Boston College
Computer Science Department

Non-Invasive BCI through EEG

An Exploration of the Utilization of Electroencephalography to Create Thought-Based
Brain-Computer Interfaces

Senior Honors Thesis
Daniel J. Szafir, 2009-10
Advised by Prof. Robert Signorile

Contents

List of Figures and Tables

1.   Abstract

It has long been known that as neurons fire within the brain they produce measurable electrical activity. Electroencephalography (EEG) is the measurement and recording of these electrical signals using sensors arrayed across the scalp. Though there is copious research in using EEG technology in the fields of neuroscience and cognitive psychology, it is only recently that the possibility of utilizing EEG measurements as inputs in the control of computers has emerged. The idea of Brain-Computer Interfaces (BCIs) which allow the control of devices using brain signals evolved from the realm of science fiction to simple devices that currently exist. BCIs naturally present themselves to many extremely useful applications including prosthetic devices, restoring or aiding in communication and hearing, military applications, video gaming and virtual reality, and robotic control, and have the possibility of significantly improving the quality of life of many disabled individuals. However, current BCIs suffer from many problems including inaccuracies, delays between thought, detection, and action, exorbitant costs, and invasive surgeries. The purpose of this research is to examine the Emotiv EPOC© System as a cost-effective gateway to non-invasive portable EEG measurements and utilize it to build a thought-based BCI to control the Parallax Scribbler® robot. This research furthers the analysis of the current pros and cons of EEG technology as it pertains to BCIs and offers a glimpse of the future potential capabilities of BCI systems.

2.   Introduction

Who wouldn't love to control a computer with their mind? Interfaces between the brain and computers have long been a staple of science fiction where they are used in an incredible variety of applications from controlling powered exoskeletons, robots, and artificial limbs to creating art envisioned by the user to allowing for machine-assisted telepathy. This space-age fantasy is not quite real yet, however simple BCIs do currently exist and research and public interest in them only continues to grow. This research explores the process in creating a novel BCI that utilizes the Emotiv EPOC System to measure EEG waves and controls the Parallax Scribbler robot.

2.1   Electroencephalography

EEG waves are created by the firing of neurons in the brain and were first measured by Vladimir Pravdich-Neminsky who measured the electrical activity in the brains of dogs in 1912, although the term he used was "electrocerebrogram."[1] Ten years later Hans Berger became the first to measure EEG waves in humans and, in addition to giving them their modern name, began what would become intense research in utilizing these electrical measurements in the fields of neuroscience and psychology.[2]

EEG waves are measured using electrodes attached to the scalp which are sensitive to changes in postsynaptic potentials of neurons in the cerebral cortex. Postsynaptic potentials are created by the combination of inhibitory and excitatory potentials located in the dendrites. These potentials are created in areas of local depolarization or polarization following the change in membrane conductance

---

1   Swartz, B.E; Goldensohn, ES. "Timeline of the history of EEG and associated fields." *Electroencephalography and Clinical Neurophysiology.* Vol. 106, pp.173–176. 1998. <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6SYX-4FV4S6H-1-1&_cdi=4846&_user=10&_orig=browse&_coverDate=02%2F28%2F1998&_sk=998939997&view=c&wchp=dGLbVz z-zSkWb&md5=47fbbe7e51a806779716fba415b96ab7&ie=/sdarticle.pdf>.
2   Millett, David. "Hans Berger: from psychic energy to the EEG." *Perspectives in Biology and Medicine,* Johns Hopkins University Press, 2001. 44.4 pp. 522–542. <http://muse.jhu.edu/journals/perspectives_in_biology_and_medicine/v044/44.4millett.html>.

as neurotransmitters are released. Each electrode has a standard sensitivity of 7 µV/mm and averages the potentials measured in the area near the sensor. These averages are amplified and combined to show rhythmic activity that is classified by frequency (Table 1).[3] Electrodes are usually placed along the scalp following the "10-20 International System of Electrode Placement" developed by Dr. Herbert Jasper in the 1950's which allows for standard measurements of various parts of the brain (Figure 1).[4] The primary research that utilizes EEG technology is based on the fact that this rhythmic activity is dependent upon mental state and can be influenced by level of alertness or various mental diseases. This research commonly involves comparing EEG waves in alert and asleep patients as well as looking for markers in abnormal EEG waves which can evidence diseases such as epilepsy or Alzheimer's. One of the historical downsides of EEG measurement has been the corruption of EEG data by artifacts which are electrical signals that are picked up by the sensors that do not originate from cortical neurons. One of the most common cause of artifacts is eye movement and blinking, however other causes can include the use of scalp, neck, or other muscles or even poor contact between the scalp and the electrodes.[5] Many EEG systems attempt to reduce artifacts and general noise by utilizing reference electrodes placed in locations where there is little cortical activity and attempting to filter out correlated patterns.[6]

3   Nunez PL, Srinivasan R. *Electric Fields of the Brain: The Neurophysics of EEG*. Oxford University Press. 1981.
4   Niedermeyer, Ernst and da Silva, Fernando Lopes. *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields, Fifth Edition.* Lippincott Williams & Wilkins, 2005. pp 140
5   Rowan, A. James. *Primer of EEG.* Elsevier Science, Philadelphia, PA. 2003.
6   Ludwig, Kip A. et al. *Employing a Common Average Reference to Improve Cortical Neuron Recordings from Microelectrode Arrays.* Journal of Neurophysiology, September 3rd, 2008.
     <http://jn.physiology.org/cgi/reprint/90989.2008v1.pdf>.

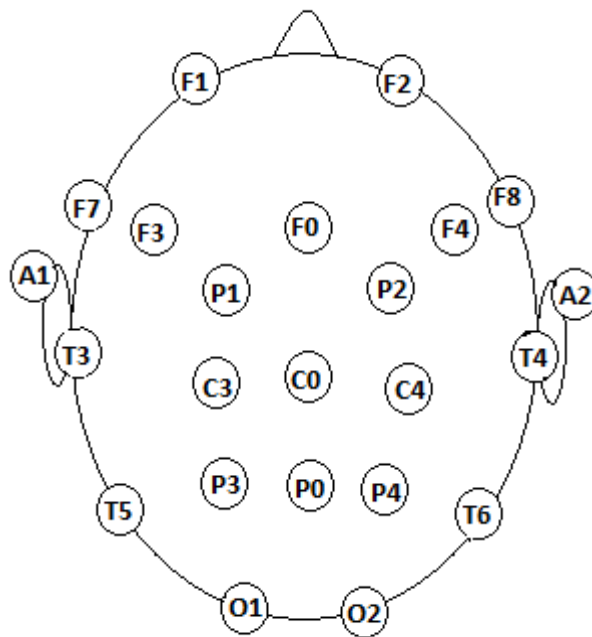| Band	Frequency (Hz)
Delta	1-4
Theta	4-7
Alpha	7-13
Beta	13-30
Gamma	30+

Table 1: *EEG Bands and Frequencies*

Figure 1: *Electrode Placement according to the International 10-20 System. Odd numbers on the right, even on the left. Letters correspond to lobes – F(rontal), T(emporal), P(arietal), and O(ccipital). C stands for Central (there is no central lobe).*

2.2   Brain-Computer Interfaces

The term "Brain-Computer Interface" first appeared in scientific literature in the 1970's, though the idea of hooking up the mind to computers was nothing new.[7] The ultimate goal of BCI research is to create a system that not only an "open loop" system that responds to users thoughts but a "closed loop" system that also gives feedback to the user. Researchers initially focused on the motor-cortex of the brain, the area which controls muscle movements, and testing on animals quickly showed that the natural learning behaviors of the brain could easily adapt to new stimuli as well as control the firing of specific areas of the brain.[8] This research dealt primarily with invasive techniques but slowly algorithms emerged which were able to decode the motor neuron responses in monkeys in real-time and translate them into robotic activity.[9,10] Recently, a system developed by researchers and Carnegie Mellon University and the University of Pittsburgh allowed a monkey to feed itself via a prosthetic arm using only its thoughts.[11] This research is extremely promising for the disabled, and indeed by 2006 a system was developed for a tetraplegiac that enabled him to use prosthetic devices, a mouse cursor, and a television via a 96-micro-electrode array implanted into his primary motor cortex.[12] Despite these achievements, research is beginning to veer away from invasive BCIs due to the costly and dangerous

7   J. Vidal, "Toward Direct Brain–Computer Communication." *Annual Review of Biophysics and Bioengineering.* Vol. 2, 1973, pp. 157-180. <http://arjournals.annualreviews.org/doi/pdf/10.1146/annurev.bb.02.060173.001105>.
8   Fetz, E E. "Operant Conditioning of Cortical Unit Activity." *Science.* Volume 163, February 28, 1969, pp. 955-958. <http://www.sciencemag.org/cgi/rapidpdf/163/3870/955.pdf>.
9   Kennedy, Philip R. et al. "Activity of single action potentials in monkey motor cortex during long-term task learning." Brain Research, Volume 760 Issue 1-2, June 1997, pp. 251-254. <http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6SYR-3PM7MHB-14&_user=521319&_coverDate=06%2F20%2F1997&_rdoc=1&_fmt=high&_orig=search&_sort=d&_docanchor=&view=c&_searchStrId=1275477491&_rerunOrigin=google&_acct=C000026018&_version=1&_urlVersion=0&_userid=521319&md5=7690dc204a5a471e27a26a151b0d158d>.
10 Wessber, Johan et al. "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates." *Nature.*Vol. 408, No. 6810, 2000. pp. 361-365. <http://medev.kaist.ac.kr/upload/paper/ij01.pdf>.
11 Carey, Benedict. "Monkeys Think, Moving Artifiacl Arm as Own." *The New York Times.* May 29, 2008. <http://www.nytimes.com/2008/05/29/science/29brain.html>.
12 Hochberg, Leigh R. et al. "Neuronal ensemble control of prosthetic devices by a human with tetraplegia." *Nature.* Vol. 442, July 13, 2006, pp. 164-171. <http://www.nature.com/nature/journal/v442/n7099/full/nature04970.html>.

nature of the surgeries required for such systems. Non-invasive alternatives for BCIs include EEG technology, Magnetoencephalography (MEG), and Magnetic Resonance Imaging (MRI), as well as the "partially invasive" Electrocorticography where sensors are placed within the skull but outside the gray matter of the brain. Non-invasive methods are limited in that they are often susceptible to noise, have worse signal resolution due to distance from the brain, and have difficulty recording the inner workings of the brain. However more sophisticated systems are constantly emerging to combat these difficulties and non-invasive techniques have the advantage of lower cost, greater portability, and the fact that they do not require any special surgery.[13]

3.   Previous EEG BCI Research

Though the idea of using EEG waves as input to BCIs has existed since the initial conception of BCIs, actual working BCIs based on EEG input have only recently appeared.[14] Most EEG-BCI systems follow a similar paradigm of reading in and analyzing EEG data, translating that data into device output, and giving some sort of feedback to the user (Figure 2), however implementing this model can be extremely challenging.[15] The primary difficulty in creating an EEG-based BCI is the feature extraction and classification of EEG data that must be done in real-time if it is to have any use.

