 [3].

Out of the six MPU-6050, we had decided, one stopped working and we could not replace it, so we just worked with five on the fingers.

2.2.2. Software

2.2.2.1. Kalman Filter [4]

The Kalman filter works by recursively processing set of measurements containing noise and combining the measurements to estimate the actual state of the unknown variables. The filter first makes a prediction of the different states. Then, it makes a weighted average based upon the certainty of the prediction between the predicted state and the newest measurement to produce the estimate of the underlying states. The Kalman Filter assumes that the error and measurements have a gaussian distribution.

In our situation, the Kalman Filter was used to filter out the noise from the accelerometer data to determine orientation, and it would've also been used to combine the gyroscope and accelerometer data to obtain a real time observation of the orientation, but there was not enough time to achieve this implementation. The Kalman Filter process has two stages: predict and update.

<u>State-Space Model:</u>
Kalman Filter assumes that the true state of the system, x_k, as ahown in Equation 1.6

$$x_k = F_k x_{k-1} + B_k u_k + w_k \qquad \text{… (1.6)}$$

where, x_k is the true state of the system assumed by Kalman Filter,
F_k is the model of the system,
B_k is the control input,
u_k is the gyroscope data,
$B_k u_k$ is [gyroscope data × Δ(t); 0], and
w_k is the process noise, which is modeled as Gaussian with mean zero and covariance Q.

Kalman Filter assumes that the observation of the system, z_k, as shown in Equation 1.7.

$$z_k = H_k x_k + v_k \qquad \text{… (1.7)}$$

where, z_k is the observation of system assumed by Kalman Filter,
H_k is the observation model,
x_k is the state of the system, and
v_k is observation noise which is modeled as Gaussian with mean 0 and covariance R.

<u>Stage 1: Estimation</u>
Equations 1.8 and 1.9 show the estimation by the Kalman Filter

$$\hat{x}_k^- = F\hat{x}_{k-1} + B_k u_{k-1} \qquad \text{… (1.8)}$$

$$\hat{P}_k^- = F\hat{P}_{k-1}F^T + Q \qquad \text{… (1.9)}$$

where, x_k is estimated state of system,
P_k is estimated covariance of the system,
F is the model of the system [1 - Δ(t); 0 1],
Q is the variance of the process noise w_k.

The control input Bu comes from the gyroscope measurements [gx;0] × Δ(t),Δ(t) is the time between samples in the system, but the project was not able to run in real time, and stationary sign language signs were used, so the variance was changed to make the gyroscope impact negligible. Q is the variance of the process noise w_k.

Stage 2: Correction
Equations 1.10, 1.11 and 1.12, can model the update by the Kalman Filter.

$$K_k = \frac{\hat{P}_k^- H_k^T}{H\hat{P}_k^- H^T + R} \quad \text{... (1.10)}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k \hat{x}_k^-) \quad \text{... (1.11)}$$

$$\hat{P}_k = (I - K_k H_k)\hat{P}_k^- \quad \text{... (1.12)}$$

where, K_k is the optimal Kalman gain.
$\hat{x}_k$ is the resulting state determined by Kalman Filter
$\hat{P}_k$ is the resulting covariance determined by Kalman Filter
Z_k is the observed measurements.
H_k, H is the observation matrix [1;0].
I is the identity matrix.
R is the covariance of observation noise.

Originally, the Kalman Filter was tested using Matlab. The simulated results for Kalman filter acceleration and Kalman filter distance can be seen in Appendix A Fig. 13 and Fig. 14 respectively. The original signal was an object moving with a constant acceleration of $1m/s^2$ with initial position at 0. Gaussian noise of mean 0 and standard deviation .03 was added to the position measurement. The Kalman Filter was applied to the acceleration and position measurements to predict the original values based upon the noisy measurement inputs. As can be seen from the figures, the kalman-filtered output (green line) was smoother as the noise from the data has been filtered. The original unfiltered data was more jagged (blue line). Thus, the results of the filtering are evident from the figures.

2.2.2.2. Perceptron Algorithm[5]

Perceptron algorithm is a classification algorithm. It matches the vector model value to the input value to check if the input is correct. We decided to use the perceptron-learning algorithm as our classification software; we picked this for a few reasons. It is perfect for an n-dimensional linear functions and using this we can map any n-dimensional linear separator that (in our case) passes through the origin to an n-dimensional weight vector such that $W_x > 0$ for positive points and $W_x < 0$ for negative points.

The algorithm consists of two parts. First the algorithm learns via supervised learning. It is during this time that the algorithm comes up with a model that fits the one at hand and assigns weights. After about 100 data sets of each symbol, the algorithm should be able to correctly match the gestures performed by the user up to 97% accuracy.
There are two important equations used in this algorithm, one to update the weights and one to determine the error.

$$w_j(t+1) = w_j(t) + n(d-y)x \quad [3] \qquad \ldots (1.13)$$

Where,

- d is the desired output
- t is the iteration number and
- n is the gain or step size, where 0.0<n<1.0

$$E = \frac{1}{2}\sum_{i=1}^{p}\left|y^{(i)} - d^{(i)}\right|^2 \quad [3] \qquad \ldots (1.14)$$

Where, p is the number of input/output vector pairs in the training set.

The perceptron-learning algorithm is as follows [3]:

- Initialize weights and threshold to small random numbers.
- Input vector x to the neuron inputs and calculate the output
- Update the weights according using equation (1.13)
- Repeat steps 2 and 3 until:
    - The iteration error (calculated by equation (1.14)) is less than a user specified error threshold, or
    - A predetermined number of iterations have been completed.

The data flow for Perceptron can found in Appendix A Fig. 15. During initial tests, the simulations for Perceptron were achieved by implementing single neuron Perceptron in 2 dimensions on Matlab. The simulations are based on two sets of random data divided by the line x=y. In the first simulation, shown in Appendix A Fig. 16, the training data size is much larger (400) compared with test data size (84), but the total errors are zero. In the second case, shown in Appendix A Fig.17, the training size is more comparable (500) to our test data size (469) and there is only one error. From the figures we can see that the algorithm has been able to successfully differentiate between the two data sets. This goes to show that even with relatively small training data sets this algorithm performs well and that it is suitable for our implementation.

3.0 Verification

3.1. Power Supply

The Lithium Backpack (5V) and the 9V Energizer battery were tested using a digital multimeter to make sure the batteries were providing the rated voltages. Thereafter, the main requirement that was satisfied by both the batteries that were used was to provide 5V to the microcontroller and 3.3V to all the other hardware components. Both batteries supplied the 3.3V by using the voltage regulators on the Arduino Uno. The Energizer battery also provided 5V to the microcontroller using the same method. As mentioned earlier, the Lithium Backpack was shorted so the Energizer 9V battery was used as a substitute. It was a perfect backup as it met all the requirements.

3.2. Microcontroller

Just as a precautionary measure, we tested the microcontroller. We made the Arduino Uno send out a high signal to the LED. After that we confirmed that the LED lights up. The Arduino was able to successfully receive data from the sensing unit and communicate well with the Bluetooth module. However, our project underwent a design change and we decided to implement the kalman filtering as well as the machine-learning algorithm on the computer instead of the microcontroller. The feedback unit also worked with the microcontroller but not in real time.

3.3. Bluetooth

To verify if our Bluetooth module is functioning correctly, the device was initialized by connecting to the Arduino and then the computer was paired with the Bluetooth module. Once computer was connected to it wirelessly, we sent a test signal to the Arduino that turned on the on-board LED. Two-way communication was established by programming the microcontroller to send data through Bluetooth to the computer and thereafter verifying that the same data was received on the computer serially through the USB port.

3.4. Flex Sensors

The flex sensors could not be used in the final design because they broke down due to excessive heat while soldering. As a result, the resistance output from the sensors not within the desirable range of 10kΩ to 40kΩ. As a consequence of the flex sensors breaking down the data received from the sensing unit was insufficient which decreased the accuracy with which different gestures could be differentiated. The accuracy diminished greatly as we tried to implement the gestures for more alphabets. Ultimately, the project had to be limited to only five alphabets (A, B, V, L and Y).

3.5. Accelerometers & Gyroscopes (MPU6050)

Once the mini finger (MPU6050) PCB's were ready, they were at a specified orientation angle and data was collected from the sensors using the microcontroller. The data was filtered using the Kalman filter. Thereafter the gyroscope and accelerometer data was plotted to check if it was approximately 0°/s (+/- 2°/s) and at the specified angle (+/-2°) respectively. The resultant graphs are shown in Appendix B Fig. 1.

3.6. Feedback Unit (LED's)

The LED's were tested on a breadboard at 3.3V. Thereafter, the LED's were working as a feedback unit with the microcontroller but not in real time.

