NeuroController

Authors: Jonathan Ke, Wendy Mo, and Jean Udompanyawit

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—For individuals with difficulty controlling their muscle functions, performing tasks on a desktop computer is virtually impossible. Given that the traditional mouse and keyboard would be very difficult to use in this scenario, we aim to create an alternative platform that allows a more accessible control through brain signal acquisition techniques, specifically electroencephalogram (EEG) and electromyography (EMG). This platform increases computer accessibility through an integrated control interface system that is easily customizable for users.**

***Index Terms*— Arduino, Bluetooth, EEG, EMG, Emotiv, Fast Fourier Transform (FFT), Logistic Regression, Python, Random Forest, Support Vector Machine (SVM)**

I. Introduction

Nowadays, the computer is prevalent and critical for people in various ways, ranging from working to easing general daily lives tasks. Even though computers are being more accessible to the general body of people, one of the limited user groups are those who lack physical control abilities. For individuals who suffer from physical disabilities such as amputees, there is no platform in the market that offers an alternative controllable interface on the computer for this user group. Therefore, our project will aim to increase accessibility of desktop devices for individuals who do not have control of their upper limbs or have difficulty controlling muscle functions. We will create an application that serves as the mouse and keyboard interface and will be controlled through electroencephalography (EEG) and electromyography (EMG) signals.

Even though there is various research studying the control of cursor movement, there are currently no competing technologies in the market that offer an integrated interface allowing a full interaction on the computer desktop. Therefore, we are developing this platform to allow the user to both control a cursor as well as to use a keyboard on a device, through mental and augmented physical control. Our goal is to provide the individuals with a minimum shoulder movement ability and with full mental control abilities to be able to use this platform to navigate tasks on the computer wirelessly. This will provide full and unhindered access for people with disabilities to use a traditional desktop device such as a laptop on a user-friendly control.

II. Use-Case Requirements

Our main goal in this project is to provide an interface for those with disabilities to use desktop devices. The most important requirement is that the platform works with low latency, in real time, and with high accuracy for unhindered control on a desktop application. One use-case requirement we have is to aim for 75% of users to open the Chrome browser within 60 seconds. This will be tested through one of our testing methods, called the Task test. To ensure a satisfying speed of our platform across the software and hardware implementations, another requirement is that a user, on average, can score 500 ms on reaction time in the human benchmark test using our device. Further, we will require that our user hit three targets within 60 seconds with a 75% accuracy. This user accuracy and speed requirement will be tested through a point-and-click test. Another use case requirement comes from our signals; for our application, two types of control data are required: continuous control data and single-time control data. Continuous control data will be obtained from the EMG, which measures the muscle movement signals from the shoulders. On the other hand, four types of distinct single-time control data will be acquired from the EEG device. These distinct signals will be used to serve different features on our interface. For example, the EMG sensors will allow for control in either an up to down or left to right direction and left winking will mimic clicking. These feature mappings will be set in this manner when the user puts on our device for the first time but will be customizable to allow for maximum flexibility. Finally, the last use-case requirement is that we will have an onscreen accessibility keyboard that provides the ability to send all standard keyboard inputs, modified with customizable widgets based on the user preferences.

III. Architecture and/or Principle of Operation

We are building our application using Python as the programming language. From there, we are using various packages and libraries, including Tkinter and PyAutoGUI. For our backend signal processing, we are creating and training ML models using the Scikit machine learning library. Python provides a development environment that can be used and accessed by both the developer and user on any laptop. After the user puts on the EEG headset, is attached to the EMG sensors, and opens our application, the user will be prompted to a screen that has an accessibility keyboard. For the EMG sensors, we will connect an Arduino with a Bluetooth module to transfer the data wirelessly. Both the headset and sensors serve as our data acquisition platforms. When the user performs various actions to accomplish tasks (eg. double blinking to click), the data will be processed and classified through ML pattern recognition, and the application will trigger the mapped action within the desktop.

The EMG shoulder movement will allow the user to control moving the cursor or going across the keyboard either in a left to right motion or in an up to down motion. The user will be able to switch between these two methods through one of our extracted features. The gestures used to control different functions will be made customizable based on our user preferences. Further descriptions of our system architecture are described with our figures.

There were many changes to our overall system from the design report. First, we removed the FFT processing to get the frequency domain information because the features we had chosen to build our ML models to parse for an action had already produced a satisfying result of 90%. Moreover, adding extra processing algorithms would also slow down our system as well as parsing through a greater number of features would mean that EEG classification would take longer.

We also removed the mapping of double blinking to an action because we faced the problem of falsely predicting that event happening, existing when it did not occur, or not recognizing the movement when it occurred. When we ran tests during our live-sampling, double blinking gave us the greatest error. The left and right winking actions are distinct enough such that there was the smallest amount of error when we did our live sampling. Our final implementation had the left wink mapped to changing modes and right wink mapped to a left click.

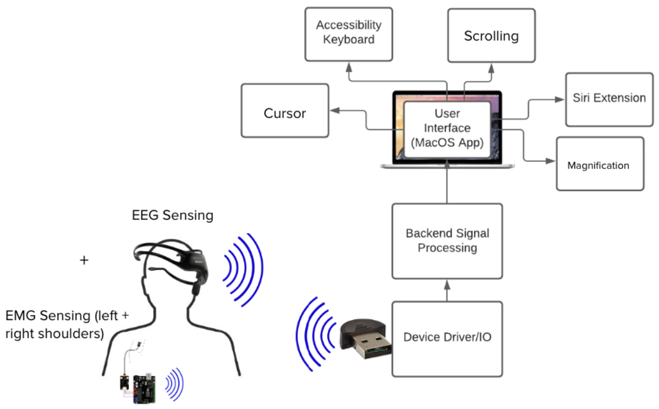


Fig. 1. Overall System Layout. Users will generate signals via EEG and EMG. The signal is sent wirelessly to backend processing that funnels recognized signals to the user interface.

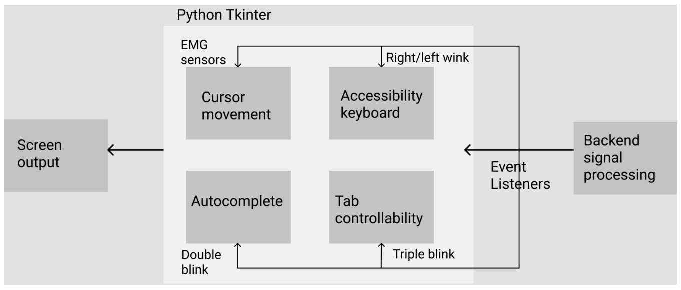


Fig. 2. Frontend Block Diagram. The processed signals from the backend will be connected to event listeners that the frontend can respond to. Users will activate certain features of our application, which will be displayed on our screen output.