Feature extraction deals with separating useful EEG data from noise and simplifying that data so that classification, the problem of trying to decide what the extracted data represents, can occur. There is no best way of extracting features from EEG data and modern BCIs often use several types of feature extraction including Hjorth parameters (a way of describing the normalized slope of the data), wavelet transforms, Fourier transforms, and various other types of filters. The major features that

13 Fabiani, Georg E. et al. *Conversion of EEG activity into cursor movement by a brain-computer interface.*
      <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.128.5914>. 2004.
14 J. Vidal, "Toward Direct Brain–Computer Communication."
15 Omidvarnia, Amir H. et al. *Kalman Filter Parameters as a New EEG Feature Vector for BCI Applications.*
      <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.1388>.

EEG-BCI systems rely on are event-related potentials (ERPs) and event-related changes in specific frequency bands. The "P300 wave" is one of the most often used ERPs in BCIs and is utilized because it is easily detectable and is only created in response to specific stimuli.[16,17]
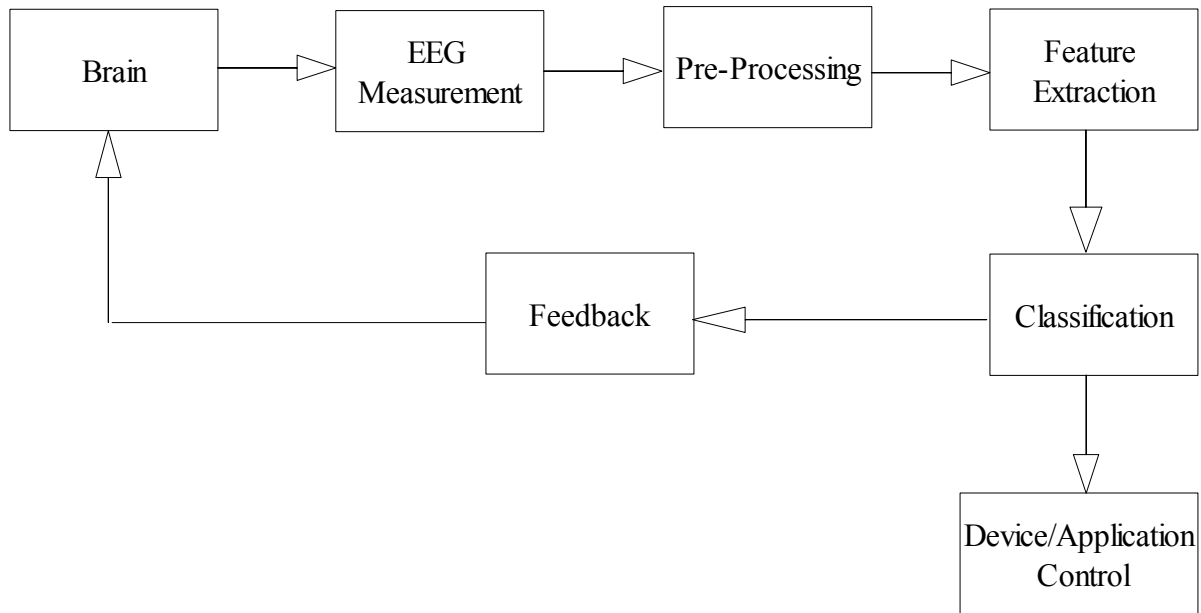

Figure 2: *Brain-Computer Interface Design Pattern*

BCI systems are further complicated by the fact that there is no standard way of classifying the extracted data. Various means including neural networks, threshold parameters, and various other types of pattern recognizers are employed to try to match the input data to known categories of EEG archetypes.[18] Furthermore, researchers have also relied on unsupervised learning algorithms to find natural clusters of EEG segments that are indicative of certain kinds of mental activities with varying

---

16 Niedermeyer. *Electroencephalography.* pp. 1265-1266.
17 Sellers, Eric W. et al. "A P300 event-related potential brain–computer interface (BCI): The effects of matrix size and inter stimulus interval on performance." *Biological Psychology.* Volume 73, Issue 3, October 2006. pp. 242-252. <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6T4T-4KGG642-1-N&_cdi=4983&_user=521319&_pii=S0301051106001396&_orig=search&_coverDate=10%2F31%2F2006&_sk=999269996&view=c&wchp=dGLbVzW-zSkWA&md5=4e9076e39d5e96823ba4f50e3e38588d&ie=/sdarticle.pdf>.
18 Adlakha, Amit. *Single Trial EEG Classification.* Swiss Federal Institute of Technology. July 12, 2002. <http://mmspl.epfl.ch/webdav/site/mmspl/shared/BCI/publications/adlakhareport.pdf>.

degrees of success.[19,20]

Feedback is essential in BCI systems as it allows users to understand what brainwaves they just produced and to learn behavior that can be effectively classified and controlled. Feedback can be in the form of visual or auditory cues and even haptic sensations, and ongoing research is still attempting to figure out the optimal form feedback should take.[21]

EEG-BCIs can be classified as either synchronous or asynchronous. The computer drives synchronous systems by giving the user a cue to perform a certain mental action and then recording the user's EEG patterns in a fixed time-window. Asynchronous systems, on the other hand, are driven by the user and operate by passively and continuously monitoring the user's EEG data and attempting to classify it on the fly. Synchronous protocols are far easier to construct and have historically been the primary way of operating BCI systems.[22]

EEG-BCI systems have made incredible progress in recent years. By 2000, researchers had created a thought-translation device for completely paralyzed patients which allowed patients to select characters based on their thoughts, although the character selection process was time consuming and not perfectly accurate.[23] By 2008, researchers collaborating from Switzerland, Belgium, and Spain created a feasible asynchronous BCI that controlled a motorized wheelchair with a high degree of accuracy though again the system was not perfect.[24] Today, the 2010 DARPA budget allocates $4

---

19 Lu, Shijian et al. "Unsupervised Brain Computer Interface based on Inter-Subject Information." 30th Annual International IEEE EMBS Conference. Vancouver, British Columbia, Canada. August 20-24, 2008. <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04649233>.

20 Niedermyer. *Electroencephalography.* pp. 1240.

21 Kauhanen L, Palomaki T, Jylanki P, Aloise F, Nuttin M, and Millan JR. "Haptic feedback compared with visual feedback for BCI". Proceeding of 3rd International BCI Workshop and Training Course 2006. Graz Austria. 21-25 Sept 2006, Pp. 66-67. <*www.lce.hut.fi/research/css/bci/Kauhanen_et_al_conf_2006.pdf*>.

22 Niedermeyer. *Electroencephalography.* pp. 1265.

23 Birbaumer, N. et al. "The Thought Translation Device (TTD) for Completely Paralyzed Patients." *IEEE Transactions on Rehabilitation Engineering.* Volume 8, No. 2, June 2000. pp. 190-193. <www.cs.cmu.edu/~tanja/BCI/TTD2003.pdf>.

24 Galán, F. et al. "A Brain-Actuated Wheelchair: Asynchronous and Non-invasive Brain-Computer Interfaces for Continuous Control of Robots." *Clinical Neurophysiology.* Volume 119, Issue 9, September, 2008. pp. 2159-2169. <http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6VNP-4T08054-5&_user=521319&_coverDate=09%2F30%2F2008&_rdoc=1&_fmt=high&_orig=search&_sort=d&_docanchor=&view=c&_acct=C000026018&_version=1&_urlVersion=0&_userid=521319&md5=433c13acfed7171c6385ecefa6fe6431>.

million to develop an EEG-based program called Silent Talk which will "allow user-to-user communication on the battlefield without the use of vocalized speech through analysis of neural signals."[25] State-of-the-art EEG-based BCIs are a cutting-edge emerging technology and researchers are constantly developing newer and more accurate algorithms to aid making BCIs that are simpler and more effective than ever before.

4.   The Emotiv© System

The Emotiv System, whose tag-line is "you think, therefore you can," bills itself as a "revolutionary new personal interface for human computer interaction." It is based around the EPOC headset for recording EEG measurements and a software suit which processes and analyzes the data. Emotiv offers both a consumer headset for $299 which only works with approved applications and a developer headset which supports product development and includes a software bundle for $500. Both headsets are wireless and utilize a proprietary USB dongle to communicate using the 2.4GHz band. This research used the developer EPOC headset which contains a rechargeable 12 hour lithium battery, 14 EEG sensors (plus CMS/DRL references), and a gyroscope, and has an effective bandwidth of 0.16-43Hz (Figures 3, 4). Emotiv offers 6 different software development kits which grant various control over the EPOC headset API and detection libraries and come with up to 4 different programs (Table 2). This research originally used the Development SDK and later upgraded to the Research Edition. The Research Edition includes the Emotiv Control Panel, EmoComposer (an emulator for simulating EEG signals), EmoKey (a tool for mapping various events detected by the headset into keystrokes), the TestBench, and an upgraded API which enables the capture of raw EEG data from each individual sensor.[26]

---

25 Drummond, Katie. "Pentagon Preps Soldier Telepathy Push." Wired Magazine. May 14, 2009.
    <http://www.wired.com/dangerroom/2009/05/pentagon-preps-soldier-telepathy-push>.
26 Emotiv Website. <http://www.emotiv.com>.