3.7. Kalman Filtering

Once the glove was ready, the Kalman filtered was tested to make sure it worked with real data. The filter was used on training and testing data sets for each finger. Examples of the test results for Symbol A can be seen in Fig. 1 in Appendix B. The graph in Fig. 1 shows that the Kalman filtered pitch follows the measured values of pitch from accelerometer while rejecting some of the noise. From the figures, it can be seen that the filter values (red line) is less noisy as compared to original data from sensing unit (blue line). Graphs in Fig. 2 and Fig. 3 are an example of the set of graphs verifying that the training data matched the test data with minor differences. The graphs show the roll measurement on the thumb between the learning set of 'A' and the testing set of A. Both graphs show that the Kalman filtered roll stayed within 5 degrees of 70 degrees, so the testing and learning set closely matched.

From Fig. 4 in Appendix B, it is evident that Sensor 3 differed from the other sensors by having especially noisy accelerometer measurements that the Kalman Filter could not adequately reject. As a result, the output pitch of the Kalman Filter varied between -30 and 20 degrees. The excess amount of noise in sensor 3 contributed to some of the error in the decision-making.

3.8. Perceptron

Initially we trained the algorithm on test data for five gestures form the alphabet. The machine-learning algorithm was tested by collecting unseen samples of filtered sensor data. Our training set size was 100 samples and we received an accuracy of 75%. We wait for the algorithm to classify 10 samples, once that is complete, the gesture which was, classified the most number of times. This improved our gesture recognition accuracy.
We aimed for an accuracy of 98% but were only able to achieve an accuracy of 75% this might be because of our missing flex sensor data. With more data to classify each gesture by the algorithm can better assign vector weights to each gesture from the training set.

The Requirements and Verification table can be found in Appendix B Table 3.1.

4.0. Cost Analysis

The cost analyses for the project was divide into two parts. Firstly, it can be divided into the labor costs the number of hours that we as a team put in. Secondly, it can be divided into the cost of the parts used to complete the project. After analyzing the numbers together the project seems to be very costly, however large-scale production of the glove would certainly drive the costs down. Moreover to reduce costs cheaper alternatives to the parts can be used. The total cost of the project was $47,635.00.

4.1. Labor

The labor cost was calculated assuming that each of us worked approximately thirteen hours a week earning thirty five dollars for an hour of work, even though the hours put in towards the end of the project were a lot higher. Based on the above numbers the total labor cost for the entire length of the project was $47,250. A detailed breakdown of labor costs is shown in Appendix C Table 1.

4.2. Parts

A lot of different parts were used to complete the project. The total cost of these parts was $385. The break down of the expenditure on parts is shown in Appendix C Table 2.

5.0. Conclusion

At the end of the project due to various components breaking down we were not able to achieve the goal we had set out for. However, even with our limited resources we were able to demonstrate five different sign language gestures, for the alphabets A, B, V, L and Y. We had to choose accuracy with which the device was able to detect gestures over the number of gestures it was able to recognize.

5.1. Accomplishments

Even though the device was not able to identify all gestures due to a decrease in accuracy as the gestures were increased, we can certainly take away a few positives from this project.

- We were able to get optimum amount of data from the accelerometers and gyroscopes.
- We successfully implemented the Kalman Filter on the data from gyroscopes and accelerometers to get more accurate and reliable data.
- We successfully implemented the Perceptron Algorithm to check gestures.
- We were efficiently able to detect and differentiate among 5 gestures (A, B, V, L and Y).

5.2. Uncertainties

In our project uncertainties were present in the wireless communication between the computer and the glove's Bluetooth device. The glove's Bluetooth device is unreliable and whimsical, but it functions correctly sometimes but other times it doesn't pair with the computer's Bluetooth at all. This can be attributed to a broken antenna, or the Bluetooth chip isn't being programmed or it may be some other hardware issue.

5.3. Future Work

Even though our project was a partial success there is a wide scope for improvement in the future. With the correct packaging the device can even be marketed in the future. The following steps could be taken in the future to better the device.

5.3.1. Replacing Parts

The first step we would take is to replace the broken flex sensors and test our design because with more data we will be able to attain more accurate data for each gesture which would also make it easier to differentiate between gestures. With this the vocabulary can also be increased.

5.3.2. Haptic Feedback

Vibration motors can be placed at the fingertips to alert the user when a hand gesture is incorrect by vibrating only for the fingers that are incorrectly placed. Motors can be controlled by the Arduino, which after checking the input gesture alerts the motors when a gesture is wrong. Currently our design checks if the entire gesture is correct not each finger.

5.3.3. Computer Software

In the future we can expand the functionality to include computer software that will be tutorial software that helps to teach sign language in real time. The computer software sends the data for the current gesture to the Arduino and receives data from the gyroscope and accelerometer along with the output of the perceptron algorithm, this way it tracks the user's progress. The computer software can record the user's learning curve, by recording data from numerous training sessions.

5.4. Ethical Considerations

The purpose of this project is to create a device that can recognize and give the user feedback on ASL gestures, which helps the user learn sign language actively through feedback mechanisms. We will make sure our device is safe to use. We will take the necessary precautions to make sure that no one is harmed while using the device. The following IEEE code of ethics were relevant to this project.

- *To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment [6];*
- *To avoid injuring others, their property, reputation, or employment by false or malicious action [6];*
  Safety is one of the highest priorities for the glove device. The device should be carefully made to not expose the user to any current from the circuit. We also need to take necessary precautionary procedures incase the device components get too hot, since they are close to the skin and may cause harm. During the development process safety precautions must be taken while soldering and using the hot glue gun.
- *To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations [6];*
- *To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others [6];*
- *To be honest and realistic in stating claims or estimates based on available data [6];*
  We feel that this is a vital part of the learning process and will ensure that I adhere to it. I will uphold IEEE code of Ethics and Academic Honesty.

