EPOC-alypse Mind Controlled Car

Senior Design II Project Documentation

GROUP 21

Group Members:

Christopher Perez

Lee Sully

Kathryn Morales

Michael Strobridge

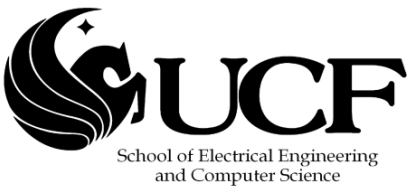


Table of Contents

1: Executive summary

The idea of having the power to control physical objects with your mind has been a fantasy of many people ever since first watching Star Wars. So for all of the people who have ever wanted to use the force, this project is right up your alley. The project that will be undertaken will be a small remote controlled car with an onboard computer that will interpret EEG (electroencephalography) readings from a headset worn by the user, and, depending on the type of brain activity detected, either move the car forward, backward, left, or right. The reason for incorporating EEG is to explore the emerging field of brain computer interface (BCI). Until recently the field of BCI has been primarily focused on that of neuroprosthetics applications that aim at restoring function to damaged parts of the body, but now, commercially available headsets make it possible for the field to broaden its view. Theses commercially available headsets are intended for use with videogames and integrating with the user's ordinary computer allowing for endless possibilities.

The main objective of the project is to design a functioning car that will respond to a user's brain activity and react accordingly. Being used will be an Emotiv EPOC EEG headset to gather the necessary brain activity to drive the car. To run our custom architecture we will be using the Intel DH61AG mini-ITX motherboard with in Intel core i3 processor to run the Emotiv software and run a translation program to convert the EEG signals into a 6 bit format that will be able to be read by the custom PCB on the car. The car itself will house a custom PCB that will contain an Atmega 328 chip, dual h-bridge motor driver, and an Xbee series 1 for Bluetooth communication between the car and the motherboard.

This paper describes how each of the components listed above were researched and how they were implemented, including a budget and a timeline for finishing the EPOC-alypse mind controlled car for the final senior design presentation. In order to make the car respond as accurately as possible training on the Emotiv headset is essential. The ability to focus your mind and activate certain areas of the brain on command is the key to making this entire project work.

No experts or guidance from anyone who has used the headset before or who has in depth knowledge on the brain were referred to during the course of this project. This resulted in a well refined knowledge of how specific areas of the brain worked and what the thoughts that controlled the car actually meant.

There are also requirements from the hardware and software interfaces that will be dealt with, these are listed in the body of this paper. All these requirements led us to our budget, which since our group is unsponsored, was initially a fairly small number due to the financial situation of team members. This, however, changed dramatically due to the

hardware that was required to run all of our systems properly. This is outlined more in the budget section of the paper.

2: Introduction

2.1: Motivation

Brain computer interface seems to be narrowly focused on the medical applications, but a much broader applicability of BCI exists than just medical uses. This project aims to expand into the realm of alternative uses for BCI by applying it to everyday activities. As more is understood about the brain and how it works, the more its power will want to be harnessed. Manipulating physical objects just by thinking is the ultimate goal for this area of interface. As a group of electrical and computer engineers the prospect of this is too enticing to pass up, therefore a mind controlled car was the perfect choice. The ability to apply this project to people who no longer and move their arms and legs was also a driving factor, just by simply replacing the remote for the car with the joystick for a wheelchair someone who is confined to a bed or otherwise immobile can become mobile.

2.2: Project Objective/Goals

The objective of this project, on paper, is quite simple: to build a remote controlled car that is controlled by your mind by using the Emotiv EPOC EEG headset. We came up with this project due to a unanimous interest by all members in the group.

Our goal is to control the car using thoughts via Emotiv EPOC.

- Control the car using a mini ITX board
- Get the commands from Emotiv EPOC and process them.
- Design an architecture to connect both the ITX and Emotiv, and that is extendable to incorporate multiple devices.
- Establish adequate connections and fine tune the signals for smooth controlling of the car.

The initial purpose of this project is to practically solve a way to manipulate physical objects with your mind. The main goal will be to navigate the car using the headset through a simple obstacle course. This course will consist of left and right turns on a circular track.

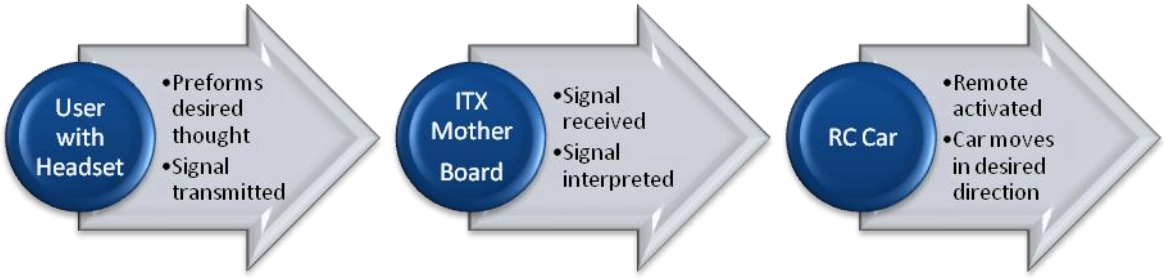


Figure 2.1 Basic flow diagram of how the EPOC-alypse car will work.

There are also many personal goals associated with this project as well. Many of which included getting hands on experience with the various topics and technologies, which would like to be pursued further in future careers. The more experience that is gained while working with a brain computer interface such as Emotiv's, the more we can explore the emerging technologies that incorporate the human brain.

2.3: Specifications

Below is a brief list of specifications that are a necessity for the project. These guidelines are the main aspects of the design that will have to be obtained for the project to work. Everything else can be changed.

Headset proficiency	Control four different actions Mastery of isolating thoughts
Motherboard	2.4 GHz processor 1 GB RAM 50 MB disk space USB 2.0 port

Car	Range of 40ft Full range of motion

(Table 2.1) Specifications table

2.4: Previous Works

Using the power of the human mind to control everyday objects had always been a fantasy for people. The possible applications and benefits of reading and interpreting brain-waves are endless. Examples of possible uses include medical devices for doctors and patients, new ways to control devices without a remote control, etc. Recently, however, several projects have arisen that have started to make this fictional desire into a reality. Projects such as the BrainDriver, the SWARM Extreme, RC Mind Control, and Brain-Controlled NXT Robot are beginning to use brain-waves as a means of allowing a user to control devices and applications. These projects are very similar to this project because they incorporate the use of an EEG headset and use data from it to control a device with brain waves or facial expressions.

2.4.1: BrainDriver: A Mind Controlled Car

At the Freie Universitat Berlin, Raul Rojas an AI professor have demonstrated how brain interface can drive an actual car. The project uses the Emotiv neuroheadset that feeds the commands received from the driver to the drive-by-wire system installed on the vehicle. The thoughts control the engine, breaks, and steering. While nifty the BrainDriver application is still a demonstration and not road worth because there is a two to three second delay in the response time. But the researchers say that future applications could be an autonomous cab ride, where a passenger could decide which route to take when more than one possibility exist.

2.4.2 The SWARM Extreme

The SWARM Extreme was a project carried out at Northeastern University in 2011 (Bothra). This project used the Emotiv EPOC headset to control an AR Drone without the use of a remote control. The processor used for this project was an ARM9 processor which was embedded in the AR Drone and interpreted signals sent from the laptop running the Emotiv software. The software design consisted of a client/server

architecture that used the signals received from the headset and processed them into usable signals for the drone (Bothra). A buffer was used in the client in order to prevent over-whelming the Drone with too many commands. The project used a library written in Python to hack the Emotiv software. By using this software, it was possible to attain raw EEG data from the EPOC headset.

2.4.3: The RC Mind Control Project

The RC Mind Control project was carried out in The Center for Research in Arts, Technology, Education, and Learning (CRATEL) at Wichita State University. Initially, this group attempted using the NeuroSky headset, but found that its capabilities were not sufficient for this project. They found that the NeuroSky headset could only read brainwaves from one side of the brain. This made it very difficult to get usable and accurate data. After discovering this, they decided to use the Emotiv EPOC headset due to its more mature development and more accurate readings. A very important feature of the EPOC headset was that it provided them with enough data points to incorporate research done by Jonath R. Wolpaw and Dennis J. Mcfarland by using an equation that was determined by these two authors. The equation allowed them to calculate the direction in which the RC car was supposed to move by using the amplitudes of the signals sent by the EPOC headset. This group decided to use an Arduino board to communicate with the remote control of the car. Instead of connecting the Arduino to the circuit on the car, they decided to connect the Arduino to the actual remote control. The Arduino then sent processed signals to the remote control which were then sent to the car.

The operation of this prototype required the user's level on concentration to be above a level of 0.599. The level of concentration as well as the position of the user's head were determined using the various suites included in the Emotiv SDK as well as the built-in accelerometers. Once the level of concentration was interpreted to be above .599, the Arduino processed the received signal and sent the proper signal to the remote control. The direction in which the car moved was determined by the position of the user's head. For example, to move the car forward, the user had to tilt his head up. After a signal was received, the software waited for the next signal from the headset to reach the Arduino board. When the user wished to stop the car, he simply relaxed his level of concentration and tilted his head back to a centered position.

2.4.4: The Brain-Controlled NXT Robot

The Brain-Controlled NXT Robot was a project carried out by a PhD student at Madeira Interactive Technologies Institute. Although this project does not control a car, it is a similar project to ours because it uses the EPOC headset to control a Lego Mindstorms NXT Robot. This project used an ARM7 microprocessor which was embedded in the robot. This project consisted of two prototypes. One of the prototypes used the NeuroSky headset and the other used the EPOC headset. This was done in order to show the different features and functionalities of the two distinct headsets. The NeuroSky headset used an algorithm provided by Neurosky in order to get attention and meditation levels of the user. The patterns that were found by the headset were then used to send commands to the robot. The Emotiv EPOC headset, on the other hand, used facial expressions as well as frustration, excitement and other emotions in order to produce data.

The software used on the ARM processor included a Java Virtual Machine called LeJOS. This was used as the OS for the project because it provided the "extensibility and adaptability of JAVA" (NXT ROBOT). This was important for this project because Java was the programming language used in this project with the use of Eclipse as the IDE. In order to connect the computer and the robot, a Bluetooth connection was used along with the pccomm and bluecove libraries found in the LeJOS API. This project used a very similar technique as the RC Mind Control project because they used a threshold value to determine when the signal sent by the headset was strong enough to be considered intentional. In order to determine the threshold value, readings were taken for concentration levels while calm, normal and concentrated. These values were used to determine a level that would certainly determine when the user was concentrated. This experiment was conducted with five different users, thus making the results more general for all users. When the concentration level dropped below the threshold, the robot was instructed to stop moving. In order to monitor the attention levels of the user, the NeuroSky headset used the Mindset development tools along with a software program called LiveGraph, which plotted the concentration levels of the user with real-time data.

After researching many projects, these three examples were most relevant to the project that this group will be designing. Different headsets were analyzed as well as different devices such as cars, drones, boats, etc. This was very useful for this project because it helped minimize time wasted on ideas and procedures that would not have worked. By looking at the implementations of other projects, ideas were found that will allow this project to be more successful.

3: Project Definition and Scope

3.1 Definition and Scope

The EPOC-alypse mind controlled car is a project that combines the efforts of both computer software and hardware components. Both areas of study work together in this project in order to control a remote control car by interpreting signals sent from the human mind as well as facial expressions. The final functionality of this project is to be able to run the developed software and control the car without having to use any remote controls or having any other human interaction with the car. The scope of the project is a final product that can function as stated above and that can be finished within one semester of research and one semester of work.

3.2 Span

The span of this project consists of many fields of study including device communication, signal processing, hardware manipulation and many others. Device communication is used in order to send the signals from one component to another. Without this knowledge, it would have been impossible to accomplish the main task of the project. Signal processing was used in the software aspect of the project. The signals received from the headset were processed and made into useful signals for the car to use. Hardware manipulation is a very important aspect of this project. The car that we used in this project has a custom PCB attached to it which contains many components that are specifically used for the intentions of receiving signals from the motherboard and transmitting the correct voltages to the correct wires on the car. This PCB controls the motors which drive the car.

3.3 Assumptions

As with any project, there were some assumptions made while making this project. First of all, it was assumed that the environment in which the system is used will not cause any significant interference with the signals being sent by the headset. The signals being sent to the motherboard need to be considered valid and if any signals are corrupted or lost, there is the possibility that the car will behave unpredictably. Also, it was assumed that the user will have a complete mastery of the headset prior to using the project. Although it would be beneficial to assure that anyone can use the project, unfortunately the Emotiv EPOC headset requires much training before the signals sent by the user can be considered reliable. This is, of course, is something that we cannot change and therefore we made the decision that the one of the group members would go through the necessary training and wear the headset during any presentations and demonstrations.

3.4 Requirements

There were a couple of specific requirements that needed to be fulfilled by the end of the project. The most basic requirement was a functional car that would be able to respond to the signals sent by the EPOC headset. The other requirement for the project was that it would be able to work as long as it was kept in range. The receiver on the car and the transmitter on the motherboard have a maximum range that ensures reliable signal transmission. The car needed to be able to function as long as it remained within this range, meaning that there could not be any bugs or unhandled exceptions in the software. There were no requirements for the speed of the car.

3.5 Measuring Success

The overall success of this project was measured by how accurate the car's actions were compared to the signals sent by the user. This meant that both correct actions and response times were taken into account. In order to achieve these results, it was important to have hardware components that met all the requirements necessary to run the Emotiv software as well as transmit all the signals from the headset to the motherboard and from the motherboard to the car. It was also important to develop software that would be able to interpret and process signals efficiently enough to produce real-time execution.

4 Research

Brain-computer interface is a direct communication pathway between the brain and an external electronic device. The goals of BCI are often directed at assisting or augmenting human cognitive or sensory motor functions. Recently neural interface devices have become available on the market for gaming and other virtual uses. Neurosky, a company who offers one of these headsets gathers raw EEG data to use with applications on iOS and Android platforms. Their headset contains one sensor on the frontal lobe to collect multiple mental states. As the company puts it “the physics of brain waves is virtually identical to the physics of sound waves where a single microphone can pick up the complexity of the concert.” Another company who is in the same market is Emotiv, the headset this project utilizes. This headset contains 14 different sensors that are divided up into 7 pairs, which makes it better for reading the cognitive thoughts of a person.

So what are EEG (Electroencephalography) signals exactly? EEG is essentially the recording of electrical activity across the scalp, measuring the voltage fluctuations resulting from ionic current flows within the brain. These ionic current flows are maintained by neurons which obtain their charge from membrane transport proteins that act as simple pumps transporting ions across the cells membrane. So the way the headset picks up readings is that when one neuron releases a large amount of ions, these ions can push against other neurons, which push against others and so on. This is known as volume communication, and when that wave reaches the electrodes on the EEG detector they can exert a force on the metal inside each electrode. This difference in pushing or pulling on the metal between two electrodes is recorded as the EEG reading.

The Emotiv EPOC EEG Neuroheadset has 14 saline felt based electrode sensor receivers. Each sensor is ideally mapped and conveniently placed in the 14 different areas and lobes of the brain for optimal measurements. All of these 14 lobes and areas are divided into specific regional areas each has different functional aspects.

- The FRONTAL LOBE : Planning, Language, Expression and Speech contains the Motor cortex area involved in movements (movement, conscious thought , and controls voluntary movement of body parts),
- PARIETAL LOBE: Touch, Taste contains the Somatosensory cortex areas (receives and processes sensory signals from the body) ,
- OCCIPITAL LOBE visual area (contains the visual cortex) receives and processes signals from the retinas of the eyes,
- TEMPORAL LOBE :Language Reception.

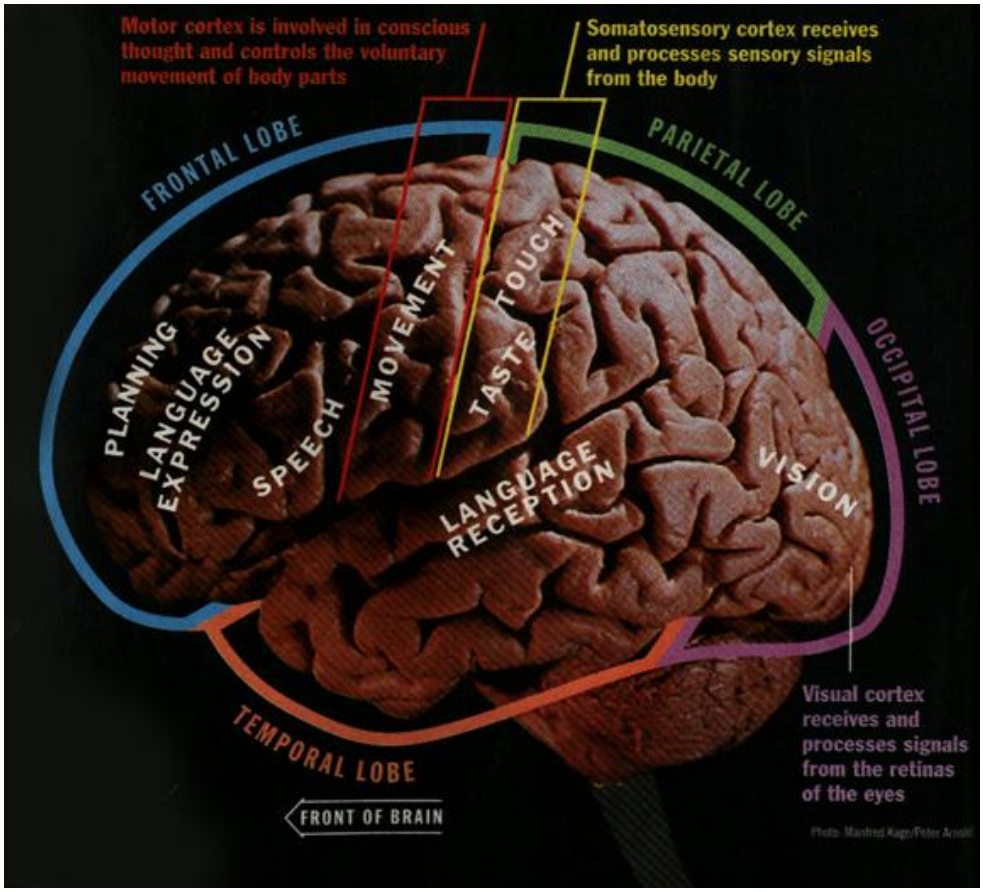


Figure 4.1 Lobe locations on the brain.

The 14 channels AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4 shows real time brain wave activity divided into common brain wave frequencies Alpha, Beta, Theta and Delta. The sense 14 locations are given in the Hardware overview of the Headset section which has a diagram of each of the sensors to be used and how they are each mapped to specific areas for different types of readings in the brain based on activity.

4.1 Brain Waves

These waves that are produced from the volume communication between neurons can be classified into; Delta, Theta, Alpha, Beta, and Gamma waves. There are also Mu waves but for or project these are irrelevant. The table below describes the frequency of each wave and what state of mind each occurs at.

Wave Type	Location	Frequency (Hz)	States of Mind
Delta	Frontal cortex	0 - 4 (high amplitude)	Asleep
Theta	Locations not related to task being preformed	4 - 8	Drowsiness, Idling, Arousal
Alpha	Posterior regions, either side of the brain	8 – 13	Relaxed, eyes are closed
Beta	Either sides of the brain but mostly in frontal region	13 – 30	Alert, working, anxious, busy
Gamma	Somatosensory cortex	30 – 100	Cross modal sensory processing (i.e. combining sight and smell)

(Table 4.1) Brain wave types and characteristics.

The project utilizes mostly Beta and Theta waves, thought delta waves seemed to present themselves more in the evening when the mind starts to start its night cycle.

While the type of brain waves that are exhibited are useful for initial training to try and obtain the proper mind set it is not what is being quantified for the EEG readings to control the car. This comes from the locations of the sensor pairs oriented around the skull. These areas are referred to as Brodmann areas.

4.2 Brodmann Areas

A Brodmann area is a region of the cerebral cortex defined based on its cytoarchitectonics, or structure and organization of cells. Many of these areas have been defined solely on neuronal organization and have since been correlated closely to a wide range of cortical functions. Below in figure 4.1is a rough diagram of where each area is located in the brain.

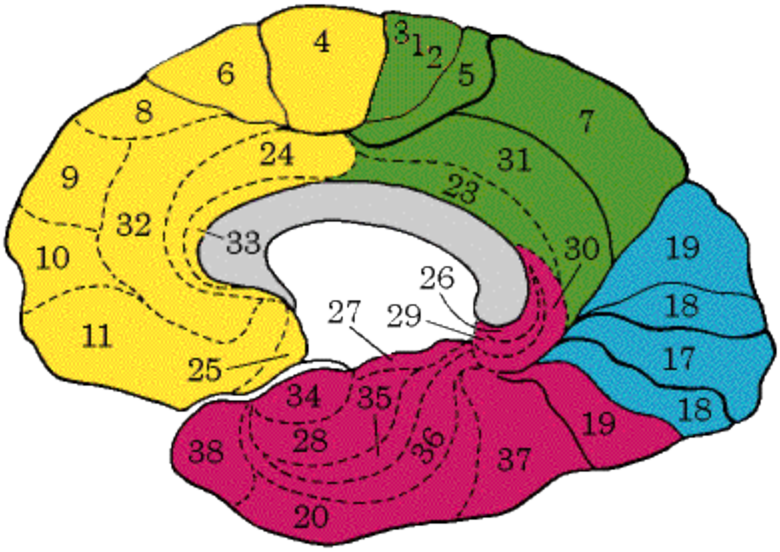


Figure 4.2 Brodmann areas.

