EMG Computer Interface

Aidas Jakubenas, CSE, Ryan Dewsnap, CSE, Samuel Worrell, EE, and Berke Belge, CSE

Society has been moving away from physical interfacing for some time now. From eye tracking to voice control, we are always finding better and more intuitive ways to accomplish everyday tasks on computer devices. Compounded by the recent pandemic, touchless interactions have become somewhat coveted not only for health reasons, but also ease of use. Our design aims to incorporate electromyography (EMG) sensing technology to detect arm/hand gestures to carry out common tasks such as purchasing a train ticket, facilitating a class, or navigating a presentation. We see our device being applicable to large companies or campuses that could benefit from faster, more efficient, hands-free interactions. Similarly, the device could be used by speakers and teachers to aid in presenting or utilized in a hospital where custom gestures for those with disabilities could be used to call for a nurse or other important functions.

I. INTRODUCTION

A. Significance

In 2020, the Covid-19 pandemic took the world by storm and initiated major shifts in the way society as whole functions. With this shift came a need for novel methods to minimize the transmission of this virus, and in turn the problem that our project aims to solve is being able to minimize touchless interactions among common surfaces, specifically public computers. In fact, a Princeton Review Study notes that there is "*a high probability of observing pandemics similar to COVID-19 (probability of experiencing it in one's lifetime currently about 38%), which may double in coming decades"* [1]. This suggests that touchless interfaces may soon become the norm rather than the exception in our society. Not only this, there are far more advanced applications that could be explored in the future.

B. Context and Competing Solutions in Marketplace

Similar solutions for minimizing touch interactions with computers exist in the market, however they do not cover the breadth of features that exist in our design. There are products that do allow for touchless interactions, but they either do not allow for gesture customizability, or do not allow for accurate readings based on user review. Many also do not have an ergonomic design as with our project. Our design aims to be a wireless, ergonomic sleeve that allows for a full range of interaction with a public computer. For example, the Air Keyboard is a novel undergraduate thesis project that utilizes EMG sensing along with predictive text to use finger movements as a keyboard [2]. The downside of this product is that they never released a commercial or viable product, and their device exclusively emulates a keyboard. Similarly, the Tap Strap 2 focuses on keyboard interaction however it is a commercially released product [3]. Its downside is that the device, according to user reviews, is hard to handle (i.e. do something else while wearing the device such as using a phone) and even harder to learn which gestures do what. The final product we examined was the Real Time EMG-Based Assistive Computer Interface for Upper Limb Disabled that utilized non-arm gestures to provide assistance for those with upper limb disabilities [4]. However, this product lacks customizability for those with other disabilities and is hardwired for a specific set. The pros and cons of all three devices compared to our device can be best summarized by Table 1.

	Compatible with any computer	Non-Intrusive	Customizable profiles	Movements + Voice
Real Time EMG Interface				
Tap Strap 2				
Air Keyboard				
Our Solution				

Table 1: Competing Solutions Analysis Matrix. Movements + Voice refers to the concept that we intend to use built in speech-to-text functions that are inherent in all Apple and Windows systems to avoid having to configure all 26 letters of the alphabet in our device.

C. Societal Impacts

The main constituents of our project will be organizations that aim to minimize the spread of Covid-19, or any virus in the case of another pandemic arising. Our product is specifically designed to be utilized within an organization, such as a college campus or private enterprise. Thus, these entities will be able to take an extra precautionary measure to prevent the spread of viral material among individuals.

D. System Requirements and Specifications

Any system that is designed will need to have system

requirements and specifications to align with the goals of the engineers. These allow for tangible targets to reach for out of our envisioned EMG computer interface. The following system specifications and requirements were determined.

S1. *The system shall sense gesture movements with a reasonable accuracy*. This ultimately means that the design of our system must be able to categorize 5 distinct gestures with the EMG sensors used. And we must be able to do this in a reasonable accuracy which was determined to be a net 90% true positive accuracy.

S2. *The system must be reliable*. This specification requires the EMG sensing interface to be reliable from person-to-person, and for daily shifts in placement on the forearm. We need to ensure that a user can put the sleeve on to a relative area on the arm and be able to use it properly.

S3. The system shall be ready to use in a reasonable amount of time. This specification requires the system to be powered-on and in a usable condition in less than a minute.

S4. *The system shall have reasonable power consumption*. We have designed this specification around our products battery life. As this device is not constantly used, a minimum of 3 hours of active operation time on one battery charge is expected.

S5. *The system shall be ergonomic*. Overall, our device will need to be ergonomic enough for daily use without much hackle and stress. It should be easy to put on and adjust to one's forearm. This should take only a maximum of 1 minute to adjust.

S6. *The system must be usable within a reasonable distance*. As the system will have a plug-in version and a Bluetooth version the device must be usable as a computer interface up to 3 meters away from host computer.

S7. *The system must have some customizability*. The user must be able to pair 5 distinct muscle movements with 5 different gestures using the GUI interface developed as a part of our product system. We arrived upon 5 distinct muscle movements by looking at the anatomy of the muscles on the forearm and seeing that with how muscles overlap and influence each other, we are limited by the human biology rather than the accuracy of our sensors [5].

Requirement	Specification	Value
portable	weight	< 2.5 lb
	size	< 4 cuft
	battery powered	1.5hr between charge
responsive	latency	< 50 msec
safety	Auto shut down	within 1 sec after fault
-	-	-
-	-	-

Table 2: Requirements and Specifications

II. Design

A. Overview

The design primarily utilizes a microcontroller with four EMG sensors to capture the voltage difference across the arm as time series data which is sent to a host computer where the time series data is compared against an ML model that classifies the data according. If the gesture is bound to a keystroke, that keystroke is executed on the host computer. As see in Figure 1, the software is evenly divided between our communication in C on the ATmega328p [6] and Python on the host computer.