6.0. Appendix

6.1. Appendix A – Design

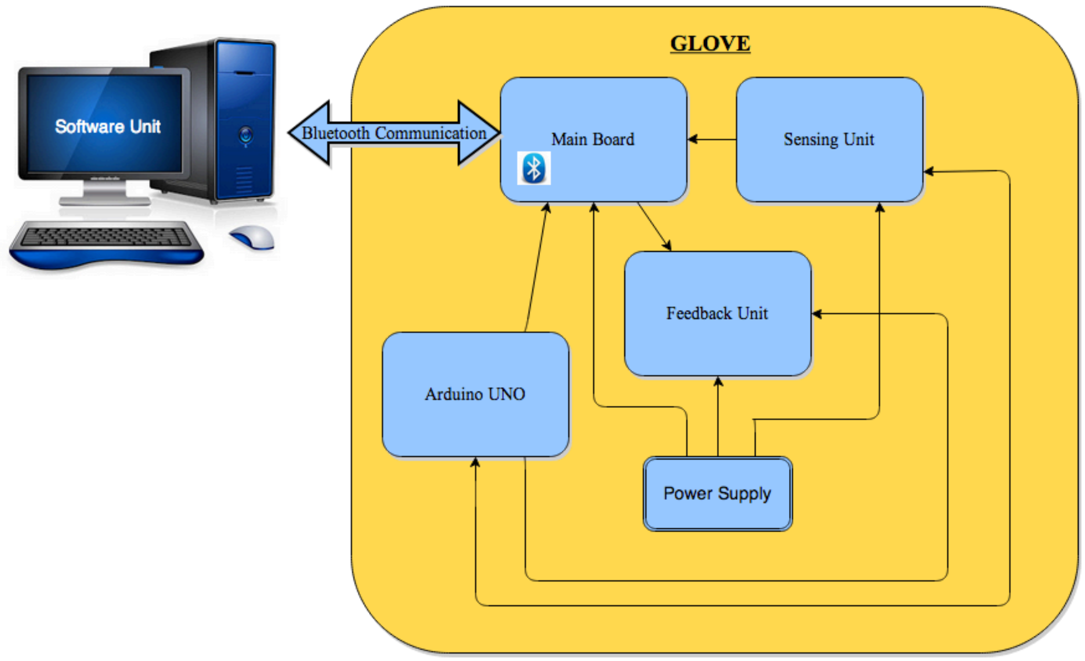

Figure 1: Top Level Block Diagram

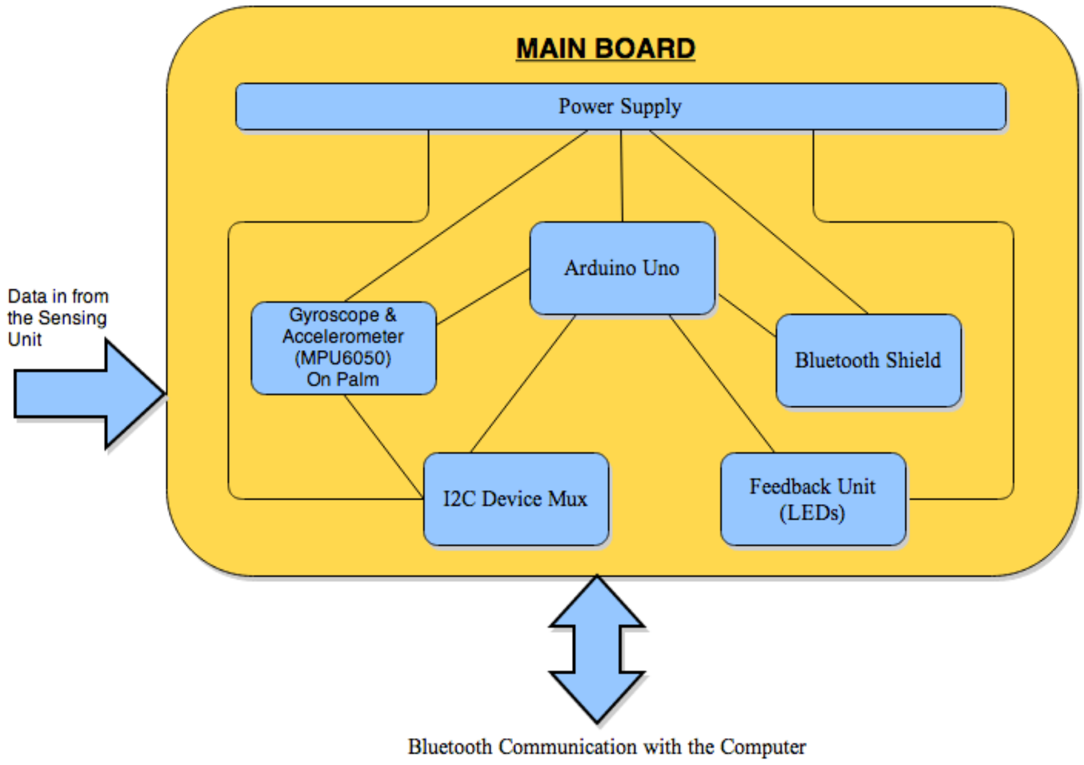
MAIN BOARD
Power Supply
Arduino Uno
Data in from the Sensing Unit
Gyroscope & Accelerometer (MPU6050) On Palm
Bluetooth Shield
I2C Device Mux
Feedback Unit (LEDs)
Bluetooth Communication with the Computer

Figure 2: Main Board Block Diagram

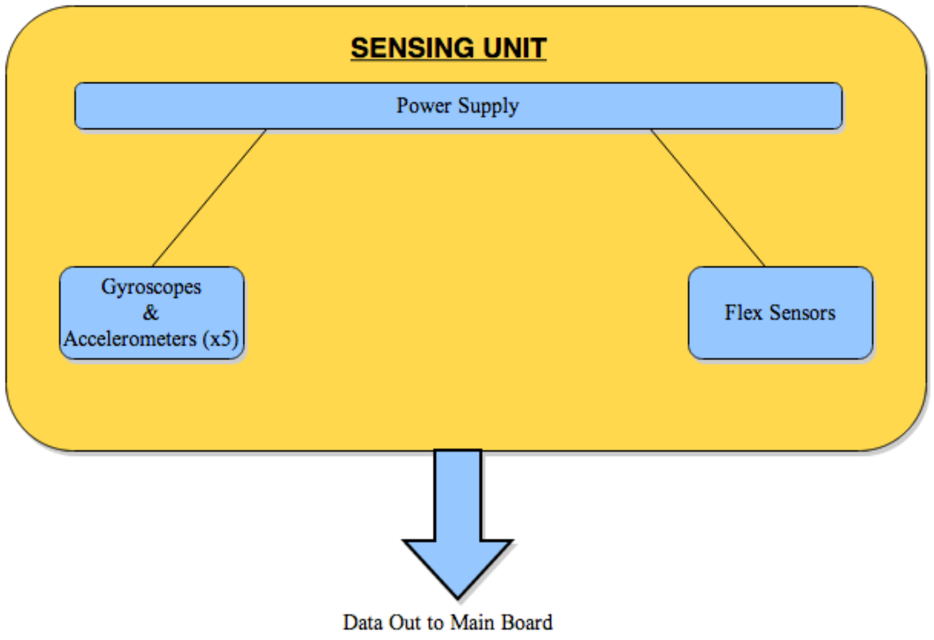

Figure 3: Sensing Unit Block Diagram

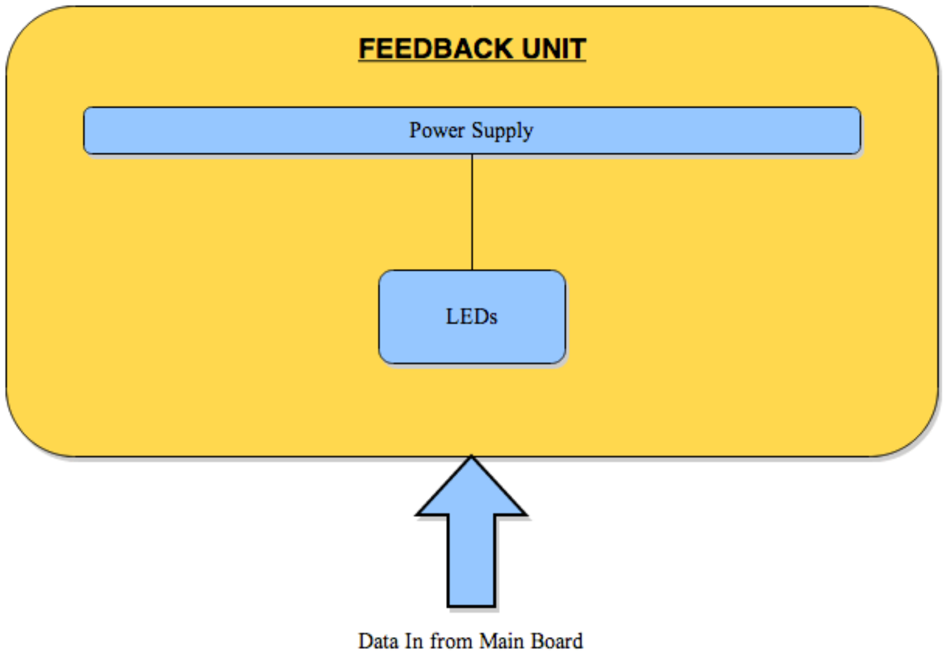

Figure 4: Feedback Unit Block Diagram

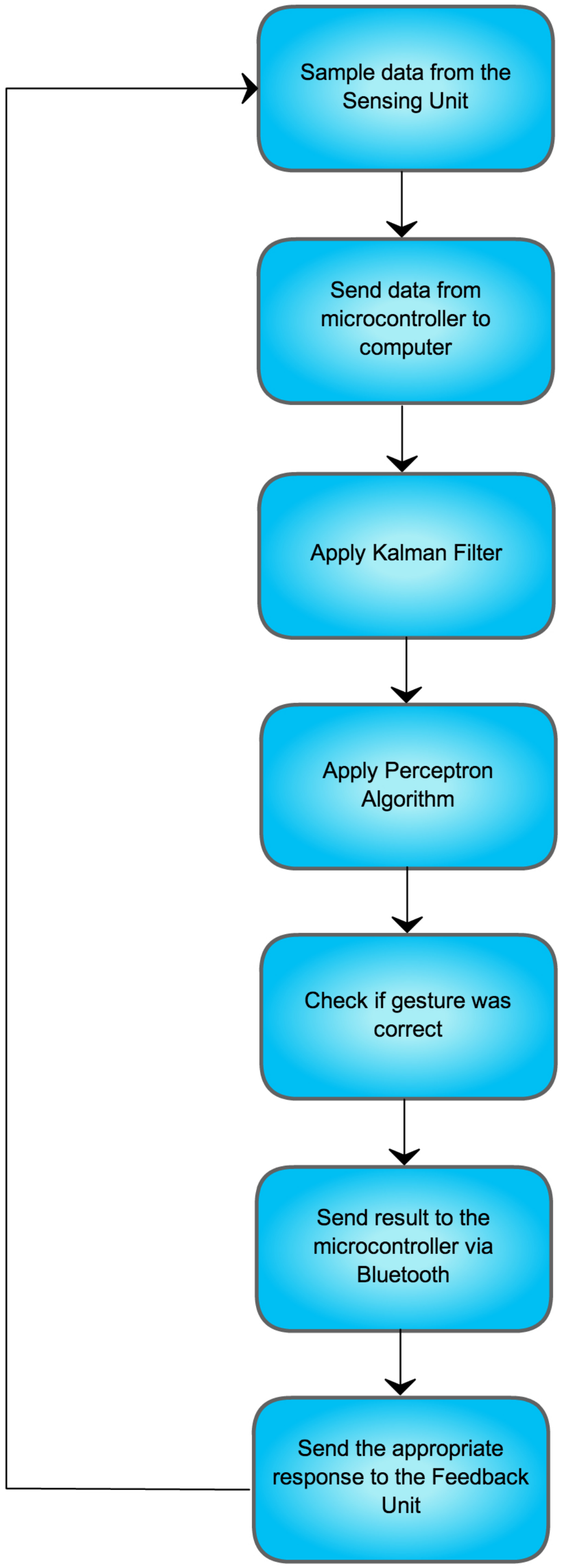
Sample data from the Sensing Unit
Send data from microcontroller to computer
Apply Kalman Filter
Apply Perceptron Algorithm
Check if gesture was correct
Send result to the microcontroller via Bluetooth
Send the appropriate response to the Feedback Unit

