Boston College
Department of Computer Science

Neuroprosthetics: An Investigation into Utilizing EEG Brain Waves to Control a Robotic Arm

By

Jake St. Germain

Computer Science Honors Thesis
May 2015
Advisor: Dr. Robert Signorile

Table of Contents

Abstract

Medical advances have created a society where more people are surviving life-threatening injuries, although at a physical cost. Individuals who are handicapped due to these enormous medical developments are becoming increasingly more common, especially ones who are living with limited motion of their extremities, paralyzed from the head or waist down, or are paraplegic or quadriplegic. Improvements in prosthetic technology have not yet caught up to the medical world, often limiting the options for these individuals to assimilate back into society. This thesis proposes that it is possible to directly connect the brain, through the measurement of EEG waves, to a robotic arm to increase functionality and movement. This union will require the frequency of EEG brain waves and the transfer and manipulation of this information to a robotic arm, all in real-time. This research will utilize the capabilities of the Emotiv EPOC system to record and parse brain waves, which will then be transferred to a Lego NXT robot set that resembles a prosthetic arm. That system simulates a BCI, or Brain Computer Interface, where the brain is able to communicate with an external source, or the Lego arm.

1. Problem

Some of the most impressive technological advances in the 21st century have been in the field of medicine, especially in combat and emergency situations. These developments have extended life for thousands of people, but sometimes it comes at a significant cost. Increasingly this results in veterans and other individuals trying to return to normal lives without complete function of their entire body. As the United States military starts to wind down active operations abroad, the number of amputees among the soldiers will significantly decrease. According to a study done by the Department of Veteran Affairs in conjunction with the National Institute of Health, the average cost projection for a veteran of Iraq or Afghanistan with an upper arm amputation could be more than $800,000 in his or her lifetime.[1] The government will continually be paying out this enormous sum to better the life of the soldiers protecting this country's freedom. Economically, more viable prosthetic solutions will drastically lower this cost, saving the government money in a time when every penny counts.

Prosthetic arm research is continually lagging behind work done on prosthetic legs, mainly due to the assumed impact that a leg amputation has over an arm.[2] It seems that, however, veterans are choosing instead to continually not use their prosthetic arm. They often cite that it just gets in the way, and that it is simply easier to learn tasks without them. Over the years there has not been much improvement over the 'pirate-hook' design, as scientists have instead focused on significantly improving technology for prosthetic legs. Since the number of veterans, and individuals overall, is higher for amputated arm compared to leg amputees, the need for an enhancement in design is quite noticeable. Current designs in leg prosthetics mimic motions, muscles, and movements in actual legs, so a similar idea could be shifted to imitate an arm. This could result in a greater range of motion, more flexibility in hand movement, and the outcome of actually having four fingers and a thumb instead of a claw design.

One of the most difficult parts of receiving a prosthetic arm is the usability and functionality of it. This greatly differs from the function of a leg, as you lean with your body

[1] Blough, D., Hubbard, S., McFarland, L., Smith, D., Gambel, J., & Reiber, G. (n.d.). Prosthetic cost projections for servicemembers with major limb loss from Vietnam and OIF/OEF. Retrieved May 5, 2015, from http://www.ncbi.nlm.nih.gov/pubmed/20803406

[2] Martino, H. (2014, December 18). War Vets Cast Aside Costly Prosthetic Arms, Citing Usability. Retrieved May 5, 2015, from http://www.nbcnews.com/news/investigations/war-vets-cast-aside-costly-prosthetic-arms-citing-usability-n271211

weight on a prosthetic leg. The wearer will eventually adjust their balance subconsciously to fit the prosthetic so that he or she can walk again. Since there is little movement in prosthetic arms, the usability of them comes into question, hence why more veterans are just stopping wearing them. This can easily be answered by modifying the arm to respond to stimuli that the wearer contributes. Recently, scientists have been researching the possibility of directly attaching a prosthetic arm to the nerves in the amputated arm through different sensors and electrodes.[3] Since the brain is still able to send signals down that path, a prosthetic arm that connects directly into those nerves would be able to respond immediately to the brain's inputs. This would allow the wearer to control more fine motions of the arm, such as grip strength and locomotion. While this research is extremely promising, it is fairly expensive, so more cost-effective measures need to be investigated.

Quadriplegics are now also entering the scene, mainly due to increased medical efforts to save those with possibly mortal injuries. One of the difficulties with designing prosthetics for these individuals is that many do not have much control over their entire body, including cases of people with limited head motion. An extreme case is seen in patients with ALS, or Lou Gehrig's disease, as their brain is still able to send out signals to the motor neurons located in the spine and other muscles, but these motor neurons start to break down. ALS usually results in full paralysis before death, even though the brain is still able to normally function. There is no indication of any kind of prosthetic advancement for these individuals, as it could dramatically increase their quality of life. A potential answer for this conundrum is to directly link prosthetic movement to the signals that the brain sends or electrical impulses in the brain, especially since the brain is still intact in ALS patients. This could allow for a more comfortable lifestyle for ALS patients, along with the possibility of becoming a complete prosthetic and attempting to overcome the disease and return to a normal life.

[3] Laursen, L. (2014, February 5). Amputee Successfully Feels Prosthetic Grip Strength Via Arm Electrodes. Retrieved May 5, 2015, from http://spectrum.ieee.org/tech-talk/biomedical/bionics/sensitive-prosthetic-hand-gets-a-grip

2. Overview

2.1 EGG

EEG, or electroencephalography, is the measuring of a slight electrical signal that is given off through your brain.[4] The brain is made up of billions of neurons that fire when a person thinks. This firing results in an action potential within the neuron, which can impact other neurons that are connected to the initial one. This series of action potentials is called volume conduction, and the result is every activity that occurs in our brain. A participant would place an EEG device, which usually consists of a hat or helmet that has numerous electrodes on it, on their scalp. An electrode is basically a piece of metal that has electrons on the tip of it, allowing it to conduct electricity. While individual action potentials are actually too small to measure, an EEG device measures the millions of neurons that are in a similar spatial region, giving an output that is an electrical reading. This electrical reading is usually conditioned over multiple minutes in conjunction with new stimuli to the patient. This allows the examiner to notice different levels of electrical output of the brain and how they correlate to different motivations, which has directly resulted in most of the scientific knowledge of the brain.