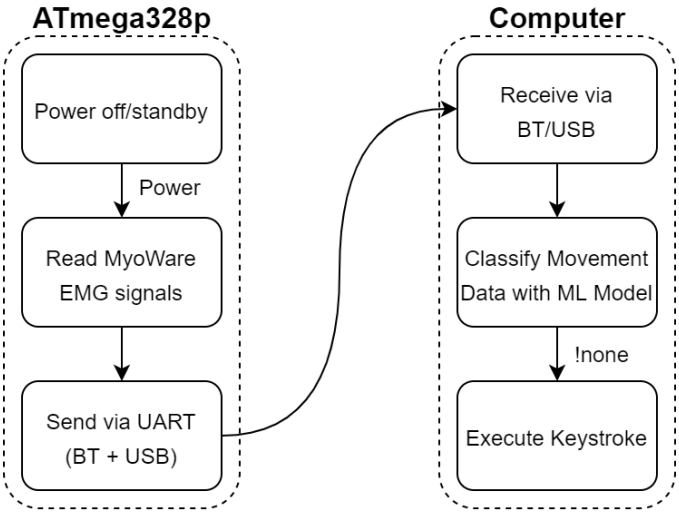

Figure 1: Software Block Diagram

Initially we planned to use a custom EMG sensing circuit rather than the MyoWare, however as discussed in Appendix A, the custom circuit did not aid our accuracy and performed far worse than the MyoWare sensor. We also attempted to initially set simple threshold values in the software to process signals, however the variability of the skin to electrode contact made this unreliable. We then pivoted to using machine learning (ML) on the host computer to obtain our desired 90% accuracy. Our HC05 Bluetooth Module allows us to adhere to IEEE 802.15.1 standard Bluetooth protocol. The device is also FCC compliant with under 1.6 W/kg of RF exposure (watts over kilograms of body tissue).

Finally, for our wired communications our device is compliant with IEEE RS-232 standard format.

B. The Hardware

The majority of our hardware is invested in our PCB. As see in Figure 2, the ATMega328p [11] does the brunt of our data collection and communication.

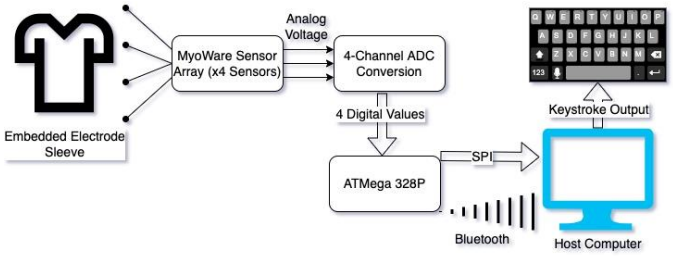

Figure 2: Hardware Block Diagram

The MyoWare 1.0 [7] and MyoWare 2.0 [8] sensors are housed on the outside of the PCB enclosure and multicore wires with snap leads are connecting the electrodes in the sleeve to the sensors. Our FT232R USB to serial UART interface [9] is used to send the time series data through a wired connection. The HC05 Bluetooth module [10] is used to send the data wirelessly and the entire system is powered and charged by our LiPo micro-USB shield [11]. In Figure 3, the schematic for each breakout is defined outside the PCB.

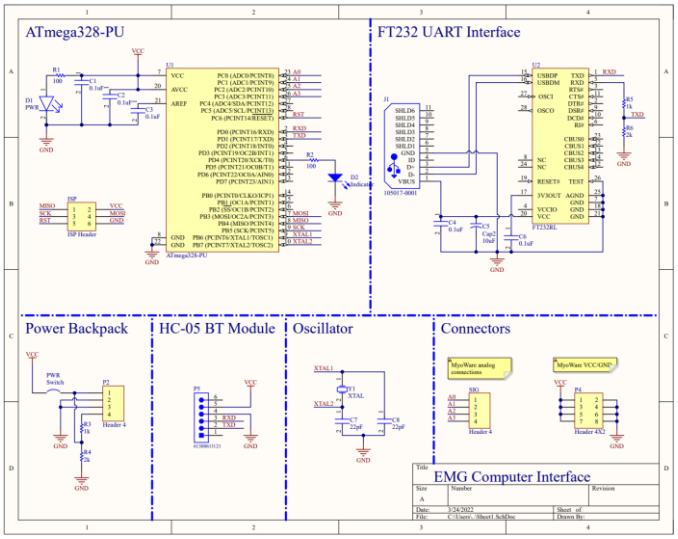

Figure 3: PCB layout along with breakout components.

C. Embedded Software

The embedded software for this project was originally going to involve classifying gestures and performing all the logic. However, our team decided it would be more efficient for specifications such as battery life to just perform simple tasks on the embedded hardware.

The embedded software was written in C and revolves around some built-in functions in the ATMega328P. The software included functions for collecting data from the MyoWare sensors and running those analog inputs through the built in Analog-to-Digital Converter (ADC) which then leave us with digital datapoints that we can use more appropriately.

Next the software included UART serial communication code for both the USB and Bluetooth modules. This part of the software determined the size of the buffer we would send across UART and accurately displayed sensor data in readable form for our host computer. This software then sent the data across UART communications to the host computer which has a separate host software running which decodes and processes the data from the hardware.