German anatomist Korbinian Brodmann published his maps of cortical areas in humans, monkeys, and other species in 1909, but a more detailed map was published by Constantin von Economo and Georg N. Koskinas in 1925.

The Emotive headset is located over 7 of the most prominent of these Brodmann areas, the Dorsolateral prefrontal cortex, frontal eye fields, anterior prefrontal cortex, primary gustatory cortex, middle temporal gyrus, primary motor cortex, and the somatosensory association cortex. These areas are displayed in table 4.2 in relation to their location in the brain, and in table 6.2 referencing which sensors are over which area.

Area Name	Brodmann Area Number
Dorsolateral prefrontal cortex	9
Frontal eye fields	8
Anterior prefrontal cortex	10
Primary gustatory cortex	43 (not show on diagram)
Middle temporal gyrus	21
Primary motor cortex	4
Somatosensory association cortex	7

Table 4.2 Brodmann areas and locations

4.2.1 The Dorsolateral Prefrontal Cortex

Also known as DL-PFC is part of the “primate” part of the brain and actually consists of Brodmann areas 9-12, 45, 46, and 47. It acts as the highest cortical area responsible for organization, motor planning, and regulation. “It is responsible for integration of sensory and mnemonic information and the regulation of intellectual function and action.” It is involved in all complex mental activity but requires the additional cortical circuits that are connected to other regions.

.

4.2.2 Frontal Eye Fields

This area is located in the prefrontal cortex which is ultimately connected to the frontal cortex of the primate brain close to the skull. Also referred to as FEF the frontal eye fields control eye movements and visual attention. The area is activated during the initiation of eye movements such a blinking, rapid eye movements, and twitches. There is also evidence to suggest that the FEF plays a role in a purely sensory processing way as well. But together with the supplementary eye fields, the intraparietal sulcus, and the superior colliculus, the FEF controls all eye related movement.

4.2.3 Anterior Prefrontal Cortex

The anterior prefrontal cortex which is part of Brodmann area 10 which also includes the rostral and frontopolar prefrontal cortex, is the largest cytoarchitectonic area in the brain and not much is understood about this region. Best guess is that this area is involved in strategic processes in memory retrieval and executive function. It is proposed that ‘cognitive branching’ occurs here which enables a previously running task to be maintained in a pending state while the task at hand is comleted.

4.2.4 Primary Gustatory Gyrus

One of the least understood areas of the brain. No official information could be found.

4.2.5 Middle Temporal Gyrus

Located on the temporal lobe, the middle temporal gyrus’ exact function is unknown but has been connected with processes ranging from contemplating distance, recognition of known faces, and accessing word meaning while reading. It should be noted that this area is for reference while using the Emotive headset.

4.2.6 Primary Motor Cortex

The primary motor cortex work in unison with the premotor cortex, the supplementary motor area, and posterior parietal cortex to send synapses down the spinal column. It contains large neurons called Betz cells, which send long axons sown the spinal cord. The PMC contains a rough map of the body with different parts of the body controlled by partially overlapping regions of the cortex arranged from toe to mouth.

4.2.7 Somatosensory Association Cortex

Also known as SAC, the somatosensory association cortex has been linked to involvement with speech language and also movement. It has been seen that a reduction in SAC activity is a response to speaking, so the area appears to be activated by simple rather than complex sensory stimuli.

4.3 Reasons for Specific Testing Reactions

4.3.1 Temperature Change

During headset testing method tried to induce difference brain activity was drastically changing the temperature of an extremity. This was thought to produce a different pattern to help control a second action for the 3D cube in the Cognitiv suite in the Emotiv software, which is explained more in section 6 and 11. The guess was that by altering the temperature of a specific body part, a change in brain activity would present itself but to the tester's surprise, it did not.

Temperature responses in the brain are in the preoptic are of the hypothalamus which is in the midbrain tegmentum (split). It has been seen that "a mechanism is initiated to deliver more blood to this region to compensate for the temperature induced drainage of blood out of the brain. A compensatory reaction is generally observed in mammalian brains to supply oxygenated blood to accommodate increased oxygen consumption by cells." (Ogawa et al. 1992)

The problem with this was that the hypothalamus is too far inside the brain to register its activity on Emotiv's, sub-medical grade equipment. Therefore no stable or strong response was seen.

4.3.2 Music

Another method tested for trying to induce a different kind of response in the brain was having the user listen to different kinds of music. First off it has been that people like listening to music for the same reason they like eating or having sex. All 3 of those activities release the chemical called dopamine. Had this been researched before testing it could have been deduced just from that statement that music was not a wise choice for stimuli because it has been seen that brain patterns of a person eating or having sex are, while elevated, quite erratic in nature.

Researchers at McGill University in Montreal conducted a study with eight people who consistently felt chills from particular moments in some of their favorite music. PET (positron emission tomography) showed that the participant's brains pumped out a profuse amount of dopamine while listening to favorite pieces of music as opposed to just a slightly elevated amount when listening to other music. "It was seen that

dopamine surged in one part of the striatum during the 15 seconds leading up to a thrilling moment, and a different part when that musical highlight finally arrived."

This information backed up the tests conducted by our group when increased brain activity for specific genres of music showed up compared to others. The fact that this region of the brain is always elevated while listening to music was promising but eventually led nowhere because of the extreme changes in the elevated activity due to the uses specific taste in music.

4.3.3 Pain

On of the most least anticipated methods tried was that of inflicting pain on the user by the insertion of a thumb tack into a shoe and then having the tester put on the shoe.

First off, nociception is what the sensation of pain is called, and while pain seems to be related to the sense of touch, the two register on a completely different neurological level. Once pain is detected its sent to the spinal cord and up to the brain, (it is fascinating to note that the signal crosses over to the opposite side of the spinal cord and then is sent to the brain, so a feeling of pain on the right side of the body is a signal that climbs up the left side of the spinal cord) once in the brain the sense of pain is registered by many areas but most predominantly the somatosensory association cortex.

As stated above in the previous section, the somatosensory cortex is toward the top of the brain and close to the skull which makes is easily detectable by the Emotiv neuroheadset. Once this was seen in testing more was done to provoke a more intense response. It was found that when pain that is induced by tissue damage (i.e. cuts and bruises) the nociceptors become even more sensitive to pain, a phenomenon known as hyperalgesia. When cells are damaged histamine, serotonin, and prostaglandin are released into the area of the injury making the neurons send the sensation to the brain quicker. Ultimately the neurological response is the same, at least for minor injuries.

So because of the Emotiv Neuroheadsets position of sensors O1 and O2 right over the somatosensory association cortex , described more in section 6, the external stimuli was decided on being pain.

What was more fascinating that the very predominant display that pain had on an EEG, was that the brain started to mimic the pain response on the EEG without any pain being inflicted.

What could be gathered about this phenomenon was that, with the tester having conscious knowledge of how the pain of the tack in the shoe feels, and physically looking at the shoe with the tack in it, the brain knew it was supposed to be registering pain to perform the action on the 3D cube in Emotivs Cognitiv suite. But the tester felt no pain what so ever, the adaptive ability of a trained brain was absolutely remarkable. Regrettably no published paper on this phenomenon could be found.

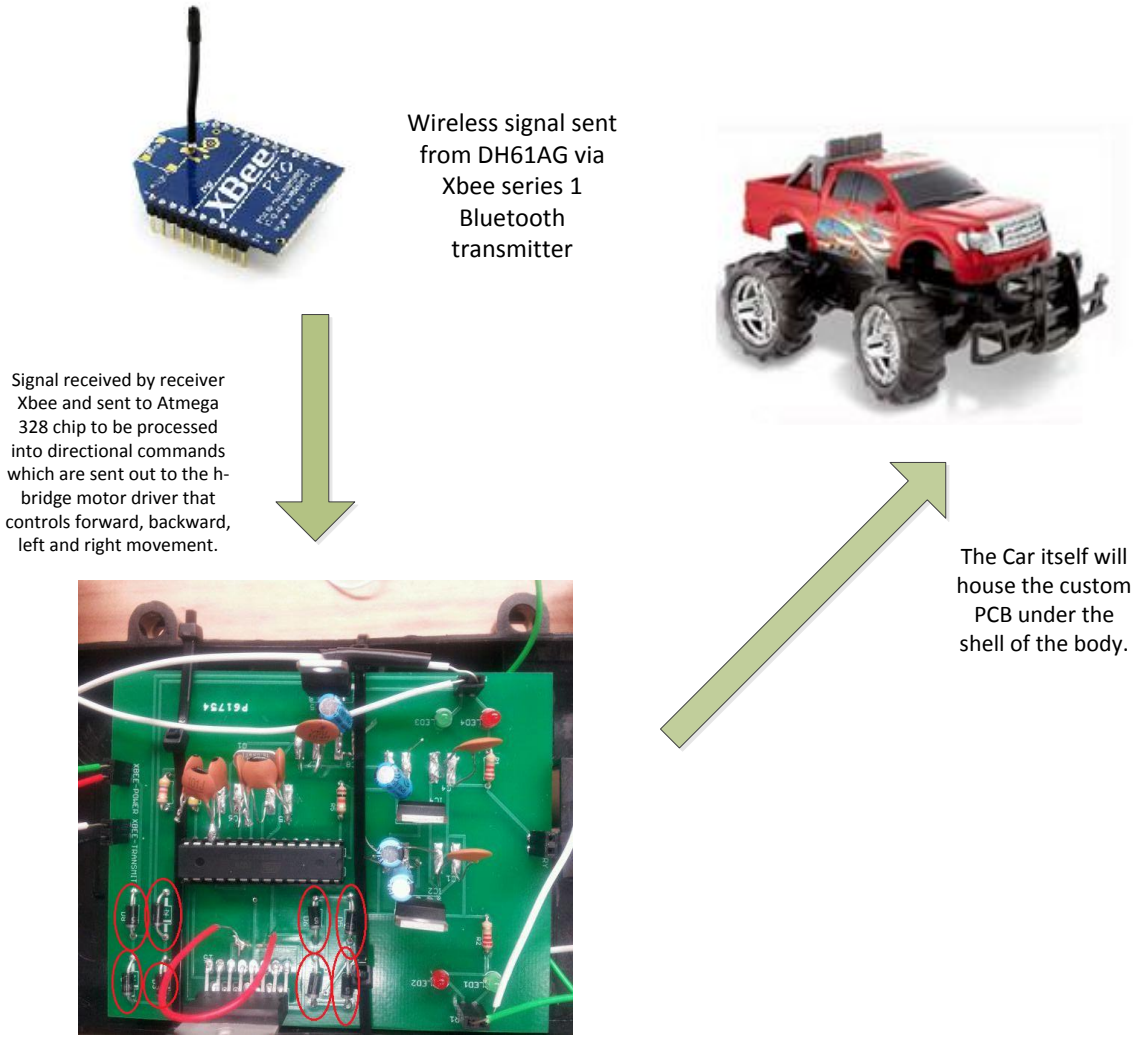
INFLICTING PERSONAL INJURY WAS NOT DONE FOR CONDUCTING ANY OF THE TESTS PREFORMED.

4.3.4 Different Languages

Language is one of the most important uses of the brain, and yet very little is known about how it is interpreted. It is know that for written language, the right side of the brain tends to interpret most of it. But for spoken language, there seems to be a much longer track that the brain takes to interpret and make sense of the noises. And as pertaining to internal monologue, there is even less information available on that. It was discovered in testing thought that thinking in a language other than ones native tongue activates activity in the somatosensory association cortex which can be picked up by the headset. As it was seen this area is separated by a distance from any other areas to make the signals produced distance enough to make the car move more reliably than any other method.

5. Design Overview

The EPOC-alypse controlled car will have 5 major parts which consist of; the Emotiv EPOC neuroheadset, the Intel DH61AG motherboard, 2 Xbees, and a custom PCB located on the car, All 5 of these components will work together to achieve the ultimate goal of making the car move. Shown in the following figures are 2 flow diagrams(figures 5.1 and 5.2); these are a visual representation of how each component looks like and a summary of what each part does.



(figure 5.1) Flow diagram of how onboard electronics will work

Emotiv Neuro-headset gathers EEG signals and transmits the data to the Bluetooth dongle plugged into a USB port on the DH61AG motherboard.

The EEG signals are received and interpreted by custom written software that determines what kind of signal is being received and what that signal corresponds to. Another program will be running in the background to convert the identified commands into a 6 bit binary string to be sent out via and Xbee series 1 plugged into another USB port.

The signal is received by another Xbee onboard the car that is connected to the custom PCB which contains an Atmega 328 processor and a dual h-bridge motor driver. Depending on the 6 bit string received the car will move in the corresponding direction.

(Figure 5.2) Flow diagram of how entire system will work.

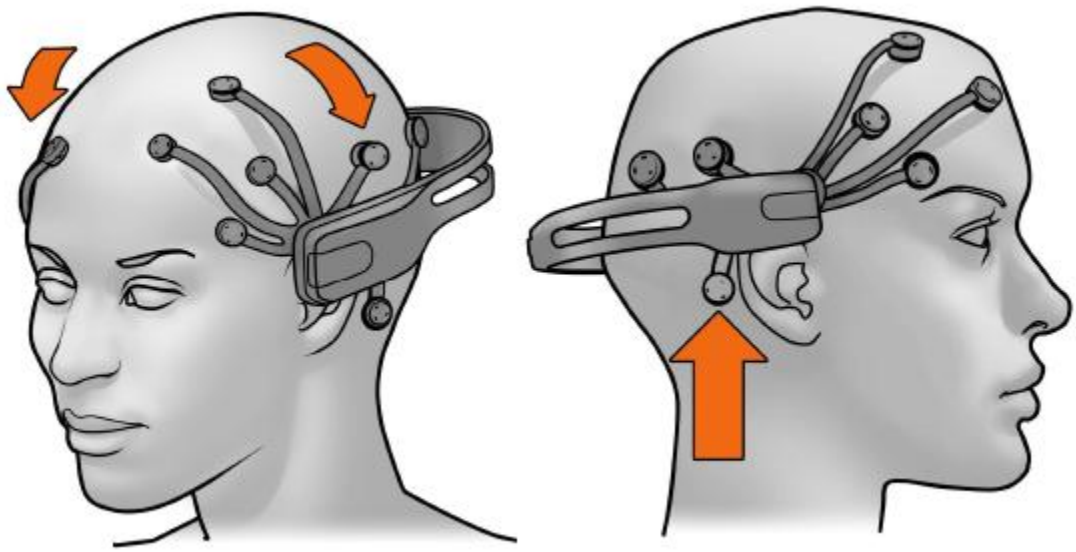
6: Hardware Overview

6.1: Headset

The Emotiv EPOC headset is an affordable easy to use marketed EEG recording device. Targeted at the emerging BCI video games market, Emotiv aims to enhance the gaming experience. Emotiv's design also has attracted the interest of neuroscientists due to the setups low price, running inexpensive experiments from one's own computer.

The Emotiv headset is the key to the entire project, being what obtains and transmits the neuro-signals. The headset comes with 14 independent sensors that consist of felt pads with gold connections to increase the sensitivity of the pickups. These felt sensors need to be moist at all time to conduct the potential difference across the skull, this is done by using a saline solution.

The placement of the headset on ones scalp is also an integral part to the acquisition of signals. As the headset is carefully slipped on it is key that to place the sensors with the black rubber insert on the bone just behind the ear lobe, as shown below in figure X



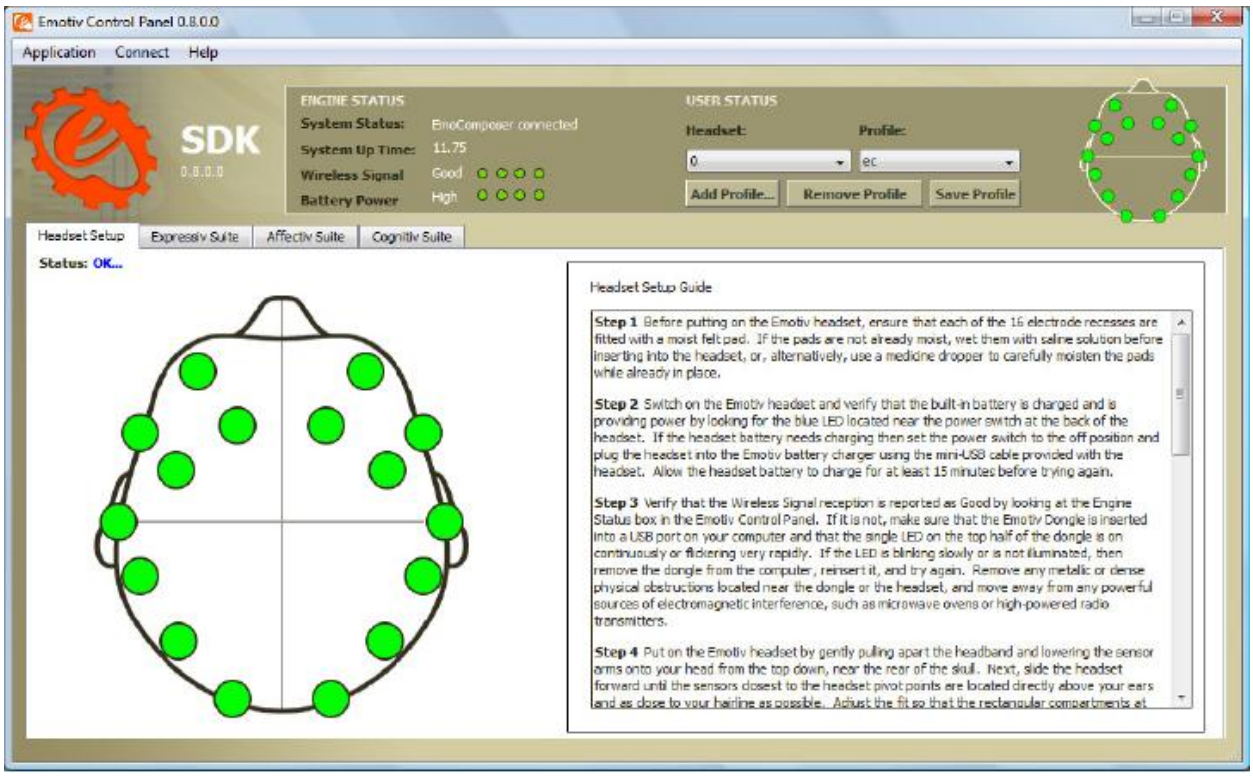
(Figure 6.1) Shows the correct placement of the headset with respect to the rubber reference sensor.

It should be noted that the two front sensors should be approximately at the hairline or three fingers above the eyebrows. After the headset is in position check to see that

there is good contact by the reference nodes, this is essential because a bad connection with these sensors will not produce any readings.

Once the headset is connected via the wireless USB receiver the headset setup panel is displayed. The main function of this panel is to display the contact quality feedback for the neuroheadset's EEG sensors.

The EPOC headset has sixteen electrodes to measure the potential difference across the skull. However there is no official reference for the user wearing the headset so the electrodes are actually paired up, and the difference between a pair is used as the measured signal. So when the user is training a certain action to manipulate a 3D cube with the Cognitiv suite, it is comparing how the values of a pair of electrodes change. Therefore whenever it sees a similar change, the software recognizes that you are trying to perform a specific action on the cube.



(Figure 6.2) A screenshot of the headset setup panel showing all good connections for all 14 sensors.

This image represents the sensor locations as seen when looking down from above onto the user's head. Each circle represents one sensor and its approximate location when wearing the headset.