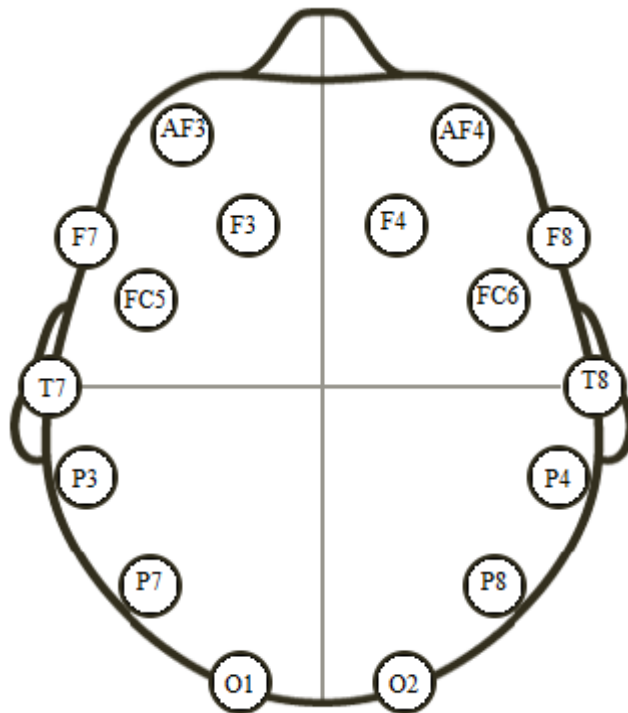Figure 3: *Emotiv EPOC Headset*

Figure 4: *The 14 EPOC Headset 14 contacts. In addition there is a Common Mode Sense (CMS) electrode in the P3 location and a Driven Right Leg (DRL) electrode in the P4 location which form a feedback loop for referencing the other measurements.*

SDK Edition	Cost	Development License	Included Software
Lite	Free	Individuals	Control Panel, EmoComposer, EmoKey
Developer	$500.00 (comes with developer headset)	Individuals	Control Panel, EmoComposer, EmoKey, Basic API
Research	$750.00 or $250.00 for upgrade from Developer SDK	Individuals	Control Panel, EmoComposer, EmoKey, TestBench, Raw EEG data API
Enterprise	$2,500.00	Enterprises	Control Panel, EmoComposer, EmoKey, Basic API
Enterprise Plus	$7,500.00	Enterprises	Control Panel, EmoComposer, EmoKey, TestBench, Raw EEG data API
Education	$2,500.00	Educational Institutes	Control Panel, EmoComposer, EmoKey, TestBench, Raw EEG data API

Table 2: *Emotiv SDKs*

4.1   Control Panel

The Emotiv Control Panel is a graphical-user interface which functions as a gateway to using the EPOC headset. It oversees connecting with the headset, preprocessing and classifying the input signals, giving feedback to the user, and allows the user to create a profile and train thoughts and actions. The Control Panel includes the Expressiv, Affectiv, and Cognitiv suits as well as a Mouse Emulator which allows the user to control the mouse by moving their head and utilizing the headset gyroscope. The Expressiv suite is designed to measure facial expressions based on reading EEG/EMG

and is an innovative way in utilizing artifacts that are usually simply filtered out of EEG systems. The

Expressiv suite can recognize 12 actions: blink, right/left wink, look right/left, raise brow, furrow brow,

smile, clench, right/left smirk, and laugh. It gives feedback by matching the incoming signals to a

simulated face avatar which mimics the user's expressions (Figure 5).
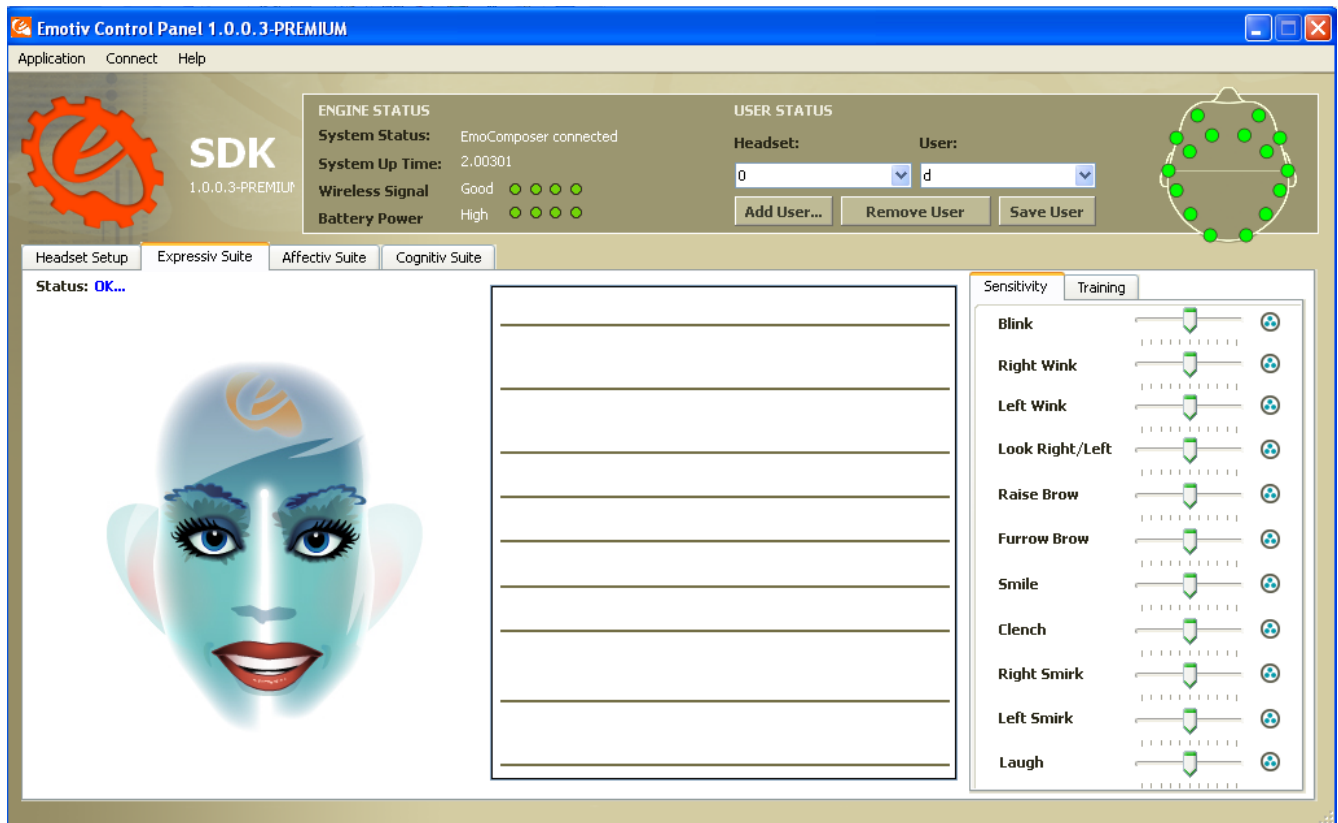
Figure 5: *Expressiv Suite*

The Affectiv suite monitors the user's emotional states. It can measure engagement/boredom,

frustration, meditation, instantaneous excitement, and longterm excitement (Figure 6). The Cognitiv

suite monitors and interprets the user's conscious thoughts. It can measure 13 active thoughts: push,

pull, lift, drop, left, right, rotate left, rotate right, rotate clockwise, rotate counterclockwise, rotate

forward, rotate backward, and disappear, as well as the passive neutral state. The Emotiv Software

detects these thoughts using built-in proprietary software. This software works by running the input

from the electrodes through a neural network and attempting to classify the signals as one of the 13 built-in "prototype" action thoughts. The data for these "prototype thoughts" was gathered prior to the release of the headset based on hundreds of test cases and serve as a base for classification. Action thoughts must be trained before use and the user can tie different thoughts to the built-in actions (i.e. training the "push" command by thinking "blue"), however doing this can lower the accuracy of recognizing the thoughts and increase the training time since these thoughts will not match up to the prototype thoughts the software initially expects. The software gives feedback in the form of a floating cube that responds to the recognized thoughts (Figure 7).
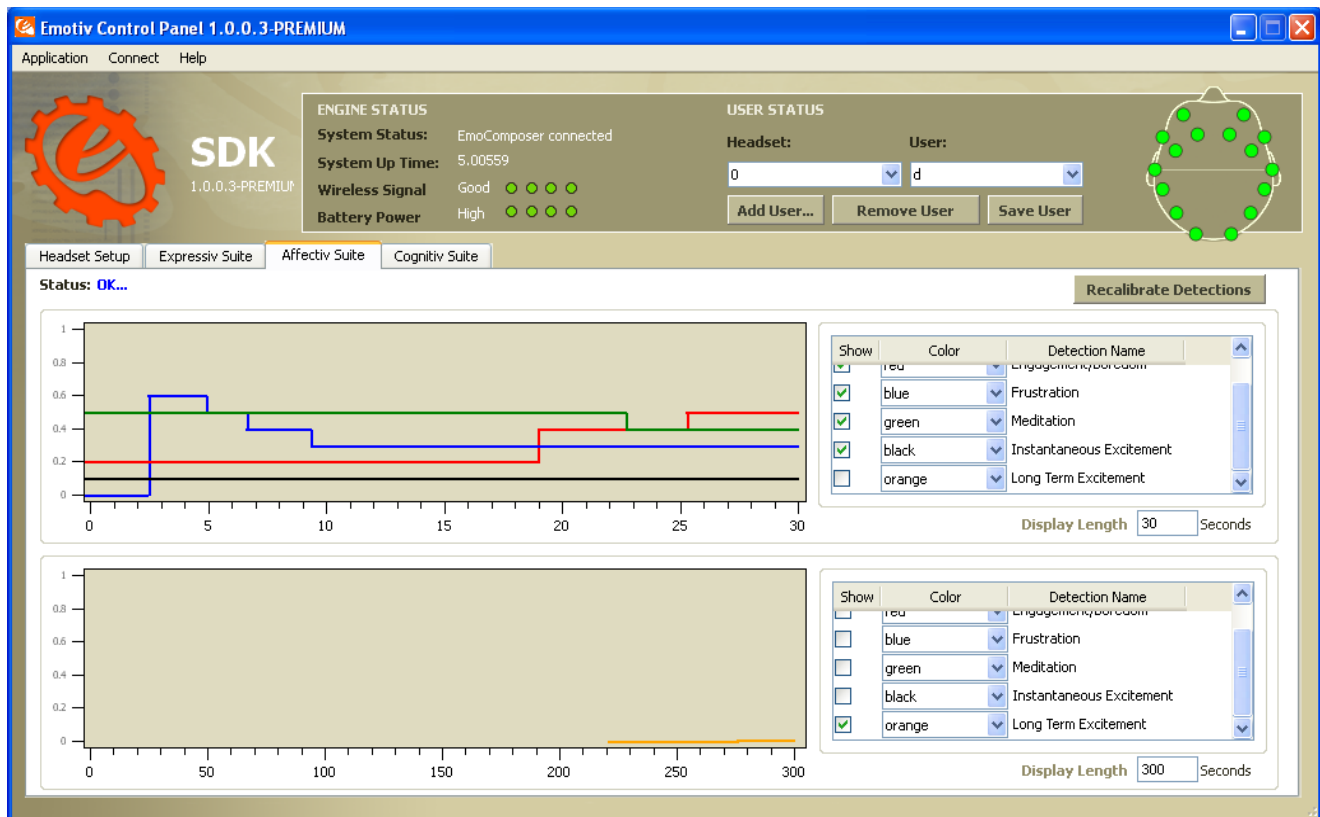

