IR Signals for Finger Movement Detection

Sahil Gupta (skg73)   Mashrur Mohiuddin (mm889)

December 19, 2015

1 Introduction

We created an armband that used infrared (IR) diodes to capture finger movements from the the extensor digitorum communis muscle. From here, we used a machine-learning model to classify these signals, in order to create a real-time predictor. The idea was to shine infrared light on the user's upper forearm. As the user moved different fingers, muscles on the upper forearm would move, causing the IR light to diffract in differing directions. This reflection was detected using photodiodes, digitized through an ADC, and then processed via machine learning in order to predict exactly which finger the user was moving. Using two channels, we were able to achieve almost perfect prediction accuracy when distinguishing between three fingers in real time. While a third channel was built, due to time constraints we were not able to acquire three channels of data. Given the time, we believe we would be able to accurately predict four different finger movements, and possibly the thumb (the thumb does not really seem to cause much muscle movement in the upper forearm).

2 Hardware

The circuit used was adopted from a similar project by Raghav Subramaniam who compared the use of IR sensors and electromyography to detect muscle contraction.

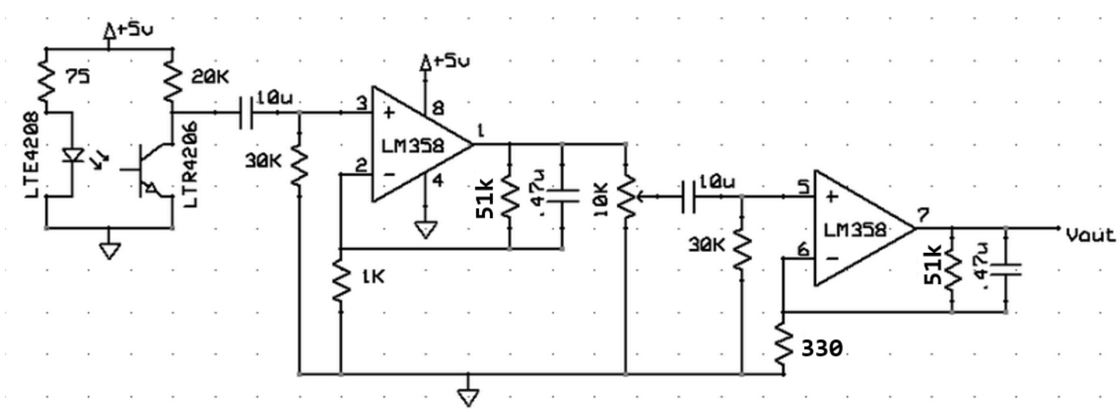


Figure 1: Schematic of IR circuit.

Figure 1 shows a schematic of the circuit used. The circuit was powered by the 5V and GND rails of an Atmega1284p microcontroller. An infrared light was biased with a 75Ω resistor and shined on the user's arm. The reflection of the infrared light was then picked up by a LTR4206 photodiode. The output of this was high pass filtered with a cutoff frequency of:

$$f_c = \frac{1}{2\pi(30k\Omega)(10\mu F)} = 0.53Hz$$

This was to remove any DC offset present. The signal was then sent through an LM358 for amplification with a gain of:

$$G = 1 + \frac{51k\Omega}{1k\Omega} = 51$$

To reduce noise in the system, the signal was low passed filtered with a cutoff frequency of:

$$f_c = \frac{1}{2\pi(51k\Omega)(0.47\mu F)} = 6.6Hz$$

The output was fed through a 10kΩ trimpot before the second amplifier stage. This served as a voltage divider so that only a fraction of the first stage would receive further amplification, providing a variable gain control. Because discrete amplifiers introduce DC offset, our signal was again high pass filtered at 0.53Hz and then amplified further. The second stage had a gain of:

$$G = 1 + \frac{51k\Omega}{330\Omega} = 155$$

The output of this circuitry was then fed into the ADC ports of an Atmega1284p to be digitized and processed through software.

Three channels were built and soldered as shown in Figure 2. A picture of the armband is also shown in Figure 3. The wires from the IR light and photosensor were twisted in order to reduce noise artifacts from moving wires. Foam was placed in between each sensor and light pair in order to isolate each sensor so that it only picked up IR light reflecting off the user's arm.