D. Host Software

The host software is defined as the software that will held and used by the interfacing computer. In this case, when the EMG Interface device is paired with a computer to interact with. The computer will hold the host software which will allow the device to work properly. Because our scope is generalized to a network dedicated to a certain company, institution, or campus, we assume firmware will be easily flashed on all machines and updated as an update to the network.

The host software will consist of one main GUI that performs as the main “hub” for the user to configure and use their device. The GUI allows users to do the following: maintain a live graph of EMG signal inputs, train the ML model (Section II, Part E), configure various gestures, and ultimately allow for easy profile swapping. An example of what the gesture configuration screen looks like is shown below in Figure 4.

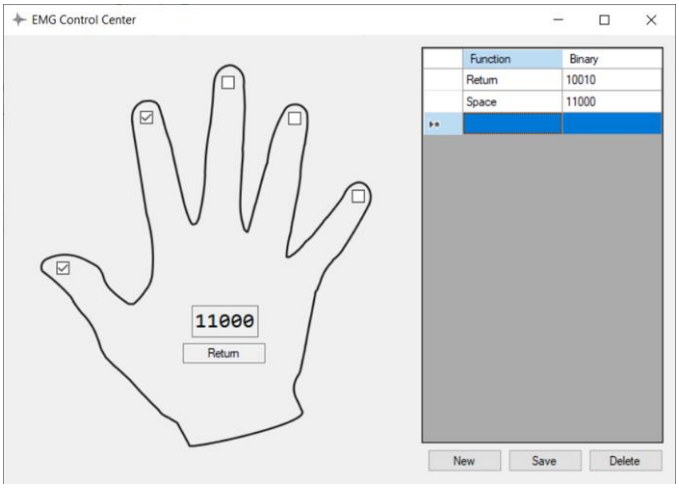

Figure 4: Gesture configuration GUI example. Each finger can be custom mapped to a keystroke. We plan on adding more gestures beyond simple finger selections in the future.

The host software is all written in Python utilizing multiple libraries to help ease implementation. The GUI is written in PySide6 which is a GUI/User interface library to help create an interactive python experience.

Once the user intends to interact with the computer, the host software begins by sampling UART serial data stream from our devices Bluetooth/wired connection. It reads inputs that reveal all MyoWare sensor data inputs in one string format. The Host software then runs the data through an analysis portion which will calculate the Root Mean Square (RMS) value for the last 20 samples it receives. It proceeds by running these RMS values as inputs to the trained prediction model. The model will then determine if the output is a gesture and activate a keyboard

shortcut to perform the paired gesture-to-action task.

E. Machine Learning Software

Machine learning is a hot topic in today's computing literature and projects. It is popular for a reason other than being technically interesting, but it is quite accurate at making decisions based on context.

In our case, the EMG Computer Interface utilizes a k-Nearest Neighbors (kNN) clustering algorithm to determine what gesture a user is performing with real-time data taken from our MyoWare sensors.

For context, we developed our kNN cluster utilizing Python sklearn library which was developed to simplify the mathematical workings of the models and let users utilize ML learning as a part of their project rather than learning how to re-invent something that has already been done. We then sample data serially in real time and use these data points as test points for our kNN clustering scheme.

As mentioned, our system uses RMS to classify distinct gestures, rather than using the raw data. It is important to understand how kNN works with our data to produce a working classifier. Our model takes RMS values that have been calculated from a pre-gathered dataset that has been classified. Each classification is a gesture and classifies that combination of RMS values from each of the sensors. We then give the kNN cluster this training dataset and it will determine cluster groups for each of the classifications. In this case if we have 5 gestures, our model will determine 5 cluster areas of data and define these general spaces as a gesture. Once we feed live RMS data from our sensors into the algorithm it will simply determine which cluster this data point belongs to. The more distinct and repeatable the RMS values are, the more accurate our model will be.

It is important to realize that training the algorithm is the most intense portion of this machine learning process. This is typically done off-site and takes from minutes to hours depending on the model size. However, once we have a trained model, we just simply determine which category the data falls into which is why our system is so fast. We are classifying gestures in real-time. As seen in Figure 5 below, when using a subsection of training data as test data, we achieve extremely high accuracy.

```
[[20  0  0  0]
 [ 0 27  0  0]
 [ 1  0 22  0]
 [ 0  2  0 80]]
              precision    recall  f1-score   support

   come_here       0.95      1.00      0.98        20
       flick       0.93      1.00      0.96        27
      middle       1.00      0.96      0.98        23
        none       1.00      0.98      0.99        82

    accuracy                           0.98       152
   macro avg       0.97      0.98      0.98       152
weighted avg       0.98      0.98      0.98       152
```

Figure 5: K-Nearest classification accuracy. From the training dataset provide, 70% was used to train the model and the other 30% was used as test data and classified to obtain the results seen above.

III. The Refined Prototype

A. Prototype Overview

The refined prototype consists of an interchangeable PCB which has all our hardware embedded except for the MyoWare sensors themselves. This interchangeable PCB is sometimes swapped out with a custom-built proto board. In the refined prototype state, we are still testing proto-board and PCB compatibility with the MyoWare sensors, and our ML model descripted in Section II Part E.

The refined prototype however contains all the working parts described in the block diagram, excluding the MyoWare Sensors. The final prototype is housed in a PCB enclosure on our embedded electrode sleeve in an ergonomic fashion. As seen in Figure 6 below, the embedded electrode sleeve allows for easy application of the wet electrodes to ensure placement is relatively consistent.

Figure 6: Final prototype of EMG electrode sleeve with enclosed PCB strapped onto bicep using Velcro.