Figure 6: *Affectiv Suite*

Currently, at any given time the Cognitiv suite can distinguish between four separate thoughts on-the-fly, however the user can recognize additional thoughts concurrently by having multiple Control Panels open simultaneously, each looking for different thoughts. The four thought limit is currently in

place due to usability concerns as adding additional thoughts can greatly increase the difficulty of using the Cognitiv suite effectively. Despite this, Emotiv is currently considering upping the concurrent thought recognition limit to beyond four.
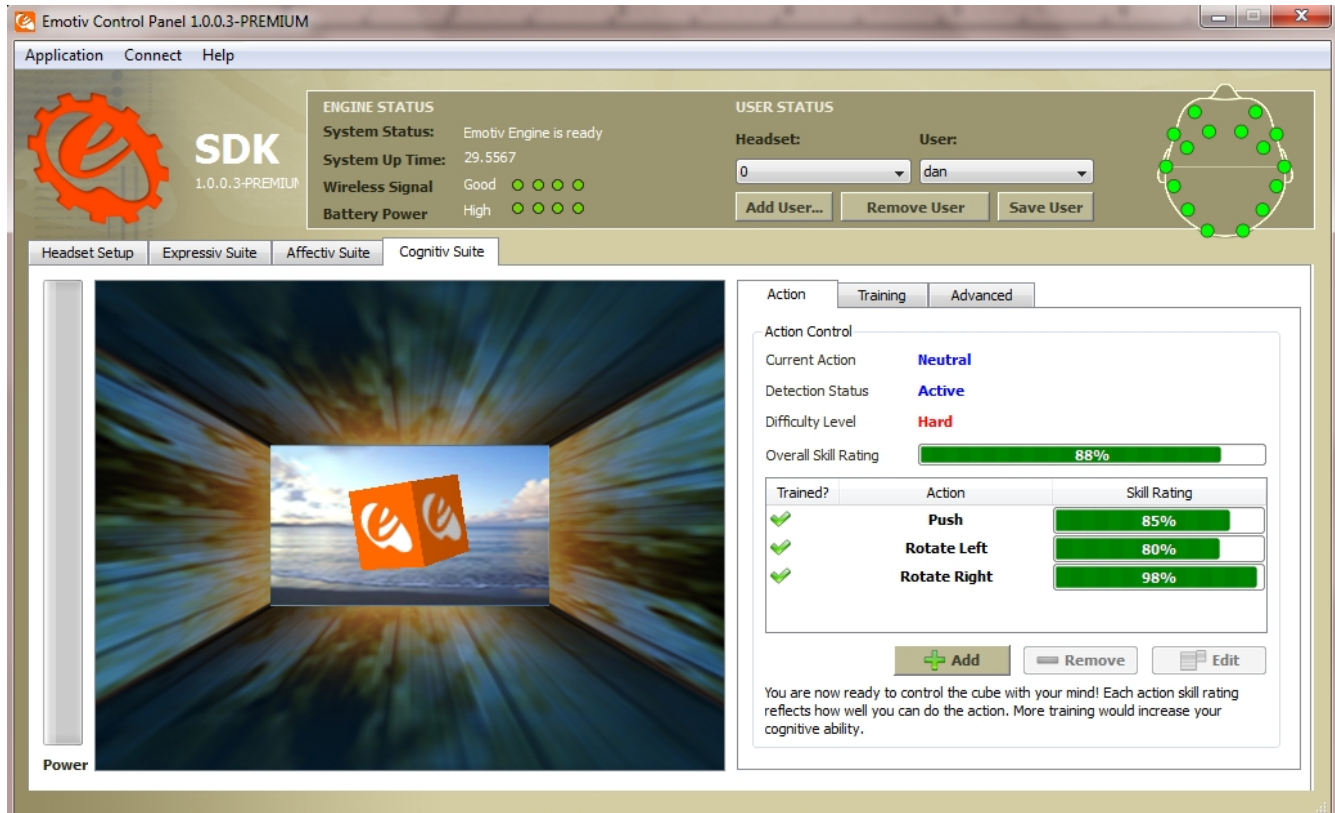
Figure 7: *Cognitiv Suite*

4.2   TestBench

The TestBench program provides a real-time display of the Emotiv headset data stream. It allows the user to see the EEG contact quality and actual data coming in from each sensor, as well as the gyroscope data, wireless packet information, and battery life (Figure 8). Furthermore, the program can display a Fast Fourier Transform (FFT) of any incoming channel and can display the Delta, Theta, Alpha, and Beta bands as well as a user-defined custom band (Figure 9). Finally, the TestBench can record, save, and playback data in European Data Format (.edf) and can convert saved EDF data to Comma-separated Value format (.csv) and was used to record and save data for blink analysis.

Figure 8: *Real-time EEG Measurements in TestBench*
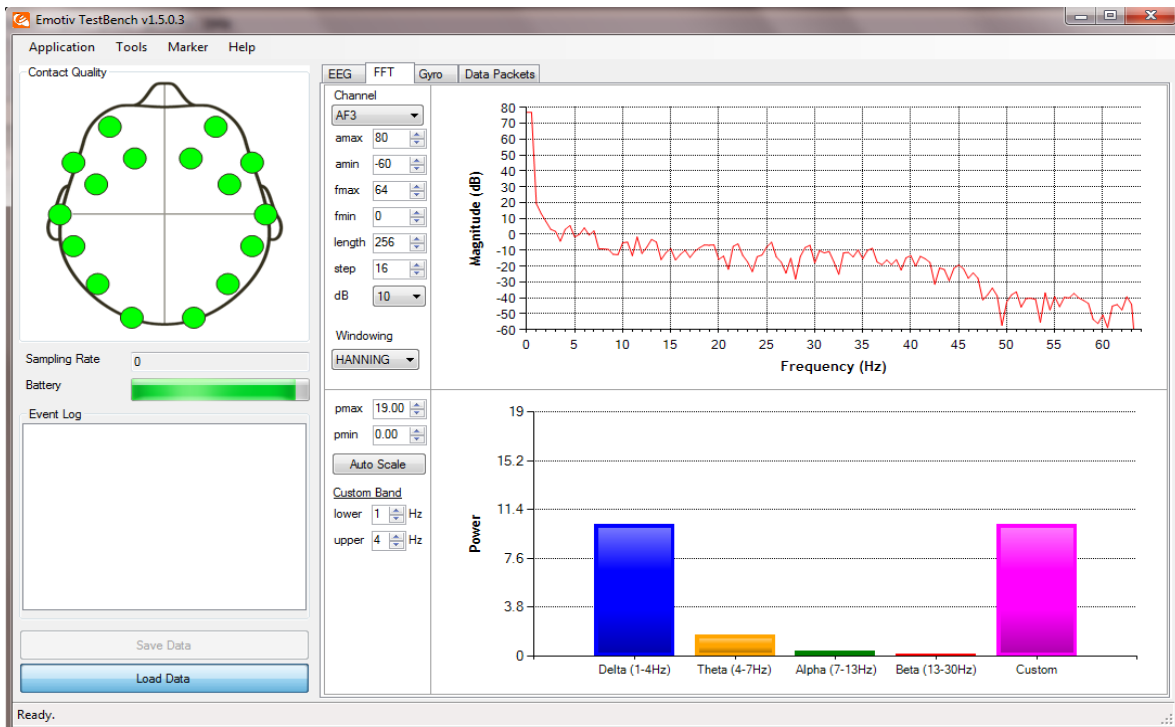
Figure 9: *FFT Measurements in TestBench. TestBench also measures Delta, Theta, Alpha, Beta, and a user-defined custom band.*