Number of channels	14 (plus CMS/DRL references
Channel names	AF3, AF4, F3, F4, F7, F8, FC5, FC6, P3 (CMS), P4 (DRL), P7, P8, T7, T8, O1, O2
Sampling method	Sequential sampling, Single ADC
Sampling rate	128 Hz (2048 Hz internal)
Resolution	16 bits
Bandwidth	0.2 – 45 Hz, digital notch filters at 50 Hz and 60Hz
Dynamic range	256 mVpp
Coupling mode	AC coupled
Connectivity	Proprietary wireless, 2.4 GHz band
Battery type	Li-poly
Battery life	12 hours
Impedance measurement	Contact quality using patented system

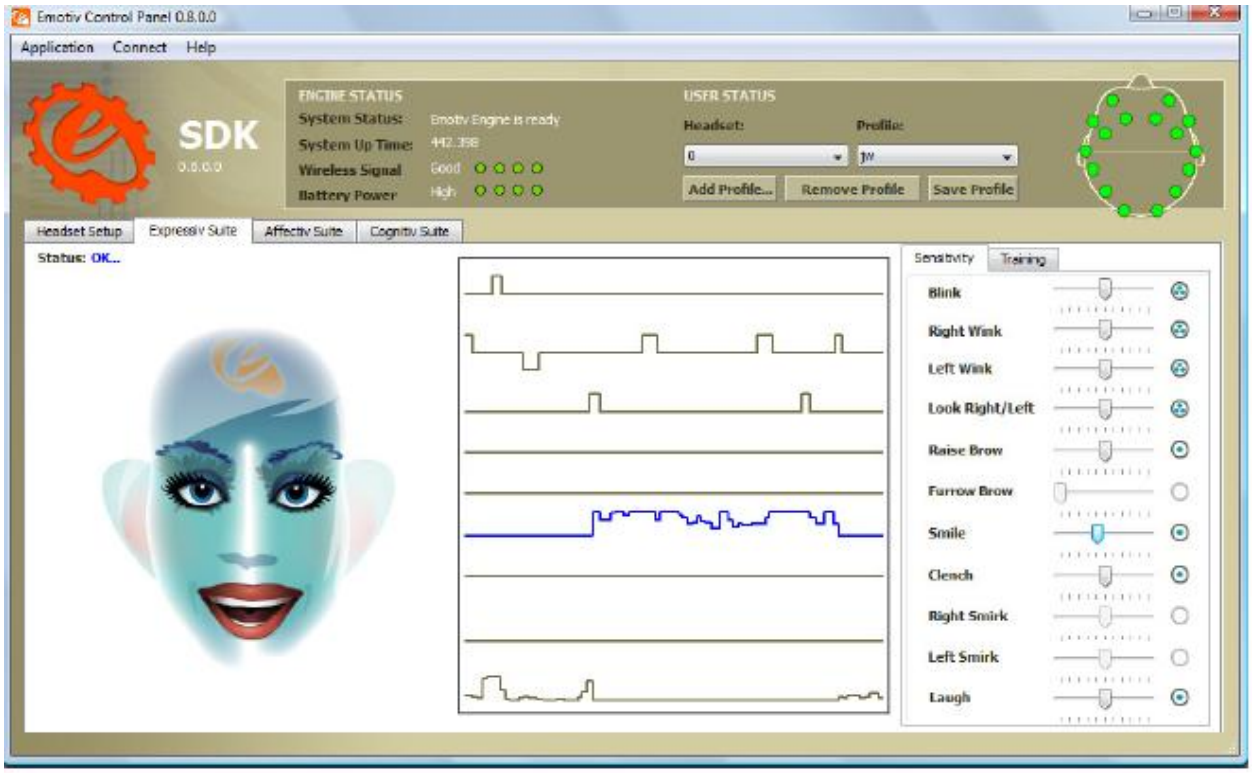
(Table 6.1) Emotiv neuroheadset specs.

Source: Emotiv. Emotiv Software Development Kit User Manual for Release 1.0.0.5.

6.1.1 The Three EMOTIV Acquisition Suites

There are three Emotive acquisition suites; the Expressiv suite, the Affectiv suite, and the Cognitiv suite. Each of these uses a different kind of interpretation of the sensor's signals to display different kinds of reading.

6.1.1.1 The Expressiv Suite



(Figure 6.3) The EXPRESSIV suite control panel.

While in EXPRESSIV suite, the avatar on the left of the screen will actually mimic the user's facial expressions. While at the same time the graphs to the right of the avatar will indicate which of a various amounts of facial expressions are being registered.

These graphs show short history of the detections listed, and can be interpreted as follows:

- **Blink**: low level indicates a non-blink state, while a high level indicates a blink.

- **Right Wink / Left Wink**: these two detections share a common graph line. A center level indicates no wink, low level indicates a left wink and high level indicates a right wink.

- **Look Right / Left**: these two detections share a common graph line and a single sensitivity slider control. A center level indicates eyes looking straight ahead, while a low level indicates eyes looking left, and a high level indicates eyes looking right.

- **Raise Brow:** low level indicates no expression has been detected, high level indicates a maximum level of expression detected. The graph level will increase or decrease depending on the level of expression detected.

- **Furrow Brow**: low level indicates no expression has been detected, high level indicates a maximum level of expression detected. The graph level will increase or decrease depending on the level of expression detected.

- **Smile**: low level indicates no expression has been detected, high level indicates a maximum level of expression detected. The graph level will increase or decrease depending on the level of expression detected.

- **Clench**: low level indicates no expression has been detected, high level indicates a maximum level of expression detected. The graph level will increase or decrease depending on the level of expression detected.

- **Right Smirk / Left Smirk**: these two detections share a common graph line. A center level indicates no smirk, low level indicates a left smirk and high level indicates a right smirk.

- **Laugh**: low level indicates no expression has been detected, high level indicates a maximum level of expression detected. The graph level will increase or decrease depending on the level of expression detected.

This control panel also includes sensitivity adjustments by moving the sliders on the right for each of the corresponding graphs.

EXPRESSIV supports 2 types of signatures that are used to classify input from the headset as indicating a particular facial expression, a preprogrammed universal signature or a trained signature. The foremost being what the average readings for blink, wink, etc. would be and the latter being the use performs an action, such as blink, and the program remembers it for future use.

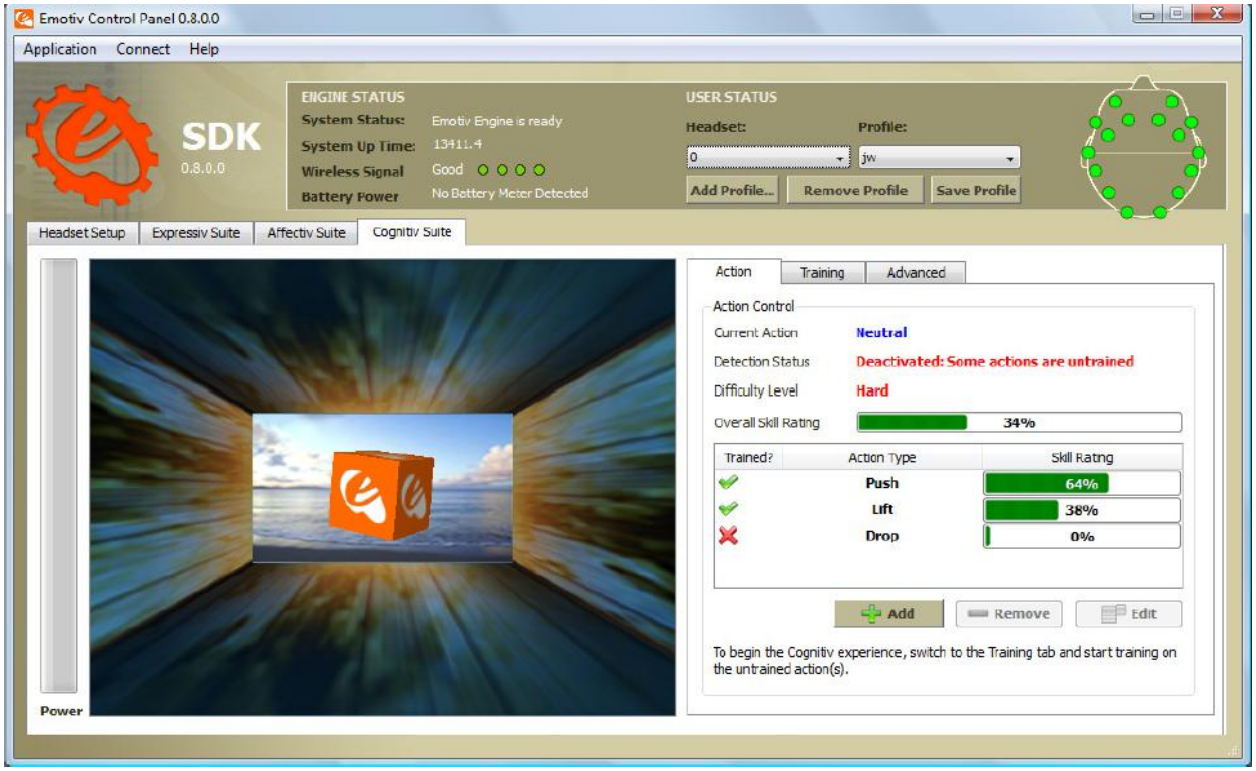
6.1.1.2 The Cognitiv Suite

This suite detects and evaluates a user's real time brainwave activity to discern the user's conscious intent to perform distinct physical actions on a real or virtual object. The detection so designed to work with up to 13 different actions including directional movements and rotational movements and an extra action that exists only in the users imagination which is to make something disappear.

The suite allows the user to choose up to four actions that can be recognized at any given time. The detection reports a single action or neutral, which would be no activity at a time, along with an action power which represents the detections certainty that the user has entered the cognitive state associated with that action.

The tricky part being that increasing the number of concurrent actions increases the difficulty in maintaining conscious control over the Cognitiv detection results. This is where training would come into play. New users gain control over a single action quite quickly but learning to control multiple actions requires practice and adding more actions quickly increases the difficulty,

The Cognitiv suite control panel uses a virtual #D cube to display an animated representation of the detection output. This cube is used to assist the user in visualizing the intended action during the training process. In order to enable the detection, each chosen action, plus the neutral action, must be trained. The suite enables the EmoEngine to analyze the uses brainwaves and develop a personalized signature which corresponds to each particular action as well as the background state of neutral. As the engine refines the signatures for each of the actions, detections become more precise and easier to perform.



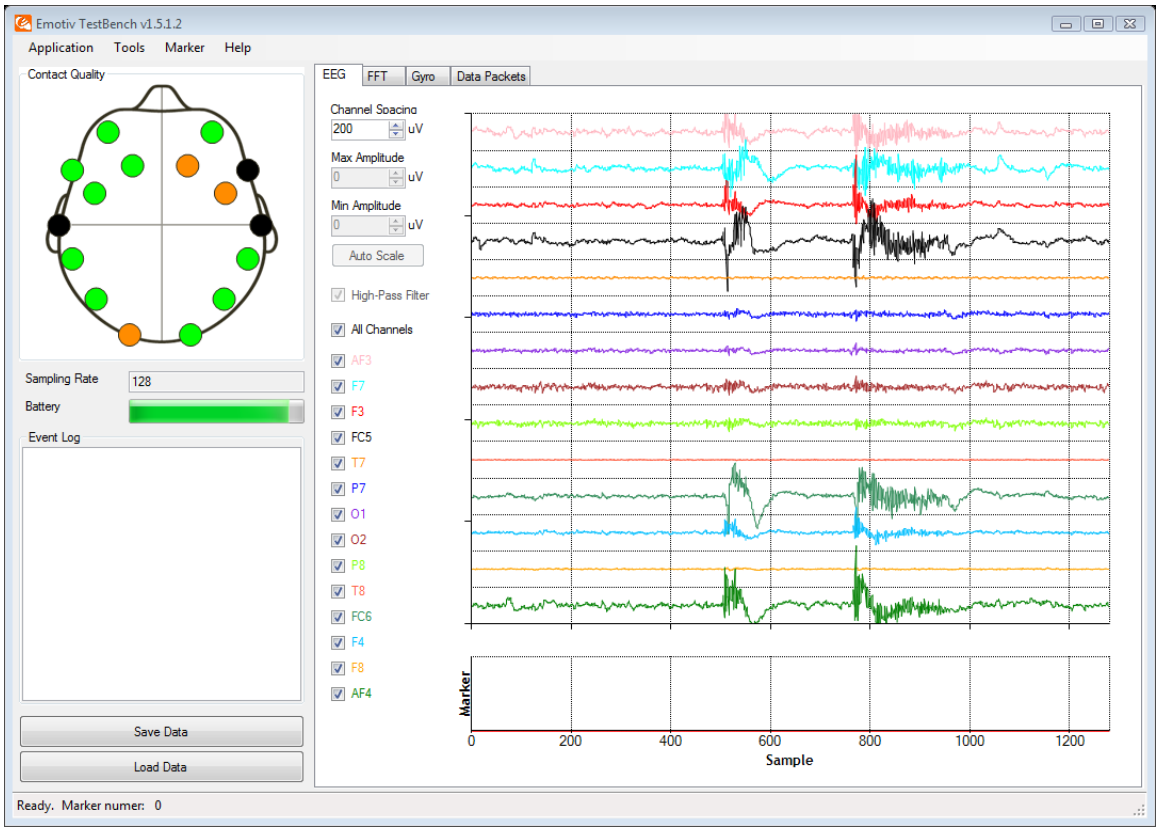
(Figure 6.4) the Cognitiv control panel with interactive 3D cube to simulate the action that the user is intending to accomplish.

6.1.1.3 The Test Bench

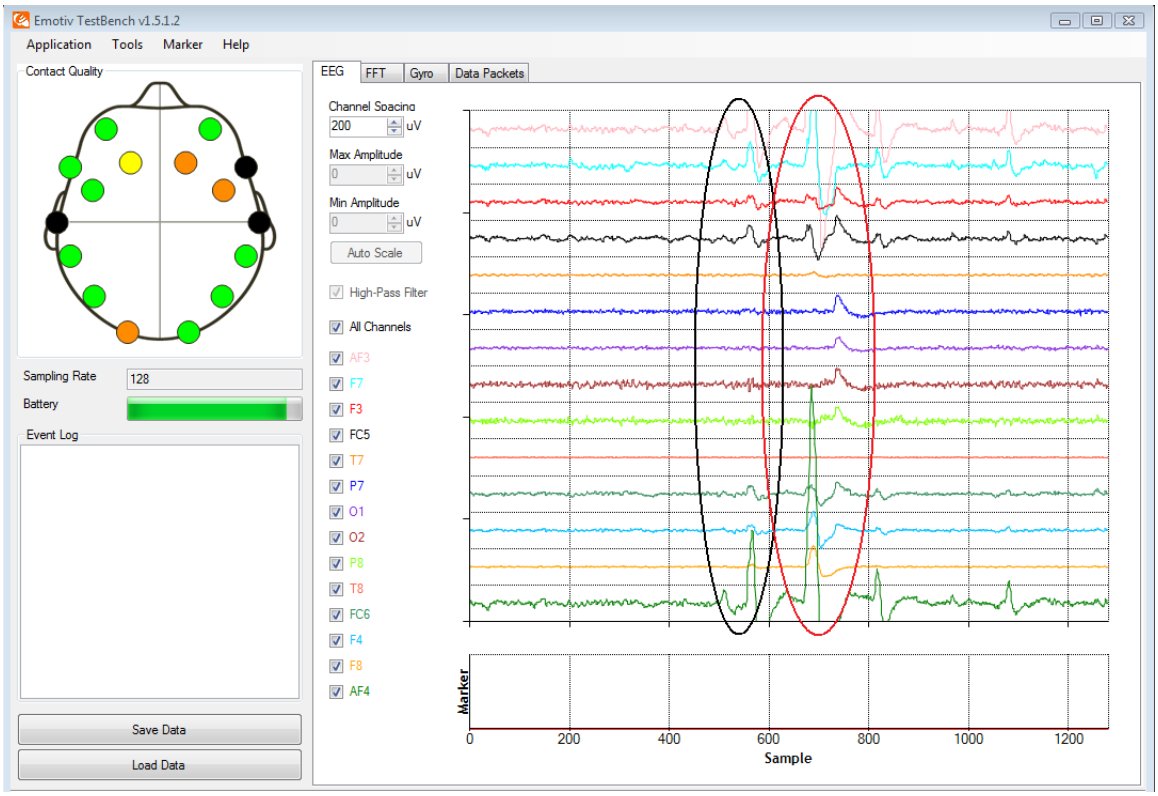
The Test Bench software is a real time EEG reading that displays the output of all 16 sensors. It can be noted early on that facial expressions such as blinking, smiling, and teeth clenching, are all very discernible due to the fact that the brain activity that is signaling these actions is very close to the top of the skull.

It is from the test bench that it can be seen where the increase in brain activity is coming from for training the Cognitive actions for the car. By being able to see the level of activity in a specific area of the brain, it is much easier to isolate ones thoughts and learn to activate the region on demand.

The figures below illustrate a few of the most noticeable readings from a few facial expressions. Though the sensitivity of the headset limits the detection of thoughts that can affect the Cognitive suite in manipulating the 3D cube.



(Figure 6.5) This is the EEG reading from clenched teeth. The active areas are areas that are toward the front of the head. The short spike is from a short clench and the longer one is from prolonged clenching.

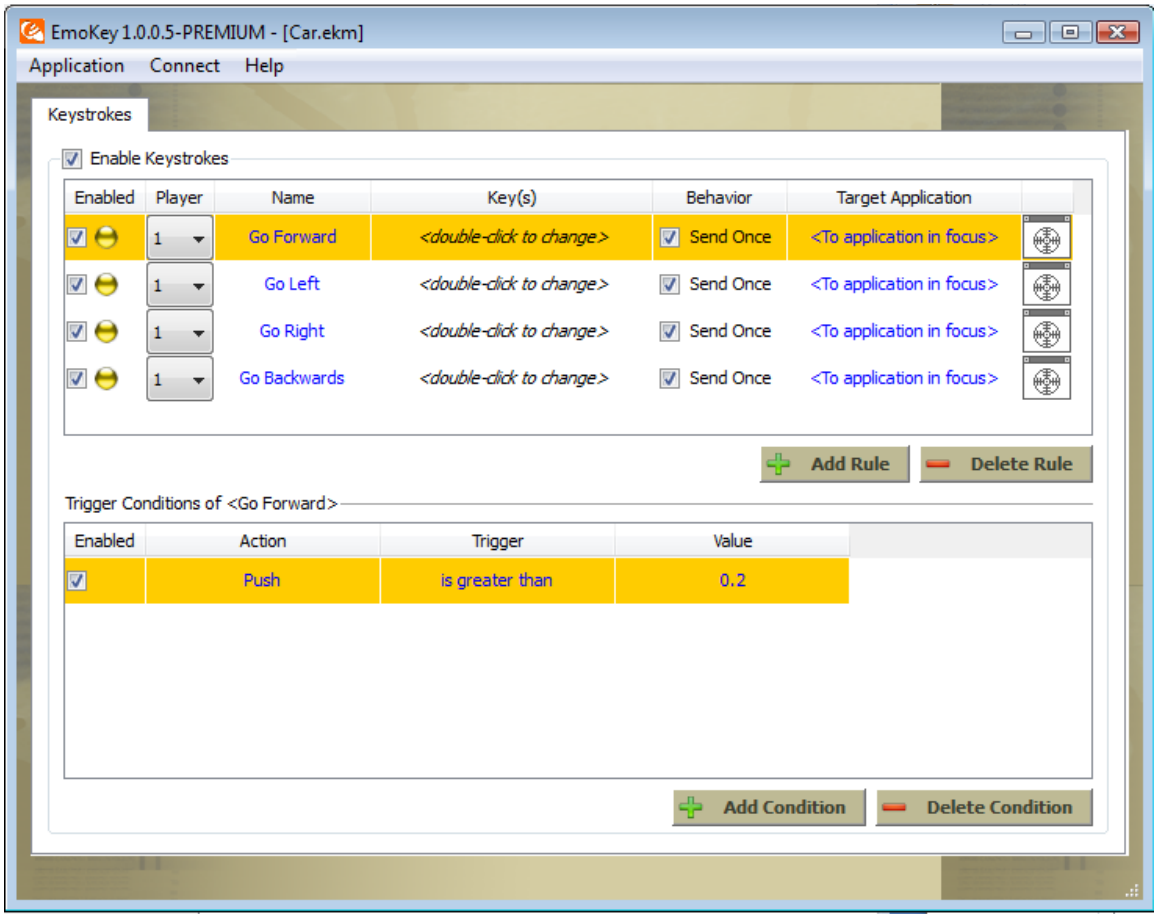


(Figure 6.6) Here, blinking is shown. The spikes in the black circle reflect a normal blink while the spikes in the red circle reflect a much harder and intense blink.

6.1.1.4 Emokey

For our project we combined both the Expressiv suite and the Cognitive suite. Using the Expressiv suite's facial cue detection to control lateral movement (left and right) and using the Cognitiv suite to control forward and backward movement. By winking with ones left eye the car will turn left and by winking the right the car will turn right. And by using the Cognitiv "push" and "pull" actions for the 3D cube, we will associate the push action with forward movement and the pull action with reverse.

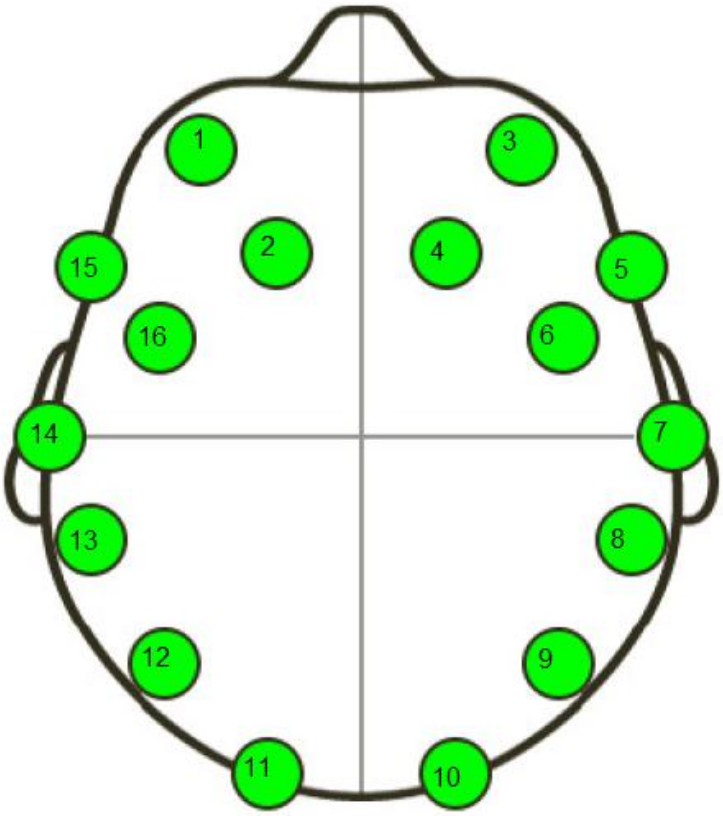
To convey these instructions from the motherboard to the arduino board the Emokey program will be used. Emokey maps out routes for instructions by associating a specific action with a command. The figure below shows a preliminary mapping for the car. The commands forward, backward, left, and right are triggered by push, rotate right, wink left, and wink right respectively. For instance with the push command, the Emokey will translate the push events generated by the Cognitiv suite into forward motion as long as the Cognitive suites push detection is reporting a score > 0.2



(Figure 6.7) Emokey mapping for controlling the car.