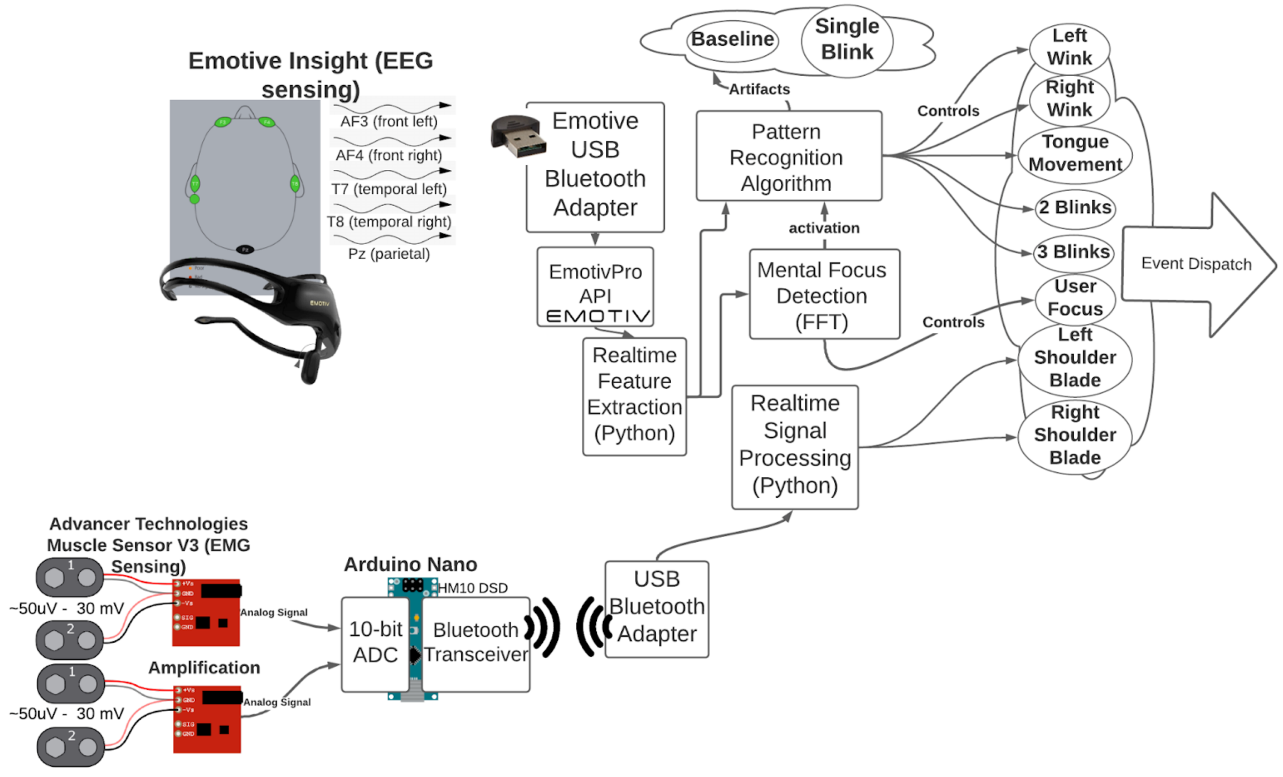


Fig. 3. Backend System Layout. Both EEG and EMG signals from the two sensing systems are funneled through separate wireless transmitters and unified under the same backend layer.

IV. Design Requirements

Speed and accuracy are our two most important requirements. As mentioned in the use-case requirements section, we aim to have over 500 ms latency from acquiring EEG and EMG signals to the end process of the computer input. Our data will be transmitted via Bluetooth, which generally has a latency of approximately 100 to 300 ms. We require that our Python backend processing does not have a latency over 50 ms. Python backend processing refers to the signal processing, machine learning data classification, and communication between the data acquisition device to backend program as well as to the user interface. Our accuracy will be speculated based on two aspects: signal classification accuracy and overall user control accuracy. The first design requirement is to have our signal classification accuracy of more than 90% using a combination of support vector machine (SVM), random forest classification, and linear classification models. We require that the collected training data be contained from over 200-300 samples data sets per each signal data type. The data will be sampled at 128 Hz. Since our window of interest will be running through the samples of approximately 3 second span as our longest data. EEG data will be analyzed in real-time to detect the most fitting classification for double blink, left wink, right wink, normal baseline, and unclassified noisy signals. Real-time EMG data from the two shoulder blades will be our main controlling method for the two directional control planes: left/right and up/down. Therefore, we aim to have the detection of the shoulder movement to be used as the continuous controlling data. The design requirement for our interface is to offer our user both the cursor and the mouse control on the desktop by EMG, with additional controls through EEG. The user would be able to navigate through the entire desktop page easily with the cursor triggered by EMG signals from the shoulders or send key inputs like tabbing through a webpage using a combination of EEG signals.

V. Design Trade Studies

A. *Emotiv Insight Model 1.0*

The Emotiv Insight is a 5 channel, wireless headset that records the wearer's brainwaves and translates them into meaningful data. The Emotiv Insight is preferable to other EEG headsets (both produced by Emotiv and other companies) for its lower price point and its ability to provide the data necessary for our modeling. While other headsets contain more channels and provide more information, after looking at our project budget, the other EEG headsets would have cost more than the money allotted. Lastly, there was already an Emotiv Insight available in the capstone inventory, so we did not need to spend any money. This device also provides an out-of-the-box platform for interfacing through the Emotiv app for our application development.