B. List of Hardware and Software

- ATmega328p [6]
- MyoWare Sensors x4 [8]
- C code, analog read/UART transmit
- FT232 breakout [9]
- HC-05 Bluetooth Module [10]
- LiPo charging breakout [11]
- Breaker switch
- USB Micro-B receptacle
- 8 MHz crystal
- LEDs
- Caps, resistors
- Reusable Wet EMG Electrodes

C. Custom Hardware

For our custom design, we went with the ATmega328p for our microcontroller. This MCU is somewhat of a standard in the automotive industry for its versatility while maintaining low power draw.

For Bluetooth communications, we switched from the BLE112 to the HC-05. We made this decision because of a design change regarding the designation of our device as HID compliant. Initially, we planned on configuring the Bluetooth module exactly as a wireless keyboard would be. After dabbling in machine learning, we realized that if we were going to process data on the host computer, we may as well execute keystrokes from the host computer as well. In turn we are using a cheaper Bluetooth module with an older protocol (low energy vs. classic). This came with its own tradeoffs in terms of power consumption and response time, but we ultimately concluded that the switch was justified.

To communicate over USB, we used a FT232 based breakout. FTDI is an industry leader in TTL converters, and while this part was difficult to acquire, it was well worth it in terms of compatibility and reliability.

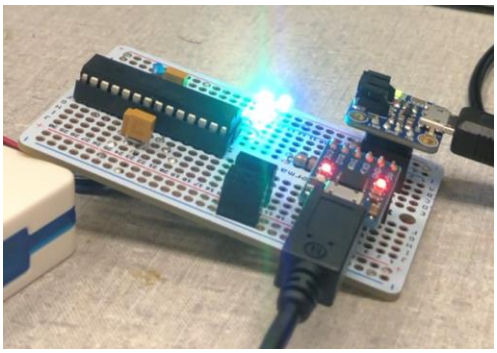

Figure 7: Prototype board used for rapid testing and deployment.

As you can see in Figure 7, we made use of female pin headers and a DIP adapter to easily swap out parts should something go wrong. This helped greatly with our testing process as things inevitably went wrong and we were able to troubleshoot piece by piece. We also used female headers to connect MyoWare sensors and debug/program via ISP. We utilized 2 LEDs, one for power and one as a general indicator. We tested this circuit by reading analog values from each port we were utilizing (A0-A3). We then ensured we could send these values over UART through USB and Bluetooth with reasonable speed and accuracy. Since most of our processing will be done on the host computer through Python, our hardware design was kept relatively simple and essentially just reads and transmits our data.

Figure 8: PCB with populated parts, front and back view.

After verifying our prototype, we ported the design to a PCB as seen in Figure 8. For our PCB we obviously used the SMD versions of the breakouts we had previously used, apart from exempted hardware. For USB communications, we had to design our own circuit using the FT232RL and Micro-B receptacle. The PCB functions perfectly on the first revision, of which the routing layout can be seen below in Figure 9.

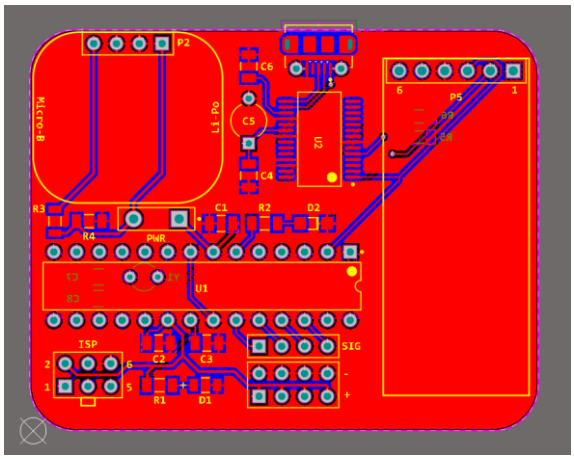

Figure 9: PCB routing without components.

The PCB functioning on the first revision is something that we are very proud of and wouldn't have been accomplished with the extensive planning and preparation we did prior to ordering the first revision.

D. Prototype Functionality

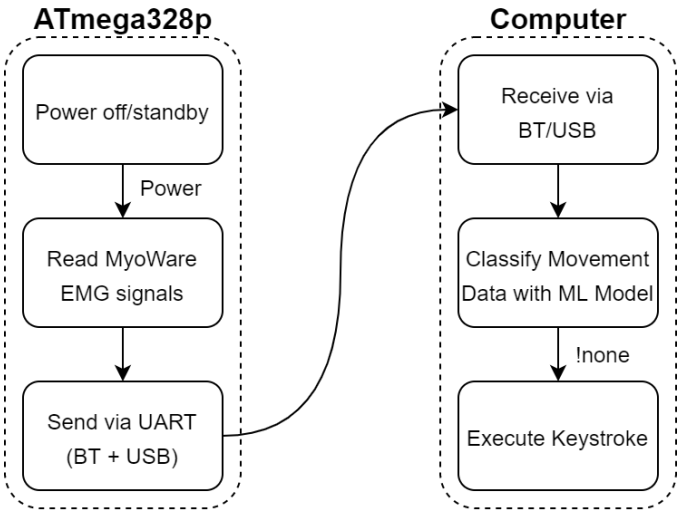

Figure 10: Software Diagram.

As seen in Figure 10, our software is divided in half with the signal capture and communication done on the ATmega328p while the processing and keystroke input is done on the host computer. There is a power switch available on the device that allows it to be powered off when not in use which is crucial to preserve the battery for three hours of active use. The signal can be sent over Bluetooth or a wired USB as the UART data line is common. The signal sent is four comma separated values as a time series so that we capture the entire signal. Once on the host computer, our Python program calculates the root-mean-square (RMS) for every 10 values which makes the impulse of the gesture much easier to characterize. Then, this RMS series is compared against the ML model for each point in the series. The corresponding keystroke is executed once there is a change in gesture (i.e. from no gesture being detected to a fist, because to the ML model its getting a bunch of fist values from the time series).