6.1.2 Headset Configuration

There are 16 different sensors on the headset that correspond to 7 different areas of the brain as shown in figure 6.8. These sensors are arranged into 7 pairs; AF3 and AF4, F3 and F4, F7 and F8, FC5 and FC6, T7 and T8, P7 and P8, O1 and O2, and DRL and CMS which are reference nodes.



(Figure 6.8) Sensor layout as viewed from the top of the skull.

Sensor Number	Sensor Name	Region of Brain Sensor is Located Over
1	AF3	Dorsolateral prefrontal cortex
2	F3	Frontal eye fields
3	AF4	Dorsolateral prefrontal cortex
4	F4	Frontal eye fields
5	F8	Anterior prefrontal cortex
6	FC6	Dorsolateral prefrontal cortex
7	T8	Primary gustatory cortex
8	DRL (reference)	Middle temporal gyrus
9	P8	Primary motor cortex
10	O2	Somatosensory association cortex
11	O1	Somatosensory association cortex
12	P7	Primary motor cortex
13	CMS (reference)	Middle temporal gyrus
14	T7	Primary gustatory cortex
15	F7	Anterior prefrontal cortex
16	FC5	Dorsolateral prefrontal cortex

(Table 6.2) Sensor names and relative locations.

6.1.3 Headset Transmission

The headset actually transmits over Bluetooth and Bluetooth dongle, the Emokit gets a connection to the Bluetooth device using a standard HID interface. Once the connection is made it gets 32 byte reports from the device that are encrypted to the Emokey language, these are then decrypted by using AES. The Emokit then parses out the gyroscope data and the actual sensor data and sends it to a queue that can be read in whichever manner is desired.

(Figure 6.9) Emotiv USB Bluetooth receiver.

6.2: The car

The vehicle that was used was relatively small in size. It ended up being 9.8 x 9.3 x 17.7 inches. It also only weighed 3 pounds. Although a smaller vehicle is preferred, it could not be too small as it needed to house multiple pieces of hardware on it in order for it to function as intended. The vehicle will need to house:

- a DC drive motor
- a DC servo motor
- several batteries
- many small DC voltage regulators ranging between 5 and 10 volts
- L298N H bridge
- Xbee receiver/transmitter between the motherboard and the vehicle
- Atmega 328p-pu microcontroller
- Several pin heads

A small Remote control vehicle was used as a base that included some of these components. There was also a printed circuit board that was created to hold the rest of the necessary parts needed for the scope of this project.

The proximity sensor was to be placed on the front of the vehicle using a small bracket or clip. The sensor was to be used to ensure that no damage would be done to the vehicle due to hitting or bumping into other objects. The sensor was to be small enough to be housed on the vehicle without interfering with other components and without being interfered with itself. It was preferably to be a capacitive sensor that will be able to detect metallic and non-metallic objects such as liquids, plastics, or woods from a distance of 35 cm to 1 m. Otherwise it was to be an inductive sensor that can only detect metallic objects from the same distances.

A lightweight DC drive motor with an operating voltage of 6 volts was used to move the vehicle, the motor has a two wire connection where all the power for the motor used in the vehicle is delivered between the two wires from the battery. For the purposes of this project, the DC motor was to have a maximum spin of at least a few thousand RPM's. The driving force of the vehicle will be a variable DC voltage that is small and light enough to fit on the back of the vehicle and does not interfere with the performance. The variable DC voltage was to preferably be in the range of 0-9 volts which should be strong enough to move the vehicle at an acceptable speed.

The DC servo motor is made through the assembly of a normal DC motor, a gear reduction unit, a position-sensing device, and a control circuit. The servo motor receives a control signal that represents a desired position of the servo shaft, in turn power is applied to the DC motor in the servo motor and it rotates within 200 degrees back and forth until the shaft is in that position. The servo motor has a three wire connection respectively, the power, ground, and control wires. The power wire must have a 5 volt high current power source constantly applied to it for it to function properly.

The four double A batteries that were used to power the vehicle were at least 1.5 volts each in order to properly power the DC motor and the motherboard. In addition to these requirements, the batteries were at least 500mA/H in order to give an adequate amount of time to test the vehicles operation.

Voltage regulators were used in order to properly power the Atmega 328p-pu microcontroller, the L298N H Bridge, the DC drive motor, and the DC servo motors. This will be done by maintaining a constant voltage level so as to avoid any damage to the processor in any way. One of the voltage regulators controlled the DC servo motor to change in the range of 0 to 5 volts to determine the extent of which the vehicle will turn.

The other voltage regulators controlled the DC drive motor, Atmega 328p-pu microcontroller, and the L298N H Bridge.

There will need to be a system designed to control both the DC drive motor and the DC servo motor. The most simple way to adequately control each of the motors was to use an Atmega 328p-pu microcontroller. This microcontroller was needed to control the circuit for the DC drive motor and DC servo motor.

The chosen motherboard needed to have a processor with an operating frequency of at least 2.4GHz and 1Gb of random access memory, (RAM) and also have at least two USB ports so that the transceiver can be inserted as well as a mouse so that the programming can be done. The motherboard was used to connect the USB transceiver and run the EMOTIV software as well as the wireless transmitter used to connect to the printed circuit board on the vehicle.

The USB transceiver received the information from the headset which in turn was to be sent to the written program and used to control specific actions on the vehicle. For instance, when a certain brain wave or facial movement is recognized, that in turn will be linked to an action to move the vehicle in the forward, backward, left, or right directions. This in turn was to be sent via the wireless Xbee transmitter/ receiver to the vehicle to be used to control the various actions. The transmitter operated at a frequency that is free to use, 2.4 GHz, as to not violate any laws instituted by the FCC.

6.3 The Motherboard

In order to run the Emotiv SDK as well as the developed software for this project, a motherboard or mainboard was needed which would be wirelessly connected to the headset as well as the car. While searching for a correct board, there were many specifications that needed to be considered. The Emotiv software has a few minimum requirements that we had to fulfill with our motherboard and processor. First of all, the processor needed to have a clock rate of at least 2.4 GHz. This meant that we had to look for more modern processors that had higher clock rates and more processing power. We also needed to find one that had at least 1 GB of ram and 50 MB of disk space. This was much easier to find, due to the large availability of memory with modern technology. These specifications, as well as price, were the most prominent requirements we needed to find in our motherboard.

After finding many suitable boards for our project, we decided to use the Intel DH61AG Core i3/i5/i7 Desktop Thin Mini-ITX Motherboard. This motherboard has many of the specifications we needed for our project as well as a few others that made the project easier to implement. Some of the more important specifications for this board were:

- 2.5 GHz processor clock speed

- 2 GB Memory

- Expansion slots for PCI Express

- 10 USB ports

- 1 HDMI port

These features of the motherboard made it easier to not only run the Emotiv SDK and the developed C++ software, but the HDMI port also allowed us to connect it to a monitor and view the Emotiv GUI while running the Emotiv software. One problem we did find with the motherboard was that it did not include a significant amount of disk space, which meant we needed to find an external source of space. In order to make up for this, we purchased an external hard drive which was used to contain the Emotiv SDK as well as the developed software for the project.

7: Hardware Design

7.1: Headset Hardware Design Overview

The hardware for the project consisted of four components; the Emotiv neuroheadset, the Intel DH61AG core i3 desktop thin mini-ITX motherboard, a custom built Arduino PCB, and a remote controlled car.

7.1.1 Emotiv Neuroheadset

The Emotive neuroheadset, as described in the overview, is an array of 7 sensor pairs, 14 sensors total arranged mostly towards the front of the skull. These sensors measure the dissimilarity between a pair and this difference is recorded as the signal. The signal is then transmitted via blue tooth to the receiver which is plugged into the DH61AG motherboard.

7.1.2 DH61AG Motherboard

The DH61AG core i3 mini-ITX motherboard is a second generation Intel core platform with integrated HD graphics, HDMI, and onboard DC power. It features the 2-chip Huron River platform which is capable of supporting Intel's Core i3/i5/i7 (sandy bridge) processors. The exact processor that has been chosen is the Intel core i3-2100T Sandy Bridge LGA1155 that runs are 2.5 GHz which is perfect for that task at hand.

7.1.2.1 Intel Core i3-3220

With the i3-3220, one gets only two physical cores, but the 3220 features hyper threading which adds an additional two virtual cores. This makes a sizeable difference in performance over the Pentium models. Due to budget constraints, the 3220 does lack turbo boost, thought, with its frequency limited to 2.5 GHz it supports a cache configuration of 2x 256 KB of level 2 cache, 3MB of level 3.

7.2: Vehicle Hardware Design

7.2.1: Battery and Power

In order for the vehicle and all the components on it to be properly powered, there were many things taken into consideration. One of which is what is the total amount of voltage necessary for all the components being used on the vehicle to function as required. After looking at the requirements for each of the components being used, at least four double A batteries will be used. Rechargeable battery packs offer many benefits that regular non rechargeable batteries do not offer. One of which being that they can often last up to five times longer on each charge and are environmentally friendly. Although this is the case, it ended up being more cost efficient to use non rechargeable double A batteries instead of rechargeable ones.

There are many different types of batteries available that can be used for this project. Many modern batteries use a variety of chemicals to power the reactions that ultimately make the batteries work. There are Zinc Carbon, Alkaline, Lithium Ion, Nickel Metal Hydride, and Nickel Cadmium, and Lead Acid. Zinc Carbon batteries are very common in cheap AAA, AA, C, and D dry cell batteries. Alkaline batteries are also very common in AA, C, and D dry cell batteries and gets its name from the potassium hydroxide electrolyte. The lithium Ion rechargeable battery is used in high performance devices such as cell phones and digital cameras and Lead Acid rechargeable batteries is the last type and is most commonly used in car batteries.

Focusing on the rechargeable batteries, the most commonly used is the Lithium Ion battery. These are the most popularly used on the market for many reasons. One of which being the fact that they hold their charge long term. It only loses about 5 percent of its charge per month. Another benefit of this type of battery is that it has no memory effect. In essence, this means that it is not necessary to fully discharge the battery before recharging. Lithium Ion batteries can also be used several hundreds of charging and discharging cycles without loss of use. There are a few disadvantages to these types of batteries though. Once they are fabricated, they only last between two to three years whether they have been in use or not. They are also extremely heat sensitive. If they are in an environment where there are high temperatures, the Lithium Ion battery packs will degrade much faster.

Nickel Cadmium batteries were one of the first rechargeable batteries available on the market. One of the greatest disadvantages to its use is the problem known as the memory effect. In order for the battery to not lose its capacity, it needs to be fully

discharged every single time. This is when Nickel Metal Hydride batteries came into favor. These batteries have a higher capacity in comparison and are only minimally affected by the memory effect.

After much consideration, the four chosen batteries used were the zinc carbon batteries with at least 500mA/H to allow for adequate time to test the vehicle and all components involved. Nickel Metal Hydride was not chosen because the batteries start out around 1.2 volts, but as it discharges it drops only to about 1.1 volts before the battery is fully depleted. Even though the NiMH battery starts at a lower voltage it provides a more usable voltage for a longer duration of time. It also has a 30 to 40 percent higher capacity over a standard Nickel Cadmium (NiCd) battery and are simple and easy to store and transport. The current ratings of the batteries is very important as to have at least 30 to 40 minutes of drive time while powering both the serial to parallel converter as well as all the vehicle components.

The dimensions of the batteries will most likely be 14.5 x 50.5 inches because it will technically be four AA batteries connected together in each battery pack. The four batteries will be connected in series with the common node of the circuit being that of where the positive of one battery meets the negative of another. Many of the components in the vehicle will be run off of the positive terminal of the batteries, so this in turn requires that the drive and servo motors will be run off of the negative terminals. This will make it so that the other connected components will still remain active when the battery that controls the vehicle has been depleted.

It was also extremely necessary for there to be several voltage regulators in place in order to control the various voltages that are needed to be supplied to each circuit component. Voltage regulators are designed to regulate a constant voltage level to all components involved. Since all of the components needed about 5 volts to properly work, the LM7805 voltage regulator will be used. See figure 7.2.

Figure 7.2

The LM78XX features a simple three terminal regulator with 5, 12, and 15 volt regulator options. Each of the three options being the LM7805, LM7810, and the LM7815 respectively. It also includes internal thermal overload protection so as to prevent damage to the component, due to overheating. With adequate heat sinking the LM7805 can provide up to 1.5A of output current within the temperature range of 0 to 125 degrees Celsius. Three separate voltage regulators with different output supply voltages will be used to power all of the subsystems needed to fully operate the project. One LM7805 for the Xbee transmitter/receiver and two LM7805s for the motors, one for each.

7.2.2: Sensors

In order to preserve the condition of the vehicle and to prevent any damage to it due to foreign objects, one proximity sensor was to be used on its front side. This sensor will take information and relay it to the processor, where it can in turn determine how to adjust its position accordingly. The proximity sensor needed to be small in size and relatively inexpensive. There are many different types of sensors, each with their own pros and cons. There are inductive, capacitive, photoelectric, and ultrasonic sensors available.

Inductive proximity sensors are becoming more widely used in detection, positioning, and counting of ferrous and nonferrous metal substances. They can also detect metal through a layer of non metal material. The sensor itself is comprised of an oscillator circuit, which is the actual sensing part, and an output circuit with a switching device that is completely housed in a resin encapsulated body.

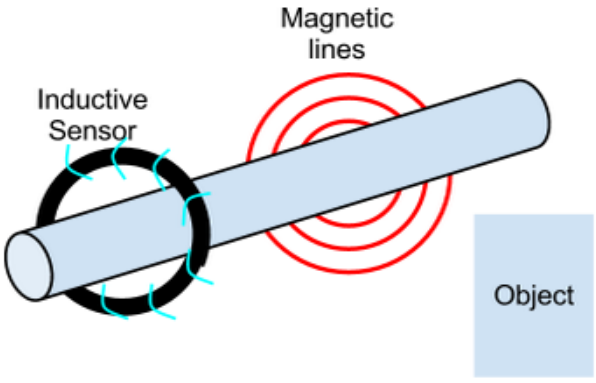


Figure 7.2.1
Inductive Sensor

The way that the inductive sensor actually works is when the inductance coil that creates a magnetic field is disturbed, the circuit responds by closing the output switch. The magnetic field is created on the front of the sensing face and is the essential part in making the sensor work properly.

Capacitive proximity sensors are usually used as a last resort when all other sensing techniques can not be used. They operate by responding to a change in the “dielectric medium surrounding the active face and can thus be tuned to sense almost any substance”. They can respond to all substances that have a high dielectric constant, such as water, oil, fuel, or sugar.

Another benefit to using capacitive sensors is that they can sense a substance through a layer of glass, plastic, or thin carton as well. These sensors are most often used in level control of non conductive liquids, granular substances, or sensing objects through a protective layer such as glass. A disadvantage of using the capacitive sensors is the fact that deposits of excessive dust and dirt on or around the sensing face can cause an erratic response and causes a need of periodic cleaning.

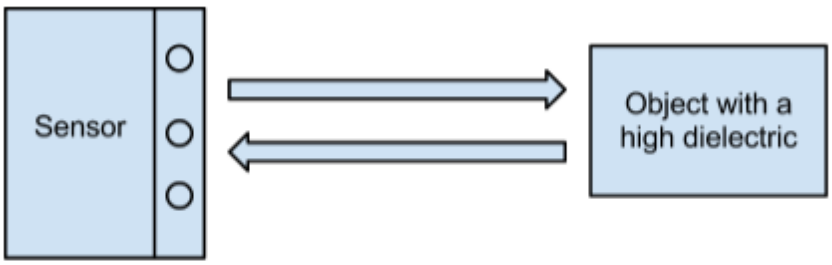


Figure 7.2.2
Capacitive Sensor

The way that capacitive sensors actually work is based on an internal oscillator with two capacitive plate electrodes. These are then tuned to respond when a substance with a high dielectric is approaching the sensors face. If an object is sensed, the output switch will then either close to activate a load for a normally open option, or open for a normally closed option. After all of this has happened, the LED light on the sensor will then illuminate indicating the the switch has made a change.

Photoelectric sensors offer many benefits that other sensors do not. They have non-contact sensing of almost any substance or object up to a range of 10 meters. Photoelectric sensors function by using a light source, usually a light emitting diode in infrared or visible light spectrum.

A huge advantage that photoelectric sensors have over capacitive sensors is that these can operate better under dusty or dirty environments. These are also often used because of the focused beam and long range which helps in sensing distance and accuracy. There are many different types of photoelectric proximity sensors. Of these types there are infrared proximity (diffused reflective), transmitted beam (through beam), retroreflective (reflex), polarized retroreflective (polarized reflex), fiber optic, and background rejection. The infrared proximity sensor works by detecting light reflected by the intended target. This type of sensor is most commonly used when the object that needs to be detected is only accessible from one direction.

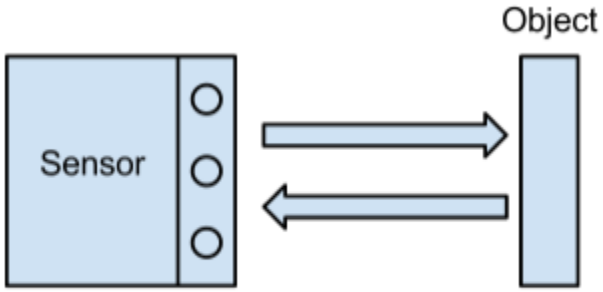


Figure 7.2.3
Diffused Reflective

Another type of sensor is the transmitted beam photoelectric sensor. This uses separate infrared transmitters and receivers so that when an object passes through the infrared beam, it causes the receiver to output a signal. This signal either closes a normally open switch, or opens a normally closed switch.

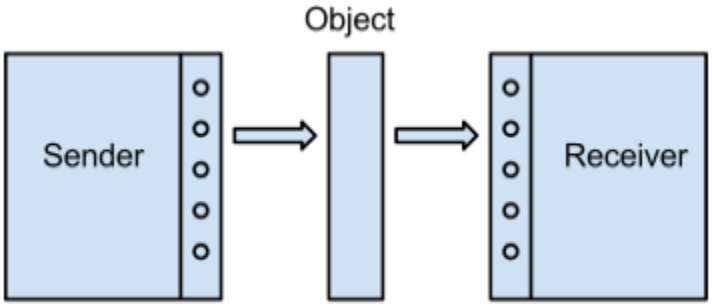


Figure 7.2.4
Through beam

The next type of sensor is the retroreflective photoelectric sensor. These types of sensors operate by sensing the light beam that is reflected back from a target reflector. Just like the transmitted beam sensor, if an object interrupts the beam, an electronic output is activated. Polarized retroreflective sensors work almost the same as retroreflective sensors. The only difference being that it uses a polarizing filter, designed so that shiny objects are easily detectible, in front of the transmitter and receiver optics.

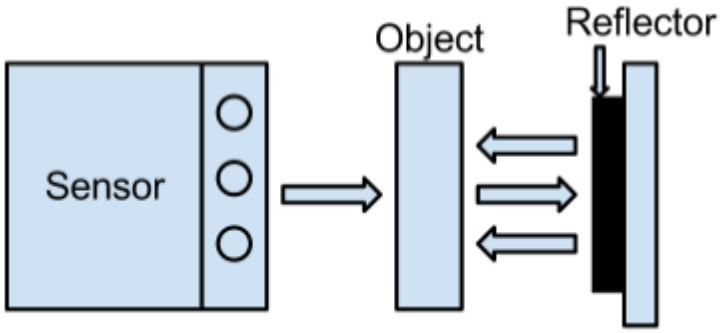
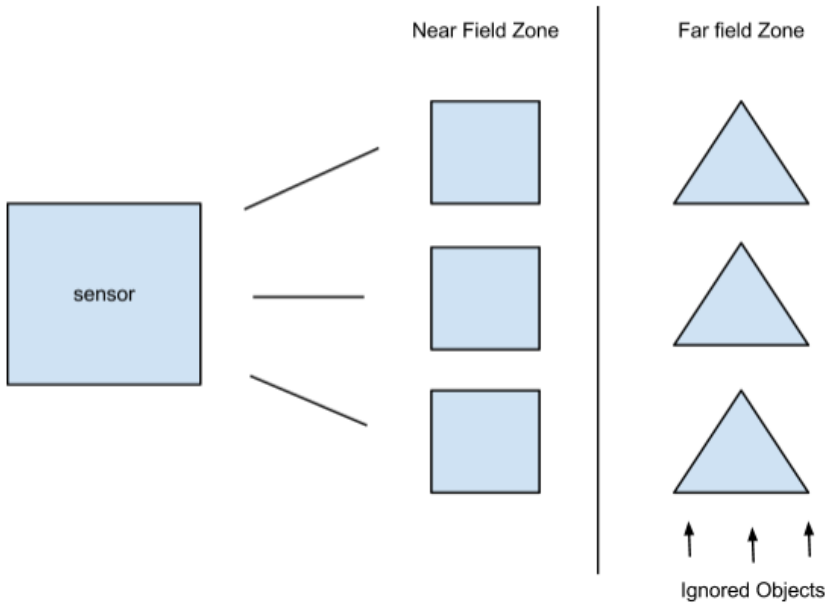


Figure 7.2.5
Retroreflective

The next type of sensor is the fiber optic sensor. These operate like the name indicates, by using fiber optic cables to conduct light from the light emitting diode to the sensing area. Then another cable is used to return the light from the sensing area to the receiver. This type of sensor offers great benefits for sensing in extreme conditions as well as confined spaces. For instance if the sensor is used in an environment that has temperature extremes or is exposed to harsh chemicals, it can still be protected and work properly.