B. *Hardware Design*

A major and early decision made in the designing of our application is to use the commercially available Emotiv headset. With this, we are somewhat limited in how we can pursue low latency and easy operation within our system. We initially considered adding a RaspberryPi to offshore the ML and signal processing to a second computing device. However, the tradeoff here is that we desire our system to be wireless with respect to the headset. The Emotiv headset uses a Bluetooth wireless dongle to send data from the headset. To reach the target of 500 ms between a user's response to a stimulus and generating the corresponding output, we already need to account for a latency of around 200 ms for the Emotiv Bluetooth receiver to acquire from the headset, process, and setup for API access [1]. If there exists another remote device like a RaspberryPi to acquire, process, and send data again through a wireless system like Bluetooth or WiFi, we expect around an additional latency of 200 ms. This leaves approximately 100 ms for the user to react to a stimulus and for our signal processing and application inter-communication to react to the user's signal, which is a lower flexibility time window for the user. We need more slack in meeting the latency requirements so instead are opting for our primary compute to be performed within the desktop or laptop which the application is being used from to ensure that the majority of the latency is from the Bluetooth connection of the Emotiv Insight and wired latencies within the desktop operating system, providing greater slack for users to react. This will include some work to make sure our software critical path is short and inexpensive as to prevent our application from overburdening the desktop hardware.

C. *Model Selection*

With the headset chosen for acquiring EEG signals, the Emotiv Insight also comes with software packages to process and interpret signals, which we opt to not use and instead to construct our own.

From experimentation of the EmotivBCI package, which is provided with the Emotiv Insight, we found very low reliability and minimal ability to tune or update the package to meet our accuracy needs. Primarily due to the lack of tuning and the closed source software licensing of EmotivBCI, which shrouds the ability to tune or test the system to meet the 90% accuracy metric, we decided to instead ingest raw EEG data from the headset and process the data using our own models and filters to ensure we have the flexibility to meet the metric. Furthermore, since we only have one headset on hand, we desire our recognition system to be trained specifically for this headset.

Given that the software will be hosted on a desktop computer potentially running other applications, we desire to find models that are lightweight, quick to train, and deploy. Hence, we will be using an assortment of random forest, SVM, and logistic regression models over more complex and expensive models like a neural network. Neural networks are not optimized for CPU processing and require a large amount of data to train successfully. Our project is limited by the small samples of data we need to manually collect. To allow for rapid retraining and use, we cannot deploy a neural network. In comparison, random forests and logistic regression are cheap to train and quick to build given we are choosing a set of features that should be easy to differentiate on and more complex feature sets can be modeled using SVM.

D. *Python Tkinter*

Our user interface is an essential part of our project because it influences a user's experience while using our application. An inviting, easy to use interface will increase a user's satisfaction.

Initially, we looked into various desktop application development platforms and had decided on using Flutter for its reduced code development time, "hot reload" feature, and its ability to deploy visually appealing apps from a single codebase. However, we decided to go against this option because none of us had experience building an app from scratch and there was a steep learning curve.

After settling on using Python as the language for our project, we considered using Tkinter for the creation of our user interface for its various built-in features. This will allow us to easily interface the frontend with the backend.

VI. System Implementation

A. *Signal Processing Algorithm*

Our data was sampled at 128 Hz, processed through temporal analysis, and fed into an ML model. On the pattern recognition side, the incoming real-time signals will be analyzed continuously and will be classified into different pattern recognition according to our signal input. Since we have real-time data running constantly, the data will be kept in a 30 second data buffer, in which we will analyze the data using a sliding 1.5 second window. Our ML model will classify the data into double blinking, left wink, right wink, normal baseline, and unclassified noisy signals. Outstanding features of the patterns in terms of amplitude and data variance will be used to classify the signals into the following categories. A more detailed explanation of the procedure training and classification will be explained in section C, *EEG Classification*.

B. *Signal Acquisition*

The brain signals we will be acquiring are AF3, which is located on the front left of the forehead and AF4, which is located on the front right of the forehead. These signals will be acquired through the Emotiv Insight headset which sends signals via Bluetooth to a locally installed EmotivPro desktop application accessible through the Emotiv API interface.

A custom Python script will be created with a thread directly responsible for calling the Emotiv API to set up a connection to the Emotiv device and communicate connection issues to the frontend. Once a connection is established, the same thread will poll sensor and device data from the API stream, strip context data, and dump the relevant sensor data into a local buffer which will have the capacity to hold the previous 30 seconds of polled data to handoff to the classification subsystem. A second thread within the running process will use the currently buffered dataset for classification and logging. Another thread within the process will be actively Fourier transforming incoming data.

C. *EEG Classification*

The classification subsystem running on a second thread will classify raw EEG data stored within the buffer into left and right wink. To do this, the classification thread maintains a sliding 1.5 second window on the previous 30 seconds of data within that shared buffer and live computes features from the 1.5 second window. The features computed will be described in more detail below.

The process for distinguishing different blinking and winking patterns is done by extracting features including maximums, slopes, average signal strength, frequency strength, variance, and differences in amplitude in the AF3 and AF4 electrode data and using a random forest model to predict the occurrence and number of blinks and winks.

The models described above are trained using manually collected data from the Emotiv headset before deployment. Over 1700 unique samples have been collected for training the various models across 8 different individuals. The ML models will be trained in Scikit-Learn to optimize algorithm performance over smaller data sets. Some visualizations of the sample data we will use for training are provided in Appendix A. This data has already been cut to only have the feature we are detecting within the sample window. The resulting outputs from the model will be dispatched to a data bus shared by event listeners on the frontend.

The full system is then implemented for real-time as an ensemble FSM decider which takes each smaller model into consideration when making final predictions. The model is shown below.

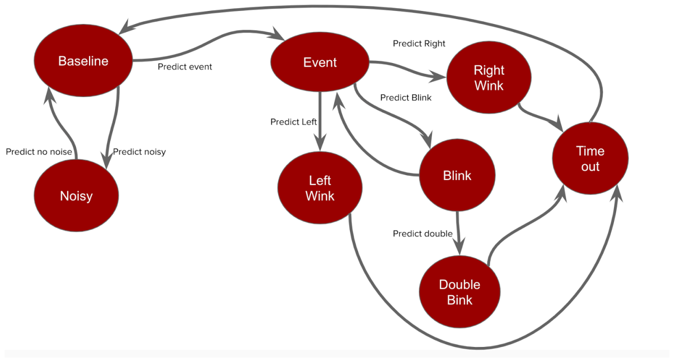


Fig. 4. Finite state machine for EEG classification

The full model uses multiple smaller ML models to make an educated prediction about whether a user triggered event actually occurred. Validation between different models is done to ensure an ensemble prediction yields better results than not. This also mitigates false positives as if there is a mismatch in prediction, the ensemble model will hedge on the side of no event occurring. The model times out after each valid event is predicted as to avoid repredicting the same event within the time window.