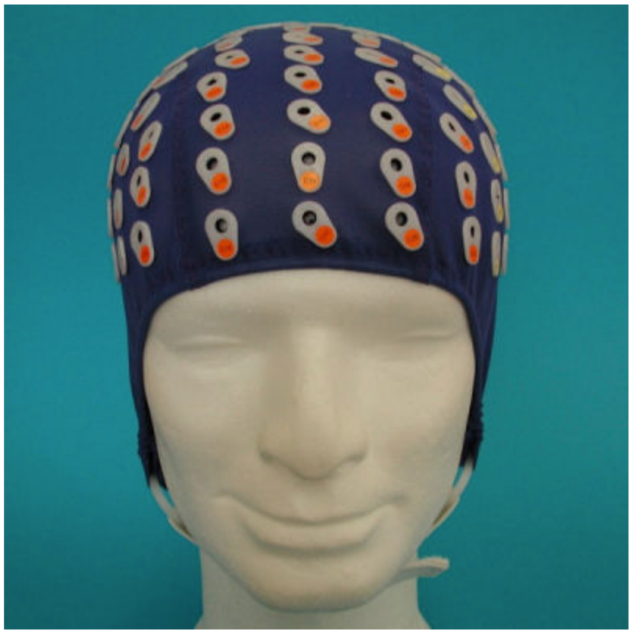


Figure 1: An EEG cap

[4] Campellone, J. (2014, February 10). EEG: MedlinePlus Medical Encyclopedia. Retrieved May 5, 2015, from http://www.nlm.nih.gov/medlineplus/ency/article/003931.htm

An EEG will measure a variety of different brain waves, and the examiner will usually focus on a few specific ones in order to diagnose an issue.[5] The waves are subdivided based on frequency into four major categories: delta, theta, alpha, and beta. Delta waves have frequencies up to 4 Hertz, and are typically seen in adults or children who are sleeping. Theta waves have frequencies between 4 and 7 Hertz, are also seen mainly in children, but are indicative of a relaxed or meditative state. Alpha waves have frequencies between 7 and 14 Hertz, and are only seen in people who are relaxed with their eyes closed, as they tend to disappear when the participant starts to concentrate or open their eyes. Beta waves have frequencies between 14 and 30 Hertz, and are linked to motor activities and general thoughts, but are also signs of anxiety or depression in some patients, depending on the region of the brain that emits them.

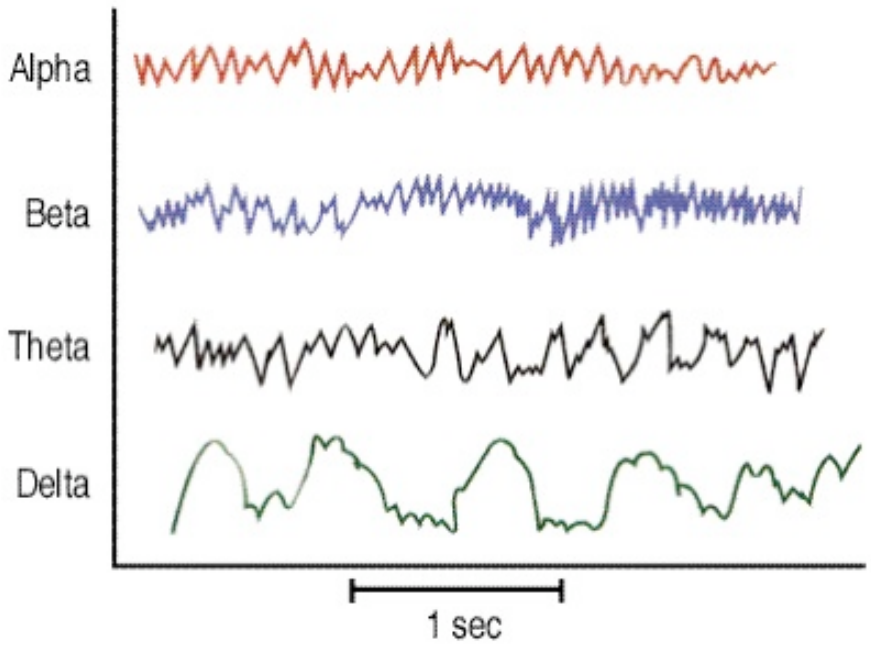


Figure 2: A chart of the different EEG waves

The EEG is one of the most important medical tests. It is the most effective test for diagnosing epilepsy, along with a number of other impactful diseases that affect the brain, like Alzheimer's or brain cancer. EEG is also useful to diagnose disorders associated with sleeping, as it is one of the few machines that will receive important data will the user is asleep. The

[5] Electroencephalogram (EEG). (n.d.). Retrieved May 5, 2015, from http://www.hopkinsmedicine.org/healthlibrary/test_procedures/neurological/electroencephalogram_eeg_92,p07655/

EEG's most important characteristic is that it is non-invasive and harmless; all that is required of the tester is that they sit still for a short amount of time wearing a modified headpiece. This gives the EEG preference over other tests, such as MRIs or a PET scan, which often require the patient to be in uncomfortable positions in an enclosed area, which can frighten people who are claustrophobic. EEG is also important in research due to its low cost, quick set up and test time, and ability to work on almost anyone. Doctors will often use EEG to confirm the presence of paralysis in a patient and to find where the initiation of it occurred, allowing for a more accurate diagnosis to be given. The data from the EEG is also given in real time to the doctors, which allows for quicker decisions to be made about patients and their health, especially when compared to other medical examinations such as blood tests, which usually need to be sent out to labs.

2.2 BCI

BCI, or brain computer interface, is a system of devices that allows the brain to communicate with some outside source.[6] The brain generally just communicates with other parts of the body through various electrical signals, which will get transferred to wherever is necessary to complete the task. The BCI field is narrowing focus mainly onto the subject of neuroprosthetics, which are prosthetic limbs that are linked to the mind. BCI is divided into two different categories based on the device being used: invasive and non-invasive. Invasive BCI includes tools that are usually implanted on top of the brain, requiring delicate surgery and a heavy amount of research before human application. While this field is the most promising due to the direct communication between the sensors and the brain, along with the longevity of the programs, the gigantic cost and risks associated with it make it incredibly hard to reproduce for a large population. Non-invasive BCI includes devices that have minimal impact on the user, such as EEG and other electrode placement procedures. BCI works through four main actions. The first is signal acquisition, which requires the device, wherever it is, to pick up and record any electrical output given off by the brain. The second action is feature extraction, which is a form of parsing and data mining, which requires the computer to distinguish between usable electrical

[6] Shih, J., Krusienski, D., & Wolpaw, J. (2012, March 1). Brain-Computer Interfaces in Medicine. Retrieved May 5, 2015, from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3497935/

signals and noise that is not relevant to the data. The third action is feature translation, which is based on an internal algorithm that decodes the changes in electrical signal and creates data that is useful to the user. The fourth action is device output, which simply consists of the travelling of the useful data to some machine, such as a robotic arm or wheelchair, which will allow it to perform an action. Usually the device will then incorporate a feedback loop to give data back to the user to increase performance.