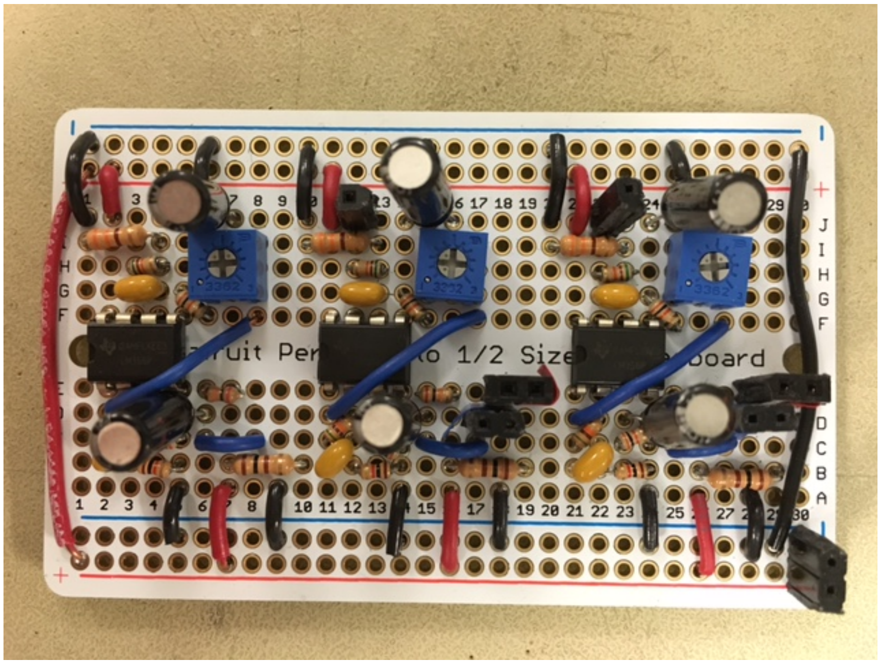
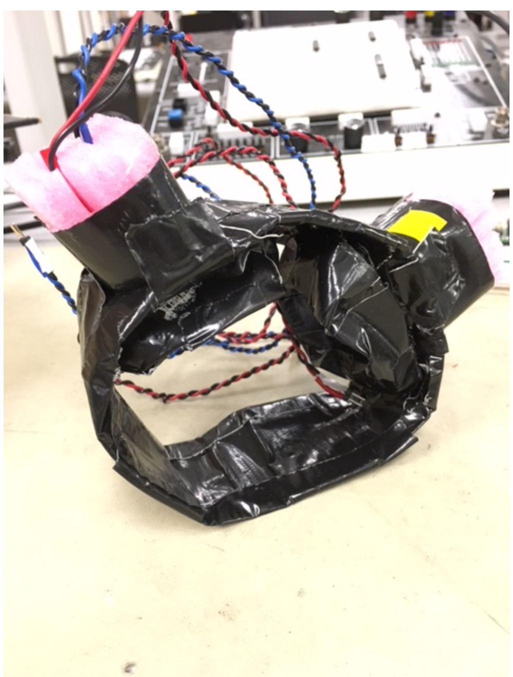
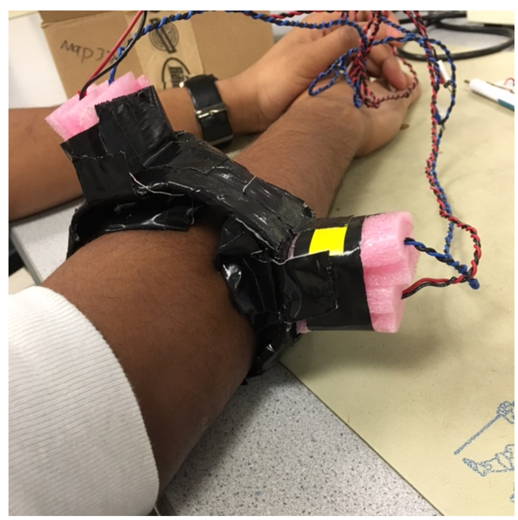


Figure 2: Picture of IR circuit.



(a) Prototype IR band.



(b) User wearing the band.

Figure 3: Pictures of the IR armband.

3 Software

There were three components to our software: data acquisition, machine learning, and real-time prediction.

3.1 Data Acquisition

To acquire the data from the signals, we fed the signals from the IR circuits into the ADC of an Atmega1284p microcontroller. This digitized our voltages within an 8-bit range (0-255). From here, we used a serial interface to send this data to a computer in real-time.

3.2 Machine Learning

In order to determine which signals corresponded to what, we needed to write some kind of program that was able to gather data from the microcontroller, format it appropriately, and feed it into a machine learning algorithm. We chose to use Python to accomplish all of these tasks. Python is a high-level language and has many modules readily available for various tasks. Specifically, we made use of the open-source scikit-learn machine learning module to handle our learning.