The last type of sensor is the background rejection sensor. This sensor uses a special arrangement of two sensing zones, the near field zone and the far field zone. The near field zone is the area where objects can be detected. The far field zone is the area where objects cannot be detected.



The cut off between these zones is very sharp and extremely precise, although this range is adjustable to meet certain needs. These sensors are mostly used for applications where the objects after a certain range need to be ignored.

Ultrasonic sensors are very useful in detecting objects of different sizes and materials. They work by using the reflection of high frequency sound waves to detect parts of the distances to those parts. There are two basic types of ultrasonic sensors, electrostatic and piezoelectric. The electrostatic sensor uses capacitive effects for longer range sensing and a wider bandwidth with greater sensitivity.

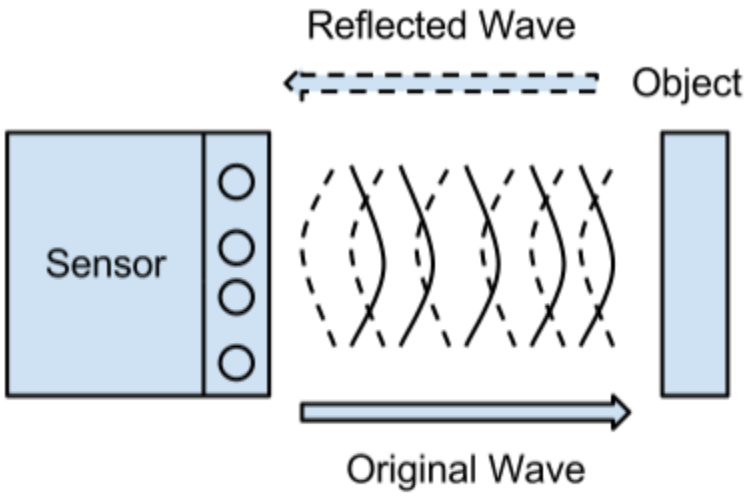


Figure 7.2.6
Ultrasonic Sensor

The piezoelectric sensor is a little more rugged looking and more inexpensive in comparison to other sensors and operates by using a charge displacement during the strain in crystal lattices. When it is necessary to detect transparent objects, such as a clear plastic, the ultrasonic sensor is the best choice in doing so.

After reviewing all of the different types of sensors and the pricing of each, the originally chosen product for the proximity sensor on the vehicle was the piezoelectric ultrasonic sensor. This was chosen because of its ability to sense any object with varying transparency as well as it being weatherproof and able to function in environments with high humidity, such as that of Florida. The particular chosen model was originally the Ultrasonic ranging module: HC-SR04.

Figure 7.2.7

This particular component is available for purchase from the iteadstudio website. It only requires a 5 volt DC power supply which works out perfectly with all of the other components on the vehicle. It has a large sensing distance of 2 centimeters to 500 centimeters. This ends up being a little over 16 feet which is more than enough sensing distance required for the scope of this project. In addition to this, the sensor also has a rectangular size of 20 millimeters by 40 millimeters. This fits in perfectly with the size of the chosen remote controlled vehicle as to not hinder its mobility in any way.

The originally chosen ultrasonic sensor has many capabilities useful to this project. If no output is detected it sends a 38 millisecond high level signal. This would have allowed the vehicle to keep operating without any due changes. It has a very large mean time to failure of 50,000 hours which should be more than sufficient enough for the scope of this project. It can also operate in the temperature range of -40 to +80 degrees Celsius, which ends up being up to 176 degrees Fahrenheit. It can detect objects up to 5 meters away and has a minimum detection range of 2 centimeters, in the range of 15 degrees in front of the sensor. It also sports a good resolution of .3 centimeters.

Although the HC-SR04 piezoelectric ultrasonic sensor would have been beneficial to the project, it was not used. Due to time constraints and money constraints, we were not able to incorporate all of this particular research into being used on the vehicle.

7.2.3: RC Car and Accessories

To begin conceptualization of the vehicle and its various accessories, it was necessary to choose a remote controlled car early on. The vehicle chosen for this project was the “New Bright RC Ford F-150”. This vehicle was primarily chosen for its being large enough to house all the extra components that are necessary to be added without interfering with its mobility and its being perfectly within our price range. The vehicle has a 1:12 scale ratio with dimensions of 17.7 inches tall, 9.3 inches wide and 9.8 inches long. It weighs approximately 3 pounds which is lightweight and allows for all of the other components to be added without hindering the maneuverability of the vehicle. It also has a great low cost of $26, which fits in nicely with our budget.

Figure 7.2.8

In order to turn the wheels left or right, a servo motor or a drive motor with gears was necessary. One of the options that we had was to switch out the servo motor that came with the vehicle, for a different one that we could buy. The servo motor that was inside of the vehicle could have been exchanged for one with a better angular movement. For instance, for the purposes of our project, as narrow of a turn possible was wanted. This would show in as clear a manner as possible that the car was in fact receiving the correct signals and going in the direction that we anticipated. An example of a servo that we could have switched out is the “Hitec 31055S HS-55 Economy Sub Micro Universal Servo”.

Figure 7.2.9

This servo is a lot more reliable in comparison to other servo motors in this reasonable price range. It comes out to being $11 with free shipping which fits in nicely with our vehicles budget. It has an input range of operation at 4.8 to 6 volts which also coincides acceptably with the output voltage of the LM7805 voltage regulator. In addition, it has a motor speed of 438 degrees per second, which is about 73 revolutions per minute (RPM). This high speed of motion is more than adequate for the purpose of this project, especially because the drive motor has a maximum angular displacement of 40 degrees.

After opening up the remote controlled car, we were able to see what kind of servo motor was actually in the car. The turning motor in the car was actually a regular drive motor with a few gears attached in a way that it would turn the wheels 45 degrees all the way to the left or 45 degrees all the way to the right. After lots of testing, we decided as a group that it was more beneficial and cost efficient to use the motor already in the vehicle. It operated in the same way that the motor that we would have bought would be used. As well as it having an operating voltage at the perfect voltage of 5 volts.

The drive motor in the vehicle was originally to be replaced because the current speed of the vehicle with its current motor was thought to be unacceptable. After much consideration, the “Mabuchi RS-380 Brushed DC Motor” was originally chosen.

Figure 7.2.10

The motor operates between 3 and 6 volts with a nominal voltage of 6 volts. It has a no load current of .8mA with a no load speed of 18000 RPM and a light weight of 71g. The motor as well has a stall torque of 771 g/cm with a stall current of 24A. For the purposes of this project, those values should never have been reached. To ensure this, a 15A fuse would have been placed on the voltage input in order to prevent damage to the components.

After much thought, the drive motor that was originally to be bought would have been way too fast. The motor that came inside the vehicle was also adequate enough for the purposes of the project. When it was operated at 5 volts it ended up giving the speed that we wanted. Although making the car go faster would have been fun to watch, a slower speed was required so that the vehicle would not become accidentally damaged by running into objects too fast. Using the motor that came inside of the vehicle also cut down the cost as a total which is always beneficial.

7.2.4 DC drive and servo motors control

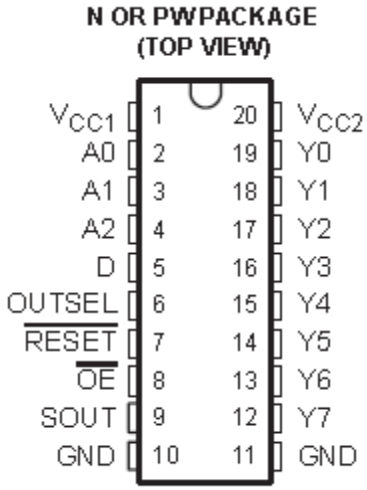
The DC drive motor creates somewhat of a complicated situation pertaining to the hardware design of the vehicle. There will need to be a system designed to control both the DC drive motor and the DC servo motor. There were many options that we could have used to control each of the motors respectively.

The first way we looked into to adequately control each of the motors was to use an eight bit serial to parallel converter. The converter would need to be used to control the circuit for the DC drive motor. The part that would have been used to accomplish this task was the 74LV8153N, shown in figure 7.2.11, and it is produced by Texas Instruments.

Figure 7.2.11

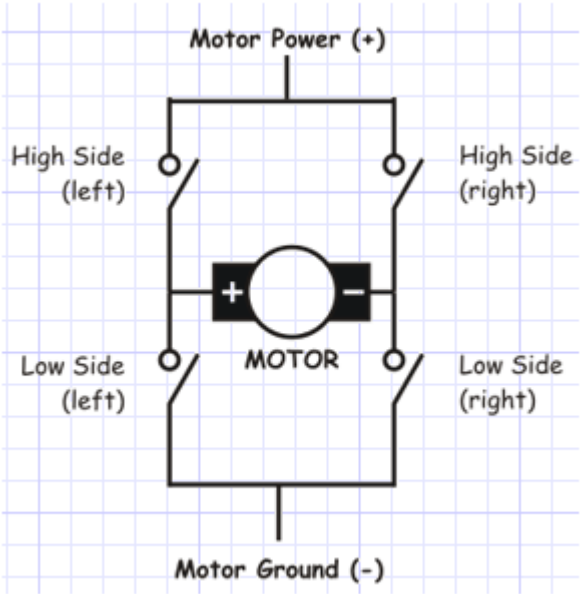
This specific part offers an automatic data rate detection feature. This eliminates the need for an external oscillator and helps lower the cost. Up to eight devices can share the same bus by using different combinations of A0, A1, and A2. Power supply levels are referenced to Vcc1 and Vcc2 and can be configured from 3 volts to 5.5 volts and 3 volts to 12 volts respectively. A pin layout of the 74LV8153N is shown in figure 2.2.12.

Figure 7.2.12



Vcc1 is the power supply pin for all inputs and outputs except Y0-Y7. Vcc2 is the power supply pin for the outputs Y0-Y7. It has a maximum data rate of 24 kbps, which was more than satisfactory for the scope of this design. The physical size of this part is .775 inches in length and .5 inches in width, with a height of about .2 inches

Another option to control the motors was the L298N H-Bridge motor control part. This particular part offered the benefit of being able to control two motors at the same time. The way that an H-Bridge works is by enabling a voltage that can be applied across any particular load.



The switches of the h bridge are turned on in pairs, the high left and low right, or the high right and low left. If two switches on one side are turned on it would create a short circuit. Depending which switches are activated, the current flows and the motor will start turning. It will either turn in the ‘positive’ or ‘negative’ direction, therefore moving the car forwards or backwards.

After much consideration, the h-bridge seemed to be the most reasonable choice. At a low price of $3, this fit perfectly into our vehicles budget. The L298N inputs 1 through 4 and enable A&B are tied directly to the output digital pin lines 14-19 of the Atmega328p-pu microcontroller. By setting the pins of the Atmega and connecting them to the L298 Driver motor, the L298 controls the two separate motors. This in turn provides control of the drive and turning motors of the vehicle. Setting the enable inputs A and B of the L298 to a high or low value enables either H-bridge device to turn on or off. If pin 14 of the Atmega were set to high then the first H-bridge, drive motor, is enabled, otherwise it is disabled. If pin 17 were set high then the second H-bridge, turning motor, is to be activated, otherwise it is disabled.

The logic table below shows how each of the motors will be controlled based upon the inputs from the Atmega328p-pu to the L298N H-bridge. In order to enable only the drive motor, we would enable only the first H-bridge. This would be done by setting pin 14 on the microcontroller to a high, or a 1. This in turn would allow us to use inputs 1 and 2 on the h-bridge that are connected to pins 15 and 16 on the microcontroller. Depending on the values sent to the inputs, the drive motor will either turn clockwise or counterclockwise. In essence this moves the vehicle forward or backward. At this time the turning motor enable bit is set as a low, or a 0 meaning that it is off for the time being. If the input values are the same, 00 or 11, it will stop the vehicle completely.

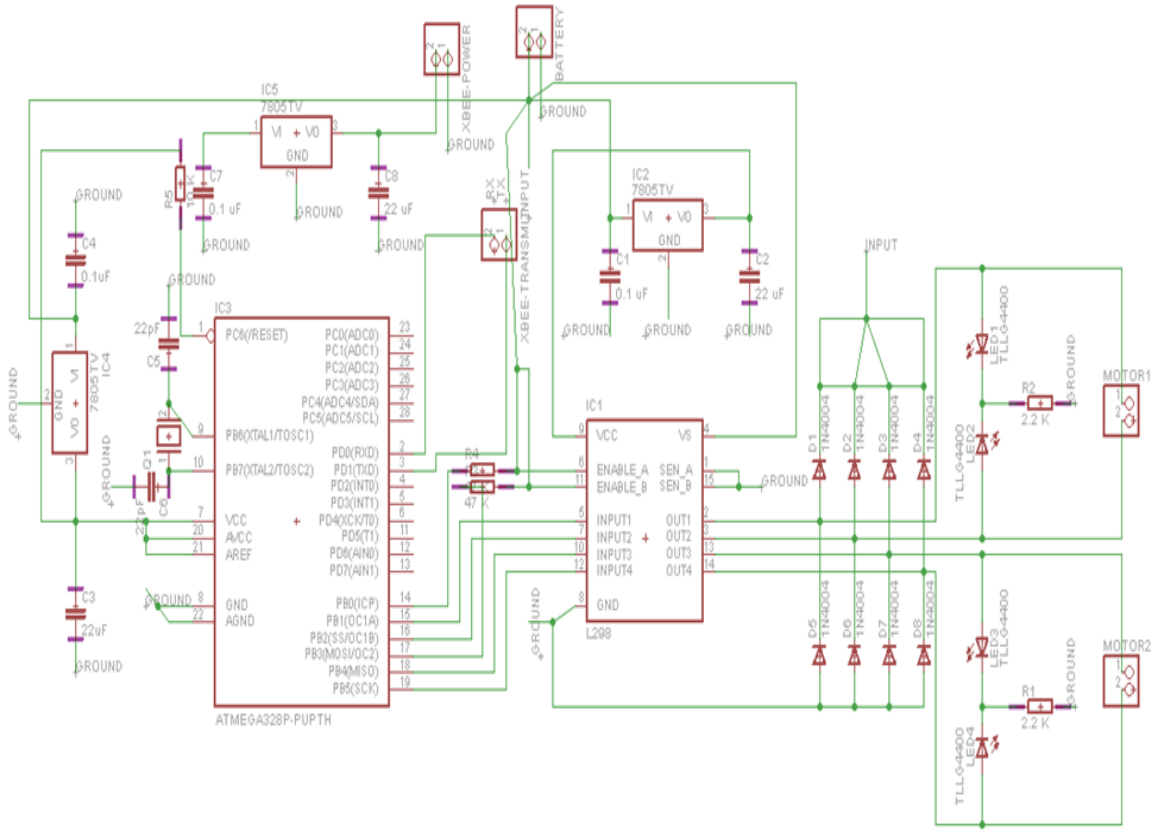
When we want the vehicle to turn left or right we need to activate both motors at the same time. This can be done by setting both enable inputs on the h-bridge, connected to pins 14 and 17 on the microcontroller, to a high or a 1. The inputs 3 and 4 for the turning motor are connected to pins 18 and 19 on the Atmega. Setting pins 18 and 19 to 01 or a 10 on the microcontroller would turn the front motor as far right or as far left as its design allows. Thus turning the entire vehicle as far right or left as it can go. This needs to be combined with the drive motor being activated forward as well. This will allow the car to be moving and turning at the same time as opposed to turning the wheels in a stopped position and then turning the drive motor back on.

Drive Motor 1		**Turning Motor 2**		
Enable Pin 14	**Input Pin 15/16**	**Enable Pin 17**	**Input Pin 18/19**	**Result**

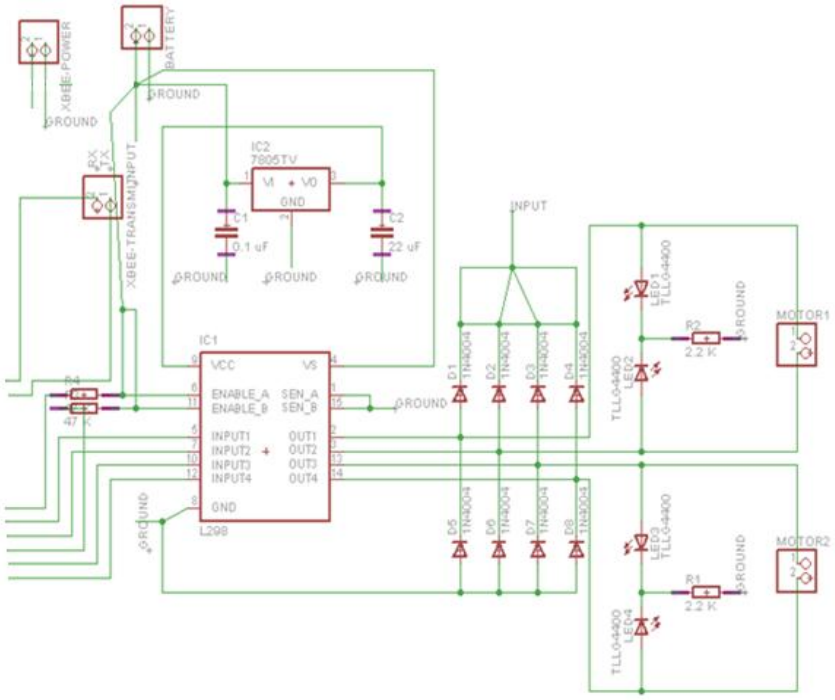
L	00/11	L	00/11	Off
L	01	L	01	Off
L	10	L	10	Off
H	00/11	L	00/11	Stop
H	01	L	00	Forward
H	10	L	00	Back
H	01	H	01	Right
H	01	H	10	Left

Table .Logic Table for Motor Control

Since the DC drive motor and turning motor that were chosen operate at 5 volts, it is necessary to create a circuit that has an output of 5 volts as well. This will help to get the most out of the motors. First, The Xbee transmitter/receiver is connected to the Atmega in pins 0 and 1. The information that the Xbee receives from the computer is sent to the microcontroller for processing. This information is then sent to pins 14-19 on the microcontroller. Those pins will then control the L298N H-Bridge. Depending on the values sent to the pins, it will enable the drive motor, the turning motor, or both. This will make the vehicle go forwards, backwards, left, or right. The figure below is what the final schematic looks like with added pinheads, resistors, capacitors, and flyback diodes.



We also added LEDs to each of the motors so that we could see that they were working properly in the direction that we anticipated. The flyback diodes that were added were done so because avoid damage to the circuitry connected to the inductive load due to flyback effects of inductive load. When the flow of current to an inductor is suddenly stopped a large voltage appears across that motor in the opposite direction. This voltage appears because energy is stored in the motors magnetic field before the current is stopped. After the current has stopped, the magnetic field in the inductor releases its energy as electricity back into the circuit but in the reverse direction that it was supplied. The following figure is a close up of the motor control circuit.



7.2.6 Wireless transmitter/receiver

In order to reliably transmit the information from the motherboard to the printed circuit board on the vehicle, we needed to use some sort of transmitter and receiver combination. We specifically looked into three different options. Those options being the RF Link 315 MHz transmitter/receiver, the Xbee series 1, and the Speedstudio 315MHz RF transmitter/receiver respectively.

The first option that we looked into was the RF Link 315 MHz transmitter/receiver combination. It has an amazing range of up to 500 feet, with perfect conditions. The transmitter costs $3.95 and the receiver costs $4.95. This offered the very beneficial low total cost of $7.90. One of the cons to using this particular product was that the signal is extremely noisy and there was no method for pairing the devices easily.

The next option for a wireless transmitter/receiver combination was the Xbee series 1. It has an indoor range of up to 100 feet and an outdoor range of up to 300 feet, line of sight. The Xbees have been used extremely often and there are plenty of resources available on the internet to help with any troubleshooting. It is also extremely reliable with little to no noisy signal, and has very simple communication. As well as it being ready to use straight out of the box. The only con to using these particular models was that they were a high price of $21.99 for each of the transmitters and receivers.