D. *EMG Signal Acquisition and Implementation*

EMG signal is the continuous control input signal that we will be using to produce continuous movement to activate the cursor. We purchased the EMG sensor that will be connected to Arduino and send the data wirelessly to the computer. EMG ranges from 0 uV to 15 mV prior to amplification [2]. Shoulder movement would increase the amplitude of the signal which will be used as the right and left digital signals. However, depending on the users, other muscle positions can be used for the detection as well, such as the back movement muscle. These signals will be sent wirelessly through the Bluetooth module that is attached to the Arduino and sent to the backend for processing. Processed signals will then be dispatched to the frontend as an event. The right signal will be used for right/up control in the front-end interface, whereas the left signal will be used as the left/down control. To create a more stabilized system, the user EMG stream will be used in the EMG calibration mode.

To calibrate, the user has to sit relaxed for approximately 5 secs to allow 1000 data points (equivalent to 4.7 seconds stream). If the buffer data has an average of less than 200 with standard deviation of less than 20, the average value will be calculated and used as the reference value for threshold detection. For muscle movement detection, we have a buffer that contains 50 data points, the average of the data points and standard deviation will be calculated. Furthermore, the movement detection signal to be sent for the event trigger on our front end if the ratio between the average real-time data and the average baseline data must be greater than 1.3 and the standard deviation should be greater than 20.

E. *Software Interface (Backend and Frontend)*

A bus system will be established in which the backend will publish detected features and event listeners on the front-end application will react and activate based on the incoming feature stream. The frontend application will dispatch interface movements accordingly. The features of our application will be displayed on our screen output. Users will be able to perform various tasks, including typing on a keyboard, moving a cursor across the screen, and performing left and right clicks. This will all be coded in Python.

VII. Test, Verification and Validation

The design of our interface is mainly broken down into three parts: signal acquisition, backend signal processing, and the front-end interface. Most of our tests will be focused on the latency, accuracy, and the user satisfaction. As mentioned in the other sections, we require our system to work on real-time data analysis, thus, we would like to ensure that the latency of our two routes of data, EEG and EMG, are within an acceptable range that the user will be satisfied with.

A. *Results for EEG signal classification accuracy*

For our EEG signal processing testing, we will test on the pattern recognition accuracy. From research paper [3], the general linear technique such as SVM has accuracy up to 97-98% of analyzing the EEG data. Therefore, our goal is to be able to have above 90% accuracy by using either the SVM and/or random forest classification. From our current data sample, we will split 75% of the collected data into a training set and 25% of the collected data into a test set such that the test and training sets both have the same proportion of each movement data point. For different models with different hyperparameters, we will use the training set to train the model and verify the model meets the 90% accuracy metric by seeing if it predicts the test data with below 10% error. Of the subset of the models we select from this process, we will add each model into the real-time prediction system, add functions to parse the appropriate features that are used by the model to predict, and then attempt to predict features live. For each signal we need to predict, we have the subject we are monitoring in the session produce that signal 20 times in sequence with at least the second pause between each and track the number of times the model predicted the movement occurring correctly. We track the number of times the model predicted incorrectly, including falsely predicting a feature existing when it did not occur and not recognizing the movement when it occurred, and divide it by 20 to get a percentage error for each movement we are testing. We then take an average across all the different movements to get an overall error prediction percentage. We aim for this number to be below 10%, meaning 90% of the time the model detected the movement in streamed EEG data or did not detect a movement that did not occur.

The logistic regression and random forest models we are using in our final system have validation errors reported below.

ML Classifier (Individual classifiers used in ensemble classifier)	Validation Error (%)
Sensor is moving/detached (Logistic Regression)	1%
Eyelid event occurred (Logistic Regression)	6.31%
Blink occurred (Logistic Regression)	2.53%
Right Wink occurred (Logistic Regression)	2.61%
Left Wink occurred (Random Forest)	20%
Double Blink occurred (Random Forest)	9.5%
Left Wink or Right Wink (Logistic Regression)	0%

Fig. 5. Finite state machine for EEG classification

All smaller models used in the ensemble classifier have a validation error under 10% as we desired except the classifier for left wink occurrence. After multiple tests with different computed features from the test data, most models have a fairly separable data set it can reliably predict on. However, in the case of the left wink model, because of the fair similarity of a left wink event compared to a regular blink event, especially factoring in the variance between different people, the prediction error on left winks is higher. However, the average model error across all models is below 10%, which meets our target for validation error. It can be observed in the appendix that signals for what we desire to classify can vary wildly and contain much noise but still be effective enough that we desire to classify them. In the appendix we can observe signals that are more ideal and cleanly represent particular movements and signals that are more noisy and much more varied that we still desire to classify correctly. This creates a major difficulty in overall classification accuracy and its ability to generalize to the real-time system.

Below are the error rates from testing the real-time ensemble classifier. We performed a certain action 20 times in a row and tracked the number of times the model predicted incorrectly, including falsely predicting a feature existing when it did not occur or not recognizing the movement when it occurred.

Left Wink	Right Wink	Double Blink
10%	15%	30%

Fig. 6. Error rates after testing our real-time ensemble classifier

The model detected correctly left wink 90% of the time, detected right wink correctly 85% of the time, and detected double blink 70% of the time. Except for the left wink detection, the error incurred by the ensemble model is significantly higher than we desired. We attribute much of this error to desensitization of the model to mitigate registration of false positives, meaning much of the error observed is from the model not detecting a signal when it was present. During user testing, we observed a high number of false positives, causing many unintentional movements when the user never issued the signal. To increase usability, we scaled our ensemble model to be less sensitive to the three kinds of signals above, so the system may not react to the actual occurrence of a meaningful signal but will not falsely emit a signal when it did not occur. The rationale for this late tuning decision is because we would rather the user retry the same command multiple times and have the system trigger on one of the repeated events than trigger unintentionally. It was observed that this made the system easier to control when performing tasks.

B. Results for EEG full system pipeline latency of <500 ms

A human benchmark test that is available online will be used to test speed of the control from the EEG signal acquisition to the output on the interface. The user will react to the stimulus, in this case clicking the screen in the test without any planar control from the EMG. This will be a good approximation to see the latency sum of acquiring the data from EEG headset, wireless transfer, backend signal processing, and the sending of data over to the front end. The entire process is expected to be less than 500 ms. To ensure the least and consistent latency from the other computer programs, we will be running only our API in the background and all other applications have to be closed.