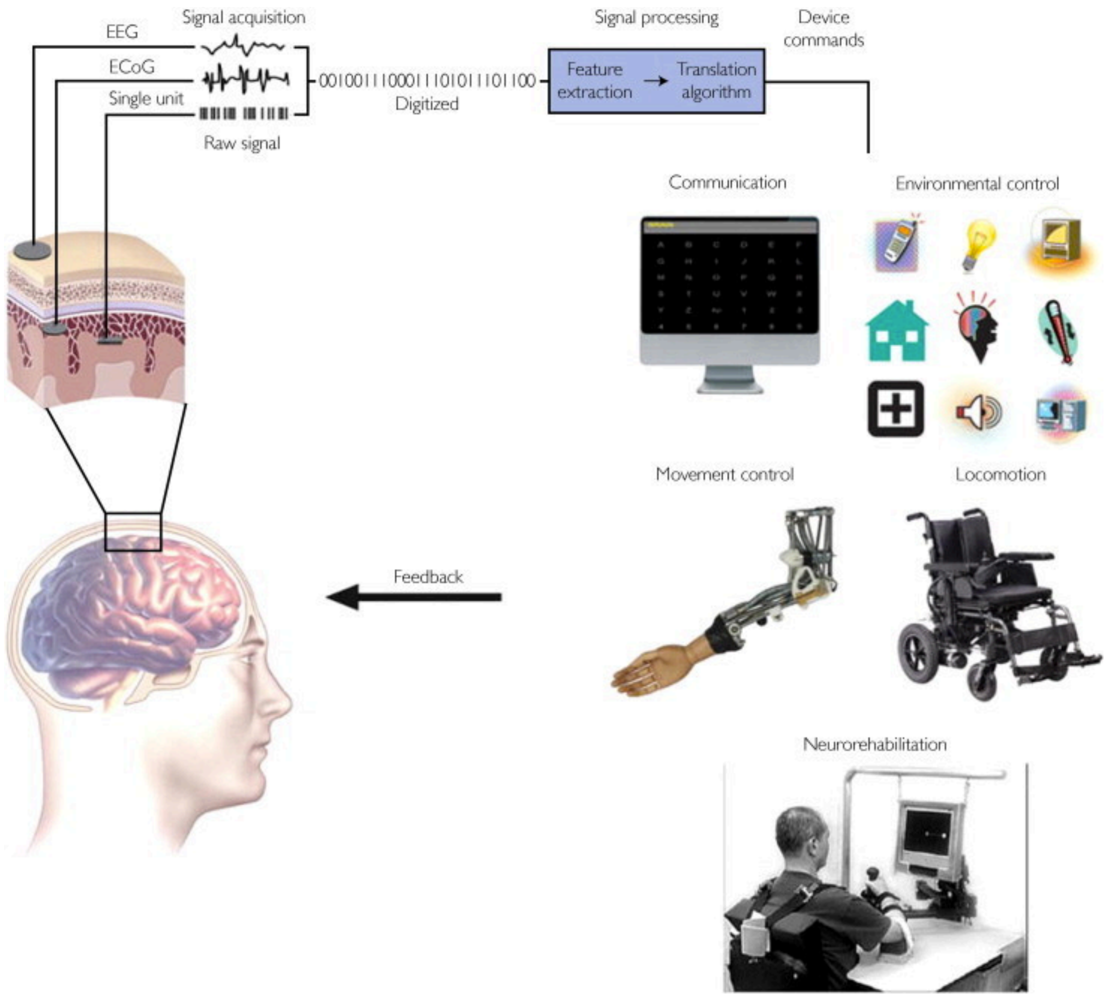


Figure 3: Common BCI protocol

BCI, when combined with EEG, provides the most promising research for people who are either disabled or are amputees. BCI with EEG input is the most investigated combination because of its non-invasive qualities, along with relatively low cost and high patient recruitment rate. The breakthrough study that linked EEG with BCI is the P300 Speller algorithm by Farwell and Donchin, which scientists have been studying ever since the original paper came out in 1988.[7] This algorithm creates a situation where the brain is able to spell out words based on a flashing 6 by 6 matrix of letters and numbers. When the row or column of the letter that the patient wants next flashes, the EEG device reads this as an event, narrowing it down so that one letter is chosen. Patients with debilitating diseases, such as ALS, have successfully used this method to spell out words with the EEG device on their head, although it does take some time to initially accurately process positives from negatives.

2.3 Emotiv EPOC

Emotiv Systems is a company that almost exclusively deals with the connection between BCI and EEG, combining the two systems to create the EPOC device. The EPOC device has 14 electrodes attached to a central piece that the user places on their head. It then is able to measure the electrical signals picked up at each electrode, exactly like an EEG, and then wirelessly transfers them to a computer so the user can utilize them. The user connects each electrode to a pad that is soaked in saline solution, which assists the electrode in picking up electrical activity by increasing conductivity. The device must be placed on a specific portion of your skull for full effect, and this placement is easily outlined in the instructions given with the item. A picture of the Emotiv EPOC is below:

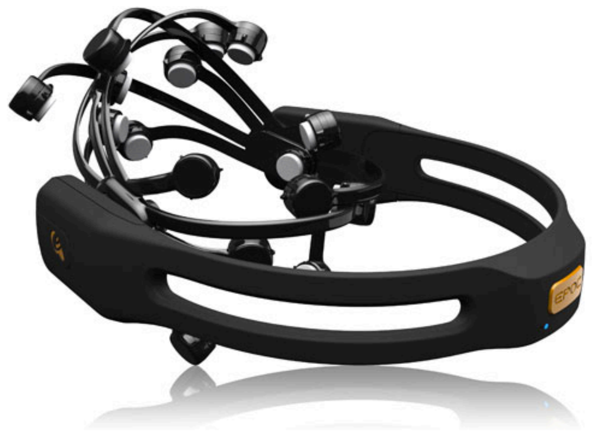
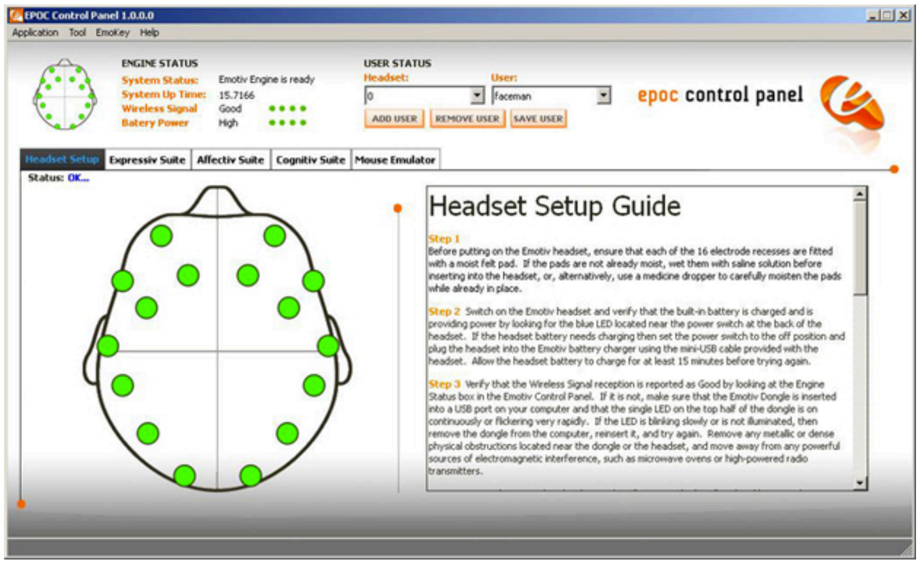


Figure 4: The Emotiv EPOC headset

[7] Krusienski, D., Sellers, E., Cabestaing, F., Bayoudh, S., Mcfarland, D., Vaughan, T., & Wolpaw, J. (2006). A comparison of classification techniques for the P300 Speller. Journal of Neural Engineering, 299-305. Retrieved May 5, 2015, from http://iopscience.iop.org/1741-2552/3/4/007

Emotiv created a few different programs that show the full usability of the EPOC system. The most important platform created is the Emotiv EPOC Control Panel, along with the Research edition that the user can purchase separately. This control panel allows the user to access the different suites that are offered by Emotiv, which includes the Expressiv, Affectiv, and Cognitiv groups. The Expressiv suite highlights the different facial reactions that the EPOC can pick up. For example, the device picks up motions like looking left or right, blinking, and clenching teeth or smiling. The user can program something to happen when the EPOC detects one of these movements, such as a specific key being pressed. The Affectiv suite measures the different emotions that are associated with the EEG. These sensations are excitement, calm, engagement, disinterest, and meditation. Again, the user can access any of these emotions and successfully program so that something happens when the device measures them. The Cognitiv suite is probably the most important package within the control panel. It allows the user to train certain actions, such as lift, pull, or push, along with training a neutral action for a base line reading. This package allows the user to theoretically think 'push,' have the EPOC device detect that thought, and then have a program make some external stimuli react to it. One current limitation with the control panel is that the user is only allowed to train up to 4 actions at once not including the neutral action, but the user can avoid this by opening up multiple control panels at the same time. A final part of the control panel is the mouse emulator that the user can utilize. This program taps into the built-in gyro in the EPOC device, and allows the operator to control the mouse simply by tilting or moving his or her head.