The last option that we looked into was the Speedstudion 315MHz RF transmitter/receiver combination. This particular wireless transmitter/ receiver combination offered the best range out of each of the options. It has an extremely wide range of about 900 feet up to 1600 feet. In addition to the huge range of operation, it was also an extremely low price of $4.99 for the pair. Although the range is large, the signal ends up being very noisy and required the addition of an encoder and decoder to remove the disturbances.

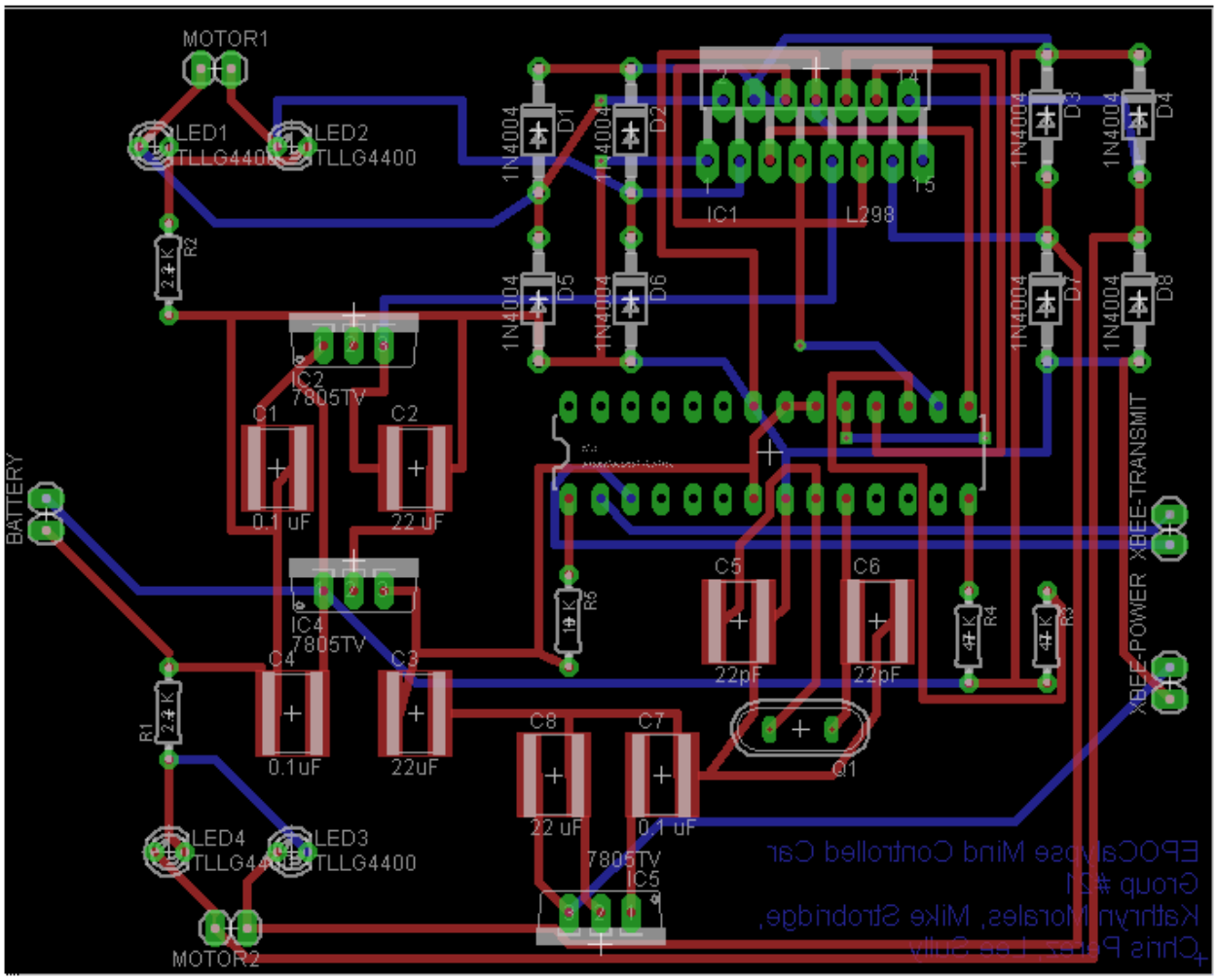
After looking onto the three options very carefully, it was decided that we would use the Xbee series 1 transmitter/receiver combination. This was done for the fact that it offers the most reliable signal and has the most resources available online for help with troubleshooting. Despite the fact that it was almost 5 times more expensive than the next option, it saved us some hassle dealing with noisy signals and having to add additional encoders and decoders. We then connected one Xbee to an Xbee Explorer Dongle dock. This could then be easily connected to the motherboard via usb. The second Xbee was then connected an Xbee Explorer Regulated dock. This could then be easily connected to the printed circuit board mounted on the vehicle.

7.2.6 Miscellaneous parts

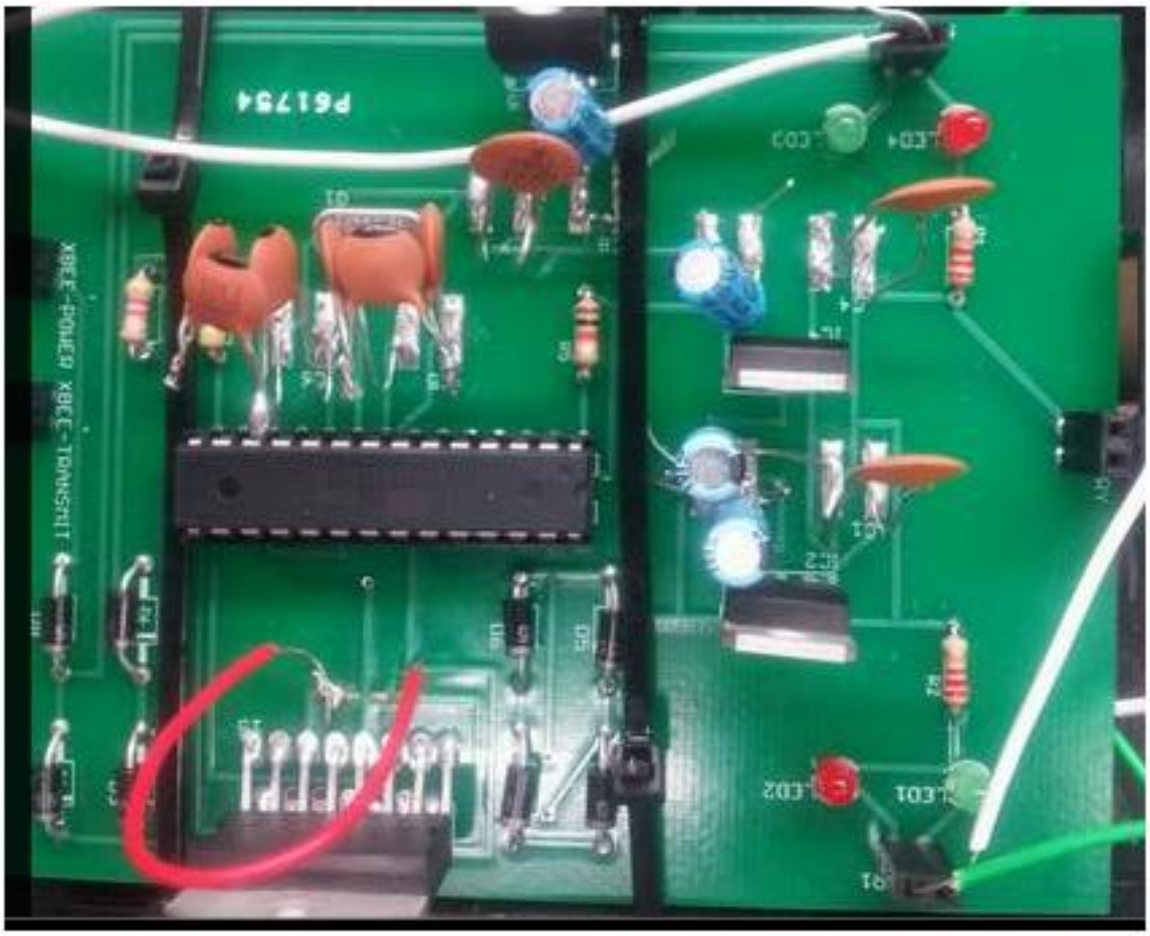
In order to properly mount each and every necessary component onto the remote controlled car, there are several miscellaneous parts that are also needed but not specifically listed. In order to properly mount the DC drive motor there will need to be a motor mount bought that fits that specific motor. As well as there needing to be multiple resistors and several wires to connect many of the components together. In order for proper biasing to help eliminate noise and create a more clear signal, biasing capacitors will be utilized as well. This is especially important so to create efficient time response to the Xbee receiver/transmitter as well as from the readings received from the headset via the usb receiver.

7.2.7 Final Printed Circuit Board

After finally putting together all of the respective components on to Eagle, we tested out the circuit on a breadboard in the lab. This was done so that we could make sure that everything was working properly before having our final printed circuit board made. After we tested the circuit on the breadboard and ensured that it was doing exactly as we anticipated, we made our gerber files. We then uploaded those files on to 4pcb.com. The following figure is a picture of the final schematic that was used.



The following figure is a picture of the final printed circuit board mounted on the vehicle with all of the respective components properly secured using soldering. All of the resistors, capacitors, diodes, voltage regulators, and LEDs are properly secured. As well as one extra wire that we had to add in because of a mistake made in our schematic on Eagle where two wires crossed. We had to cut the connection and then jump a wire over as you can see was done in the bottom left hand corner of the picture.



8. Software Design Content

8.1 Software Overview

The bulk of this project is the developed software that takes the initial signals sent from the headset and translates them into signals that are useable by the car. This software is broken into several sections. First, there is the software that resides in the motherboard. This software was written using the Emotiv API and its function is to receive signals from the headset, determine what the signal was, translate it into a 6 character signal, send the signal to the PCB and finally activate the proper motors on the car.

The other section of software was used in the custom PCB. The PCB contains the ATmega328 chip, which is also used on a typical Arduino board; therefore the software was written using the Arduino syntax. This segment of code was used to send the translated signals from the PCB to the motors on the car. These two sections of software are useless without each other because although the Emotiv software acquires the signals from the headset and filters them to the proper handlers, the software on the ATmega328 actually activates the physical motors to move the car.

Testing of this software consisted of using both the headset as well as the Emotiv EmoComposer. The reason for this was simple. Before any efforts were made to begin understanding the signals sent by the headset, it was important to make sure that the developed software was processing the signals correctly. This could be achieved without having to use actual signals from the headset.

The EmoComposer is a program distributed by Emotiv that allows developers to mimic the sending and receiving of signals. This program works exactly like the headset, meaning it requires the software to open a connection to it and also requires it to close the connection when all signals are received. This was extremely useful for this project because it allowed testing to begin before any actual signals were received. Once the code was proven to work with the simulated signals from the EmoComposer, it eliminated many possible sources of error when the actual signals were included in the translation process.

8.2 Emotiv Software

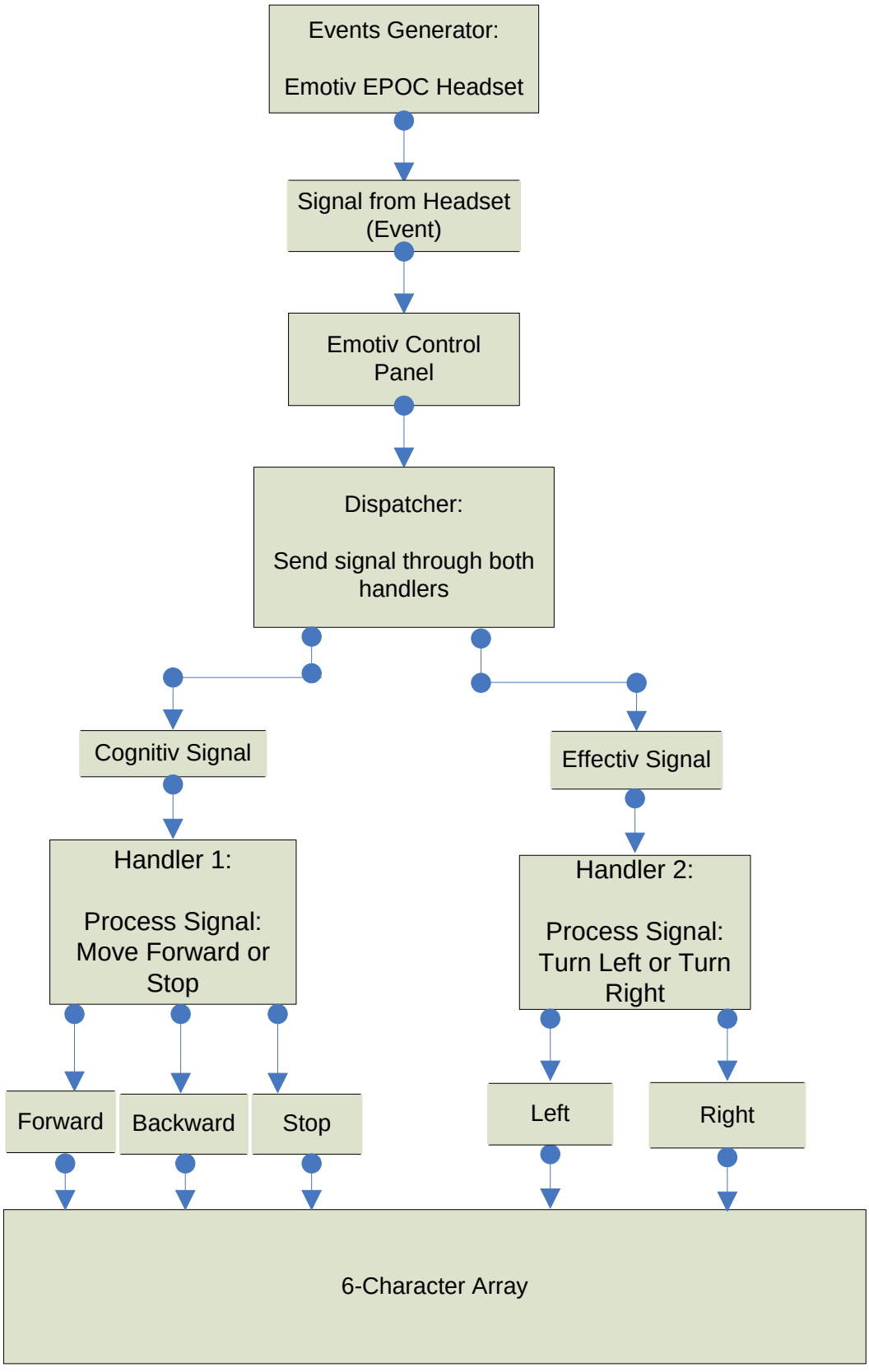
During the process of designing software for the Emotiv EPOC headset, one of the initial steps was to determine what architectural style would be used. During the design of this project, the following styles were considered:

- Client/Server
- Pipes and Filters
- Service-Oriented
- Event-Driven

After much consideration, it was decided that the Emotiv EPOC software would use an Event-Driven Architectural Style. This style was chosen for this project rather than the other mentioned styles because it is based entirely on events or an "identifiable occurrence" (Rouse). This style produces, detects and consumes events, which means each signal sent from the headset is considered an event production and is detected as well as consumed by the motherboard and RC car. This is different from the client/server style because this design creates events and these events trigger processes. In the client/server style, the client first makes a request to the server, and then the server responds with a message. This would not be useful for this project because the RC car does not request signals and the headset does not wait for requests but instead, the headset sends signals whenever the user decides to send a new command and the car executes the commands as they are processed and sent. The Pipes and Filters style was not used in this project because this style does not handle events very well. The decision making required to handle each sort of event sent from the headset would not be handled well using this architectural style. Finally, the Event-Driven style was chosen over the Service-Oriented style because the Event-Driven style was able to handle message routing. This was considered important because the events sent by the headset will need to be processed and routed at runtime, meaning there won't be a consistent data flow at all times. Due to this, the capabilities of the Event-Driven style were chosen to handle changes in the flow of the system.

A block diagram showing how the Event-Driven style was applied to this project is included below in Figure 8.1:

Figure 8.1: Emotiv Software Event-Driven Architecture



The figure above briefly shows the flow pattern that this section of software follows. The events (meaning, the signals sent to the computer) are generated by the headset and are sent to the Emotiv Control Panel. Each of these signals is referred to as an EmoState. Each EmoState triggers a new event for the software to process. From there, a dispatcher obtains the signal and sends it through both the cognitive handler and the expressive handler in order to interpret and process the signal. This means the dispatcher is only meant to send the signals to the handlers and then receive the processed signals. It is the handler's job to determine the exact command being sent by the headset. Once the necessary information is acquired from the signal, it is then combined into a single final command string. This string is then used by Converter class to make a 6 character signal, which is then sent to the Atmega328 to move the car.

The first important task that needed to be done in the software was to establish a connection between the headset and the motherboard. The connection was made with the Emotiv EmoEngine program. This meant that in order to gain access to the data sent by the headset to the processor, the software had to connect to the EmoEngine. This was easily done by calling the EE_EngineConnect() method. A successful connection was confirmed with the following block of code:

Source: Emotiv. Emotiv Software Development Kit User Manual for Release 1.0.0.5. pg 48

```
if (EE_EngineConnect() != EDK_OK)
{
    throw std::exception("Emotiv Engine start up failed.");
}
```

Reprinted with Permission from Emotiv

This block of code calls the EE_EngineConnect() method and checks to make sure that the response received from it is EDK_OK, meaning the connection was successful. The hexadecimal value 0x0000 corresponds to this response, meaning if any other hexadecimal number is received, the code throws an exception and lets the user know that the EmoEngine could not be accessed.

Once a connection is established, the next step was to begin receiving signals from the headset. This meant the code needed to run in a loop in order to get all the signals sent from the user until the connection was closed. During each iteration of the loop, the code receives a signal from the headset. If the signal represents a valid state, then a new event is made which will then be used to determine whether the received signal represents a new command. The method used for obtaining the newest event is EE_EngineGetNextEvent(eEvent), where eEvent is the current event. The result of this method call is an integer, which is then used to determine whether the new state is valid or not. If the result corresponds to the state of EDK_OK, then the event is valid and can be used. Once again, the EDK_OK response is represented by the hexadecimal value of 0x0000. Any other value means that the state is either invalid or there is no new state being received at that particular time.

After the received state is confirmed to be valid, it was then necessary to verify that the received signal represents a new command. This was necessary because a signal from the headset could be received not only when there is a new command, but also if there is a user added from the EmoEngine as well as a few other situations. Signals received from the headset are tested with the EE_EmoEngineEventGetType() method. This method returns the type of the most recent signal from the headset. The range of event types is shown in Appendix 3 of the Emotiv User's Manual and is included below:

Source: Emotiv. Emotiv Software Development Kit User Manual for Release 1.0.0.5. pg 77

EmoEngine events	Hex Value	Description
EE_UserAdded	0x0010	**New user is registered with the EmoEngine**
EE_UserRemoved	0x0020	**User is removed from the EmoEngine's user list**
EE_EmoStateUpdated	0x0040	**New detection is available**
EE_ProfileEvent	0x0080	**Notification from EmoEngine in response to a request to acquire profile of a user**
EE_CognitivEvent	0x0100	**Event related to Cognitiv detection suite. Use the EE_CognitivGetEventType function to retrieve the Cognitiv-specific event type.**
EE_ExpressivEvent	0x0200	**Event related to the Expressiv detection suite. Use the EE_ExpressivGetEventType function to retrieve the Expressiv-specific event type.**

EE_InternalStateChanged	0x0400	**Not generated for most applications. Used by Emotiv Control Panel to inform UI that a remotely connected application has modified the state of the embedded EmoEngine through the API.**
EE_EmulatorError	**0x0001**	**EmoEngine internal error.**

Table 8.1 : Event types in Emotiv software. (Reprinted with permission from Emotiv)

If the new signal received from the headset was indeed a new command, then the method returned EE_EmoStateUpdated (0x0040) as its result. This meant that a new command was being called and this new EmoState was sent to the motherboard in order to be processed by the software.

The next step to this process was to send the new EmoState to the corresponding handler. This project uses both Cognitive and Expressive signals from the headset. A new EmoState is acquired from the headset whenever an updated signal is received. The signals from each suite are handled differently and the Cognitive signals require prior training in order to process the signals correctly. Signals from each suite were handled in separate classes, CognitiveHandler for cognitive signals and ExpressiveHandler for facial expressions. These two classes have methods that are able to take the signals from the suites and extract the necessary information from them in order to send it to the car. The necessity of two different handlers arises from the fact that the two different types of commands are handled differently. There is no universal method in the Emotiv API that allows for extraction of all the possible information from the EmoState; therefore it was necessary to send the EmoState through two different handlers, where the Cogntive and Expressive data could be handled correctly. The two handlers were also used for organization. It was easier to separate the two different processes into two different classes and then simply sending the results back to the dispatcher. This allowed for easier testing as well.