Figure 5: Overall Data Flow

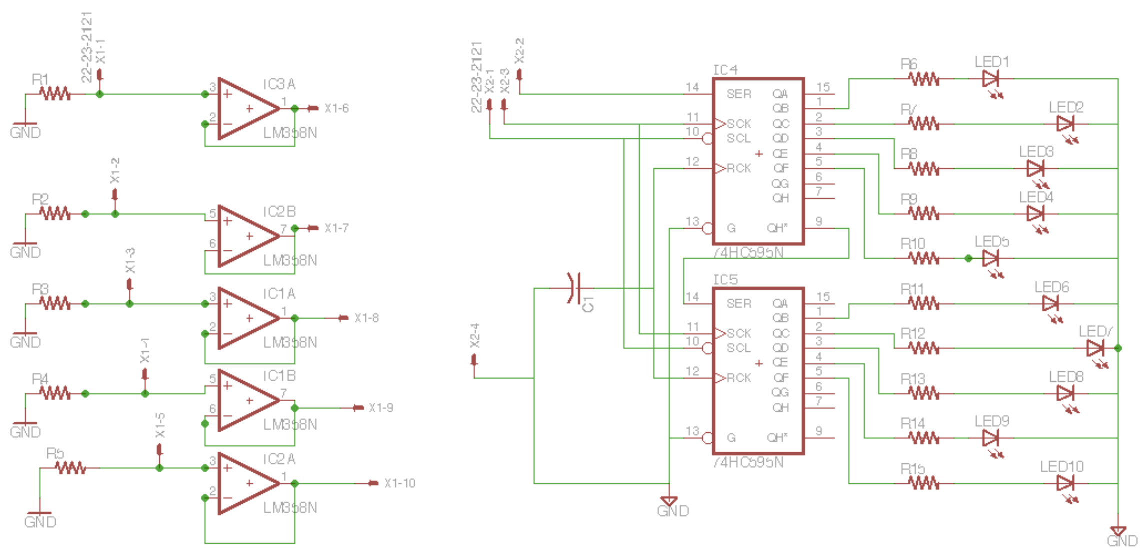

Figure 6: Main Board Schematic

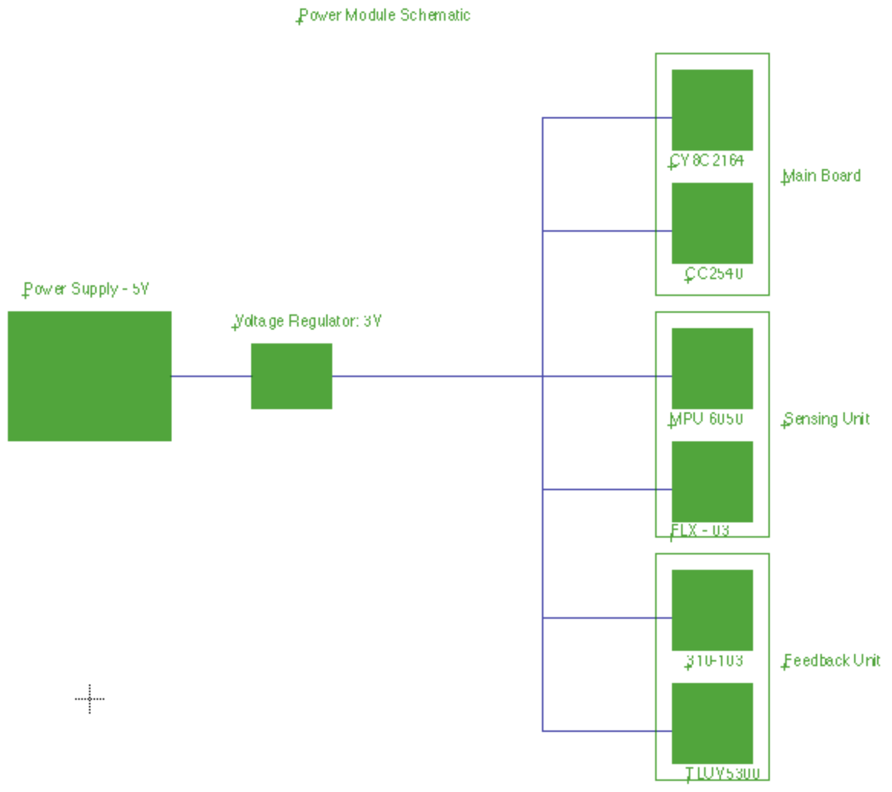

Figure 7: Power Supply Schematic

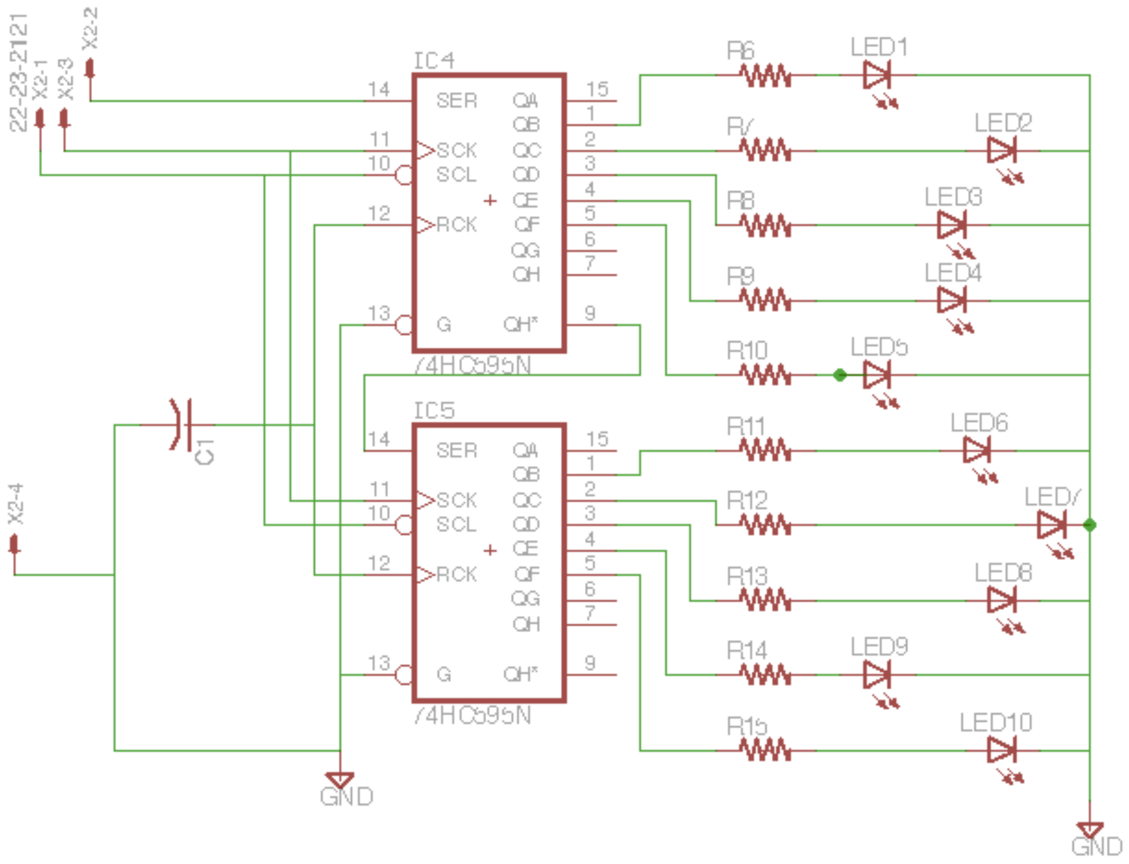

Figure 8: Feedback Unit Circuit

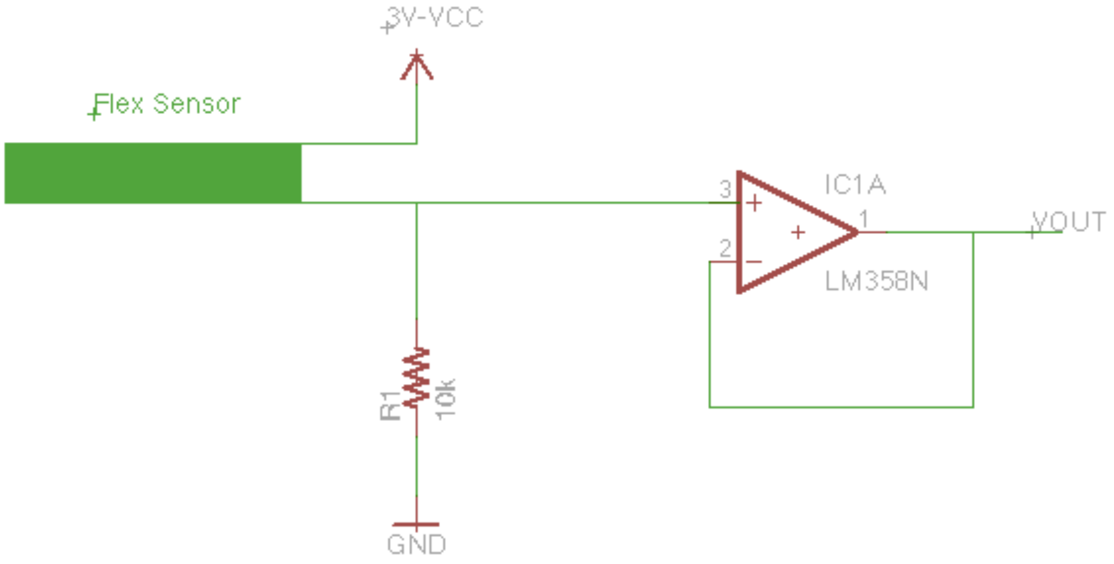

Figure 9: Flex Sensor Schematic[2]