Our system's performance on the human benchmark test was an arithmetic average reaction time of 1780 ms across three subjects. This is about 3 times the latency we expected from the system. During testing, it was observed that if the system reacted immediately to the user stimulus, the reaction time was closer to 550 ms, which is in the ballpark of what we desired from our system. The delayed time was due to the scaling down of system sensitivity to increase overall system usability. This caused the user to need to resend the same signal multiple times before the system registered the event. Our overall assessment of the system is that it should be usable, and the system does not lag significantly enough for this to impact the user experience, especially in regard to accomplishing tasks. However, random false positive signals do significantly impact the user's experience and controllability of the system, so we decided to optimize for this particular attribute at the expense of other metrics.

C. Results for Integrated system: accuracy and speed

This test was used to test the accuracy and speed of the control from the EMG signal with the clicking ability from the EEG signal. We will be using the point and click test that is created by ourselves. The test was for a one-minute period, and we aimed for the user to be able to click on three of the targets. The user had to click on these targets in a specified order. In our test, we have the subjects control the cursor to type two words in our keyboard interface, in which the words are ‘cmu’ and ‘dog.’

After testing across three subjects, each performing the task twice, participants finished the task in an average time of 74.5 seconds. This is within 25% of the target time for finishing the task. The recorded average fell slightly above our target time, but the results show promise that with some more practice and patience, users can accomplish normal point and click tasks using the system, which was the primary goal this metric measures.

Essentially, from the three tests that were taken, two were not satisfying our requirements as shown below. However, in those cases there is a huge variance in the result for the best and worst case. The best case of our user latency is at 614 ms while the worst case is 2628. This implies that our system has the potential to meet the requirement if the system is stabilized with that instance of use, meaning that the EEG were placed at the efficient spots on the head.

Requirement	Testing Strategy	Quantitative Metric	Results
User Latency	Human benchmark test	Register the time the user takes to click in reaction to a display stimulus in 500 ms	Average reaction time is 1780 ms
User Accuracy & Speed	Point and click test	User can click 3 randomly spaced static targets within 60 seconds	Average time is 74.5 seconds
Overall Intention Accuracy	Task test	75% of users can open Chrome browser within 60 seconds	Achieved in 42 seconds

Fig. 7. Testing results along the three tests with results taken on average from our test subjects.

VIII. Project Management

A. Schedule

Our schedule was broken into three main phases to distinguish between EEG data collection and model training, EMG setup, working on individual components, and integrating and testing. Throughout the semester, we each ran into various roadblocks and had to update our Gantt chart to allow more time for setup and integration. All project code is hosted at https://github.com/Jonny1003/capstone-18500-eeg.

B. Team Member Responsibilities

Our work is split into a few different categories, including signal processing, hardware components, and software environment/application. Although our project tasks are roughly split along these categories, we will work together to complete the tasks. The assignment of responsibilities serves more as an indicator for leadership of the task.

Jean has the most experience with signals and has taken advanced digital signal processing courses, so she will be mainly in charge of creating the signal processing procedure and algorithm, as well as conducting research on EEG and EMG signals. She will work on creating the signal processing structures for the model and will be working in analyzing data to extract the main features that will be used to train the ML model.

Jonathan will be working on the hardware/software interfacing. After obtaining the data from the Emotiv Insight headset, he will build pipelines for parsing the data and build a testbed for training and testing ML models. This includes testing within the data set and a live detection testbench where the model will attempt to detect continuous live inputs. He will then create the acquisition and processing production backend that will communicate to the user frontend.

Wendy will be working on the software end, primarily on the frontend side. She will create the interface and add various features to our project. She and Jonathan will work to connect the frontend and backend components together for the final application.

C. Bill of Materials and Budget

Our project has saved a huge budget by using the inventory’s Emotiv Insight headset. Thus, the other components for the EEG system that we spent on are for Emotiv’s licensing and replacement electrodes. One issue that arose during our process is connecting the wrong power supply to the EMG sensors. Thus, we had to order extra EMG sensors on our second lot in the case of an accident again. We solved the problem by instead of wiring the power source to the breadboard holes, we have cleaned up the circuits and used colored clippers for less prone power supply connection. However, in the future to lessen this risk, we could solder everything permanently on one board with the full circuit system. Other materials are Bluetooth module, batteries, EMG electrodes, enclosure box, and USB-c hub. We were given a budget of $600 and have spent just above $339. See Appendix C for more details.

D. Risk Management

One of the major risks we had is how accurately we can detect the desired features from the EEG sensors. Furthermore, we also risk having an unsmooth and difficult to use interface if the signals are difficult to pinpoint from the user. To combat this, we are already including two EMG sensing devices which only require a simple threshold cutoff for triggering display interactions. These can be rigged to generate continuous movement and be configurable as the primary controllers of desktop movement, especially the mouse, with EEG headset acquired signals used to trigger less commonly used shortcuts. As a minimum target, if our system can reliably detect two user-controllable signals from the EEG headset, we have enough flexibility to define how patterns of those signals will correspond to interface interactions. While this would be convoluted and difficult to pick-up in the beginning, we can guarantee high accuracy on detecting these signals and our system will still be usable, albeit with each action being a unique combination of the 2 binary signals we acquire. Another potential risk we may face is if the same set of features we would like to detect are very different from person to person. We can mitigate this by using the EEG headset and manually calibrating a set of data from a user and only using their unique signals to train the recognition model before it is used. We can also port this as an extra feature of our interface, so the user doesn’t have to navigate to another application before using the system. If the EEG headset acquired signals prove to be extremely unreliable, we planned on pivoting to using a desktop camera and using object detection with OpenCV to recognize facial movements that we can map to user actions. However, we were able to classify EEG data to a satisfying result as stated in the result section therefore we did not move to our back up plan on using the OpenCV.

IX. Ethical Issues

The NeuroController implementation involved a significant amount of data science to collect sufficient samples to train ML models. This involved finding participants willing to record and to share their data for this project. Having such a requirement may pose ethical concerns regarding the misuse of EEG data. However, this is mitigated by the fact that the data collected is not particularly tied to sensitive information. Sampling for winking and blinking does not directly involve disclosing any valuable private information about a user and hence we are fairly confident that the system does not pose a significant risk to users who contributed training data.