There were three modes to the program: `see`, `learn`, and `predict`. In `see` mode, the raw data from the microcontroller is simply shown on the screen. The data was acquired through serial communication with the Atmega1284p. `see` mode was mainly used for debugging purposes, so that we could know whether or not we were getting meaningful data or if the ADC was working properly.

In `learn` mode, the program would gather finger movement data from the user. For a given set of fingers (index, middle, ring, pinky, thumb, or any combination of the five), the program would ask the user to move the appropriate finger within a given time window and record data from each channel. The program would also keep track of which finger was being moved, and would store it along with the ADC data. This process was repeated for a specified number of readings for each finger.

Once we had the raw data, we began to prepare it for input into the machine learning algorithm. Machine learning is most successful when it is able to see clear patterns/trends in a data set. To improve the accuracy of the algorithm, we used several tactics. First, we removed noise from the readings by zeroing all data points below a noise threshold. Second, we shifted back all of the readings to the same time point. This way, all of the pulses brought out by finger motions would be aligned, making it easier for the machine to interpret the data. Finally, we normalized all data to be zero-mean and have a variance of one. As different fingers induced stronger signals in different channels, normalizing made sure that each channel was still weighted fairly.

After the data was formatted, we were finally able to input it into the machine-learning program. Using scikit-learn, we created a support-vector classifier (SVC) within support-vector-machine (SVM). Because we knew a priori which reading was mapped to which finger, our work was a supervised learning task. SVMs work very well for creating models for classification problems.

To verify the accuracy of our program, we used a technique called cross-validation. Cross-validation takes a percentage of the input data as the training data, and uses the remainder of the data as test data. The program learns the training data, and uses the model it builds to predict the test data. As the selection is random, different seeds can be chosen to ensure consistent results.

3.3 Prediction

The final component of our software, and mode of our program, was real-time prediction. Ultimately, our project would be used in some kind of control system, so we needed to create a real-time predictor. The predictor would use the SVM created *after* the user had already gone through the learning mode of the program once.

Our real-time predictor would monitor the inputs coming in from the microcontroller. If the sample was above a given threshold, it would classify it as a finger movement. In this case, it would start recording the data, process it in the same way the `learn` mode does, and input it into the predictor model. The predictor model would then tell us its prediction.

4 Front-End: Armband

The original design for the armband simply consisted of a band made of duct tape and foam insulation to hold the sensors. While the main goal of this project was to detect different finger movements and explore the use of machine learning, we also wanted to develop a user-friendly and aesthetically pleasing product as this project continues. The initial design was not adjustable, and so was not practical for a variety of users with different sized arms. A second iteration was designed on Autodesk Inventor and 3D printed. As shown in Figure 4, the new design takes a more flexible bracelet style structure. Each casing for the sensor and IR light pairs snaps onto the bars of the bracelet, allowing for a modular design if future iterations require different sensor types or upgrades.

The flexibility of the armband can be assessed by modeling it as series connected springs (with each spring being in between two bars). For a force acting along a spring's height, the spring constant (and thus stiffness) is given by the equation:

$$k = \frac{Ewh^3}{4L^3}$$

By reducing the height and width of the rings on either side of the band, it could be designed for increased flexibility if needed. Reducing the number of bars connected to each ring would also make each series connected spring longer, and thus less stiff to forces bending it along its height.

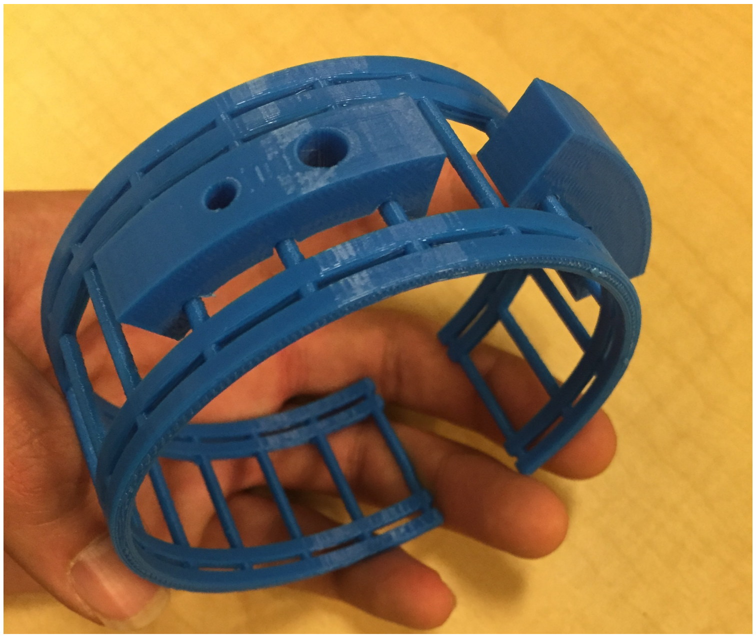


Figure 4: 3D Printed armband.