ENGINE STATUS
System Status: Emotiv Engine is ready
System Up Time: 15.7166
Wireless Signal: Good
Batery Power: High

USER STATUS
Headset: 0
User: faceman

Status: OK...

Headset Setup Guide

Step 1
Before putting on the Emotiv headset, ensure that each of the 16 electrode recesses are fitted with a moist felt pad. If the pads are not already moist, wet them with saline solution before inserting into the headset, or, alternatively, use a medicine dropper to carefully moisten the pads while already in place.

Step 2 Switch on the Emotiv headset and verify that the built-in battery is charged and is providing power by looking for the blue LED located near the power switch at the back of the headset. If the headset battery needs charging then set the power switch to the off position and plug the headset into the Emotiv battery charger using the mini-USB cable provided with the headset. Allow the headset battery to charge for at least 15 minutes before trying again.

Step 3 Verify that the Wireless Signal reception is reported as Good by looking at the Engine Status box in the Emotiv Control Panel. If it is not, make sure that the Emotiv Dongle is inserted into a USB port on your computer and that the single LED on the top half of the dongle is on continuously or flickering very rapidly. If the LED is blinking slowly or is not illuminated, then remove the dongle from the computer, reinsert it, and try again. Remove any metallic or dense physical obstructions located near the dongle or the headset, and move away from any powerful sources of electromagnetic interference, such as microwave ovens or high-powered radio transmitters.

Figure 5: The Emotiv EPOC Control Panel

Besides the control panel, Emotiv also included other programs that help with the user in understanding how the device works. The most useful of these programs is a video game created by Emotiv called Spirit Mountain, which Emotiv created in the language Unity. This game fully integrates many different aspects of the device, specifically the gyro and the Cognitiv suite. Spirit Mountain focuses on a person training different actions, such as pull and lift, while completing a mission. It provides a fun and interactive way to initially learn how to use all elements, along with giving a creative way of introducing this material. Another program included in the full package is called Neuro Mousecontrol. This application allows the user to control every function of a mouse, including clicking, moving, and scrolling. The user accomplishes this without having to train a single action, as this program relies solely on the Expressiv suite built into the application. For example, the user could control the mouse with the gyro and tie together clicking and blinking, creating a situation where the operator does not need to train any thoughts or actions to fully utilize the mouse. A third useful program is the Brain Visualizer. This platform shows the user which parts of his or her brain have increased electrical activity due to different thoughts. That gives a good demonstration of how distinctive sections of the brain fire for specific ways of thinking. The Research edition is the final addition to the package provided by EPOC. This add-in gives the user more capabilities, as it provides the user with the necessary DLLs (Dynamic Link Library) to write programs for themselves. It further gives the user an application called the Testbench, which shows the electrical reading of every node in real time. This makes it so that the user can specifically see which electrodes are seeing higher frequencies for different thoughts, which helps in understanding how the Cognitiv suite actually works.

2.4 NXT Robotic System and leJOS NXJ Software

Lego created a string of programmable buildable robots, and the second in the series was called the Lego Mindstorms NXT. The second system released in that chain was the Lego Mindstorms NXT 2.0, and it was the newest product until 2013 when Lego released their third edition. The NXT 2.0 set includes an intelligent programmable brick that houses a microcontroller. This brick is the central hub of all action of the robot. It has an LCD screen that can display text, 4 input ports that can connect to different sensors, and 3 output ports that can link to motors. The Lego Company includes these sensors with the package, providing instruments like an ultrasonic sensor, a light sensor, and a touch sensor, among others. The brick has access to Bluetooth technology, allowing it to communicate back to a computer or to other robots in real time. The brick also has a speaker, allowing the user to program the robot to emit sounds when necessary. Included with the brick is over 400 pieces of Legos, all of which the user can utilize differently. While the user can actually create very simple programs on the brick by itself, the operator needs a computer to construct more complex applications for the robot.

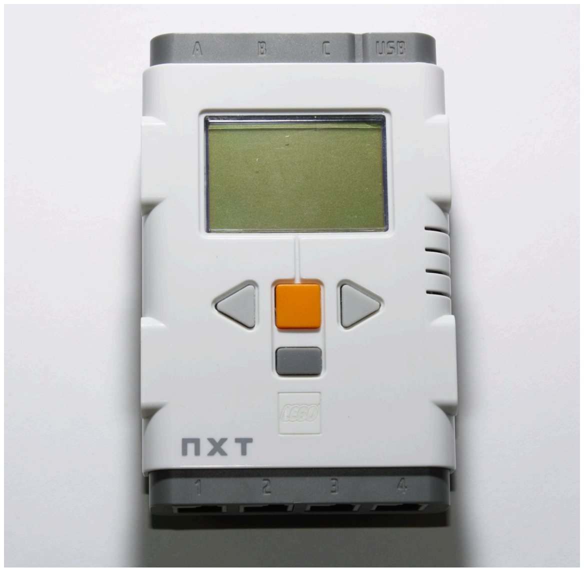


Figure 6: The NXT Brick

The leJOS NXJ software allows the NXT robots to run Java programs. The leJOS program is essentially a Java Virtual Machine that the user imports onto the NXT bricks, allowing the operator to write Java programs, upload them to the brick via USB connection, and successfully run them. This allows the capabilities of a high level language, like threading, arrays, and recursion, to impact the programming of the robot, allowing for the creation of very complex applications. Some of the created classes provided by leJOS include map creation, object detection, suites for all of the sensors, and navigation classes. In addition, leJOS also included many other standard Java and Javax classes, such as Bluetooth support, input and output standards, and support for sockets. The leJOS company also created an API (Application Program Interface) that accompanies the software, outlining the vast number of classes that the user could utilize for programming needs.

3. Implementation – Part 1

3.1 Introduction

My first goal for this thesis was to create a system that a disabled user could utilize that simulates various computer activities. The user would have to employ the various opportunities that the Emotiv EPOC provides. The program would have to be self-sufficient so that all training, moving of the mouse, and saving of data be done within the application. The overarching idea behind this was to simulate a computer that an amputee or severely disabled person could use in place of a normal computer.