Our system also requires licensing from Emotiv to use a commercial Emotiv EEG headset with our system. This requires agreeing to Emotiv terms of service and passing all collected data from the Emotiv application service before it can be forwarded to the NeuroController system. This relationship is less than ideal because now our system requires the trust of a third party service that has access to the same EEG data from potential users. This could adversely affect our users should the third party use the data inappropriately without knowledge or consent. However, to mitigate this risk, it can be noted the stakes involved with the data collected is not high, as EEG recordings with blinking artifacts is not valuable data. Furthermore, our data recordings only involve acquiring data from two of the five EEG sensors from the headset. The other sensors were mostly disconnected and noisy for the entirety of recording, meaning no meaningful data could've been collected from those other streams.

Another ethical issue that may arise from our product is ensuring that our ML models don't bias such that it is significantly more difficult for certain people to use the device than others. To mitigate this, we needed to sample multiple people when collecting EEG data so that we have different user blink signals to generalize our model predictions over and not overfit to a single person's brain signals.

Another issue arising from our product is accessibility for people with different hairstyles. EEG devices have traditionally been more difficult to use for people with a lot of hair or curly hair. Our product, which relies on EEG, suffers from the same issue. To mitigate this problem, we specifically only read the front two EEG sensors attached to the forehead, so that hair will not be an issue. This way the user only needs to ensure the ground node maintains good contact under the hair with the skull for using the device and people with different hair styles can all use the device.

X. Related Work

There are currently no other projects or products that incorporate both EEG and EMG signals in the way that we have planned. A lot of research has been done to study the 2D directional control from motor imagery for people who have fully lost their arm muscles. There are similar projects that take in EEG signals to study and control the cursor on the BCI program [4][5]. However, there is no product that is available in the market yet that allows the integration of the BCI discovery to create a platform on the desktop computer. Moreover, with how expensive a standard EEG cap costs, we chose to use an EEG headset as it is more commercialized. With the trade-off between the price and data acquisition to the spatial and temporal resolution, we are integrating EMG sensors to have precise control data from the shoulder muscle movement instead. We hope that our interface will allow amputees or people with limited physical ability to use computers on a daily basis and without much difficulty. Previous Capstone projects involving EEG or EMG signals include a game called Myorun, from Fall 2020, which allowed users to physically workout their forearm using a game and another game called Traveling Mind, from Spring 2021, which implemented a mind-controlled iOS game. Our project hopes to integrate mind-control and EMG to not just play a game but to control more general computer interfaces.

XI. Summary

The design and plan of work for our project has undergone many iterations. The systems and components have been planned and changed to best fit our design goals, as well as be feasible to implement. Our plan has also changed as we have reevaluated our skill sets and areas of expertise.

Along the course, we found challenges in multiple aspects that hindered our system to be stabilized and run smoothly. This is particularly challenging due to the variance in physiological signals that is the core of our system. Thus, we have created calibration and state transition algorithms to allow a more accurate detection and classification for EEG and EMG signals. Our system was able to reach some of the design specifications. As mentioned, we think that our calibration could be improved for the product to be enhanced in the future. If we could improve the system, we would focus on the calibration for EMG signals and enhancing the classification algorithm for the EEG signals. Having a boarded circuit would have allowed us to have less debugging issues rather than having the circuits on the protoboard.

Even though we will not be continuing this project after this semester, our group will continue to do research and remain interested in brain signal acquisition methods. Overall, our team learned a lot about EEG and EMG signals. We learned that the general shape of EEG signals for actions generally look the same for everyone, but that reading clear EMG signals depends on where the electrodes are placed. We had to collect a lot of information from our subjects because there was not a lot of public data that could be used.

We learned the importance of integration in the development and testing of our subsystems and overall. Even though we designed our schedule to include integration testing and slack time, we still ran into some troubles and had to reevaluate and re-pivot. We also learned the importance of constant communication among team members. There were times when we all faced various setbacks and being transparent about our situations made working together a lot easier.

Lastly, we learned a lot about project management and staying on track. Each week, we looked at our Gantt chart to see if we had completed everything or if we were slightly behind. Planning for slack days helped us spend the last few days working out unexpected issues and bugs without panicking.

Glossary of Acronyms

API- Application Programming Interface
CPU- Central Processing Unit
EEG- Electroencephalography
EMG- Electromyography
ML- Machine Learning
SVM- Support Vector Machine

References

[1] "The latency of Emotiv data streams," EMOTIV, 24-Aug-2021. [Online]. Available: https://www.emotiv.com/knowledge-base/thelatency-of-emotiv-data-streams/. [Accessed: 28-Feb-2022].

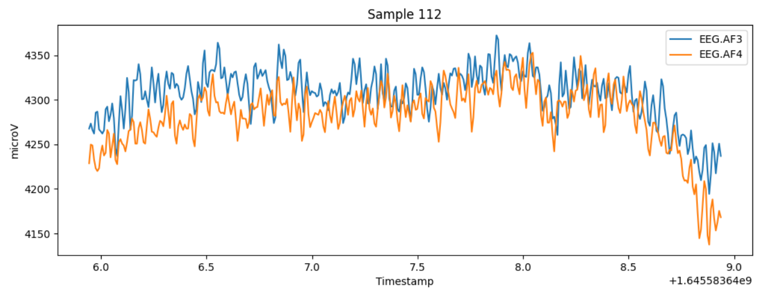
[2] M. B. I. Raez, M. S. Hussain, and F. Mohd-Yasin, "Techniques of EMG signal analysis: Detection, processing, classification and applications," Biological procedures online, 2006. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1455479/. [Accessed: 02-Mar-2022].

[3] H. U. Amin, W. Mumtaz, A. R. Subhani, M. N. M. Saad, and A. S. Malik, "Classification of EEG signals based on Pattern Recognition Approach," Frontiers, 01-Jan-1AD. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2017.00103/full. [Accessed: 03-Mar-2022].