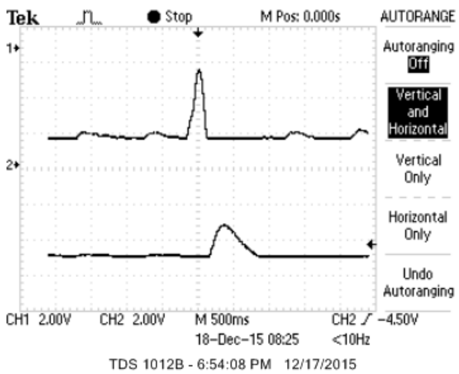
5 Results

As seen in Figure 5, strong EMG signals were captured for movement of the index, middle, ring and pinky fingers. The frequency of these signals seemed to range from about 2-4Hz. Even from the oscilloscope waveforms, each finger movement is noticeably different.

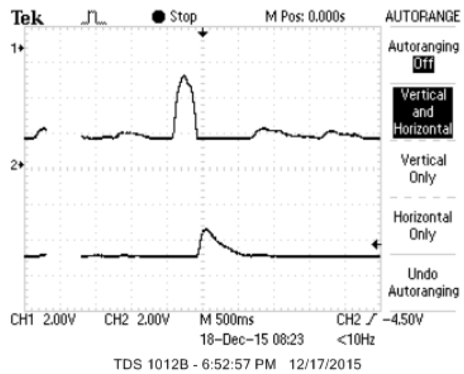
One-hundred readings were taken for the index, middle, and ring fingers. Using the default SVC kernal, our average prediction accuracy was 96.2%. This number was calculated by cross-validating our data over 100 different random seeds, and averaging the accuracy of each result. The max accuracy obtained was 100%, while the minimum accuracy observed was 80%. Table 1 shows a summary of these results, and other kernels for convenience. We tried 3 different kernels and compared the results. We also applied principal component analysis (PCA) to our data and repeated the learning comparisons. PCA attempts to take a data set and rotate/stretch it along its principal component axes, thus increasing the

	Statistic (%)		
Kernel	**Avg**	**Max**	**Min**
Default	96.2	100	80
Linear	94.8	100	83.3
RBF	43	60	30
Default w/ PCA	67.5	90	50
Linear w/ PCA	57.7	76.6	33.3
RBF w/ PCA	66.6	86.7	46.7

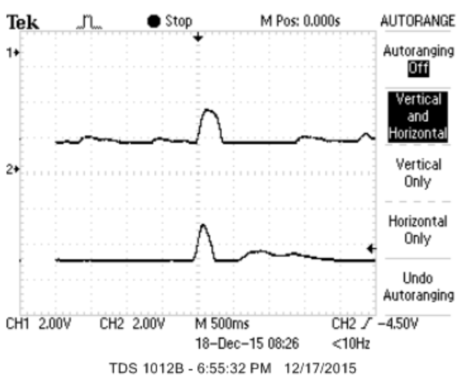
Table 1: Machine learning prediction accuracy.



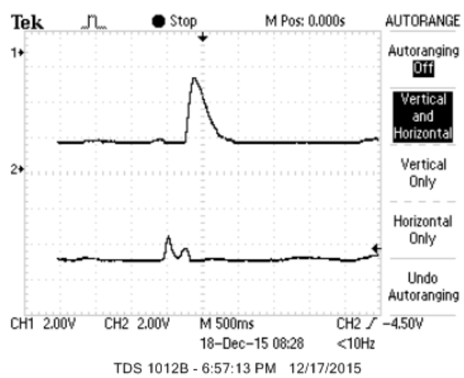
(a) Index.



(b) Middle

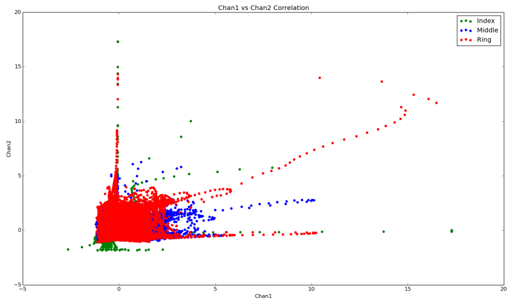


(c) Ring.