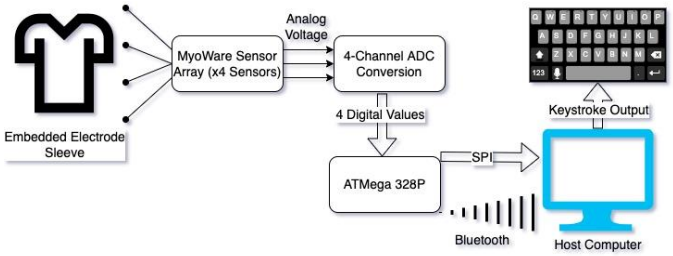

Figure 11: Hardware Block Diagram

As seen in Figure 11, our hardware is composed of our electrode sleeve, the PCB housing and sensor array, and our peripheral connections to power and USB. The electrode sleeve only contains the dry electrodes and cables to connect to the MyoWare sensors located on the PCB housing. The MyoWare sensors capture the voltage different between the two electrodes and these four analog values are converted to digital values and sent to the host computer via UART connection. We went through many iterations of the electrode sleeve design as there are many unpreventable factors that make sensing so unreliable (without taking measures). The humidity of the day, how sweaty you are, how recently you showered, and any small deviations on electrode contact can completely affect not just the values of the sensors, but the entire resulting waveform that we capture as a time series. This is why we added a Velcro strap to consistently strap oneself into the device.

E. Prototype Performance

S1. *The system shall sense gesture movements with a reasonable accuracy*. This specification entailed sensing 5 distinct movements with a net 90% true accuracy. As of the writing of this report, our current best iteration of the design can reliably detect three different movements (a cupping come-here motion, flicking five fingers out, and contract the middle finger) with an average of 85% accuracy which can be seen in Figure 12.

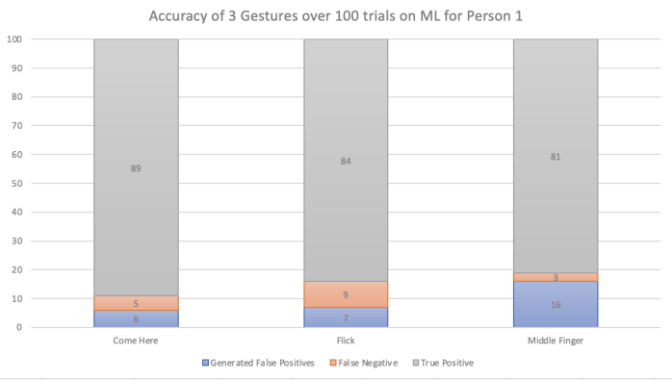

Figure 12: Percent accuracy of the three gestures, come here, flick, and retracting the middle finger, over 100 trials for Person 1 (Sam).

This accuracy was achieved using an incredibly small dataset (only 200 trials of the gesture obtained via time series collection) and with only two MyoWare sensors. A week prior to MDR, two of our MyoWare sensors broke through a faulty ground cable. This happened at the same time as the release of the MyoWare 2.0 sensor, so there was a period of two to three weeks where we were down to only two MyoWare sensors and had no ability to order more (as the MyoWare 1.0 went out of stock before the MyoWare 2.0 released). Therefore, with a large dataset (collected over multiple days, multiple people) and using our final sleeve design in conjunction with the PCB will allow us to get the accuracy we need, only 5% more, and expand the gesture count up to five.

S2. *The system must be reliable*. With the advent of our final sleeve design and PCB, we have a system that can be utilized by anyone provided they complete an initial user setup where they give the program a small training dataset (100 trials of each gesture) to correlate against the main dataset.

S3. *The system shall be ready to use in a reasonable amount of time.* The device can be powered-on and in a usable condition in an average of 15 seconds. This far outperforms our initial goal of a minute of connection time.

S4. *The system shall have reasonable power consumption*. The device can be used, powered-on, for four consecutive hours. This outperforms our initial goal of three hours of active use.

S5. *The system shall be ergonomic*. The adjustable electrode sleeve allows for reliable and repeatable use; so that a user wouldn't need to constantly order wet EMG electrode pads that are irritating to skin. Moving the MyoWare sensors off the arm and onto the PCB housing allows for one to easily use a phone, write, or even eat food while having the sleeve on.

S6. *The system must be usable within a reasonable distance*. Our device functions reliably within three meters. The Bluetooth wireless connection can be used from further than that but we include the wired connection as a backup.

S7. *The system must have some customizability*. Utilizing the customization GUI, users can manage and upload their training data, bind new keys to gestures, and select which gestures they use.

IV. Conclusion

Our EMG computer interface serves as an excellent proof of concept demonstrating that it is feasible to design a device that allows humans to interface with computers of various forms (our interface functions on Windows and Apple devices) using physical gestures. The system comprises of two Myoware sensors [8] placed on specified locations on the forearm that correspond to muscles that contract upon the flexing of different fingers. The signals obtained from these sensors are sensed by our Myoware sensor and converted to digital values (via differential amplifiers on two electrodes that is rectified and filtered) and sent to a computer through either a wired or Bluetooth connection on the UART signal line. Our Python application on the host computer is then able to classify gestures made by the user using a k-Nearest-Neighbors (KNN) machine learning model, and the user can customize which key binds correspond to each gesture on a simple GUI. As shown in Table 3, our system can identify two distinct gestures with 100% accuracy. The accuracy of the prototype did decrease with an additional gesture, however as displayed in Table 3, the correct gesture is still identified in ~90% cases with false negatives occurring more than false positives (+6% difference). With more dimensions of data (from additional sensors, an accelerometer, more EMG sensors, etc.) our system shows it is possible to produce enough accurately defined gestures to allow for touchless human interaction with computers in many different use cases.