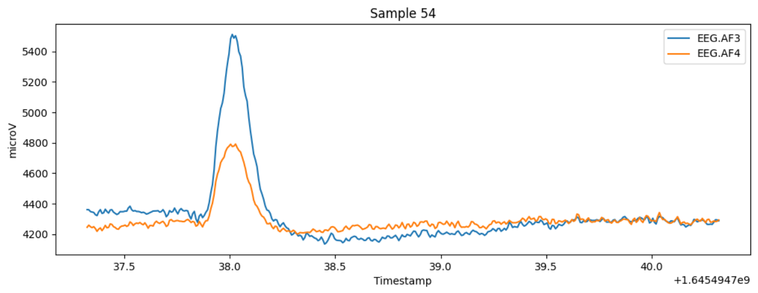
[4] Yuanqing Li, Chuanchu Wang, Haihong Zhang and Cuntai Guan, "An EEG-based BCI system for 2D cursor control," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence) [Online]. Available: https://ieeexplore.ieee.org/document/4634104?arnumber=4634104 [Accessed: 03-Mar-2022].

[5] J. R. Wolpaw, D. J. McFarland, G. W. Neat, and C. A. Forneris, "An EEG-based brain-computer interface for Cursor Control," Electroencephalography and clinical neurophysiology. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/1707798/. [Accessed: 03-Mar-2022].

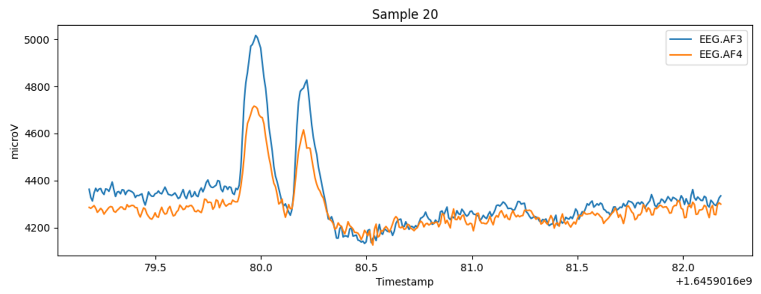
Appendix A



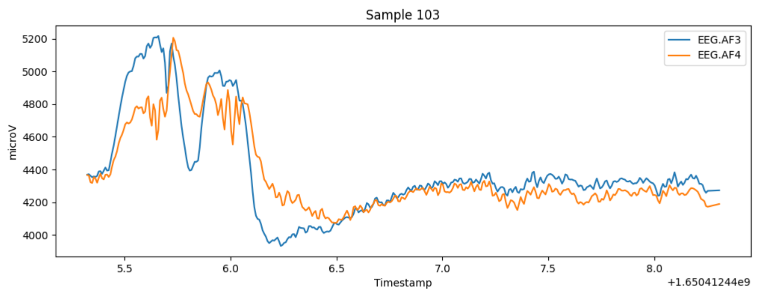
(a) Typical baseline sample with no features



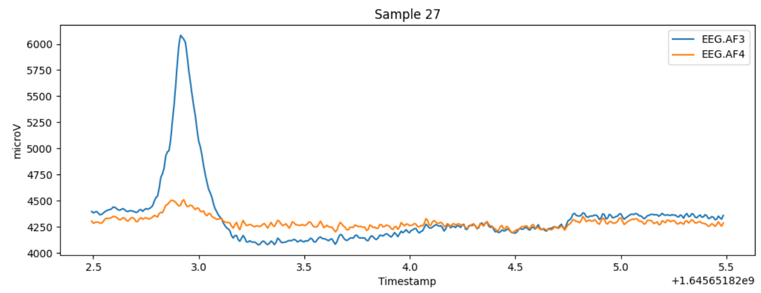
(b) Typical blink sample



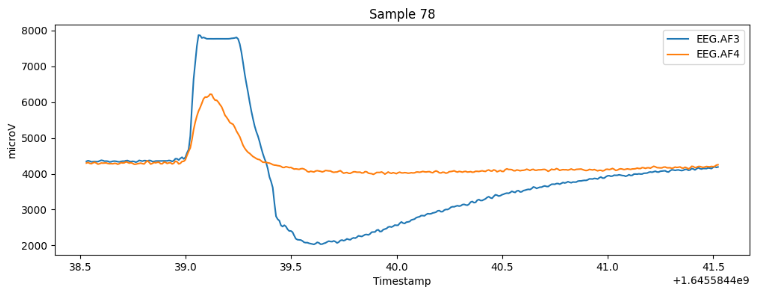
(c) Typical double blink sample



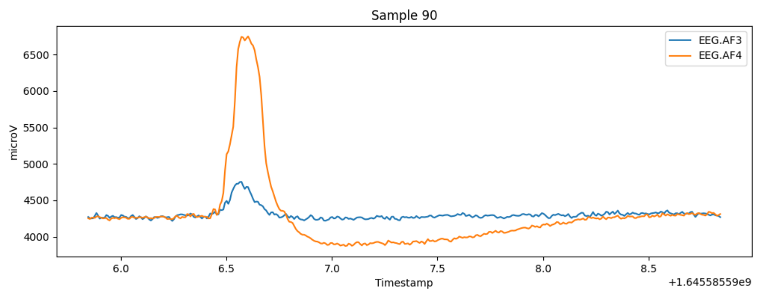
(d) Typical double blink sample, but less ideal