Table 1: Resistance and Voltage Data for Flex Sensor

R_2 (Flex Sensor)	V_{out}
9.48 kΩ	1.78 V
15.3 kΩ	1.69 V
17.0 kΩ	1.59 V
21.2kΩ	1.36 V
22.7 kΩ	1.32 V

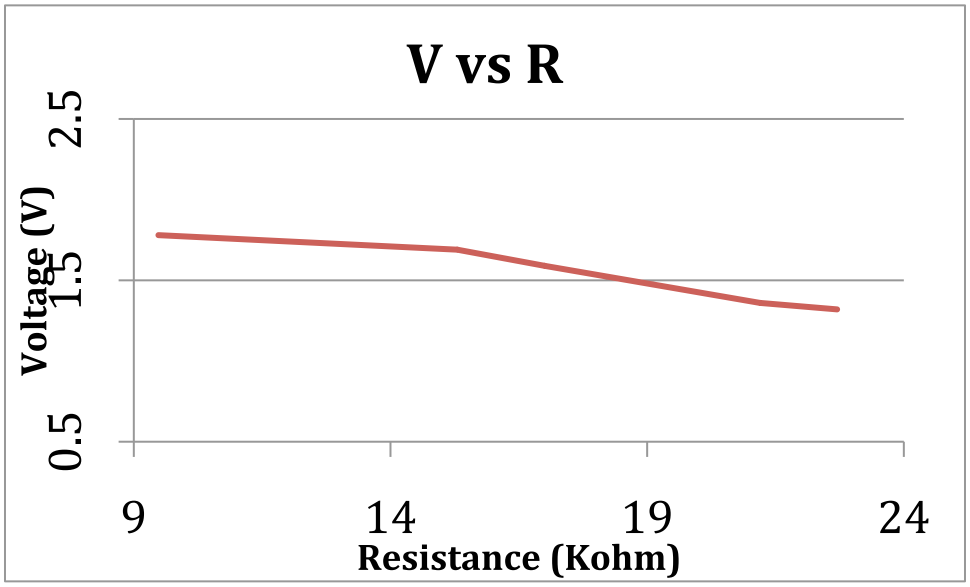

Figure 10: Voltage vs. Resistance for Flex Sensor

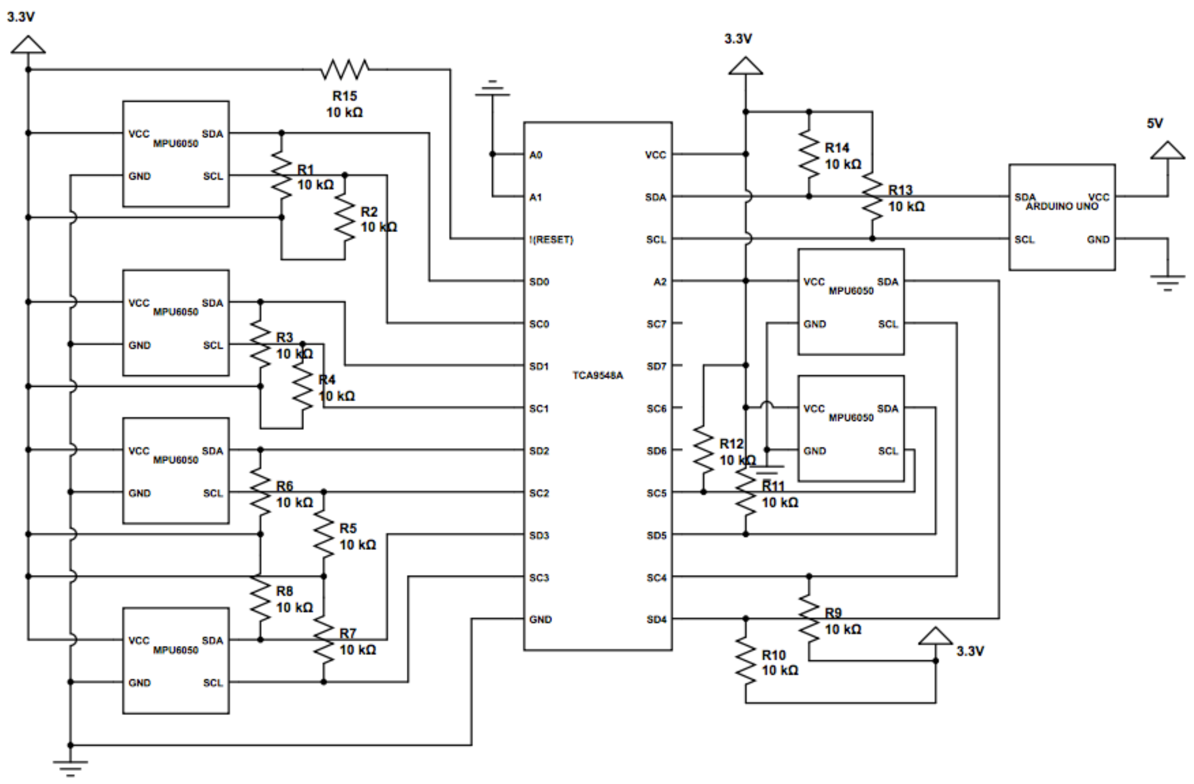

Figure 11: MPU 6050 Interface (Finger PCB's)