15

4.3 The Emotiv API

The Emotiv API is exposed as an ANSI C interface implemented in two Windows DLLs: edk.dll and edk_utils.dll. The core of the Emotiv SDK is the "EmoEngine," which is a logical abstractio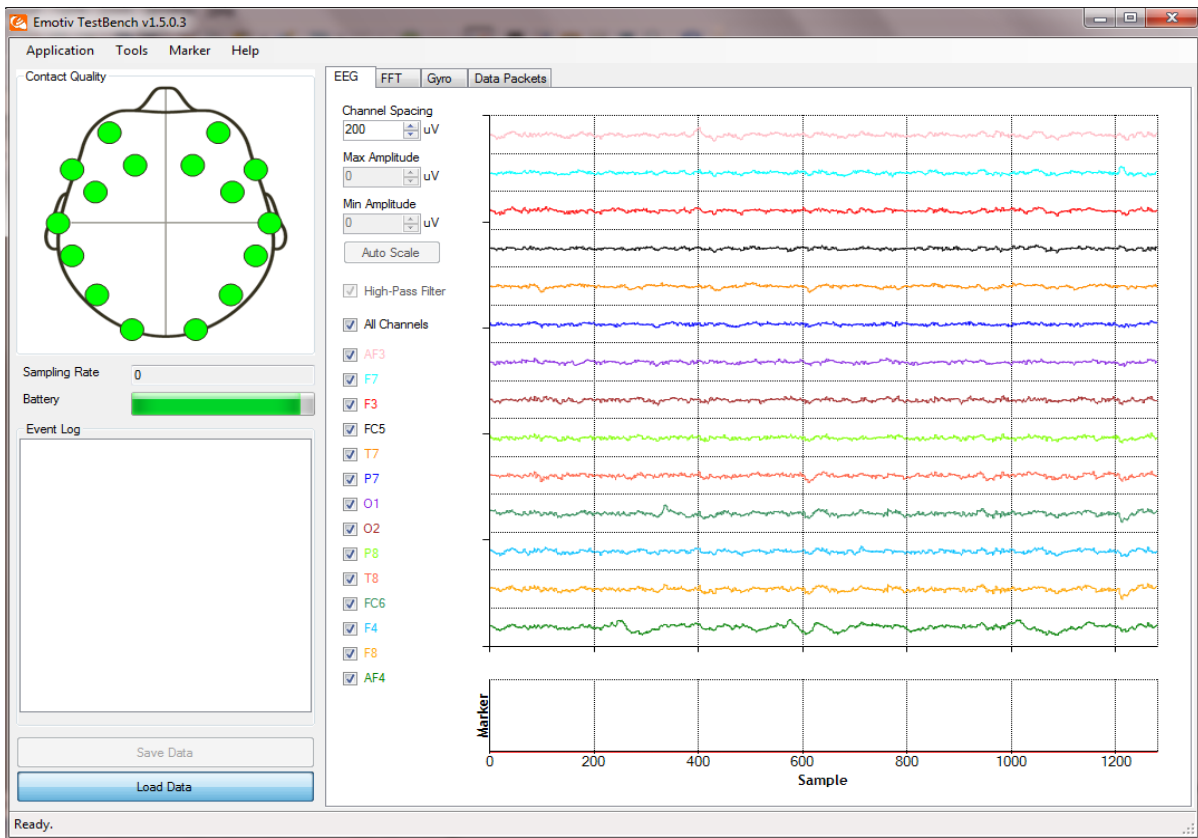n that "communicates with the Emotiv headset, receives preprocessed EEG and gyroscope data, manages user-specific or application-specific settings, performs post-processing, and translates the Emotiv detection results into an easy-to-use structure called an EmoState." Every EmoState represents the current input from the headset including "facial, emotional, and cognitive state" and, with the upgrade to the research edition, contains electrode measurements for each contact. Utilizing the Emotiv API consists of connecting to the EmoEngine, detecting and decoding new EmoStates, and calling code relevant to the new EmoState (Figure 10).[27]

Figure 10: *High-level View of the Utilization of the Emotiv API*

---

27 *Emotiv Software Development Kit: User Manual for Beta Release 1.0.x.* pp. 36-37

5.   The Parallax Scibbler® Robot and IPRE Fluke

The Parallax Scribbler robot is a fully assembled reprogrammable robot built around the BASIC Stamp® 2 microcontroller. It contains bu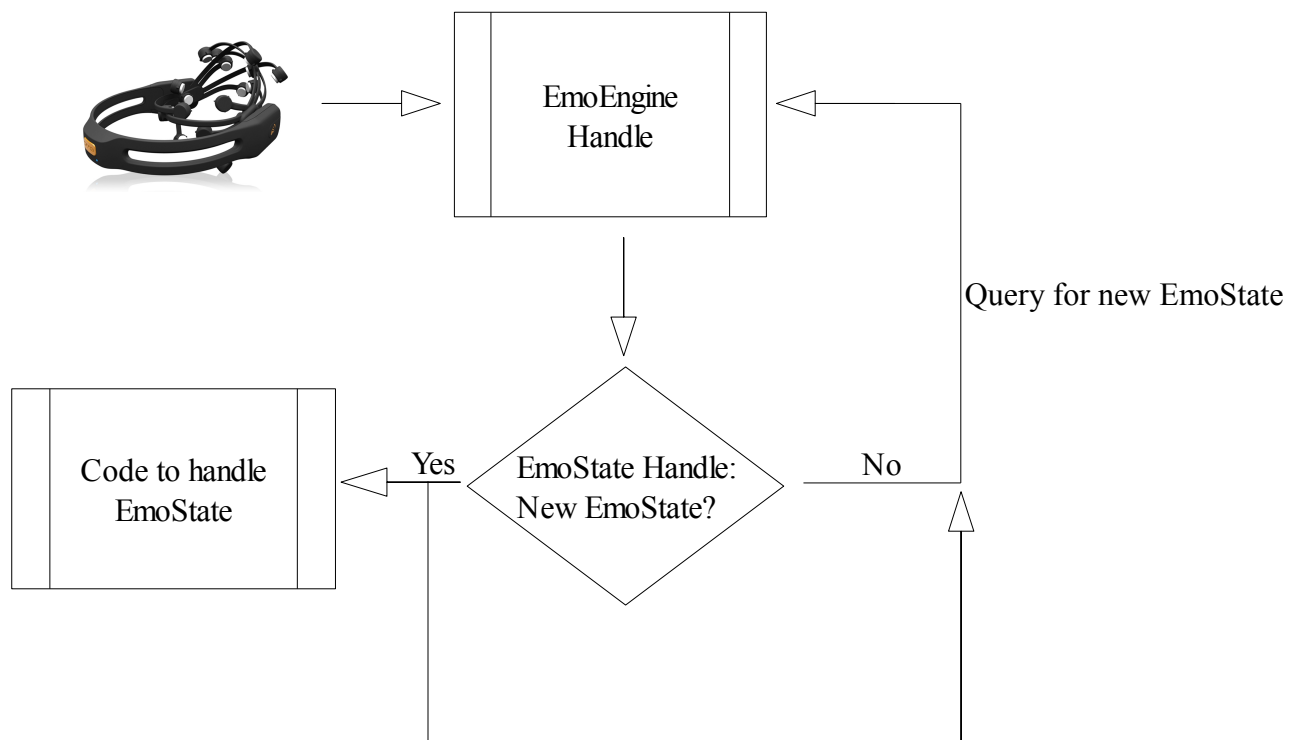ilt in photovoltaic sensors, infrared sensors, line sensors, two independent DC motors to drive the two wheels, three indicator LED lights, a speaker, and a serial port. It can be programmed using the Scribbler Program Maker Gui or the Basic Stamp Editor.[28]

The Institute for Personal Robots in Education (IPRE) Fluke is an add-on board created by Georgia Robotics that plugs into the Scribbler's serial port and adds color vision, IR range sensing, internal voltage sensing, an extra LED, and bluetooth functionality (Figure 11). Furthermore, IPRE has created the Myro (short for My Robot) software package that interacts with the Fluke via bluetooth and grants the ability to reprogram the Scribbler using Python.[29]

Figure 11: *Scribbler Robot with IPRE Fluke add-on Board*

---

28 "The Scribbler: A Reprogrammable Robot." <http://www.parallax.com/tabid/455/Default.aspx>. Copyright 2010 by Parallax Inc. Accessed 4/11/2010.
29 "Usage Guides." <http://www.roboteducation.org/guides.html>. Copyright © 2007 Institute for Personal Robots in Education. Accessed 4/11/2010.

6. Control Implementation

The control implementation for this project was written in Microsoft Visual C++. I decided on C++ as it allowed me to easily call the ANSI C libraries exposed in the Emotiv API as well as make calls to the Python/C API so that I could control the Scribbler using the Python Myro interface. The code implementing this control scheme is divided into four basic parts: connecting to the Emotiv headset via the Emotiv API, connecting to the Scribbler through the Myro Python libraries, reading and decoding Emotiv events and sending the corresponding commands to the Scribbler, and closing the connections when the user is done (Figure 12).
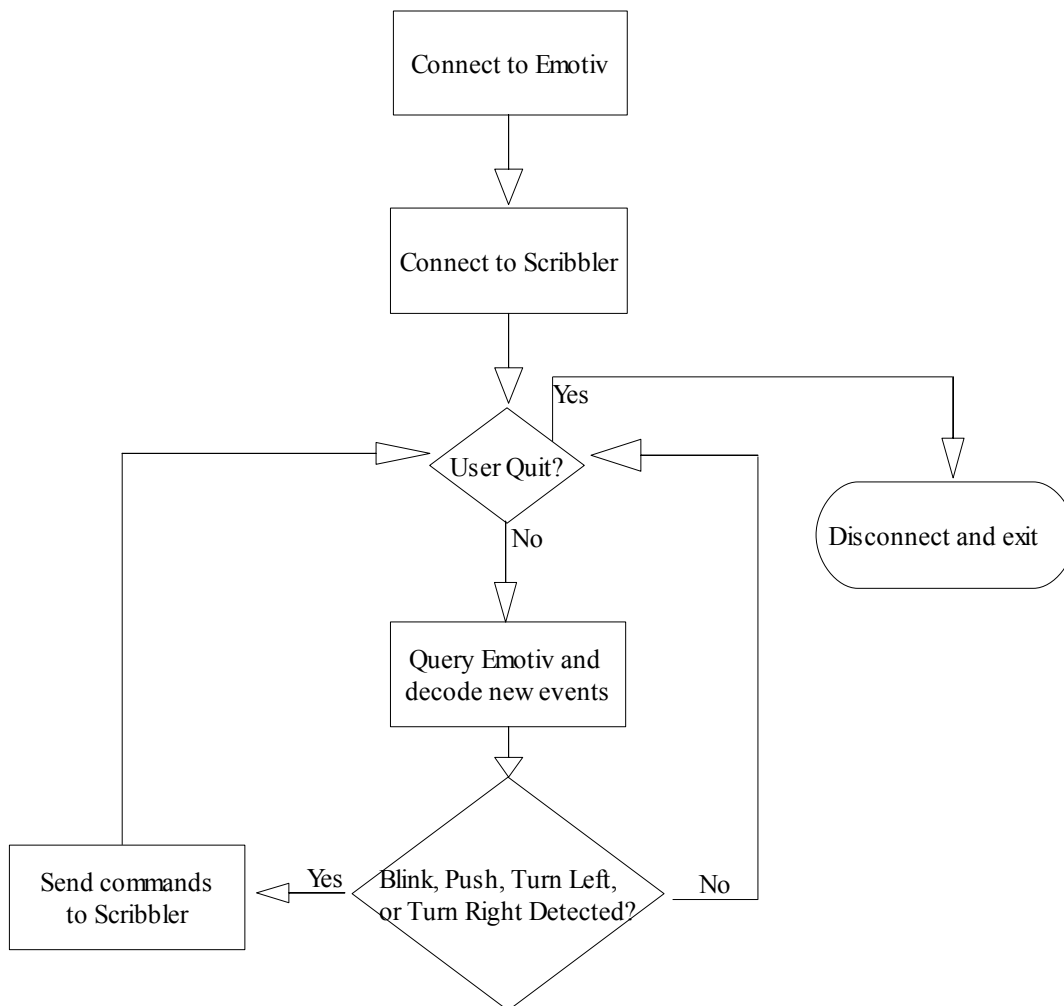