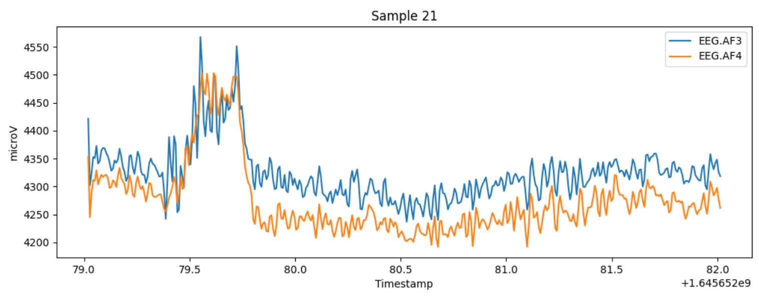
(e) Typical left wink sample



(f) Typical left wink sample, but less ideal



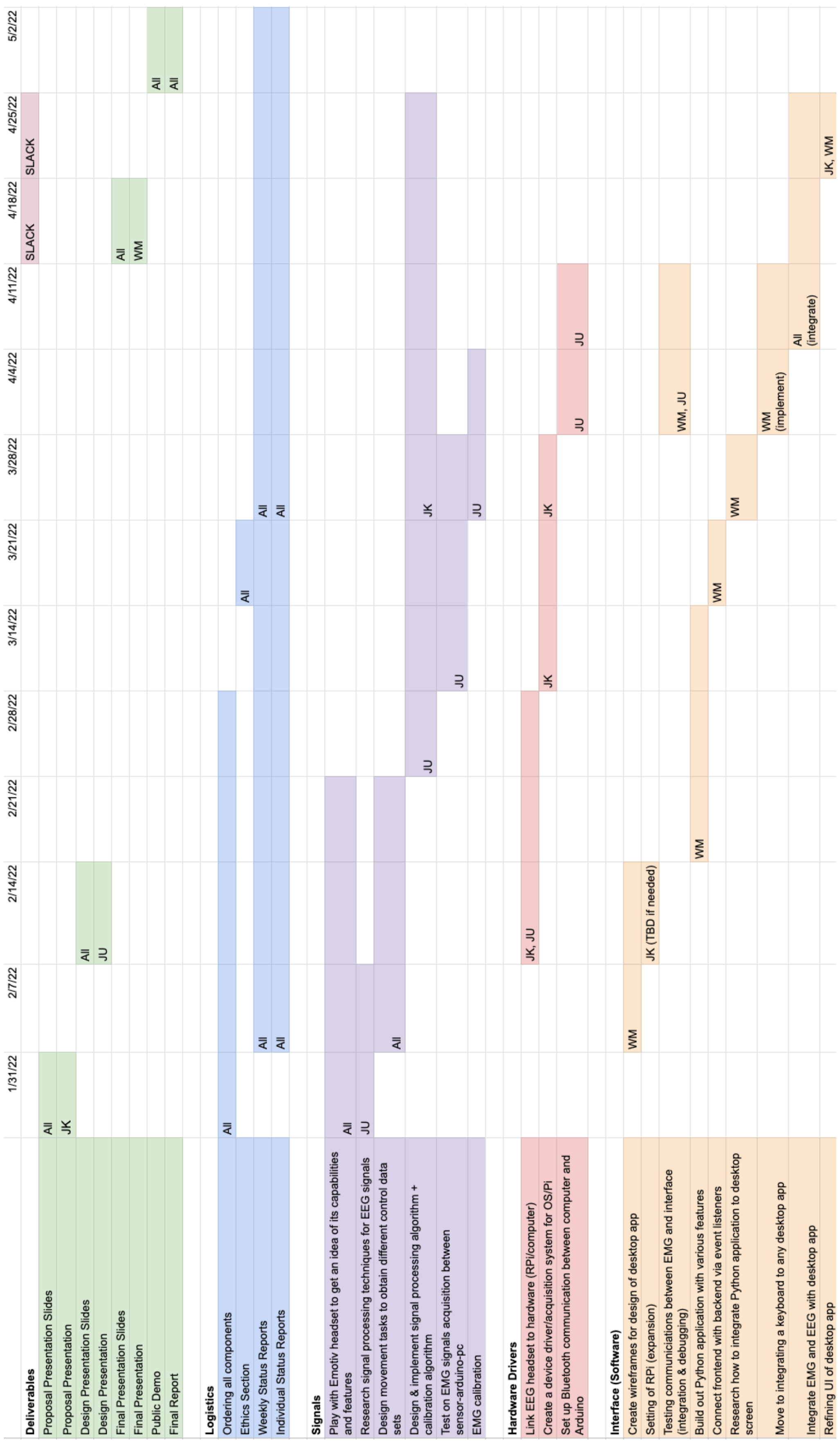
(g) Typical left wink sample



(h) Typical left wink sample, but less ideal

Fig. 9. Visualizations of the sample data from the EEG device that we are using for training.

Appendix B



	1/31/22	2/7/22	2/14/22	2/21/22	2/28/22	3/14/22	3/21/22	3/28/22	4/4/22	4/11/22	4/18/22	4/25/22	5/2/22
Deliverables											SLACK	SLACK	
Proposal Presentation Slides	All												
Proposal Presentation	JK												
Design Presentation Slides			All										
Design Presentation			JU										
Final Presentation Slides											All		
Final Presentation											WM		
Public Demo													All
Final Report													All
Logistics													
Ordering all components	All	■	■	■	■								
Ethics Section							All						
Weekly Status Reports		All	■	■	■	■	■	All	■	■	■	■	■
Individual Status Reports		All	■	■	■	■	■	All	■	■	■	■	■
Signals													
Play with Emotiv headset to get an idea of its capabilities and features	All	■	■	■									
Research signal processing techniques for EEG signals	JU	■											
Design movement tasks to obtain different control data sets		All	■	■									
Design & implement signal processing algorithm + calibration algorithm					JU	■	■	JK	■	■	■	■	
Test on EMG signals acquisition between sensor-arduino-pc						JU	■	■					
EMG calibration								JU	■				
Hardware Drivers													
Link EEG headset to hardware (RPi/computer)			JK, JU	■	■								
Create a device driver/acquisition system for OS/Pi						JK	■	JK					
Set up Bluetooth communication between computer and Arduino									JU	JU			
Interface (Software)													
Create wireframes for design of desktop app		WM											
Setting of RPi (expansion)			JK (TBD if needed)										
Testing communications between EMG and interface (integration & debugging)									WM, JU	■			
Build out Python application with various features				WM	■	■							
Connect frontend with backend via event listeners							WM						
Research how to integrate Python application to desktop screen								WM					
Move to integrating a keyboard to any desktop app									WM (implement)	■			
Integrate EMG and EEG with desktop app										All (integrate)	■	■	
Refining UI of desktop app												JK, WM	

Fig. 10. Schedule

Appendix C

Item	Quantity	Manufacturer	Source	Price	Arrived?
Emotiv Insight Model 1.0	1	Emotiv	18-500 Inventory	$0.00	Yes
Replacement EEG sensors	2	Emotiv	Emotiv	$61.94	Yes
EmotivPRO Student License	1 license ($30/month x 3 months)	Emotiv	Emotiv	$90.00	Yes
EMG Sensors	4	N/A	Ebay	$89.22	Yes
EMG gel electrodes	1 pack of 100	3M	Amazon	$18.50	Yes
9V battery	2 packs of 8	Amazon	Amazon	$11.59	Yes
HC-05 Bluetooth module	1	DSD Tech	Amazon	$9.99	Yes
USB-C hub	2	UNI	Amazon	$27.18	Yes
EMG gel electrodes	1 pack of 100	3M	Amazon	$17.51	Yes
Electric enclosure box	1	LeMotech	Amazon	$13.77	Yes
				Grand Total	**$339.70**

Table I. Bill of Materials