In order to accomplish this goal, I first looked at the provided programs and classes from Emotiv, specifically the ones written in Java. They provided 7 important classes that included most functionality to accurately connect the headset to a computer and read data from it. The first, and probably most important, class is titled "Edk.java." This class provides the backbone behind all Emotiv programming in Java, creating emulators that the user utilizes to store information, different events that can help in error processing, and the structure of the different electrodes and position of the headset on top of the head. Another provided class is titled "EdkErrorCode.java," which simply provides an enumerated list of error codes that the headset could throw. Included in this list are important events such as the gyro not being calibrated, the operator not loading the user ID correctly, and if something is wrong with the data being sent from the headset. The next supplied program is called "EEGLog.java." This is the first program that actually utilizes the main class, as the previous two were just defining different variables and conditions. This class connects to the headset, collects data from it, and then displays the collected data in the console. It applied the previous two classes and the different instances in each of them, giving the user a simple and succinct method of viewing the different electrical signals read by the headset.

The next 3 programs focus on fully developing the Cognitiv, Expressiv, and Affectiv suites that the headset is designed to measure. The first program of the mentioned triad is called "EmoProfileManagement.java." This important class gives the operator the ability to create, save, and load user profiles. This not only allows multiple users to create different profiles on one computer, it handles the possibility that one client can stop using the device one day, and pick it up the next without having to retrain all of the functions in the future. The next class is called "EmoState.java." It holds all of the enumerations of the data for the 3 described suites. This program provides the backdrop for the capacity to transform the data picked up by the headset into information that can make sense to anyone using it. The third class in the triplet is simply the practical form of the previous one, titled "EmoStateLog.java." Exploiting the created data in the preceding class, this program simply connects to the headset, and continually updates and prints the emo state until the headset is turned off. It will print certain actions like blinking or looking left or right, along with printing information about the levels of boredom, excitement, and any trained actions.

The final provided program encompassed all of the previous classes, combining them into a practical GUI (Graphical User Interface) that anyone with a headset could instantly use. Coincidentally, it is named "MainGUI.java," and is about 300 lines of code. This GUI allows the user to create, save and load profiles so that multiple people can use the same computer without having to go through the initial training every time. It allows the operator to train 3 actions: neutral, push, and lift. Training comprises of consistent and constant thinking of a motion related to that word. For neutral, this only involved of trying to think of blank space and relaxing the mind. For movements, the training consists of continually thinking of that motion being applied to something, such as a theoretical ball of fluid located in middle of your brain. As subsequent actions are added, this thinking became more difficult. As will be discussed further in the next section, different methods of thinking were explored in order to obtain the most accurate and repeatable results. The program also prints out, in the console, the name and power of the action that the user can trigger by thinking of the specific outcome. These 7 provided programs provided the basis to start my exploration of creating a simple application.

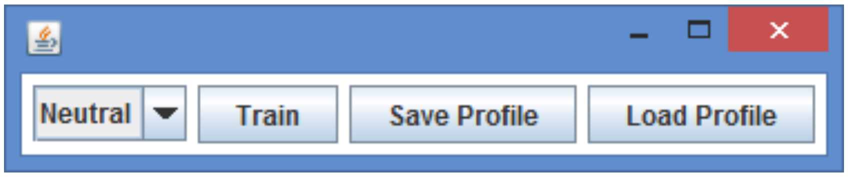


Figure 7: Provided Emotiv GUI

3.2 My Applications

I first decided that in order to simulate a simple computer for a disabled person, the principal issue that must be addressed is the mouse. Noticing that the Emotiv Control Panel has a built in mouse emulator, I first wrote a simple program that creates a button that displays a message when clicked. The mouse emulator within the Control Panel allows you to control movement of the mouse with the gyro built into the headset, but has no capability of actually clicking. Through the Expressiv suite in the Control Panel, the user can address that issue by linking a certain movement, such as blinking or smiling, with clicking the mouse. Furthermore, the Neuro Mousecontrol program written and provided by Emotiv offers both of these tasks in one. Using either one of these programs, the user is able to successfully connect blinking to a simulated click of the mouse. My next course of action was to recreate this entire procedure, just without the assistance of the created applications provided by Emotiv. I was able to write a relatively small program that fully utilized the gyro to control mouse movement, along with the Expressiv suite to associate blinking with the clicking of the mouse. Combining this with my previous simple example of pressing a button that displayed a message allowed me to test the program. Being able to control both the mouse movement and click was deemed successful, and I accomplished the first step to creating a simple computer simulator.

My next task was to simulate the function of the keyboard, or in some way allow the user to input letters and numbers just like in a normal computer. My initial thought was to create each letter as a specific action within the Cognitiv suite, allowing the user to think of a specific letter and then it be decoded and displayed on the screen. In order to do this, I had to alter the EmoState class that Emotiv provided, creating my own enumerated variable that was the alphabet. I then changed the GUI already offered by Emotiv to fit my criteria. I first made it so that the user only had to train 5 actions plus the neutral state, just to give me a sense of whether or not this system of thinking a specific letter and it appearing would actually be possible. A few problems became very apparent when I introduced this type of logic and ran the program. The most obvious issue was that the Emotiv GUI is only able to train 4 actions, not including the neutral state. Although this was apparent in the Control Panel, there was no obvious evidence in the code that this limit existed. Emotiv knows of this limit and is actively pursuing a hardware workaround, while I would pursue a software workaround the issue. The second issue that became blatantly obvious was the complexity of thinking of a word or letter. It is extremely difficult for a user to keep a constant and consistent thought about one specific letter or word in their head for up to 10 seconds. Usually a user will think the word over and over, but this in fact only produces a wave of different data, with the important changes being at the crests. This problem results in training data that is erratic across the time span, often causing an unpredictable method at which the user triggers an event for a letter. Cognitive science explains this phenomenon, as the electrical signals used to think of a word are only available for the brief time when that word is thought. The Emotiv EPOC relies on a constant thought, which is why movements such as push or lift are the actions that the user can train. The third problem that I noticed while trying this method was that the Emotiv EPOC system recorded electrical signals in a way that makes it incredibly difficult to isolate one instance. For most thoughts, there is a gradual increase then decrease in the brain waves associated with the idea, as the electrical signals start to appear, become stronger, and then gradually die down as the concept stops. Training the device to output the letter "a" when the user thinks of that letter results usually in a long series of that letter being printed out due to the nature of brain waves. These issues slightly limit the possibilities for actually creating a simulated computer application, but do not kill the idea.

To employ the idea that training the brain for an action in the Cognitiv suite generally only works for movements, I then went back to the idea of the mouse. Using the gyro to control the mouse was fairly set in stone; however, some people with limited ability in controlling blinking or in blinking at all would have difficulty with the clicking feature. To click a mouse is technically a motion, so I then set up the preparation to only train two actions: neutral and click. Once the user trained this action, they could control the location of the mouse with their head, and manage the clicking with their brain. The only limitation created by this method is that the user would initially need to click on the 'Train' button in the GUI to physically train both of the actions. This constraint can actually be circumvented, however, through the use of automation, meaning that as soon as the application would start up, training would begin. This idea is only necessary in extreme cases where the user is physically unable to click, but it may be advantageous for some.