Figure 12: *High-level View of the Control Scheme*

6.1   Emotiv Connection

Connecting to the Emotiv allows for two basic options for controlling the robot: using thoughts detected from the actual Emotiv EPOC headset or using the mock signals from the EmoComposer emulator. The EmoComposer acts as an emulator that the Control Panel can connect to and allows the user to send per-recorded built-in data signals that match possible inputs from the real headset. I coded the program to allow access to the EmoComposer as it allowed me to initially use clean signals from the EmoComposer to make a quick and testable prototype scheme before I had trained the headset, however once the headset was trained I switched to using actual headset signals as inputs. The Users Manual suggests using the EE_EngineConnect call to connect directly with the headset and using the EE_EngineRemoteConnect otherwise, however I found that the EE_EngineConnect always returned true, even if the headset was off.[30] Instead, I decided to use EE_EngineRemoteConnect to connect through the Control Panel which enables communication with the headset and querying its power and connection status. Thus my connection calls for utilizing the EmoComposer or the headset differ only in the port and IP:

```
EE_EngineRemoteConnect(EmoControlPanelIP.c_str(), headsetPort)

EE_EngineRemoteConnect(EmoComposerIP.c_str(), composerPort)
```

6.2   Scribbler Connection

Initializing the Scribbler robot consists of three steps: initializing Python, loading the Myro libraries, and connecting to the robot using the Myro's initialize() command. The Python.h library allows C/C++ code to embedding Python and make direct calls to the Python interpreter. Initializing the Python interpreter and loading the Python dictionary requires four lines of code:

```
Py_Initialize();
PySys_SetArgv(argc, argv);
```

---

30 *User Manual.* pp 37-38

19

```
main_module = PyImport_AddModule("__main__");
main_dict = PyModule_GetDict(main_module);
```

From here, any code can be sent to the Python interpreter directly using the PyRun_SimpleString

function. Furthermore, C++ code can construct PyObject pointers to reference Python functions stored

in the Python dictionary. These functions can be called via the PyObject_CallFunction which passes

back another PyObject pointer which is the return value of the called function. Thus full embedding

functionality is possible complete with function calls, parameter passing, and return values when

embedding Python in C/C++.[31]   Loading the Myro libraries consists of only one line of code:

```
PyRun_SimpleString("from myro import *");
```

however it is important to remember to update the C++ reference to the now updated Python dictionary

so that the Myro functions can be called. From there, the user inputs the Bluetooth out com port that

the Scribbler is connected to and the Myro initialize() command is called which allows the Python

interpreter to send commands directly to the Scribbler:

```
PyObject* initFunction = PyDict_GetItemString(main_dict, "initialize");
PyObject_CallFunction(initFunction, "s", port.c_str());
```

6.3   Decoding and Handling EmoStates

There are four major steps in reading and decoding information from the EPOC headset:

creating the EmoEngine and EmoState handles, querying for the most recent EmoState, deciding if this

is a new EmoState, and decoding the EmoState. The EmoEngine handle allows for queries to get direct

input from the headset including contact quality, raw electrode input, and the connection quality. New

EmoStates are constantly created by the EmoEngine which represent recognized actions such as facial

expressions, changed emotional status, and detected thoughts and can be queried through the EmoState

handle. First the EmoEngine handle to query for new EmoStates and the EmoState handle used in

---

31 van Rossum, Guido. *Python/C API Reference Manual.* Python Software Foundation 21, February, 2008.
     <http://docs.python.org/release/2.5.2/api/api.html>. Accessed 14, April, 2010.

determining what was detected are allocated using the Emotiv API:

```
EmoEngineEventHandle eEvent = EE_EmoEngineEventCreate();
EmoStateHandle eState = EE_EmoStateCreate();
```

From here, the EmoEngine handle is queried to retrieve the most recent event using:

```
EE_EngineGetNextEvent(eEvent);
```

This should be polled 10-15 per second to ensure real-time reactions to the users thoughts/actions.

Next the program determines the event type returned by the EmoEngine. There are three categories of

event types: hardware-related events, new EmoState events, and suite-related events.[32] If the event

represents a new EmoState, the code retrieves the EmoState, the user's ID number, and the time-stamp

of the event so that the event can be decoded:

```
EE_Event_t eventType = EE_EmoEngineEventGetType(eEvent);
...
if (eventType == EE_EmoStateUpdated) {
       EE_EmoEngineEventGetEmoState(eEvent, eState);
       const float timestamp = ES_GetTimeFromStart(eState);
       decodeState(userID, eState, timestamp);
}
```

In decoding the EmoState, I look for eight possible events. First, I check whether the headset

was disconnected or reconnected. If the headset was disconnected, I suspend all activity until a

reconnect EmoState is published. I originally had the program terminate if the headset disconnected,

however I decided to alter the scheme to allow for the user disconnecting/reconnecting without having

to restart the program each time since I often had trouble maintaining a lengthy connection with the

headset. I used the `ES_GetWirelessSignalStatus(eState)` and the `ES_GetHeadsetOn(eState)`

calls to determine whether or not the headset was still connected. If the event was not related to the

headset disconnecting/reconnecting, I check to see if the event was a blink using the

`ES_ExpressivIsBlink(eState)` call. I maintain a global variable that keeps track of when the last

blink is recorded and whenever a new blink is detected I compare it with the time-stamp of the new

---

32 *User Manual.* pp 38

21

blink to determine if the user double-blinked. If so, I have the Scribbler take a picture by calling the

Myro Python commands:

```
PyRun_SimpleString("show(takePicture())");
PyRun_SimpleString("stop()");
```

It is necessary to call the `stop()` command so that the picture frame returned by the Python

interpreter is refreshed.

If the event was neither a blink nor a signal that the headset had disconnected or reconnected, I

look for three possible cognitive thoughts: Push, Turn Left, and Turn Right. To do this, I cast the

EmoState as a CognitivAction, get the specific integer value of the action, and get the power of the

thought by utilizing the Cognitiv Suite:

```
EE_CognitivAction_t actionType = ES_CognitivGetCurrentAction(eState);
float actionPower = ES_CognitivGetCurrentActionPower(eState);
int power = static_cast<int>(actionPower*100.0f);
int action = static_cast<int>(actionType);
```

From here, the integer value of the action can be compared to the defined CognitivAction enumerator

stored in the EmoStateDLL.h file (Table 3). Once I have decoded which thought sparked the

EmoState, I send the appropriate call to the Scribbler (Push → Move Forward, Turn Left → Turn Left,

Turn Right → Turn Right). If the thought is not a Push, Turn Left, or Turn Right thought I ignore it. I

initially experimented with using the power of the thought as an input to the power of the robotic

action, however I found that this control scheme was too difficult to use and it ended up being far more

intuitive to use specific values for turning and moving forward no matter the thought-power. This

allowed the user to concentrate on the thoughts alone and not have to additionally worry about how

"hard" to think the thoughts.

Action	Value
Neutral	0x0001
Push	0x0002
Pull	0x0004
Lift	0x0008
Drop	0x0010
Left	0x0020
Right	0x0040
Rotate Left	0x0080
Rotate Right	0x0100
Rotate Clockwise	0x0200
Rotate Counterclockwise	0x0400
Rotate Forward	0x0800
Rotate Backward	0x1000
Disappear	0x2000

Table 3: Predefined Cognitiv Action Enumerator Values

6.4 Modifications

One of the first early problems I recognized was the disparity between the rate of input EmoStates and the time it takes the Scribbler to perform actions. The internal sampling rate in the EPOC headset is 2048Hz. This is filtered to remove artifacts and alias frequencies and then down-sampled to approximately 128Hz.[33] Any given motion input to the Scribbler using Bluetooth takes approximately 2 seconds to execute, while picture taking take slightly longer as it has to capture and send data back. My initial tests of my control scheme failed to account for this time differential in input sampling and possible output execution and consequently failed to offer a usable interface. Instead of the Scribbler responding to my thoughts, EmoStates and corresponding actions for the Scribbler almost instantly queued up while the Scribbler was still executing the first command, and the

---

33 Emotiv Research Plus SDK Headset Specifications. <http://www.emotiv.com/ResearchPlus.html>.

result was a robot that seemed to make random actions as it executed thoughts that had taken place seconds and even minutes ago instead of responding to current thoughts. To solve this problem I introduced a sampling variable to only decode one in every ten input EmoStates. The rational behind creating this variable is that the 128Hz input rate is so fast that inputs like a Push thought or even an eye blink will create numerous EmoStates. Using my sampling variable I filter out those extra states that really only correspond to one event by using a sample rate small enough that it will still capture events which send more than 10 input EmoStates while sending only one command to the Scribbler instead of queuing up 10. This system worked much better, and even had the added bonus of filtering out "noise" thoughts when the headset detected a push or turn thought for a fraction of a second. Despite this, adding the sampling variable had the unintended consequence of creating the possibility of missing a headset disconnect/reconnect as these events create only one EmoState each. To solve this problem, I moved the checking of the sample variable and settled on decoding every single EmoState so that I could catch the disconnect/reconnect while filtering thought and blink EmoStates using the sample variable to ensure the Scribbler acts on current thoughts.

One other modification I made after this scheme was created was to add an additional operation mode. I decided to add this as it would often be useful to be able to send more than three commands to the Scribbler. However, if I wanted to take the same approach as before and hook up thoughts to robot actions I had to contend with the added difficulty in recognizing more an additional thought for every new action I wanted to add. To solve this problem, I created an additional mode which remaps the same three input thoughts to different outputs in the robot. This is hugely beneficial as it does not increase the difficulty in recognizing new thoughts and also does not require the user to train additional thoughts, thus giving double the usability with only one additional input. This additional input is raising the eyebrows which toggles between the original and the new mode. I decided on utilizing the raising of eyebrows as a toggle as it is very easily trained and accurately recognized by the headset.