Despite the need for two different handlers, the end result is always a useable signal that can be sent to the car for execution. The data flow diagram below shows both handler classes and how they interact with each other as well as with the Dispatcher class:

Figure 8.2: Data Flow Diagram: Handler Interactions

The methods that will be used in each class are explained below in Table 1:

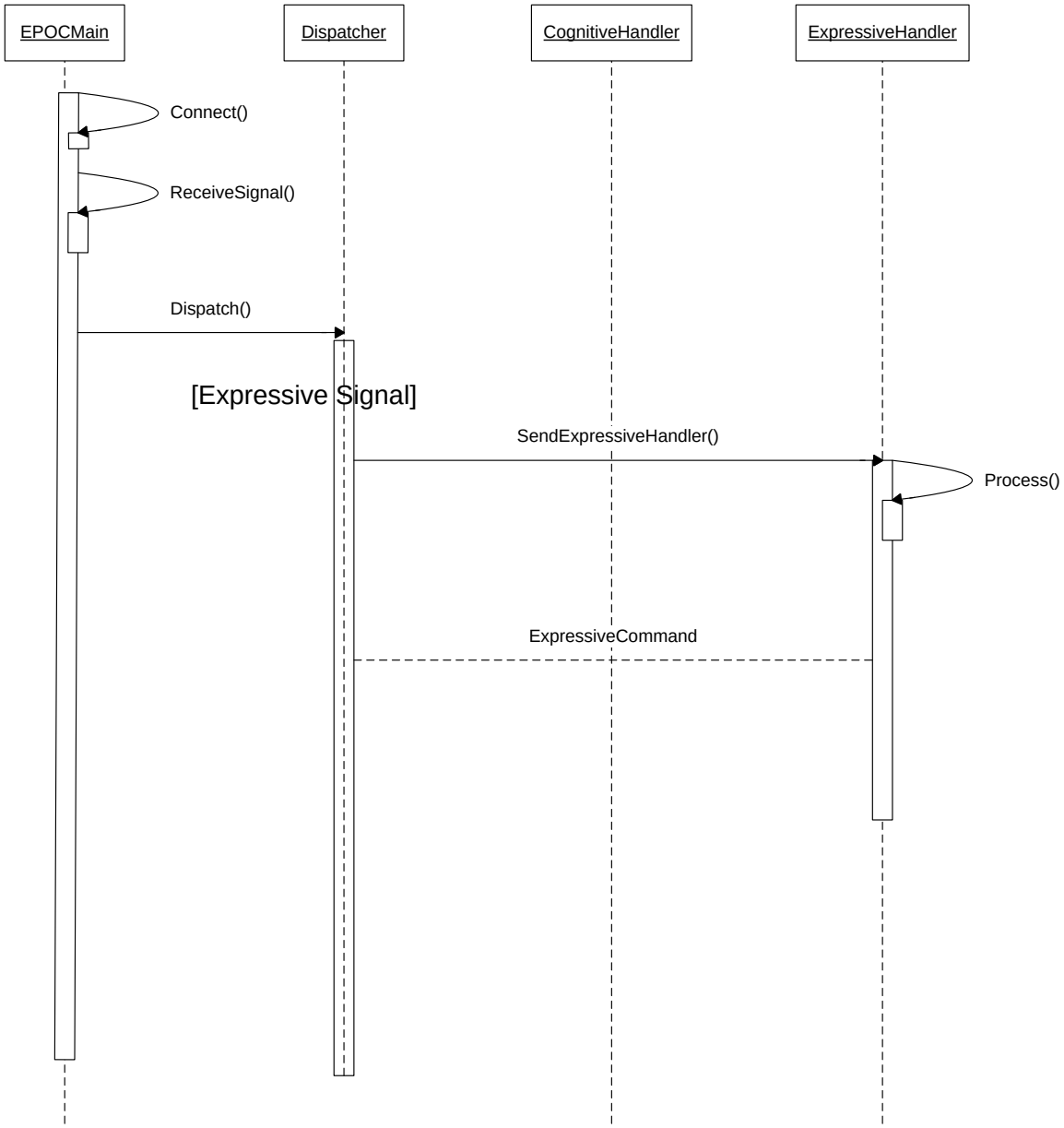
Table 8.2: Classes and Methods of Emotiv Software

Class	Method	Explanation
EPOCMain	Connect()	Makes an instance of the Receiver class and begins process of communicating with headset
	receiveSignal()	Gets signal from headset
	Dispatch()	Makes an instance of the Dispatcher class and handles the signal
Dispatcher	DetermineHandler()	Sends the EmoState to the handlers for processing
	SendStopTurn()	Sends the command to straighten the wheels
	SendCognitiveHandler()	Sends EmoState to the CognitiveHandler
	SendExpressiveHandler()	Sends EmoState to the ExpressiveHandler
	getGUIcogCommand()	Gets the processed cognitive command
	getGUIexpCommand()	Gets the processed expressive command
CognitiveHandler	Process()	Uses the EmoState to determine the cognitive command

ExpressiveHandler	Process()	Uses the EmoState to determine the expressive command
	StopTurn()	Returns the command to straighten the wheels
	getActualCommand()	Used for expressive logic (Returns processed command)
	setPrevCommand()	Used for expressive logic (keeps track of the previous command)
	getPrevCommand()	Use for expressive logic (returns the previous command)

Overall, this software has 2 separate sequences for successful signals. The first possible sequence is successfully receiving a new facial expression command. The sequence for this event is shown below:

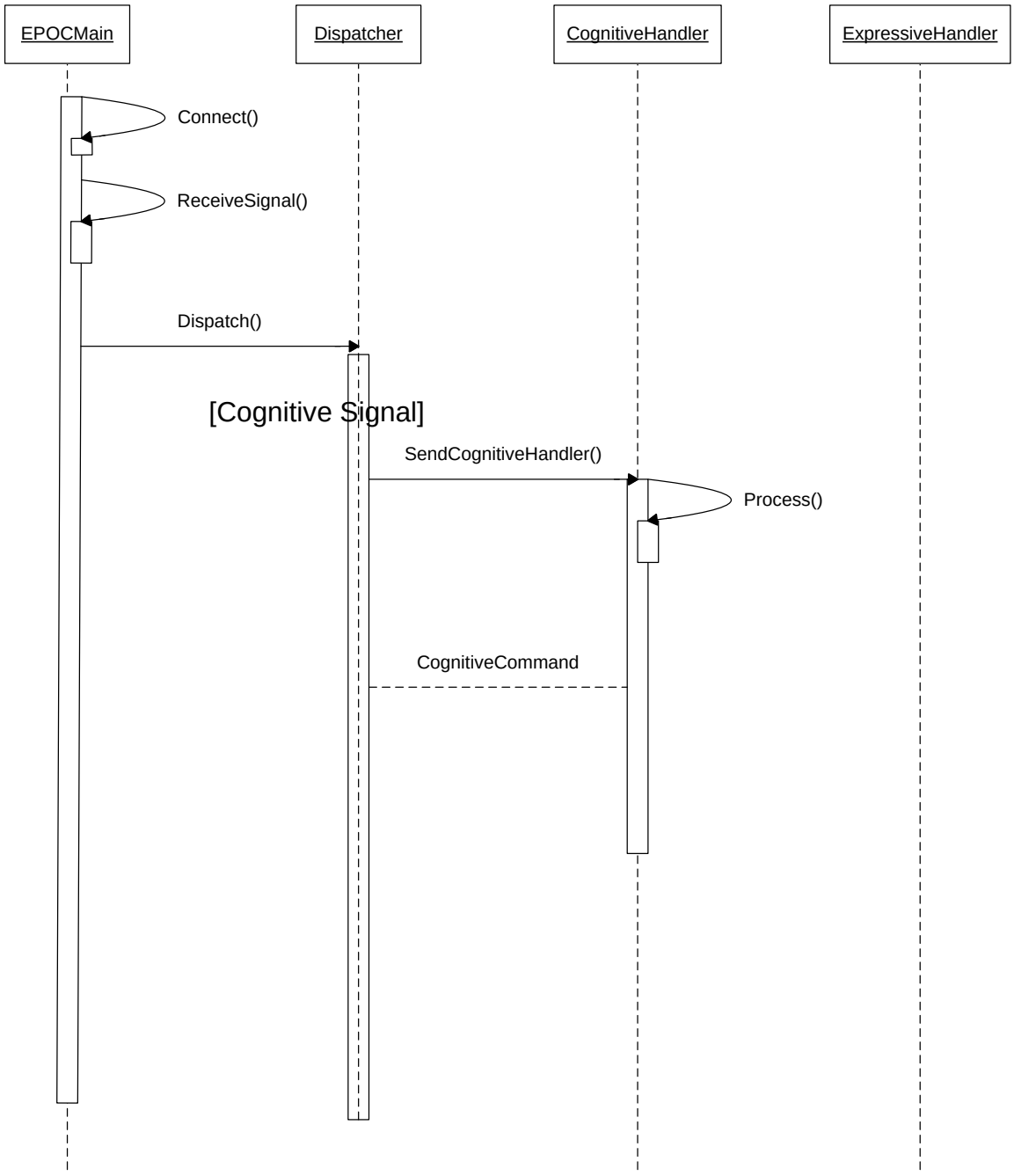
Figure 8.3: Effective Signal Sequence



In this sequence, an instance of the ExpressiveHandler class is made and this processes the signal in order to tell the car to either turn left or turn right, depending on

the command sent by the user. The second possible sequence is successfully receiving a new cognitive signal. This sequence is shown below:

Figure 8.4: Cognitive Signal Sequence



This sequence creates an instance of the CognitiveHandler class which interprets the cognitive commands and tells the car to either move forward, backward or stop.

Any expressive signals that are used in this project are used to turn the car left or right. The facial expressions that are used are a right or left wink. In the Emotiv API, facial expressions using winking can be handled by using "Eye related actions" (Emotiv, 50). Methods from the API are used at first in order to verify if the user is winking with their left or right eye. This determines which eye the user is winking by returning either true or false. When the correct eye is determined, the Expressiv action type is set to either wink left or wink right. This action type is then made into a String, which is passed to the XbeeSerial class in order to send it to the ATmega328 on the PCB. A table showing the strings assigned to each expressive command is shown below:

Table 8.3: Strings Assigned to Facial Expression Commands

Expressive Commands	Assigned String
Wink Left (Turn Left)	"WL"
Wink Right (Turn Right)	"WR"
Go Straight	“GS”
Straighten Wheels	“SW”

The Emotiv API also allows for the retrieval of the intensity of the command. This is used to measure the movements made by the user. An example of the use of this would be to measure how high the user's eyebrow is raised. This aspect is not used for the Expressive actions in this software. A simple wink always triggers the proper command.

The cognitive signals in this project are used to move the car forward, backward and to stop the car. These signals are acquired by using EEG data sent from the headset. This data is taken from each of the 14 sensors on the headset. By using the API provided by Emotiv, it is possible to use the training feature of the Emotiv Control Panel in order to

teach the software how to interpret the EEG data sent from the headset. In order to gain accurate results, it was necessary to train the desired commands several times. As more practice sessions were saved on the computer, it was easier for the signals to be interpreted correctly. Once we were able to get accurate results from the training sessions, we used the API from Emotiv to interpret the signals and process them into a String value, just like the Expressive Handler. The assigned strings for each command are shown below:

Table 8.4: String Assigned to each Cognitive Command

Cognitive Command	Assigned String
Move Forward	"Push"
Move Backward	"Pull"
Stop	"Neutral"

Once the signals have been made into EmoStates and these EmoStates have been handled, the signal commands received from each handler are then put together into a final command string. The first two characters of the final command strings always represent the expressive command. The remaining characters of the string represent the cognitive command. An example of a final command string could be WLPush. For this string, the final command that was processed was to move the car forward and turn the wheels left.

Once the final command string is made, it is then sent to the Converter class, which converts the string into a 6 character signal. After converting the signal, it is then sent to the XbeeSerial class in order to send the signal to the PCB on the car.

The data transfer between the motherboard and the PCB is initiated by the motherboard. As soon as the motherboard has new extracted data from the EmoStates, it sends the data to the PCB. The PCB then sends the 6 character signal to the

ATmega328, which analyzes the signal and activates the necessary motors in order to move the car in the desired direction.

Once this software was completed, it was packaged into a project solution using Visual Studio. This solution contains all the source code as well as the necessary header files and libraries in order to run the code successfully.

8.3 Arduino Microcontroller

Arduino is an open source microcontroller built onto a printed circuit board (PCB) to provide the necessary electronic connections. The Arduino functions as an interactive device by receiving input from sensors to manipulate devices that are connected to it. With a microcontroller embedded onto the PCB, input and output port signals are provided that enables digital information to be communicated between the Arduino and a physical real world peripheral. The Serial port on the controller, allowing processed information to be passed in a bidirectional manner sends one bit at a time in the order that they are initiated in a stream of buffer. Ports can be selected so that they are connected to a desired corresponding external device such as computers, monitors and etc. through Universal Serial Buffer (USB). I/O operations can be used for read and write or to turn off and on materials and devices by having the pins being set to high or low.

The microcontroller is preprogrammed with Arduino software environment and uses sketches which are instructions to tell what the Arduino to do. Once a sketch program is completed it is verified and compiled which debug the code and translate it into an application that is compatible with the Arduino hardware in order to upload the written software onto the Arduino board. The board must be reset either done manually (pressing a black button on the board) or automatically depending on the type of Arduino used in order to upload a new sketch onto the microcontroller. Arduino is very popular because of how it is open source and ease of use of hardware and software. There are many different types of Arduino boards and there are also many tutorials that could be found online for first timers in the initial set up process, tutorials on how to program onto the Arduino for those wishing to create, design and find new ways to interact with many objects. Arduino is highly accessibility and available for purchase with the software available to download for free online. So much ingenuity has sparked so much that one may be able to decide to build a custom Arduino PCB with simple tutorials readily available online.

Programming with the Arduino development environment is done by uploading sketches onto the microcontroller written in Arduino Programming Language. This programs the Arduino microcontroller with the instructions to follow by verifying/compiling and uploading the sketches. The purpose of using the Arduino programming language is to program the micro processor chip that is located on our PCB board on the car. The Atmel ATmega328 microprocessor chip with the Arduino Duemilanove bootlloader preloaded onto it is embedded onto the Arduino Deumilanove microcontroller PCB

development board. It is a high performance Atmel 8-bit AVR Reduced Instruction Set Computer (RISC). This microcontroller is pictured below.

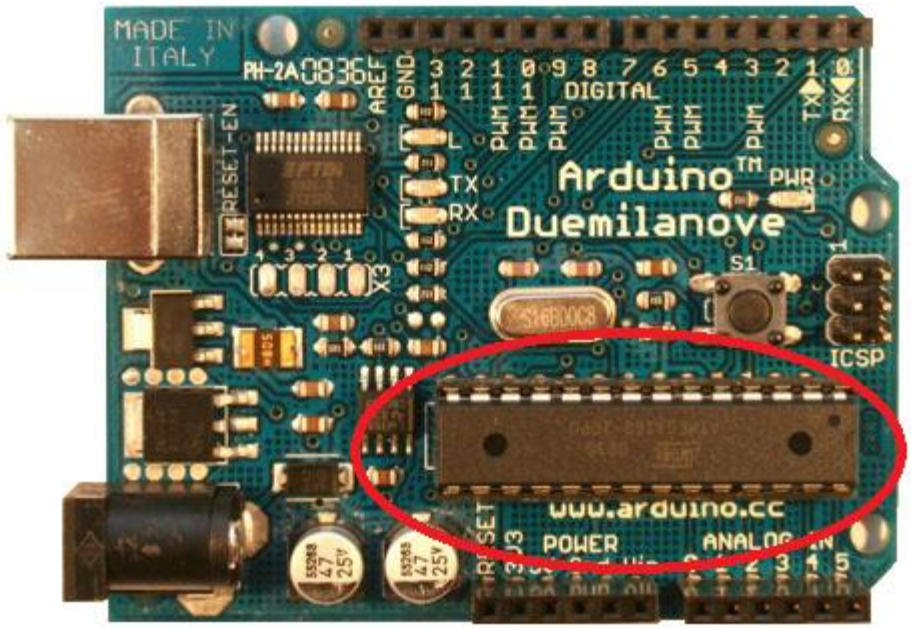


Figure 8.3.1 Reference Arduino Deumilanove Atmel AtMega328 Microcontroller

With the unique architectural design of the AtMega328 microprocessor and using the microprocessor to store our data to be sent and having Arduino sketch language code written into the microcontroller enables instructions to be followed based on the data received.

8.4 Atmel AtMega328p-pu Arduino

For our project only the Arduino Atmel AtMega328p-pu microprocessor located on the Arduino Duemilanove microcontroller development PC board was used. In order to get the minimal amount of task and processing power needed by the microprocessor for faster performance as required by the real time response of the project. The Atmel AtMega328p-pu microprocessor/microncontroller chip was chosen also because it meets the design constraints of our project. The Atmega328 has an operating voltage of 5V, 32 KB of Flash Memory and EEPROM of 2KB for storing the program. It also has 14 digital I/O pins that can be configured for either input or output. Highlighted in red are pins used in our project for connecting to external peripheral devices. Pins 0 (RX) is

used for serial communication and is used to receive serial data transmitted. Digital pins 14-19 are mapped and are manipulated by the Port B Registers for writing and reading. So with the Duemilanove bootloader already programmed onto the AtMega328 microprocessor we simply upload our Arduino language program sketch code onto it. More Specifically the programmed software on the Arduino AtMega328 processor will receive serial data in the form of a continuous set of string pairs. The AtMega328 will simply receive the binary character sequence and output each individual character to a certain pin. It will be used to store Arduino programming language code sketches and process the information sent from the Emotiv API software on the motherboard. These continuous data is received wirelessly via a transmitter receiver pair.

Source: hobbytronics. http://www.hobbytronics.co.uk/arduino-atmega328-pinout

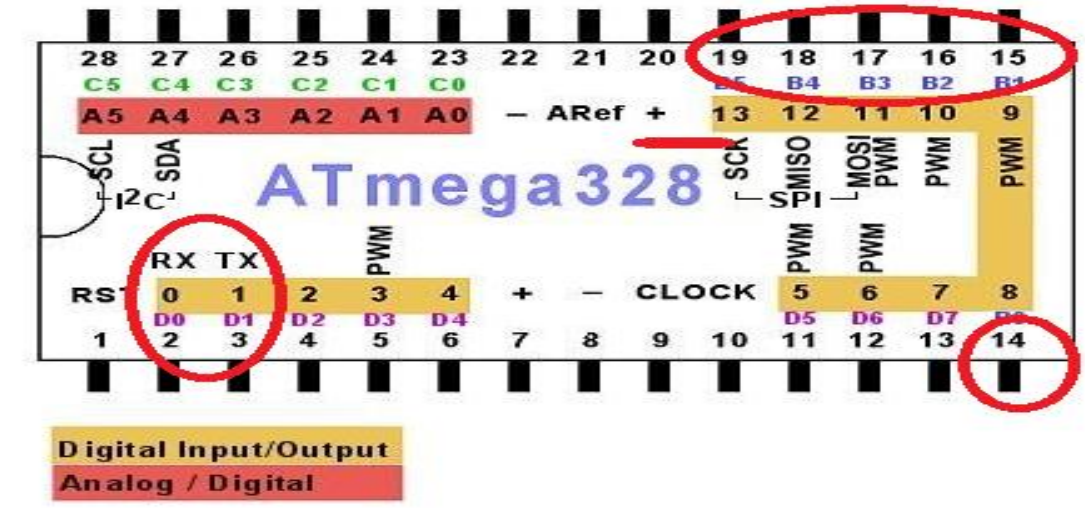


Figure 8.4.1. Atmel AtMega328p-pu pinout Arduino

The above illustration shows the Pin Mapping for the specific microprocessor used on our PCB. The Atmel AtMega328p-pu microprocessor chip is preloaded with the Arduino Duemilanove bootloader version to load the Arduino sketch code described.

8.5 Arduino Libraries and functions

Arduino contains procedures and libraries that were very useful for implementing this program.

```
void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}
```

Figure 8.5.1 Setup Command

The above figure shows a procedure called Setup ( ). Setup ( ) is similar to the main procedure in a C++ program. It doesn't take in any input and returns void. In this procedure variables are initialized and libraries begin to be used in this function. This procedure runs when a sketch starts.

```
void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Figure 8.5.2 Arduino code with adaptive response

The loop ( ) function iterates the statements within them repeatedly allowing control of the Arduino PCB board by allowing the program to change and respond as it iterates. Both the setup and loop functions are structures to the language for the Arduino programming language.

Arduino has control structures that may be used such as if, if...else, for, while , switch case, and return for conditional operations, control overflow, repeat code or loop conditionally and etc. Arithmetic are performed with = (assignment operator) storing the value to the right of the equal sign in the variable to the left of the equal sign, + / - for addition and subtraction, * for multiplication, / for division and % for modulo. These are followed by comparison operations with == for equal to,! = not equal to and etc similar to C++. The data types are void, Boolean, char, byte, int, word, string, array and etc.

The Serial communication library communicates between the Arduino AtMega328 microprocessor chip via serial ports on digital pins 0 (RX), 1 (TX) and the motherboard

through USB. In particular the Arduino environment contains many libraries one of which the Serial library contains a set of procedures as shown in the figure below.

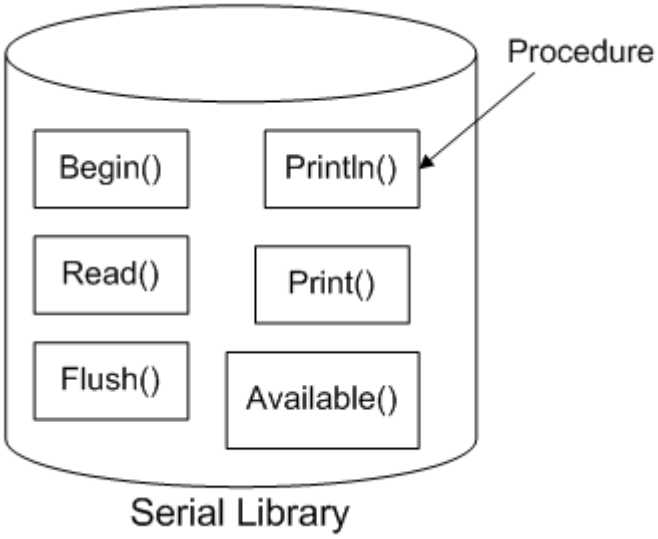


Figure 8.5.3 Serial library overview

For example this library procedural statement call is a saying that the procedure begin is in the serial library. The input to this procedure is the baud rate which is how fast the connection can read and write the bits on a wire.