With the knowledge that the user could fully control the mouse with their head, the next part of this section involved building some sample applications that resembled real tasks done on a computer, just simplified. One of these tasks is email. Sending and receiving emails is done by almost anyone that has connection to the Internet, and is probably the most common form of communication today. Sending an email requires various different parts: the email address to which you are sending, a web service that sends emails, and a return address. Most emails usually have a subject and a body as well, although both of these are technically optional. To create the email sender application, my first task was to create a keyboard that the user could control with the brain controlled mouse. It would need preferably to look like a normal keyboard, and have the necessary keys for email addresses, such as '@' and periods. I created this by simply using a 2-D array of strings, each of which contained a different letter, number, or symbol, which I then placed into a 2-D array of JButtons. The operator would be tasked with typing three different things: the recipient's email address, the subject, and the body. The user could access each of these through a drop down menu. I combined that menu with keyboard and displayed everything in one frame, so that the design was simple yet effective. The operator with the headset on would then need to move the mouse, with his or her head, over the key that he or she wants to press, and then think in the same manner which he trained the thought 'click.' If successful, a letter would appear in the text area, and the user could move on to the next letter. Once finished with a specific section, the user would hit submit, the program would save that text, and the user could choose a different option to continue the email. Once the operator submitted all three sections, he or she could hit the send button, and it would be sent to whatever email the user inputted.

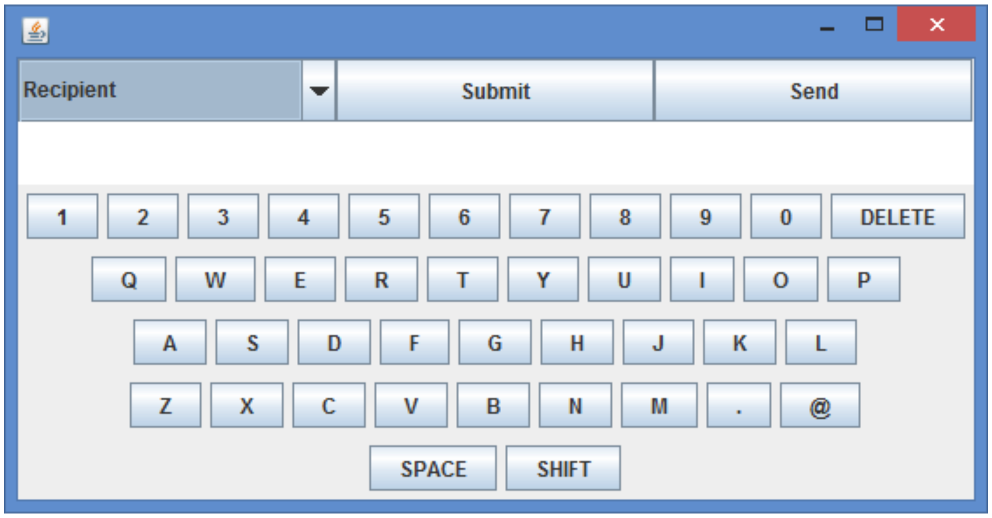


Figure 8: Email Sender

To make this entire application seem more usable just like a computer, I added a document typer in addition to the email sender. This program took many parts of the email sender, such as the keyboard, and simply replicated it. The new addition was that the user could save the document to wherever he or she wanted, giving him or her freedom to work on typing more than once. While I only added these two applications just to show the functionality of the device, I certainly could have added more complex programs, such as an Internet browser, so that the simulation was more realistic. I felt, however, that I discovered the main strengths and weaknesses of the device, which was the original intention behind this first stage of implementation.

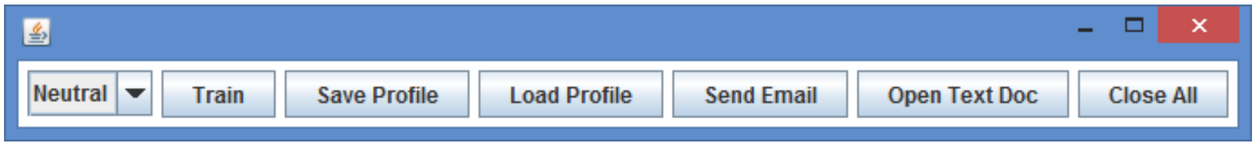


Figure 9: Finished GUI

4. Implementation – Part 2

4.1 Preparation

My main purpose in researching these topics was to try and make more headway on a solution for improved neuroprosthetics; a promising field that could impact countless lives across the world. In order to actually see how an amputee could use the Emotiv EPOC headset, I first had to build a robotic arm that which the headset could give input. To create this arm, I used instructions that someone already created for a robotic arm built with Legos from a NXT set.[8] This arm would have six degrees of freedom. It could move left and right, which simulates the arm connected at the shoulder, but just moving parallel to a flat surface. The arm could move up and down, which mimics the connection between the upper arm and the forearm at the elbow. The final two degrees of freedom were in the robotic hand, which could open and close as well. Having three fingers resulted in enough grip strength to hold one of the plastic balls included in the Lego set, along with being able to move it, and subsequently place it down as well.

I wrote three test programs to nail down the specifications of the arm before I started to work on linking the headset with it. The first experimental program was very simple, as I designed it simply to see which directions on the motors corresponded with the different motions. I discovered two critical things through this straightforward test. The primary idea I learned is that the default speed for the motors would often result in balance issues for the arm, creating issues with overextension and fluidity of motions. After some tests I deduced that the optimal speed was actually half of the default speed, and I would use this measurement for the rest of my trials. The second thing I learned was rather obvious, but vital nonetheless. I discovered that positive on the A Motor is left, with its opposite being right, positive on the B Motor is up, with its opposite being down, and positive on the C motor being open, and the opposite being close. Determining these directions and relating them to the motors was absolutely critical, as progress would have halted at that moment.

The second test program I wrote was to discover how to fully use the Bluetooth capabilities located both within the NXT robots and my laptop to communicate in real time

[8] Parker, D. (n.d.). NXT Robot Arm. Retrieved May 5, 2015, from http://www.nxtprograms.com/robot_arm/steps.html

between the two machines. In order for full functionality on the robotic arm, information and data must be sent to it as soon as the headset reads it and transfers it to the computer. This means there must be some form of communication between the computer and the robot, which generally does not happen with most robotic programs. Many of the previously created robots functioned autonomously or communicated with another robot, but simply once you uploaded the program, it would start and complete it on its own. Bluetooth communication is effectively enabled through leJOS, which creates an easy way to connect the computer with the robot. The only thing this test program accomplished was sending numbers from the computer to the robot, printing them, and sending them back. While this feat was small, it was important to enable real time communication for data sending.

The final test program that I created before starting to link the robotic arm with the EPOC headset was a remote controller for the arm. This controller would set the backdrop for controlling the arm using my thoughts, only I would be clicking a button instead of thinking a thought. The controller could control all six functions of the arm: left, right, up, down, open, and close. The main idea that I needed to determine was how to send that information to the arm, along with what that data would make the arm do. At first, I decided that the entire word displayed on the button could be sent over Bluetooth, such as sending "Up." I, however, quickly realized that this did not always produce the same result, as sometimes the arm would follow the instructions and other times it would do nothing. It turns out I was having issues in wrapping the texts in Bluetooth when sending them, which became over-complicated in the unwrapping by the robot. I then switched over to a numbering system, as numbers had previously worked in being sent over Bluetooth between the two machines. I gave each motion a different number, and hard-coded those numbers into the program uploaded into the robot. For example, if I was to click the "Left" button, the program would send the number "1" over Bluetooth, which the robot would read and then move the arm left accordingly. The next issue I had to face was how far the arm should actually move when it received this signal. I knew that the arm could receive many signals at once from the Emotiv device, which I experienced in my previous trials with trying to think of one single letter. I initially wanted the arm's movement to correspond with the power of the thought, but then I decided that to simplify things, it would just move a specific amount every time. This quantity was 30 degrees of rotation, which provided a small yet noticeable change in the arm's location.