This could enable further customization for those with disabilities by moving the sensors to places such as the neck or leg to allow for someone to call for a nurse or request water. Further, the potential for augmented or virtual reality control without the use of joysticks or buttons could lead to remote surgery advances or more accurate bomb defusal robots.

Berke			
	Thumb	Ring	Index
Correct	84	96	88
False Positive	4	0	8
False Negative	12	4	4

Sam			
	Thumb	Ring	Index
Correct	?	92	88
False Positive	?	0	8
False Negative	?	8	4

		ring	index
Correct		100	100
False Positive		0	0
False Negative		0	0

		ring	index
Correct		100	100
False Positive		0	0
False Negative		0	0

Table 3: Percent accuracy of the three gestures, moving the thumb, ring and index finger, over 100 trials for two people. The training dataset for this test was collected exclusively from Sam and is a proof of concept for robust design that it performed better when on Berke, even though the machine learning is using data from Sam's muscles.

ACKNOWLEDGEMENT

We are especially appreciative of Professor Polizzi and his time, dedication, and words of wisdom. His patient tutelage and insightful advice was a mainstay during the entire project, and we are incredibly grateful that we had the opportunity to work with and learn from Professor Polizzi. Additionally, we wanted to extend our gratitude to our faculty evaluators, Professor Taneja and Professor Jackson, for being flexible with us on rescheduling presentations due to our illnesses throughout the year and for being willing to work with us as we navigated chip shortages, sensor depreciation, and the discontinued product support of the Arduino MyoWare 1.0 Sensor mere weeks before our final presentation. acknowledgement

REFERENCES

[1] Marani, M., Katul, G., Pan, W. and Parolari, A., 2021. Intensity and frequency of extreme novel epidemics. *Proceedings of the National Academy of Sciences*, [online] 118(35). Available at: https://www.pnas.org/doi/full/10.1073/pnas.2105482118

[2] Gaba, Jacob A., "Air Keyboard: Mid-Air Text Input Using Wearable EMG Sensors and a Predictive Text Model" (2016). *Dartmouth College Undergraduate Theses*. 114. https://digitalcommons.dartmouth.edu/senior_theses/114

[3] Tap Systems, Inc., "Tap Strap 2". *Tapwithus.com.* https://www.tapwithus.com/product/tap-strap-2/

[4] C. Choi and J. Kim, "A Real-time EMG-based Assistive Computer Interface for the Upper Limb Disabled". *2007 IEEE 10th International Conference on Rehabilitation Robotics*, 2007, pp. 459-462, doi: 10.1109/ICORR.2007.4428465. https://ieeexplore.ieee.org/document/4428465

[5] Backyard Brains, "Experiment: Signal Classification". *Backyardbrains.com.* https://backyardbrains.com/experiments/RobotHand

[6] ATmega328p Microcontroller, Data Sheet, *Microchip*, https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

[7] MyoWare 1.0, Data Sheet, *Sparkfun*, https://www.sparkfun.com/products/13723

[8] MyoWare 2.0, Data Sheet, *Sparkfun*, https://www.sparkfun.com/products/18977

[9] FT232R USB UART IC, Data Sheet, *Future Technology Devices International Ltd*, https://cdn.sparkfun.com/datasheets/BreakoutBoards/DS_FT232R.pdf

[10] HC05 Bluetooth Module, Data Sheet, *ITead Studio*, https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf

[11] LiPo Battery Charger, Data Sheet, *Adafruit Industries LLC*, *https://www.digikey.com/en/products/detail/adafruit-industries-llc/4410/10673110?utm_adgroup=Evaluation%20and%20Demonstration%20Boards%20and%20Kits&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Product_Development%20Boards%2C%20Kits%2C%20Programm*

[12] Maciej Zajaczkowski, "Simple Dry Electrode EMG for Arduino". *Instructables.com* https://www.instructables.com/Simple-Dry-Electrode-EMG-for-Arduino/

[13] Sylvain Colliard-Piraud, "Using an electromyogram technique to detect muscle activity". *STMicroelectronics* https://www.st.com/resource/en/application_note/dm00356634-using-an-electromyogram-technique-to-detect-muscle-activity-stmicroelectronics.pdf

APPENDIX

A. Design Alternatives

We had to iterate through many different design alternatives when it came to the sensor, sleeve design, and layout of the final device. Firstly, we had to forgo our custom EMG sensing circuit and instead use the industry standard Arduino MyoWare sensor. We initially found several examples of projects that used custom EMG sensing circuits, such as [12] that also used dry electrodes or [13] that achieved high accuracy, which gave us hope in this approach. However, we made the decision to forgo the custom sensors because the accuracy of our custom circuit was far worse than the MyoWares. This can be seen in Figure 13 where the accuracy for three different gestures, contracting the ring ringer, twitching the thumb, and making a fist, for the custom EMG circuit over 100 trials is quite poor.