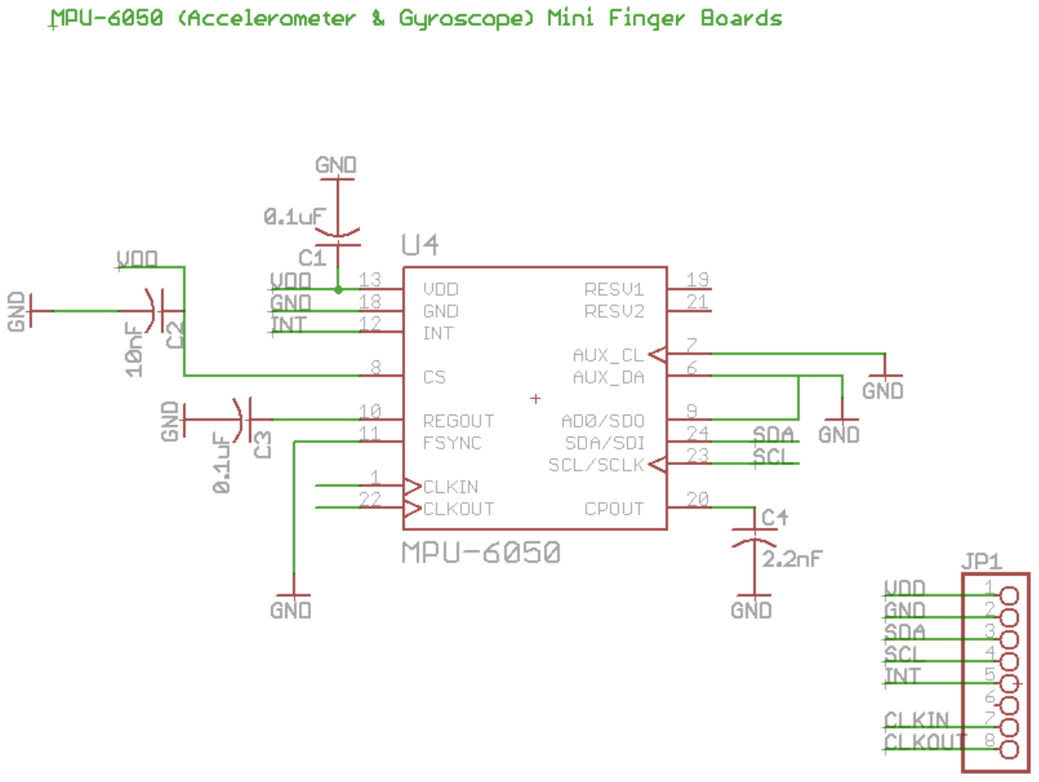

Figure 12: MPU 6050 Schematic [3]

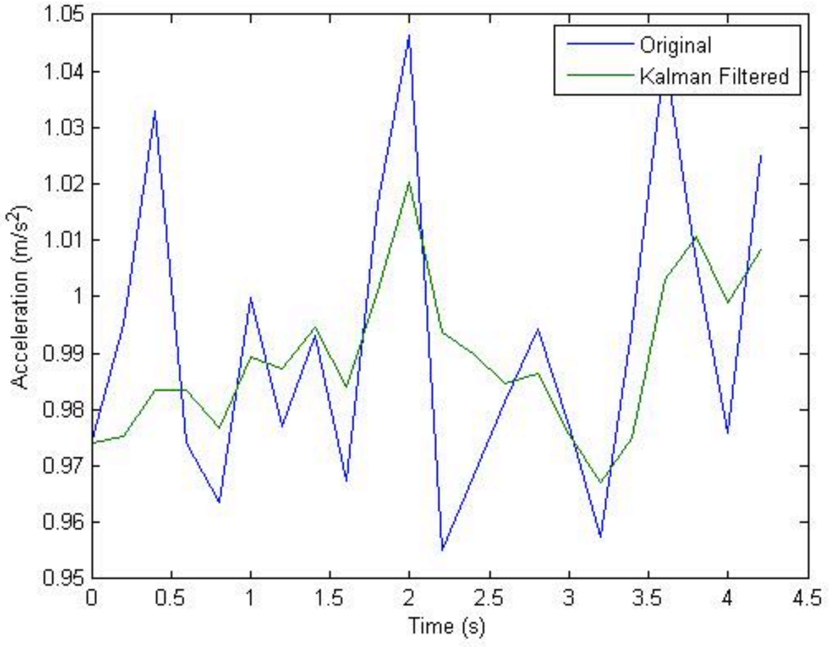

Figure 13: Karman Filter Acceleration

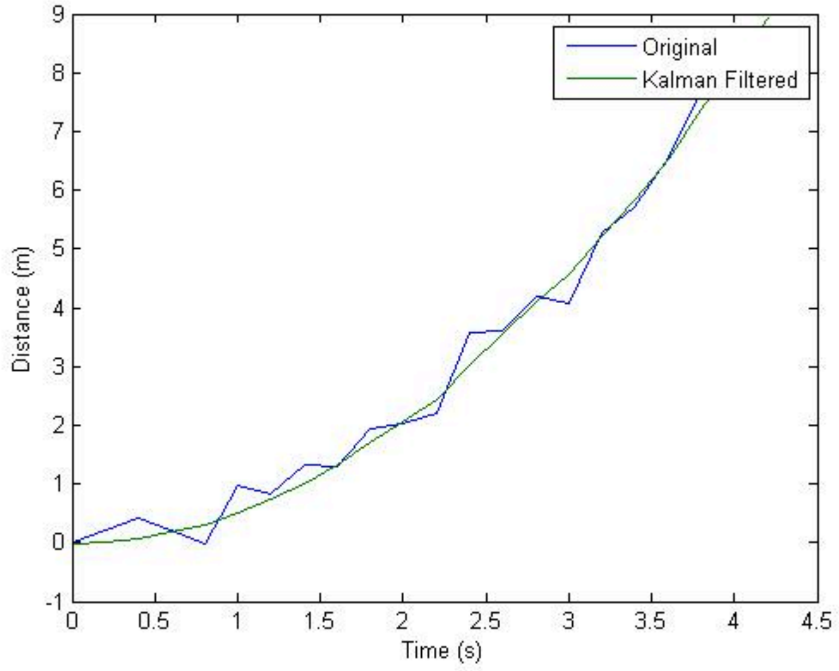

Figure 14: Karman Filter Distance

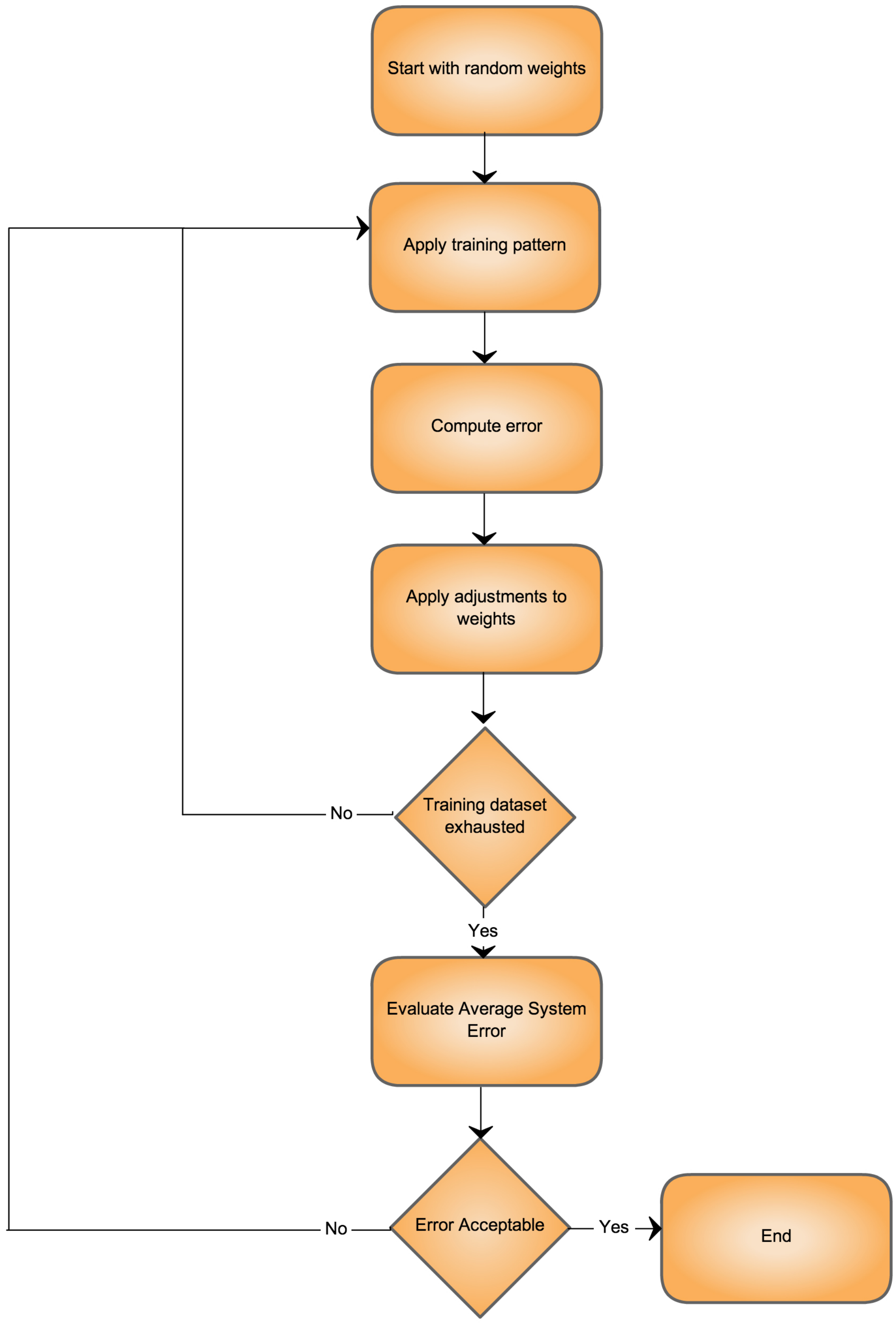
Start with random weights
Apply training pattern
Compute error
Apply adjustments to weights
Training dataset exhausted
No
Yes
Evaluate Average System Error
Error Acceptable
No
Yes
End