The new mode maps Push to the Scribbler moving forward until it detects a wall, Turn Left to rotating approximately 90° to the left, and Turn Right to rotating approximately 90° to the right. Having the Scribbler move forward until it detects a wall is accomplished by sending a loop to the Python interpreter which has the Scribbler continually use the Fluke board to check if there is an obstacle in front of it and move forward if no object was detected:

```
string command = "while(getObstacle(1)<1050): forward(.5,1)";
...
PyRun_SimpleString(command.c_str());
```

Upon seeing how successful this was I also added a check to the initial move forward command so that when the user commands the Scribbler to go forward into a wall the robot instead gives feedback to the user explaining there is an object in the way. The addition of more modes is certainly possible and is an excellent way of adding functionality without adding the cost of recognizing and learning new thoughts. In the end, it was completely feasible to control the Scribbler robot using the EPOC headset proving the viability of EEG based BCI technology.

7. Blink Detection

Having explored the Device/Application Control and the Feedback portions of the BCI Design Pattern by creating my Emotiv-Scribbler Control Scheme, I next decided to explore the Pre-Processing, Feature Extraction, and Classification of EEG data by analyzing eye blinks (Figure 2). The analysis of eye blinks is useful in BCI development for two reasons: eye blinks can be used as control inputs (as in my control scheme), and if they are not they must be filtered out lest they corrupt the useful EEG data. I furthermore decided on eye blinks as they are immediately recognizable in the AF3 channel (Figure 13). Focusing on only one channel allowed me to immediately reduce the amount of input data by a factor of 14 since I could discount the other 13 input channels.
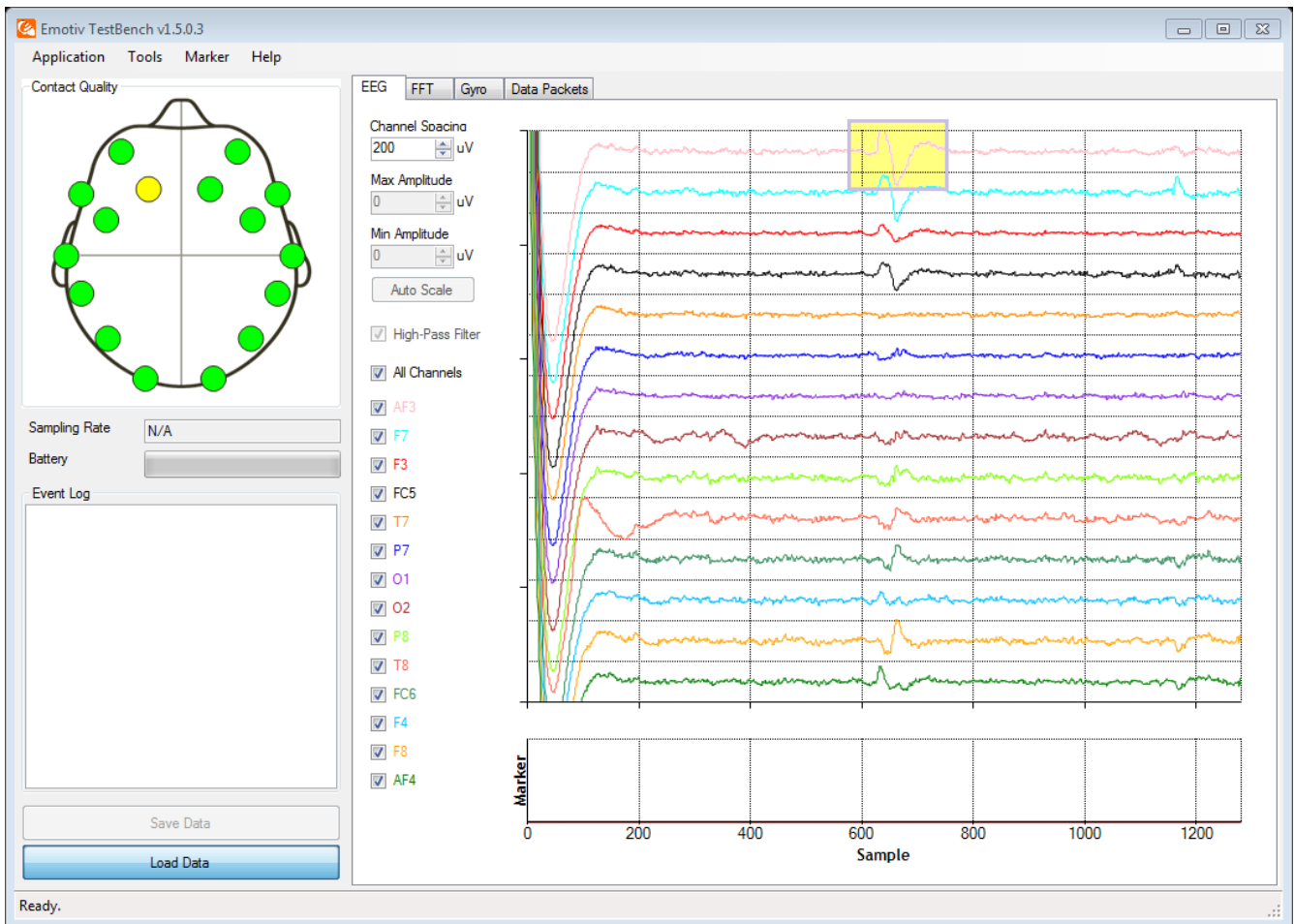
Figure 13: *A Blink Event Highlighted in the AF3 Channel*

Reducing the size of the data set is the primary focus of pre-processing and feature extraction whose goal is to get rid of extraneous and noisy data while preserving data that can best differentiate between classes. However, before I could do this I first needed a data set and thus recorded twenty ten second clips, ten of which I blinked during and ten of which I didn't. I then exported the clips to CSV format so that I could load them in MATLAB. These recordings produced a lot of data because in addition to the 14 EEG channels capturing electrode data at 128Hz the headset also records gyroscope data, battery data, packet information, etc. and each 10 second clip ended up had roughly 36000 data points and combined I recorded 720000 data points (Figure 14).
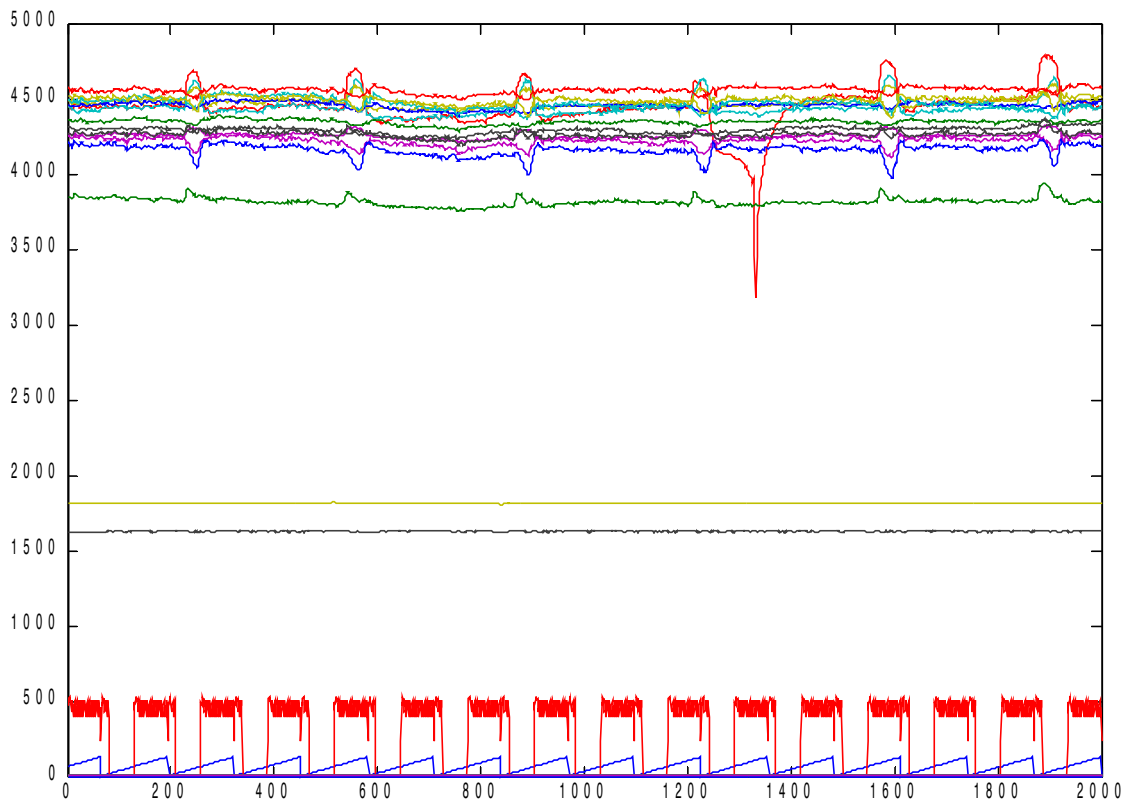
Figure 14: *One 10-second Segment with Blinks and Noise*

The first step of my feature extraction was to use just the AF3 channel where blinks are clearly

visible. I was curious as to how simply extracting this channel would work for blink classification and

decided to run the 20 test clips through a neural net to see how it would fare. The classification using

MATLABS' nprtool to create a two-layer feedforward neural network with backpropagation obtained

only a 65% accuracy (Figure 15). The reason for this became apparent when I looked at a plot of all

the AF3 channel data (Figure 16). It is clear that, though there is a certain pattern to blinks, the neural

net was thrown off because the blinks were not normalized with respect to time. The neural net treated

time as an attribute, and thus might not classify two samples that both contain blinks but where the

blinks occur at different times. Time is correlated to blinks in respect to how long the blink takes and

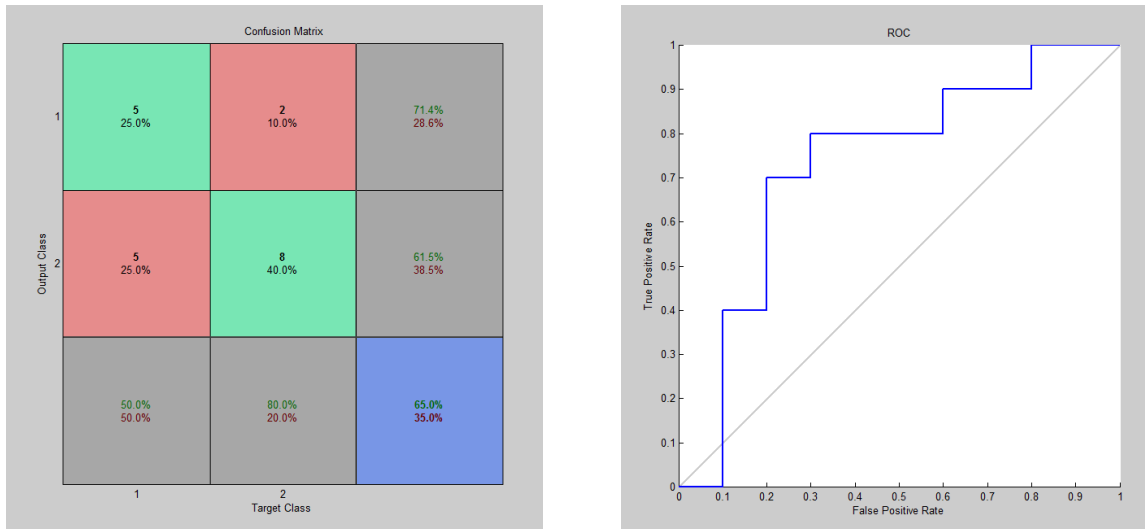thus how wide the blink spike will be, however the time that the blink occurs is not a usable attribute.