4.2 Operation

After building the prerequisite programs to test the arm, I then set to creating the programs that will interconnect the Emotiv EPOC data with the signals sent to control the robotic arm. The first task was to create my own data structures that fit within the Emotiv format, so that the headset would correctly read the data and transfer the right information to the laptop. Using the configuration generated by Emotiv in their Cognitiv suite describing data for different actions, I produced similar arrangements for the data I wanted. This entailed changing the "EmoState.java" program that Emotiv offered to add variables relating to the different motions of the arm, along with the different enumerations for each motion and the conversion back-and-forth between number and word. This made it so that the user would be training the actions of the arm directly. Furthermore, once I introduced these new structures, I had to modify all of the classes that manipulated this data to take the correct inputs. This required changing some of the methods that Emotiv provided, as well as changing different variables within my own GUI to fit the new format. This made it possible for the user to accurately train thoughts related to each movement of the arm. The major advantage behind making all of these changes was that it actually helped the user in clarifying about what to think. A possible work around would be to utilize the built in Cognitiv suite, and just relating different actions with movements of the arm. While this may give the user an easier time in the actual training of the data, it may confuse thoughts when the device is paired with the arm. For example, one could tie together the Cognitiv thought of 'push' with the arm movement of 'left.' This would make it easier to train, as the 'push' action is one of the first thoughts that the user is taught to train and is relatively simple to complete. As soon as this procedure is tied in with the arm, however, thinking 'push' and seeing the arm move left will cognitively confuse the operator, ultimately making it more difficult to complete the rest of the project.

Now that the user can consistently train actions directly related to the movements of the arm, communication must be set up between the headset, the laptop, and the NXT brick. This requires the previous knowledge from the test programs about communicating to a computer via Bluetooth, which I did successfully. I first created a button on the GUI that would call a program to establish communication with the robot. I used a program that I already wrote, namely the final test application for the arm, which was a Bluetooth remote controller for the arm. I modified the program majorly, so that it would not actually send any information, only establishing contact with the robot via Bluetooth. I could have kept this class in the main GUI class but I choose to keep it separate for simplicity, as I tried to keep all things dealing with the robot in individual classes compared to the Emotiv. I added a simple helper class to this altered program that would easily allow me to send information over Bluetooth to the robot. The button on the GUI, which I titled "Control Robot," would simply establish a Bluetooth connection to the robot, and make it possible to send information to the robot.

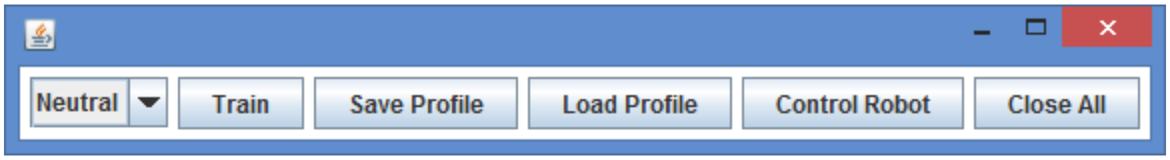


Figure 10: First GUI with robot add-in

The next question that I had to figure out was what information needed to be sent to the robot, and in what form would that information be sent and received. While the user trains actions, an infinite loop runs in the main class, which constantly updates variables that are titled events. These different events are the differences in the EEG waves from the previous time, and they are how the software is able to tell a thought from a neutral condition. If this thought is correctly read as something other than neutral, then it changes the Emotiv state. The state is current thought that the user is contemplating, and has two different factors: action and power. The action is related to the enumerated value for each thought, so it is a number, specifically a power of 2. I am not entirely sure why the Emotiv Company used powers of 2 to enumerate their values, as there seems to be no added benefit behind it. One could reduce this to simply be just consecutive integers, and that should probably work in the same fashion as powers of 2. For the variable that I created, 1 corresponded to left, 2 to right, 4 to up, 8 to down, 16 to open, and 32 to close. The power of the state is simply how strong the device is reading the electrical signal. The power is a double ranging from 0 to 1, with higher numbers meaning the electrical signals related to that thought are very strong compared to the neutral condition. Low power signals can often just be mistakes picked up by the device or errors in correctly filtering noise, so I introduced a threshold. This hurdle was 0.3, so that if the power of the thought were over this value, then it would correctly register as a thought.

With the knowledge of what the software produced during a change in thought, I could now figure out what to actually send to the robotic arm. My first obstacle was sending information too early, which would happen after the user trains the first thought, it registers within the software, but there is no connection to the robot. I created a global Boolean variable within the GUI class that was initially false, but would become true as soon as the Bluetooth connection was made to the robot. This allowed the user to train all of their thoughts before sending information to the robot, or even just train two thoughts and then start sending data. It gives the user flexibility in making this decision and is great for testing purposes. The next decision I had to make was what to do about the threshold, and how to relate it to the information sent to the robot. I still utilized the hurdle, and decided that only if power was above that value, then the program would send information to the robot to complete some movement. Any thought that resulted in power below 0.3 would then send stop signals that would override any action that the robot was taking. I decided on this as a last resort in case the operator's thoughts were being read incorrectly, and they needed a surefire way to stop motion in the robot. The final complication I had to overcome was what information to actually send over Bluetooth from the laptop to the robot. Going back to knowledge from mistakes in the preparation, I knew that words sent over Bluetooth would not always work and produce the desired outcome. Since the actions were already in integer form, I decided to send that information to the robot. That means I had to change how the robot received the information, since the data was now in the form of powers of 2. Changing this factor was relatively simple, and created a situation where an operator could train a thought, and it would correctly be sent over to the robot to perform that thought.

In order to create a prosthetic arm with the most functionality and usability, I created the robot to utilize all three motor ports and have six different motions available. However, I quickly realized that this would present a major problem, something I discovered in the first part of my implementation. Emotiv only allows for the user to train 4 actions not including the neutral state, and I wanted to train 6, not including neutral. Finding a software workaround for this hardware limitation was certainly one of the harder things accomplished, and without it, full functionality of the designed arm would not be possible. To tackle this issue, I tried a few different methods until I found one that worked the best. The first technique I tried was simply running the GUI Java program twice, which would theoretically allow me to train 4 different actions each time because the GUIs would have different data structures and therefore the data would be saved in different places. While this mainly worked in training the data, the fault in the plan came when it was time to control the robotic arm. There was a lot of trouble in actually establishing two different connections with the robot over the same Bluetooth connection, and this method simply had to be scraped. The fundamental part of one GUI had to be kept so that the same Bluetooth communication with the robot could be held in tact, as that was the most important part of the program. However, this failure did lead to some important knowledge, which was that two instances of the program could allow for the training of more than 4 thoughts.