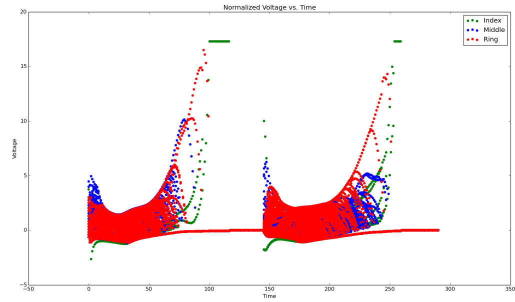


(d) Pinky.

Figure 5: EMG signals acquired from the circuit. Channel 1 is for middle (ED3), channel 2 is for ring (ED4), and channel 3 is for index (ED2).



(a) Channel correlation plot.



(b) Normalized voltage v. time of all the readings. Note: the first half of the plot (t=0 to 144) represents channel 1, while the second half (t=145 to 289) represents the second channel. In reality they are overlapping but we place them side by side here to visualize both.

Figure 6: Visualization of the machine learning data.

variance of the data set. However, as seen in Table 1, it did not seem to help our case. We include it in this report for completeness.

Figure 6 shows us a visualization of the data. Figure6a shows the channel correlation while Figure 6b shows the voltage v. time plots. From both these plots, we can see that there is much overlap between each of the fingers, but there are still distinct properties between them (ex. middle finger voltage on channel 2 takes longer to decay than the ring finger). Because there is so much overlap between the different data sets, we need a lot of data in order to be able to differentiate between them; otherwise the accuracy suffers.

Unfortunately, our real-time prediction mechanism did not yield results as accurate as the cross-validation did. The real-time system could predict middle and ring fingers fairly well, but struggled with the index finger. We believe that by rewriting our real-time predictor, gathering more training data, and utilizing a third IR channel would help improve our real-time performance.

6 Conclusions

6.1 Discussion

We successfully built a 2-channel circuit to detect the IR signals from different finger movements. Each finger tested showed distinct voltage waveforms upon moving. Through machine learning, we were able to accurately map three finger movements with 96% accuracy.

6.2 Improvements

In order to improve the accuracy for multiple targets, we must acquire more data. This would allow for greater training of the machine learning software and thus a higher prediction accuracy rate for greater than three targets. Additionally, we could utilize the third channel, which would give us another dimension of data to improve our mappings.

6.3 Future Directions

This project was initially motivated by the idea of creating prosthesis with individual finger control; however, we would like to change the scope of this project slightly. For the spring semester, we would like to focus less on finger detection, and more on building an actual control system that utilizes the types of signals we have successfully obtained so far. We would first like to fully integrate our third channel into our machine learning and potentially add an accelerometer and gyroscope to provide us information about how the user moves his/her arm as well as hands. This will provide us with more degrees of freedom to utilize for a control system.

References

[1] J.N.A.L. Leijnse, N. H. Campbell-Kyureghyan, D. Spektor, and P. M. Quesada *Assessment of Individual Finger Muscle Activity in the Extensor Digitorum Communis by Surface EMG*. J Neurophysiol 100:32253235, 2008.

[2] Qiang Li, Bo Li *Online Finger Gesture Recognition Using Surface Electromyography Signals*. Journal of Signal and Information Processing, 2013, 4, 101-105

[3] J. Rafiee, M.A. Rafiee, F. Yavari, M.P. Schoen *Feature extraction of forearm EMG signals for prosthetics*. Expert Systems with Application, 2011, 38, 40584067

[4] Sebastian Maier and Patrick van der Smagt *Surface EMG suffices to classify the motion of each finger independently*. Conference: 9th International Conference on Motion and Vibration Control

[5] Muhammad Zahak Jamal *Signal Acquisition Using Surface EMG and Circuit Design Considerations for Robotic Prosthesis*. book edited by Ganesh R. Naik, ISBN 978-953-51-0805-4, Published: October 17, 2012 under CC BY 3.0 license.

[6] O. Fukuda *A Human-Assisting Manipulator Teleoperated by EMG Signals and Arm Motions*. Robotics and Automation, IEEE Transactions on , vol.19, no.2, pp.210-222, Apr 2003

[7] *The Rehabilitation Process: Prosthetic Devices*. Ottobock. Retrieved from: https://ottobockus.com/zb2b4ob/us01/en/USD/Upper-Limb-Amputees

[8] *Targeted Muscle Reinnervation*. Advanced Arm Dynamics. Retrieved from: http://armdynamics.com/pages/tmr