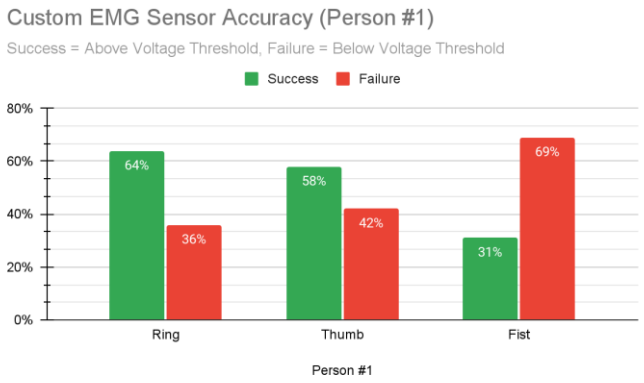

Figure 13: Custom EMG Sensor Accuracy Over 100 Trials

If we compare this to Figure 14, where we see the accuracy for the same three different gestures, contracting the ring ringer, twitching the thumb, and making a fist, of the Arduino MyoWare over 100 trials, we see the MyoWare performs far better.

The design of our electrode sleeve pivoted from using sticky, wet electrode pads to utilizing a neoprene sleeve that can be velcroid on for repeatable, reliable use. This was done to allow our design to be used on different people and achieve similar skin contact, along with making the device more ergonomic and able to be reused infinitely.

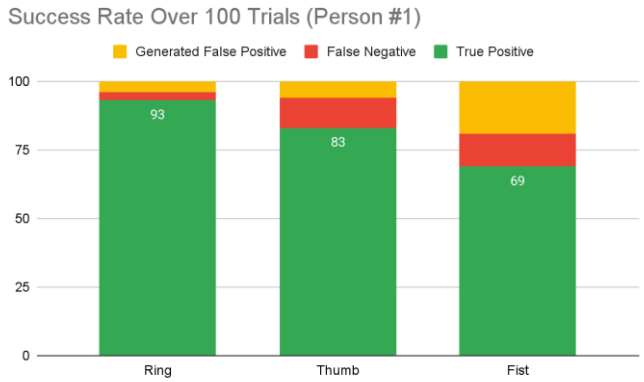

Figure 14: Arduino MyoWare Sensor Accuracy Over 100 Trials

Finally, we changed the layout of our sensors as we found that this positively impacted performance. First, we moved the sensors themselves off the arm sleeve by soldering multicore electrode snaps onto headers on the device. This allows the PCB and sensors to be housed in one ergonomic unit. Additionally, the sleeve itself is now just some light wires, electrodes, and the material itself. This can be seen below in Figure 15.

Figure 15: Custom EMG Electrode Sleeve Prototype.

B. Technical Standards

In order for our device to communicate with a host machine, communication standards must be in place so that both devices are speaking the same language. Our goal is to make the device wireless for ease of use. Additionally, a wired backup will be available in case of battery depletion or stationary configurations.

For wireless communications, we are using an IEEE 802.15.1 standard Bluetooth protocol. The HC-05 we are using is a class 2 device, meaning it can communicate in ranges up to 10 meters. This device is also FCC compliant with under 1.6 W/kg of RF exposure (watts over kilograms of body tissue).

For our wired communications, we are using IEEE RS-232 standard format to transmit data from the microcontroller. This is sent using UART protocol to our FTDI TTL converter. This

converts transistor logic (RS-232) to USB protocol allowing data to be read via COM port from a computer.

C. Testing Methods

Our team used various amounts of testing methods, however not all testing methods that we had developed could be used as the product has changed through multiple design reviews throughout the process. Our main testing methods that we used were manual collection of accuracy data. Simply put we would take our entire system and see how accurate the system is. As someone performs a movement, we categorize the movement into a class: true-positive, true-negative, false-positive, false-negative. This allows us to aggregate and determine the overall accuracy of our system throughout various trials, re-designs, and placement on the forearm.

Another main testing method was to be able to send information through our UART interface to the host computer. We made sure that we could accurately send information via the USB cable and more specifically send 3 inputs, in proper format for our system to recognize. Then the use of 3 potentiometers were used to determine the delay between turning potentiometer and receiving the input at the host computer.

Lastly, we have verification testing protocols that we will use for each of our system specifications that will be used for us to judge our final product against and determine if each specification was hit. V numbers in the following section match up with the S numbers provided in the system specifications in section ID.

V1. To verify S1, we determined that we will test that the sensing system can meet true positive/negative and false positive/negative percent specifications over 100 trials for each gesture. In other words, we will make sure that the true accuracy for our product is over 90% for 100 trials of each of the 5 gestures.

V2. To verify S2, we will intentionally misplace the device on our forearm by a few centimeters from marked locations that are known to work. We will perform this task on multiple users to determine if the accuracy remains constant despite the subtle changes in placement.

V3. To verify S3, we simply will time the power on to successful connection time to the host computer on a stopwatch 100 times and determine if all of the connection times were under a minute.

V4. To verify S4, we will demonstrate that the device can be actively used for up to 3 hours by measuring current draw from each component and calculate the theoretical active time on. We will use current draw calculations from the higher end of consumption to simulate active use.

V5. To verify S5, we will demonstrate that while holding the EMG interface device a user can actively write a paragraph on paper with a pen or pencil and then utilize a cell phone to make a phone call. These show how the system is at an ergonomic level.

V6. To verify S6, we will demonstrate the ability to connect a device from multiple distances from the host computer, using 3 increments and measuring up to 3 meters away.

V7. To verify S7, we will verify that all keyboard inputs can be customized to bind to distinct movements. Ex: Fist can bind to voice activation. Fist can then re-bind to tab input. Fist can then re-bind to space input.