Going off of that mistake allowed me to think of the correct procedure that permitted me to create a software fix to only training 4 actions. Essentially, opening up two instances of the main program generated two different data structures for the same data. I could replicate this fact by creating two different data structures myself. This would first require arbitrarily dividing the data, which I did in a set of 4 and 2. The 2 motions that I split from the rest were open and close, as they dealt with the hand, which requires different thinking than the 4 motions of the arm. In order to train these new thoughts, I then had to create new instances of the buttons on the GUI, so I generated buttons that would train these motions, along with saving and loading these new profiles. I then had to go throughout the entire program, and duplicate every instance of work being done on the first data set to make it so that the same changes were done on the second data set. This required changing most of the built-in classes to accept extra variables differentiating between the first and second set of data values so that they would not be mixed up when the user completed the training. Once I made all of the changes throughout the program, the user could successfully train the first four actions, plus neutral, using the original training buttons, along with saving and loading that profile. The user could then train the next two actions from a separate button, along with effectively being able to save and load profiles related to that data. One problem that arises from this method is that the data between the two sets is not shared, so training can be difficult and very time consuming. This is because the program may register thoughts from both sets at the same time when you are only focusing on one of them, simply due to the data not being differentiated amongst the sets. Creating a way to share this information and filter it accordingly would make this system incredibly more successful and would allow for the user to train even more actions.

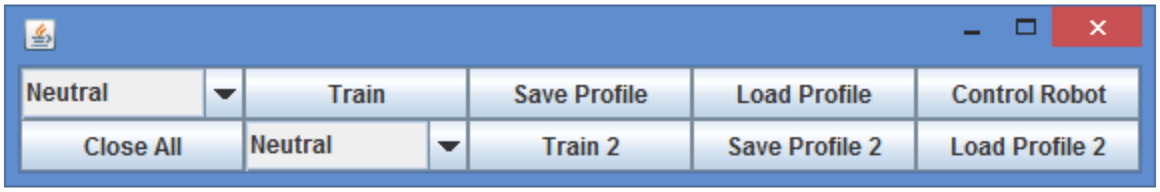


Figure 11: Final GUI

The ultimate product in the second implementation combines all of the knowledge that I have accumulated about this system. It allows any user, not just myself, to train thoughts that directly correspond with controlling a robotic arm in a simple and effective manner. While the GUI may not be the most appealing, it provides a modest way for any operator to access all of the required buttons. While not implemented, this GUI could also be combined with a previous implementation, such as the use of the headset to simulate the mouse. This would allow the user to fully control all function of the system just with their head. More specifically, if the program tied blinking to clicking like it was originally, then the user would not have to train an extra thought for the click of the mouse. This would create complete autonomous control of the GUI by the user, with the only input to the computer coming from the headset. While I did not complete this due to the ease of training with an attached mouse instead of the built-in gyro, the program can easily be changed to include that protocol.

5. Future Opportunities

The future for the interaction of EEG brain waves, BCI, and prosthetic limbs is looking very strong. The background is changing every year, as researchers are coming out with technologies that can consistently better lives. The Emotiv EPOC can help lead the charge for affordable non-invasive BCI connection to prosthetic limbs, if certain changes are made. The biggest limitation that must be overcome in order for the EPOC device to be fully effective is the training of only 4 actions. This severely restricts the impact one could have with this device, as well as arbitrarily placing a stopping point for training functions. While Emotiv is aware of this restraint and are currently working on a hardware resolution to the issue, true progress will not be made using the EPOC until that happens. Furthermore, one issue that continually popped up was the questionable necessity of the suites besides the Cognitiv program. While the Expressiv suite is useful for beginners and understanding how to tie in different aspects of the device, it is not a measurement of brain waves, simply just facial movements. The Affectiv suite is probably more detrimental to activities than it is helpful. This is due to its measurement of nervousness, which can play a role in changing thoughts based on the user and their own emotions. Focusing on the Cognitiv suite, which has the most potential out of all, would create a situation where Emotiv is completing research that could directly impact the lives of people. One feature of the device that is severely underused, and underappreciated, is the gyro. Incorporating the gyro can lead to more accurate simulations in virtual reality, and could lead to more control for the user over a robot or other device. Furthermore, I could have been used the gyro in the second part of my implementation, controlling another aspect of the arm instead of using thoughts.

One of the reasons that the EPOC device is affordable is due to it only having 14 electrodes in the headpiece. This is in comparison to normal EEG devices used at hospitals or clinics, which can have anywhere from 50 to 100 electrodes. Due to the amazing amount of wires required for a large amount of electrode, many of these devices require the user to sit in one place and be tethered to a machine that records all of the data. One of the benefits of the EPOC device is that it is completely wireless, allowing the operator to move with it on, and even control things from a distance. While adding more electrodes will increase costs and possibly limit the fact that the device is wireless, the benefits will surely outweigh those consequences. More electrodes ultimately mean the device will be receiving more data from the brain. While more advanced methods of data mining and machine learning are necessary to filter through larger amounts of data, it can create a situation where the program can register more complex thoughts. Complex thoughts will ultimately allow the user to control more complex systems, such as the full movement of a prosthetic arm, or possibly even multiple prosthetic limbs. A problem will arise when there is too much data for a computer to handle, but since that limit is not even close to being reached, more electrodes will surely create a better experience for harder thoughts.

Advances in robotic technology will also improve the experience, of both an amputee and a researcher trying to develop better situations. Simple improvements, such as the third generation of Lego NXT kits, may have a small impact in the long run, but can provide the stepping-stone for incredible influence. Better motor and hydraulic design, both by Lego and other companies, will allow for engineers to create prosthetics that amputees will actually want to wear. More helpful designs that actually mimic all the motions and muscles of a real hand and arm will undoubtedly be welcomed by wears of the prosthetics, especially since the shape has not really been upgraded for a long time. It could be as simple as changing the claw to a hand with five individual fingers, which would at least make amputees feel more comfortable wearing their prosthetic in public. These new limbs could vastly improve lives for the men and women who have protected our freedom, and any effort in enhancing their standard of living should be fully and sharply investigated.

6. Conclusions

The ultimate goal of this thesis was to make the inaccessible accessible for anyone interested. The intersection of EEG and BCI provide a promising future for people who may have a bleaker outlook on life, and could completely change their lives. I was able to successfully simulate a rudimentary computer, which was almost completely controlled from the EPOC and someone's brain. I was able to make it so that a user could control both the mouse and the keyboard simply using their thoughts. These two objects are the most basic inputs to a computer, and one can control the entire computer with just these two functions. To better simulate a created system specifically for the EPOC device, I was also able to create an email sender and a document writer, both of which utilize the built keyboard and the user-controlled mouse. These simple functions show that someone's brain could completely control a computer from boot up to shut down, opening the possibility for people to use computers who are severely handicapped or otherwise unable to operate them. The second part of my thesis was tying in EEG, BCI, and robotics, as I was trying to simulate a brain controlled prosthetic arm. For a simple Lego arm that only has six motions, this attempt was successful, as an operator could ultimately control all six movements with careful thought structures and lots of training. While this does not mean that this product is ready for the public to control robots due to the time and initial investment, scientists and engineers should research more effective and affordable ways of EEG and robot interfaces. The combination of EEG and BCI promise a bright future, and could ultimately change thousands of lives if implemented correctly.

7. Acknowledgements

First off, I would like to thank my advisor, Professor Signorile, who would consistently push me to do more. I am confident this thesis would not be successful without his guidance. Secondly, I would like to thank the entire Computer Science Department, who were kind enough to purchase the required material for me to complete this thesis. Lastly, I would like to thank my roommates and my parents, whose constant questioning about my thesis made me never forget to work on it.