Figure 15: Perceptron Algorithm Flowchart

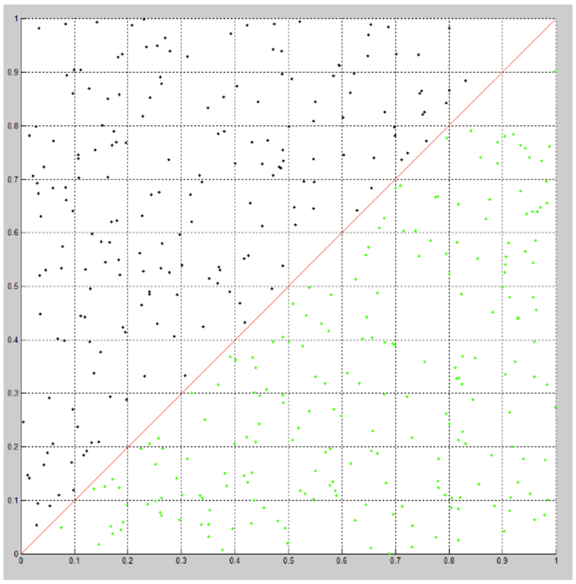
Figure 16: Simulation 1 (very small test data size) for Perceptron

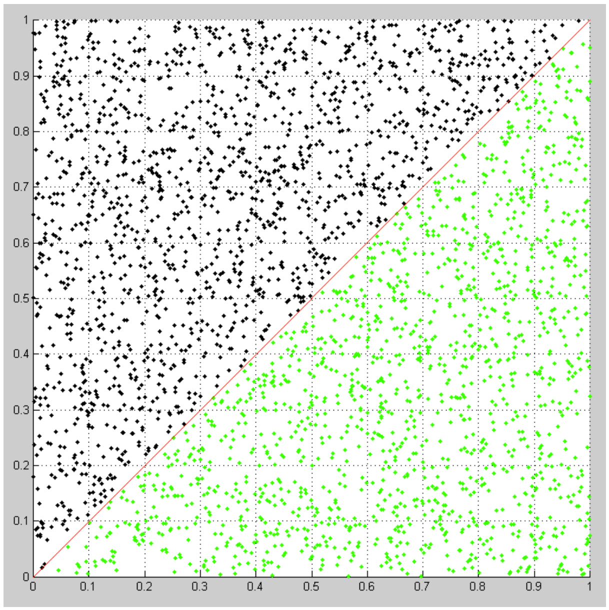
Figure 17: Simulation 2 (large test data size) for Perceptron

6.2. Appendix B – Requirements & Verification

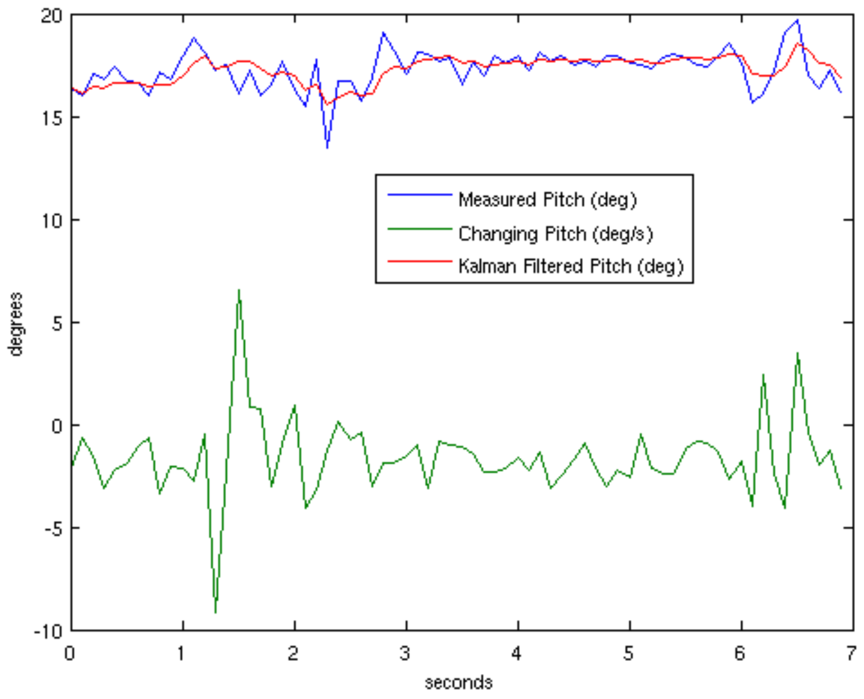

Figure 1: Test Data For Symbol ‘A’ from Sensor 1 (Little Finger)

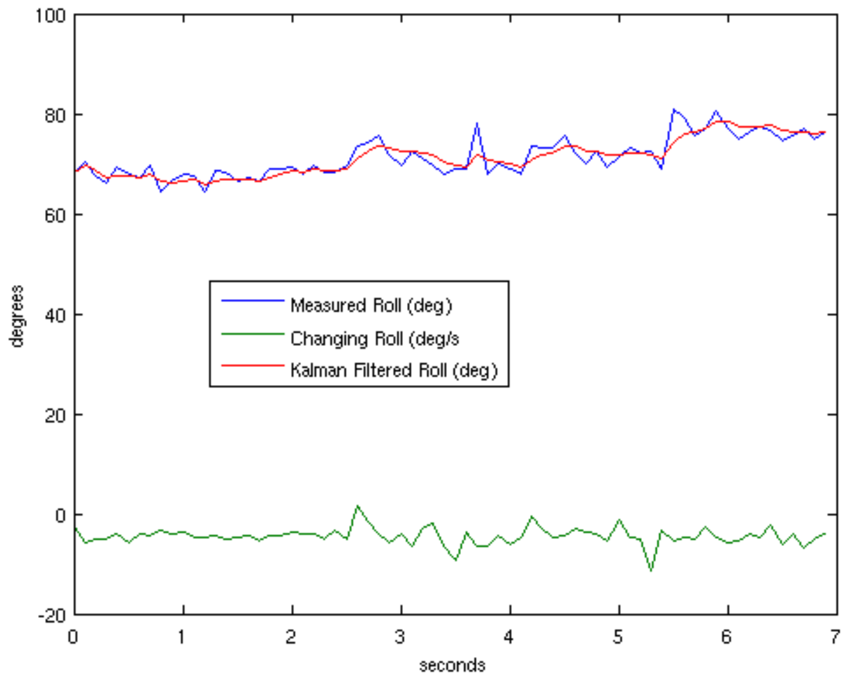

Figure 2: Train Data For Symbol ‘A’ from Sensor 5 (Thumb)

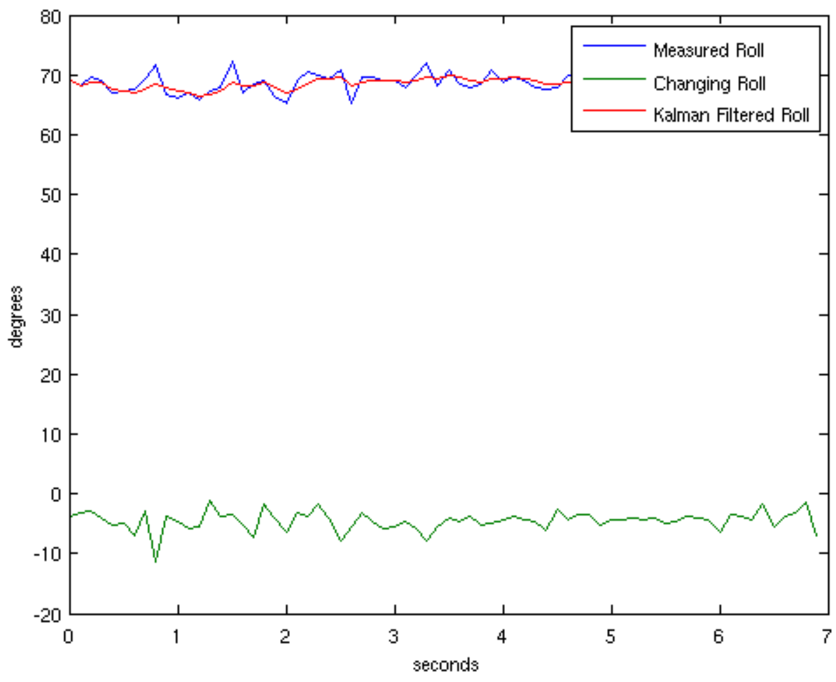

Figure 3: Test Data For Symbol 'A' from Sensor 5 (Thumb)