D. Project Expenditures

Specific Name	**Cost**	**MDR or FPR**
Reusable EMG Electrodes (80 ct)	$15	Both
Tape, Snap-pins, Reuseable Pads	$15	Both
Cables, Cable Shields	$10	FPR
HC-05	$10	Both
BLE-112	$20	Both
FT232RL USB-Serial Breakout	$25	FPR
2124 LiPo Battery Backpack	$20	Both
Misc. MDR Components (Op Amps, ADC, etc.)	$10	MDR
Sparkfun MyoWare Sensor	$160	Both
Additional MyoWare 2.0 Sensors	$80	Both
Versa-Trode Electrodes (120 ct)	$36	Both
Onboard Components (Op Amps, Resistors, Capacitors, etc.)	$30	FPR
PCB	$60	FPR
		$491.00

Table 4: Project Expenditures

E. Project Management

Although each of us certainly specialized in certain fields, we each collaborated and helped out in each and every aspect of this project. Sam Worrell functioned as the team coordinator, leading communications with our advisor, the instruction team, and our evaluators. Along with this, he devoted most of his time prior to MDR on threshold detection software and our custom EMG sensing circuit that ultimately wasn't as accurate as the industry standard MyoWare. Sam additionally led the electrode

sleeve design effort and had a hand in in prototyping the ML programming along with extensive test data collection. Aidas is the champion of our signal processing and has also been the go-to when it comes to ML. He helped in many other ways, such as helping in electrode sleeve design, protoboard manufacture, and microcontroller programming. Ryan spearheaded the PCB design, PCB manufacture, and created our initial protoboard design. He was the expert when it came to the entire PCB fabrication process and worked with Aidas to develop all of our microcontroller software. Additionally, Ryan single handedly tackled the design and creation of our team website. Berke primarily assisted in data collection and all software outside the microcontroller. He designed the custom GUI to use the device, developed Python code to speed up data collection, and was our go-to testing subject for developing new gestures for our ML training datasets. Finally, Berke helped to improve and explore Bluetooth wireless communication on our final design.

We have been communicating well and stuck regularly to meeting weekly, especially after MDR. Sam usually acts as the spokesperson for the group and provides leadership during presentations, meetings, and check-ins. Aidas uses his organizational skills to lead our lab sessions and make sure we devote our energy in areas where our effort will be best rewarded.

F. Beyond the Classroom

Aidas Jakubenas – The biggest challenges and learning that needed to be done with a project like this was time management. Ultimately, we design and build a product from the bottom up, and this requires a lot more planning and time management than I previously thought. Another big challenge was learning how ML works. None of our team had major experience with ML models and especially how to implement them, so the use of multiple research papers on similar topics was a major help as we could pick up patterns on which models to use for a project like ours.

Ryan Dewsnap – Throughout this project, I've learned a lot about the overall design process through experience as we started with an idea, rapidly prototyped, refined our specifications, and executed our gameplan. Starting with just an Arduino and a breadboard, we were able to gradually build our idea from the ground up. One major takeaway I got from this stage was the power of parallel processing through careful planning and teamwork. What seemed like an extremely difficult task, soon became much less scary as we separately developed subsystems and brought them all together.

After we had finalized our design, I got much valuable experience with prototyping on protoboard. I learned how to utilize DIP adapters, SMD adapters, and developed my hand soldering skills through building out and testing our system on this protoboard. Using the protoboard, I worked on my C coding skills as we moved from the Arduino to a standalone 328p. After verifying our circuit's functionality, it came time to design the PCB itself. With the help of Chris Caron's wonderful tutorial videos, I was able to design a compact PCB compliant with the manufacturer's specifications. I also gained valuable experience communicating with Sam to verify the design, as well as the manufacturer to ensure everything was printed as intended. Once the PCB came in, we all learned about the many ways one can solder SMD components. For our application, it was easiest to use a hotplate.

Overall, I have learned a lot this year and feel that most of it will be applicable to my professional life in the future. I learned how to design a PCB from start to finish, intricacies of communication protocols, greatly improved my soldering skills, and more. Though I've developed many technical skills, I think the most valuable takeaway from our experience this year has been teamwork and problem-solving skills. This is the essence of engineering. Facing new challenges each day has tested our resilience and drive to create something we are proud of.

Sam Worrell – The development of key fabrication, manufacture, and design skills were essential in helping my team and I develop our final prototype. In the lab, I learned different new soldering techniques to help with surface mount circuits. Additionally, I further my knowledge of circuit debugging and troubleshooting techniques by repairing our Arduino MyoWare sensors and developing our protoboard. Outside the lab, I learned how to use Altium design to develop multiple revisions of a PCB the size of a credit card. Chris Caron's YouTube tutorials were instrumental in helping us both design our PCB and manufacture it. Most importantly, our advisor, Professor Polizzi, was our most valuable source of advice when we encountered difficulties or setbacks in our project. His advice enabled our project to be much more realizable and feasible than our initial proposal. Although I may not focus on embedded programming, circuit design, or even work with PCB's, I will certainly take with me the experiences I have had working with my team on going from PDR to FPR. I learned many lessons on time management, resiliency, planning, and creative thinking that I will utilize throughout my professional career.

Berke Belge - In my opinion, the greatest challenge in developing this project has been adapting the overall design based on various extenuating circumstances. There have been many instances where we have had to make major revisions to the design and scope of our project to adapt to changing needs and obstacles that have arisen. Our advisor, Professor Polizzi was a great help to us in revising aspects of our design to account for various changing variables, as well as our evaluators, Professor Jackson, and Professor Taneja. I personally gained a substantial amount of knowledge regarding

embedded and host software as well as the general overview of all the hardware design aspects that go into creating a project such as ours. Another area that I have improved in over the course of this project is in my presenting skills. Doing various presentations over the course of these two semesters in a simulated professional environment will be useful in my career moving forward in the industry.