Figure 15: *Initial Neural Net Results showing both a Confusion Matrix and Receiver Operating Characteristic (ROC) Plot*
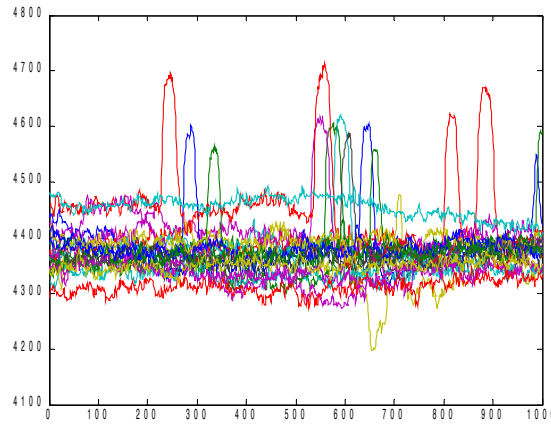

Figure 16: *Twenty 10-second Recordings of AF3 Data*

To solve this problem, I decided to further reduce the amount of data the neural net worked with along with normalizing any blinks found. Recognizing that blinks correlate to spikes in EEG data, I scanned each 10 second clip looking for the largest spikes. I found that blinks typically were represented by a surge in the 4500 to 4800 μvolt range over approximately .59 seconds and were followed by a characteristic dip of around 50 μvolts over approximately .20 seconds (Figure 17). This pattern was very clear and easily distinguishable from a non-blink state; I first noticed it when applying unsupervised K-Means Clustering to detect naturally occurring patterns in the data (Figure 18).
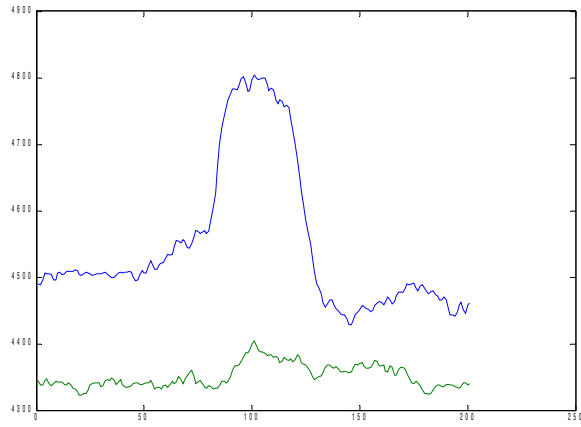
28

Figure 17: *1.6 Second Recording Showing a Blink Pattern (blue) with Non-Blink Data (green) for Comparison*
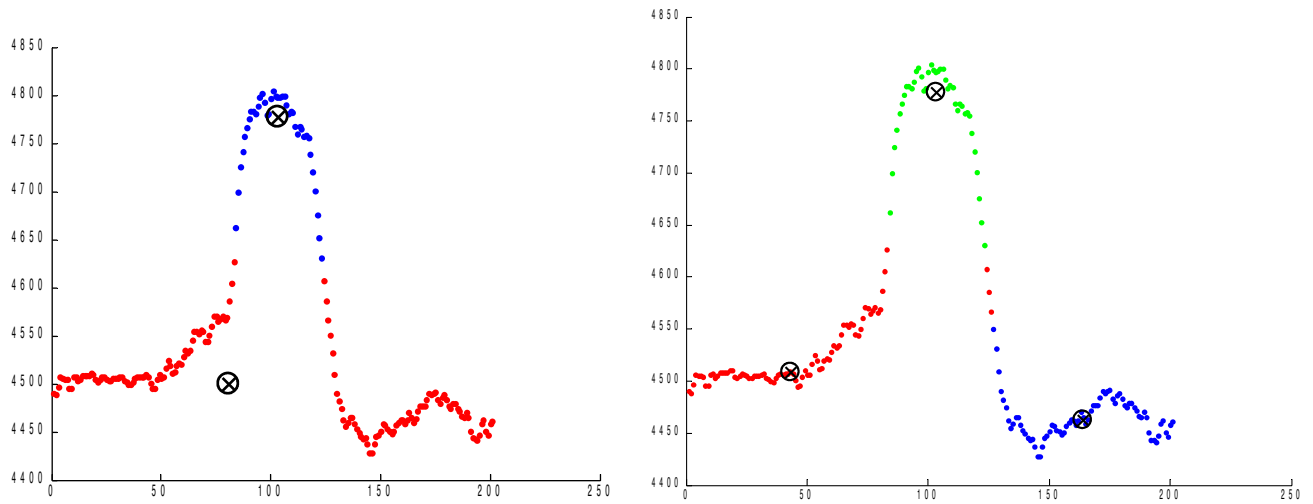

Figure 18: *Unsupervised K-Means Clustering Exposes Naturally Occurring Blink Patterns (2 Clusters Left, 3 Clusters Right)*

I used this information to further filter each of the 20 inputs down to 1.6 second segments, each of which highlighted the maximum spike of the original ten second segment. This normalized the blinks by having each blink start at roughly the same time and additionally filtered out noise that was unrelated to the blinks creating data that was much easier for a neural net to distinguish (Figure 19). Using these inputs, neural net accuracy improved to 100%, however I wanted to see if this system truly recognized blinks or was over-trained on the input data. Therefore, I recorded five more segments, 3 with blinks and 2 without, and followed the same pre-processing/feature extraction steps and fed the data to the neural net. The neural net accurately predicted all of these new inputs even though it had not been trained upon them, showcasing that it was truly was extendable and actually recognizing blink patterns. These are very promising results that prove the feasibility of utilizing a neural net to classify blinks, however it would be best to obtain a larger sample size to accurately test the classification performance of this scheme.
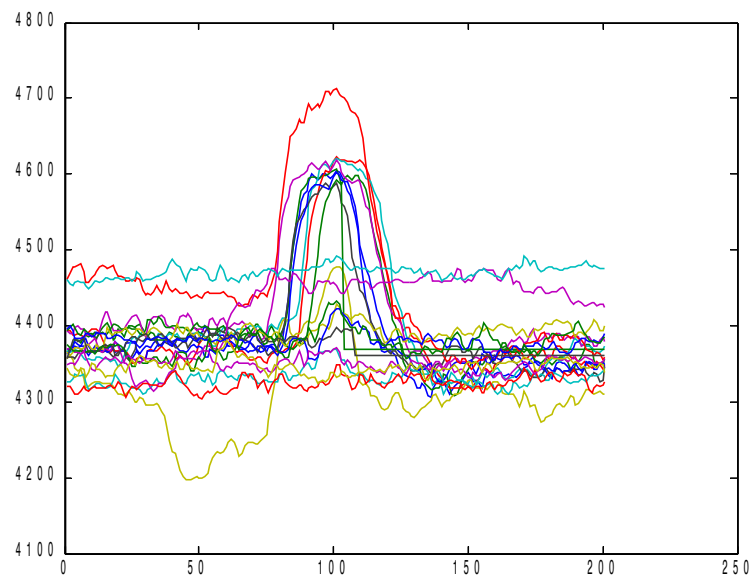

Figure 19: *Fully Processed and Normalized Input to the Neural Network*

8.   Conclusions

Having explored all phases of EEG-based BCI construction and implementation, I can safely say that it is a feasible system that will likely only improve and become more widespread in the future. The biggest flaw I encountered in the technology was interference with the wireless headset which often had difficulty connecting and staying connected, however Emotiv has informed me that this is likely a hardware problem with the headset and will be sending me a new one. I did my research using a Beta version of the EPOC headset so it is possible that the Consumer version they will send me will address this problem.

The system I constructed was largely a success as I was able to create a system whereby I could control a robot with my thoughts and further created accurate blink-recognizing software. Furthermore, my system showcases the amazing possibilities of BCI's in aiding the disabled. For instance, a person who could only move their head could certainly use my system to control a motorized wheelchair accurately using their thoughts. In addition, had they a computer built into the headset, they could easily switch modes by raising their eyebrows and then use their thoughts as an input to the computer, by using the same thoughts that had moved the wheelchair to control the mouse and double-blinking to click. An alternative would be to keep the thoughts controlling the wheelchair while utilizing the gyroscope in the headset to control the mouse, enabling the user to have simultaneous control of the wheelchair and computer.   Further research can certainly lead to direct implementation of such systems and can explore the recognition of thoughts beyond those included in the Emotiv API. In the end, the research demonstrates just the tip of the iceberg in what is truly the limitless potential of EEG-BCI systems.

9.   Acknowledgments

I would like to thank the Computer Science Department at Boston College for purchasing the Emotiv Headset and Research SDK, without which my research would have been impossible. Furthermore, I extend my gratitude towards the Emotiv Team who have developed a great cost-effective product for anyone looking to get into EEG research and who have answered many of my questions on their closely monitored forums. Finally, I am indebted to Robert Signorile who served as my adviser and consistently offered his insightful input and enthusiasm for the project.