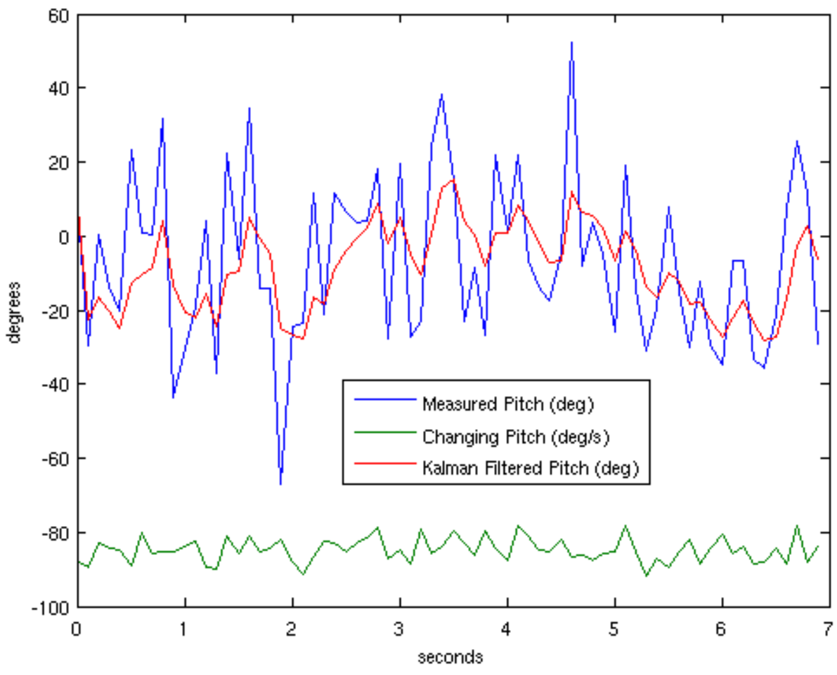

Figure 4: Test Data For Symbol 'A' from Sensor 3 (Middle Finger)

Performance Requirements And Testing Procedures

Sensing Unit

Performance Requirements	Verification	Status
Gyroscope & Accelerometer (MPU 6050)		
1. Accelerometer and Gyroscope data should be consistent.	1. The following steps will be taken to verify the consistency: a) Place MPU 6050 on a flat surface, the measurements should be 0°/s (+/- 2°/s) for gyroscope data. b) Tilt the surface and record the data from the gyroscopes and accelerometers. c) Calculate 'θ' from accelerometer data from its x-axis and z-axis. $\theta = \arctan(-\frac{z}{x})$ d) Calculate 'γ' from gyroscopes z-axis data which gives 'ω' – $\omega = \gamma \times t$ e) The value of θ and γ should be close.	Verified
Flex Sensors		
1. Check resistance range from 10kΩ - 40kΩ (+/- 1kΩ) a) Output 10k ohms when unflexed (when fingers are straight) b) Output 40kΩ when completely flexed (hand is curled to make a fist)	1. Probe flex sensors with digital multimeter to check for resistance a) When the sensor is unflexed the output resistance can be checked with DMM which should display a resistance of 10kΩ (+/- 1kΩ) b) When the sensor is completely flexed the output resistance can be checked with DMM which should display a maximum resistance of 40kΩ (+/- 1kΩ)	Failed Exposure to heat

Main Board

Performance Requirements	Verification	Status
Power Supply		
1. The power supply must be able to provide at least 5V (+/-0.5V) 2. Voltage regulators to flex sensors, accelerometers & gyroscopes, Bluetooth module, microcontroller, LED's and vibration motors should be able to output 3V (+/- 0.2V)	1. The power supply will be connected to a 2Ω load. Using a multimeter we should measure the voltage to be 5V (+/-0.5V). The same procedure can be performed with a bigger load. 2. Black multimeter lead is plugged into the "COM" socket of each regulator and red lead is plugged to the "V" socket. The multimeter should measure a voltage of 3V (+/- 0.2V)	Verified
Microcontroller		
1. Successfully receive valid data from sensing unit a) Create sample data mimicking data from gyroscopes & accelerometers, that is, 500°/s for gyros and +/-2g for accelerometers. b) Successfully receive sample analog signal mimicking flex sensors in the range of 0-2.5V, with a voltage above 1.25V considered a high. 2. Communicate with Bluetooth module by generating a 2.4GHz signal 3. Successfully filter noise from environment using Kalman Filter 4. Successfully implement Perceptron Algorithm that will give us 98% accuracy to match data.	1. Test the signals from two sources and compare to test the data a) Compare the data received by microcontroller with the sample data used for testing. b) Check the values with those received from microcontroller against the sample data 2. Code the microcontroller to send a signal of 2.4 GHz every second and graph the output with an oscilloscope. After the signal to frequency domain the frequency can be verified. 3. Check the results of the Kalman filtering in the microcontroller to the results of our Matlab simulation for Kalman Filter 4. After applying the training program to the algorithm, we	Verified Perceptron accuracy only 75%

5. Alert feedback unit with correct results	will check to see if it is capable of matching data correctly to the accuracy specified. 5. Check that the microcontroller is capable of sending a feedback signal to 6 sample loads.	
Bluetooth		
1. Check if it is transmitting the correct data	1. Compare the data sent by PSoC microcontroller with the data received by computer, if it's a match than the Bluetooth module is transmitting the correct data.	Failed /uncertain

Feedback Unit

Performance Requirements	Verification	Status
LEDs		
1. Operating voltage of a maximum 3V Typical operation for 2.4V	1. Connect LED into a breadboard and input a voltage of 3V. If the LED is on then it is working. Also the LED should be off for very low voltage inputs.	Verified (not in real time)

6.3. Appendix C – Cost Analysis

Table 1: Labor Costs

Name	Hourly Rate	Time Invested (Hours)	Total = Hourly Rate x 2.5 x Time invested
Reebbhaa Mehta	$35.00	180	$15,750
Daniel Fong	$35.00	180	$15,750
Mayapati Tiwari	$35.00	180	$15,750
Total		540	**$47,250**

Table 2: Cost of Parts

Item	Part Number	Quantity	Company	Cost ($)
Gloves	-	1	Pearl iZUmi	30.00
Flex Sensors	FLX – 03	5	Images Scientific Instruments	70.00
Accelerometers & Gyroscopes	MPU 6050	6	Invensense	105.00
LEDs	TLUV5300	5	Mouser Electronics	1.00
Microcontroller	Arduino Uno	1	Arduino Uno	100.00
Bluetooth	Bluetooth Shield	1	Seeed Studio	6.00
Battery	9V	1	Energizer	2.00
Resistors, Capacitors, Inductors	-		University	5.00
Voltage Regulator	LP2951ACP	8	Jameco Electronics	5.00
PCB	-	6	University	50.00
MUX	TCA9548A	1	Texas Instruments	10.00
Op-amps	LM 358	2	National Instruments	2.00
Total				**385.00**

Table 2: Total Cost

Section	Total
Labor	$47,250.00
Parts	$385.00
Grand Total	**$47,635.00**

7.0. Citations

[1] Texas Instruments, "I2C Multiplexer," [Online Document], 2012, [cited 2 April, 2013]
Available HTTP: http://www.ti.com/lit/ds/symlink/tca9548a.pdf

[2] Sparkfun, "Flex Sensor," [Online Document], [cited 5 February, 2013]
Available HTTP:
https://www.sparkfun.com/datasheets/Sensors/Flex/FLEXSENSOR(REVA1).pdf

[3] Invensense, "Accelerometer & Gyroscope," [Online Document], 2012, [cited 11 February, 2013],
Available HTTP: http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A.pdf

[4] G. Welch, G. Bishop, "Lecture – An Introduction To Kalman Filter," 2001
Available HTTP:
http://www.cs.unc.edu/~tracker/media/pdf/SIGGRAPH2001_CoursePack_08.pdf

[5] J. W. Vaughan, "Lecture – Machine Learning Theory," October 19, 2011
Available HTTP: http://www.cs.ucla.edu/~jenn/courses/F11/lecture8.pdf

[6] Section 7.8 IEEE Code of Ethics, IEEE. "IEEE Code Of Ethics," IEEE. Web. 25 Feb, 2013
Available HTTP: http://www.ieeeorg/about/corporate/governance/p7-8.html