```
Serial.begin(9600);           // set up Serial library at 9600 bps
```

Figure 8.5.4 Serial begin function

Information is being communicated through serial data transfer by setting pins to either high or low, this is shown in the figure below how bits 0 or 1 being transferred one bit at a time due to the USB serial connection between a computer system and a particular model of an Arduino.

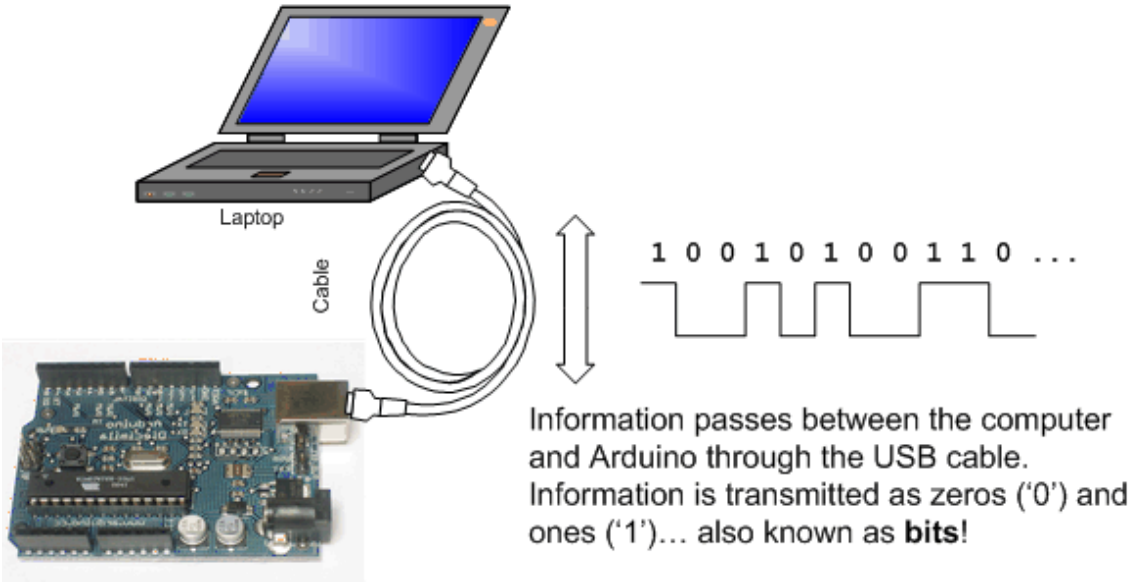


Figure 8.5.5 Information pathway from DH61AG to Arduino.

When the software is compiled it is translated into binary data. Each bit is sent one at a time through a USB cable connection to the ATmega328 microcontroller of the Arduino PCB during uploading. The RX LED emits when the Arduino receives data and the TX LED lights up once it is transferring data. The data and output can be seen on the Arduino environment serial monitor. The Arduino language has an abundance of functions and libraries.

However, since we did not use the Arduino Duemilanove microcontroller PCB development board and only used its AtMega328 microprocessor chip for simply processing the binary character sequence sent from the motherboard storing the Emotiv software and outputting each individual character to a certain pin, and the fact that the microprocessor is not directly connected to the motherboard via USB to use the Serial Library – the SoftwareSerial was used as only the AtMega328 microprocessor is used and connected to our PCB located on the car another library was needed besides the Serial Library.

The Arduino Environment supports the SoftwareSerial library for communication with peripheral devices for reading in serial data which we are sending. The SoftwareSerial Library is like the Serial library of the Arduino so they both have the same functions such as available (), begin(), read(), println() and write( ). Shown in the figure below is a sample code from this sketch is adapted from examples and tutorials from the Arduino website.

Source: http://arduino.cc/en/Reference/SoftwareSerial

```
void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(57600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  Serial.println("Goodnight moon!");

  // set the data rate for the SoftwareSerial port
  mySerial.begin(4800);
  mySerial.println("Hello, world?");
}

void loop() // run over and over
{
  if (mySerial.available())
    Serial.write(mySerial.read());
  if (Serial.available())
    mySerial.write(Serial.read());
}
```

Figure 8.5.6 SoftwareSerial library example

which is also based off of Mikal Hart examples of Software Serial.

8.6 The XBee software configuration

What was originally planned was to allow the microcontroller and the motherboard to send and receive information from each other by opening socket connections. An Xbee Series 1Transmitter/Receiver module pair was used instead to communicate between the motherboard and the car.

The Cognitive and Expressive impulse response signals measured by the Emotiv EPOC EEG reading headset being processed by the Emotiv API software on the motherboard. The Dispatcher, Converter and XbeeSerial classes in the Emotiv software extracts the information, translates, stores them into a character array signal of length six which corresponds with the impulse response that triggered the event and this character is then sent to the XBee's.

The XBee transmitter is directly connected to the motherboard via USB serial port. The character signal is written onto the XBee transmitter. The XBee transmitter sends the data written into it (via USB serial port connection) to the XBee Receiver. The XBee Receiver DOUT is tied to the Receiver (RX) pin 0 line of the AtMega328 microprocessor of the PCB on the car.

This successfully sends the character signal that needs to be sent from the motherboard and onto the car.

The XBee's are configured using the XCT-U software environment from Digi. It sets and configures parameter settings of the XBee modules. A common communication channel line was established for both XBee's to recognize each other. Also, the baud rate was set to control the rate at which the flow of data are sent across the XBee's.

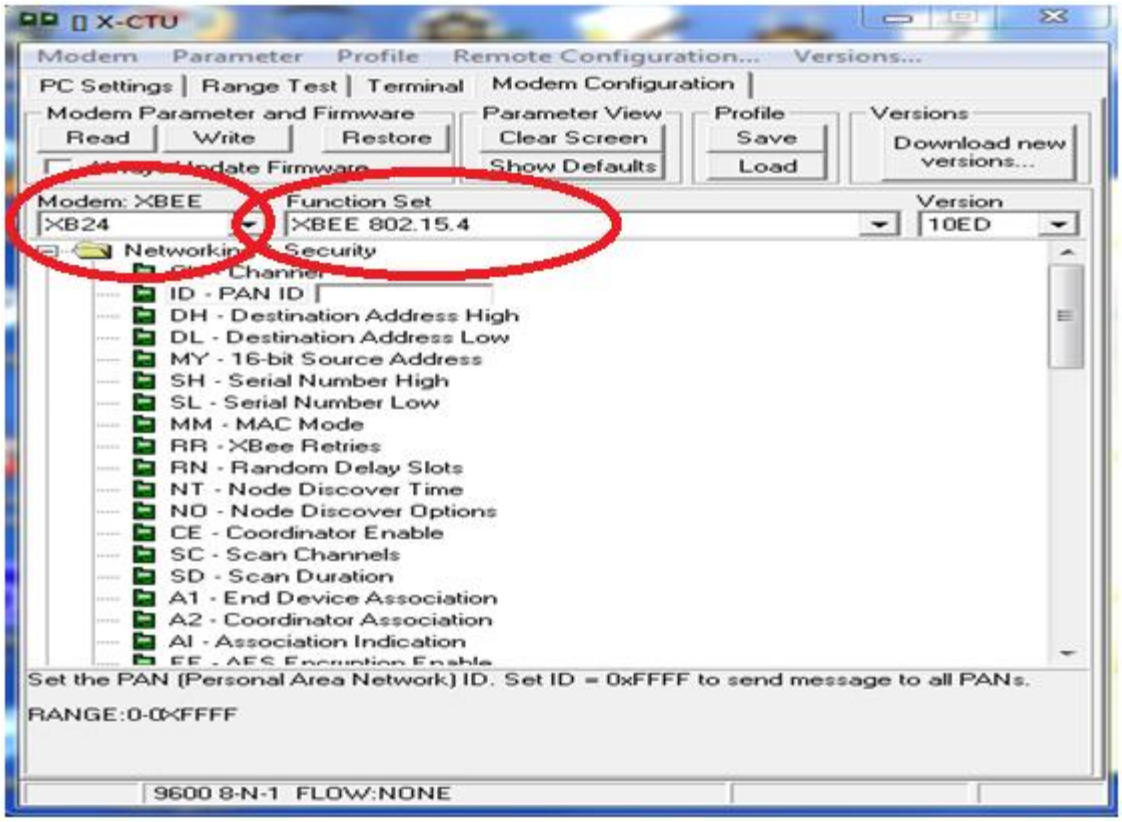


Figure 8.6.1 XCT-U software environment from Digi.

Shown highlighted in the figure above is the modem configuration for the specific XBee Series 1 module that was used in this project. The function settings and software version is based on our modem configuration is also shown. From these initial settings and setting the parameters such as the communication channel, baud rate and the PAN ID's (which each all must be set the same between the two XBee Series 1 transmitter/Receiver module) were set for proper communication.

8.7 Graphical User Interface (GUI)

Once the source code was completed for this project, the final step was to make a GUI that would make the code not only functional but also user-friendly. A window with buttons and text boxes is much more familiar and easier to use than a command window. The GUI was made using Visual C++ in Visual Studio and consists of four windows.

The first window of the GUI is the main screen, which allows the user to determine whether they will be connecting to the computer via the physical headset or the simulator (EmoComposer). The usefulness of the GUI is already seen on the main window because it allows the user to simply click the button that states the desired action. The main window is shown below:

Figure 8.7.1 Main Window of GUI

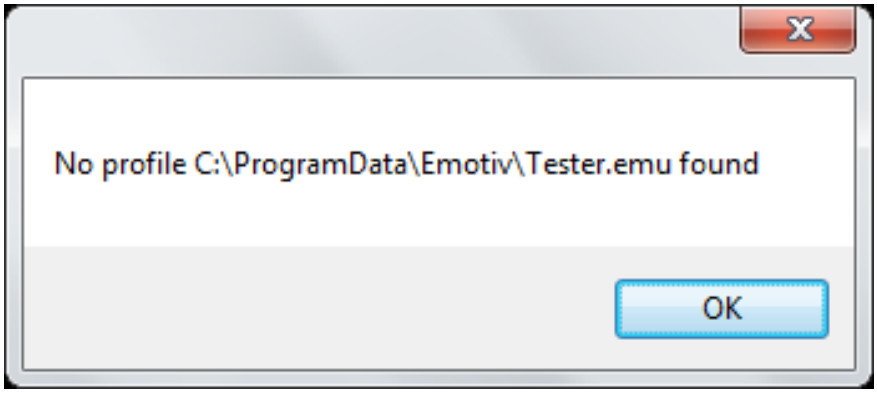
As seen on the main window, the user has two options. Each option brings the user to a new window. If the user picks to connect to the EmoEngine, this means the user is

using the physical headset. As stated before, in order to get accurate results from the software, it was necessary to train using the headset. This training information is saved locally on the computer and each user can have their own profile. Due to that, the next window the user sees if they connected to the physical headset would be a window asking for the name of their profile. The window is shown below:

Figure 8.7.2 Profile Name Window

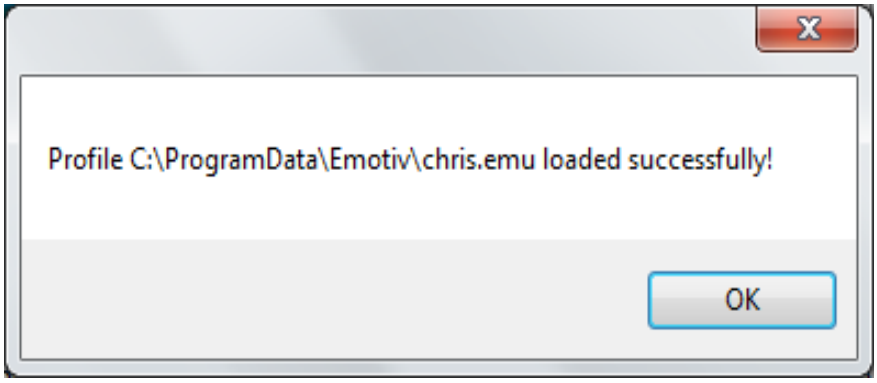
This window gives the user the opportunity to type in the name of any of the existing profiles. In order to check for any user errors, the name inserted into the textbox is quickly checked for existence in the profile directory. If no profile exists with the provided name, the user is alerted via a pop-up window. The profile name window is then closed and the user is returned to the main window. This pop-up window is shown below:

Figure 8.7.3 No Profile Found Pop-Up Window



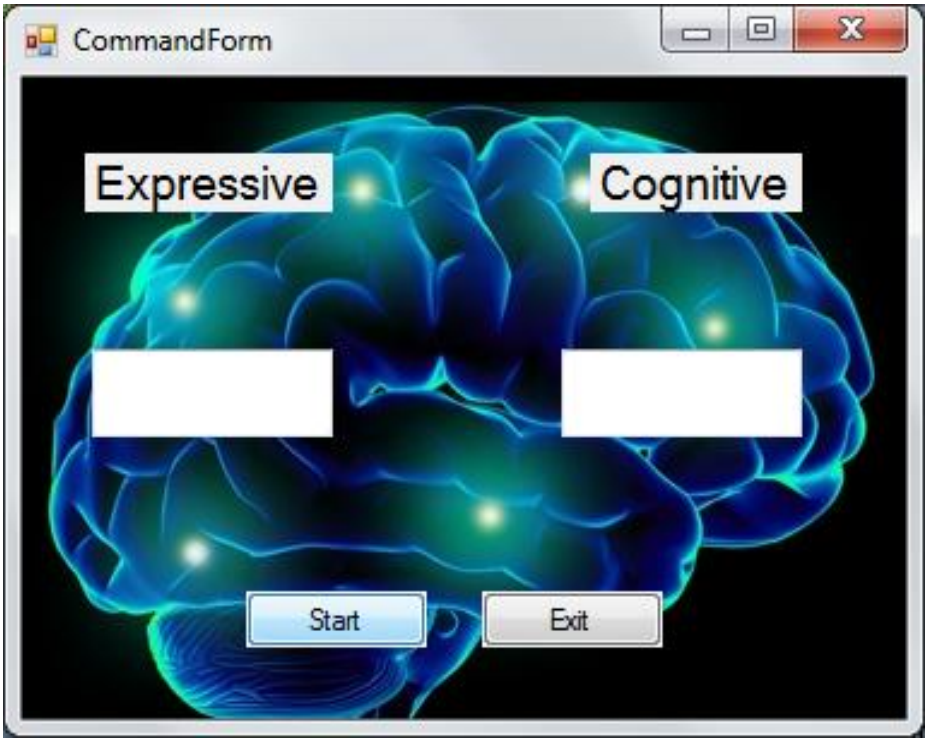
If the profile does exist, a different pop-up window is displayed to the user informing them that the profile was found and has been loaded successfully. This window is shown below:

Figure 8.7.4 Profile Found Pop-Up Window



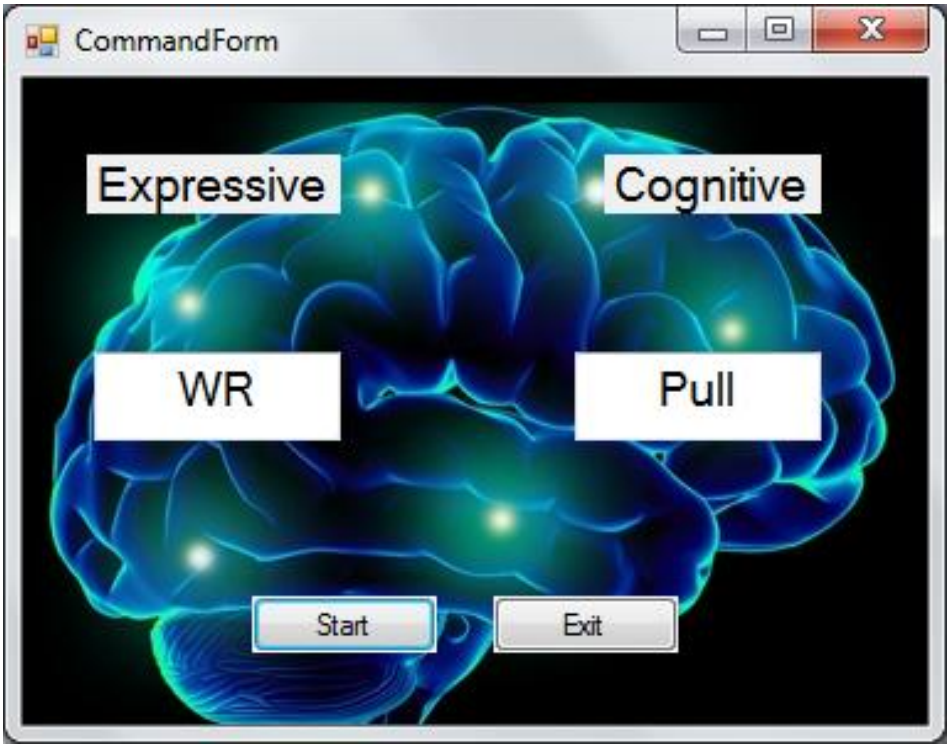
Once an existing profile is loaded, the user is then taken to the final window, which displays the commands being sent to the car. This window is shown below:

Figure 8.7.5 Command Window



The user will notice that the windows for the cognitive and expressive commands are empty when the window first opens. This is because the source code has not started running when the command window is opened. In order to begin running the source code, the user needs to click the Start button. When the Start button is clicked, the GUI opens a new thread in the software that runs the original source code in the background. Using this additional thread allows the GUI to function correctly by waiting for the user to click the Exit button. Until the user clicks the exit button, the text boxes in the window are filled with the current command being sent to the car. An example of the window is shown below:

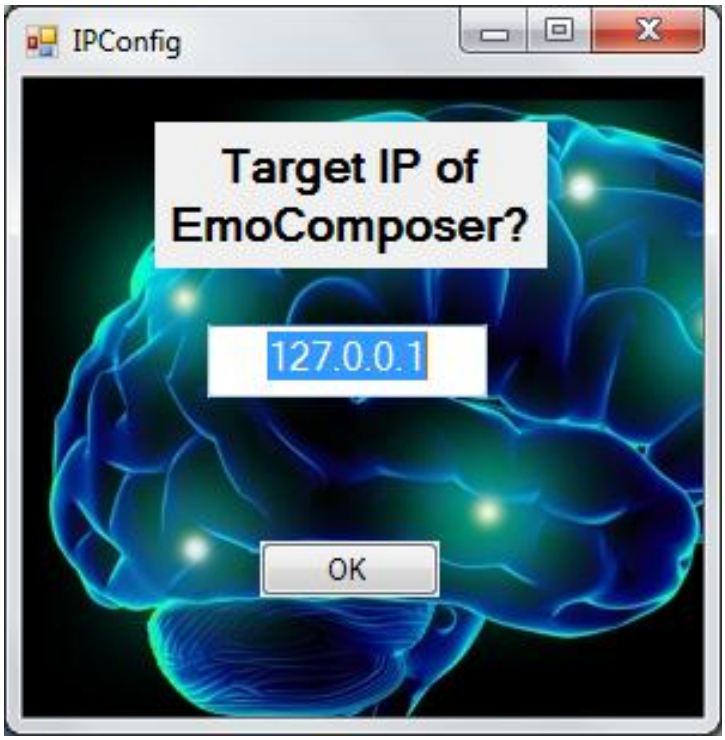
Figure 8.7.6 Active Command Window



In this example, the source code is running in the background and the current signal has been processed as a right wink (WR) and backwards (Pull) command. When the user is ready to exit the software, they can click the Exit button, which terminates the thread running in the background and also closes the active windows from the GUI. This shuts down the entire software process and signals are no longer sent to the car.

The second option the user has is to connect to the EmoComposer and send signals to the car via the simulator. This option requires the user to enter a target IP for the EmoComposer. If the EmoComposer is open on the local computer, the default IP can be used (127.0.0.1). If the IP needs to be changed, the user may do so by changing the text in the IP window. This window is shown below:

Figure 8.7.7 IP Window



Once the user enters a correct IP, they are then directed to the same command window that is seen when connecting with the actual headset. This window is navigated in the same way, regardless of which option is selected by the user.

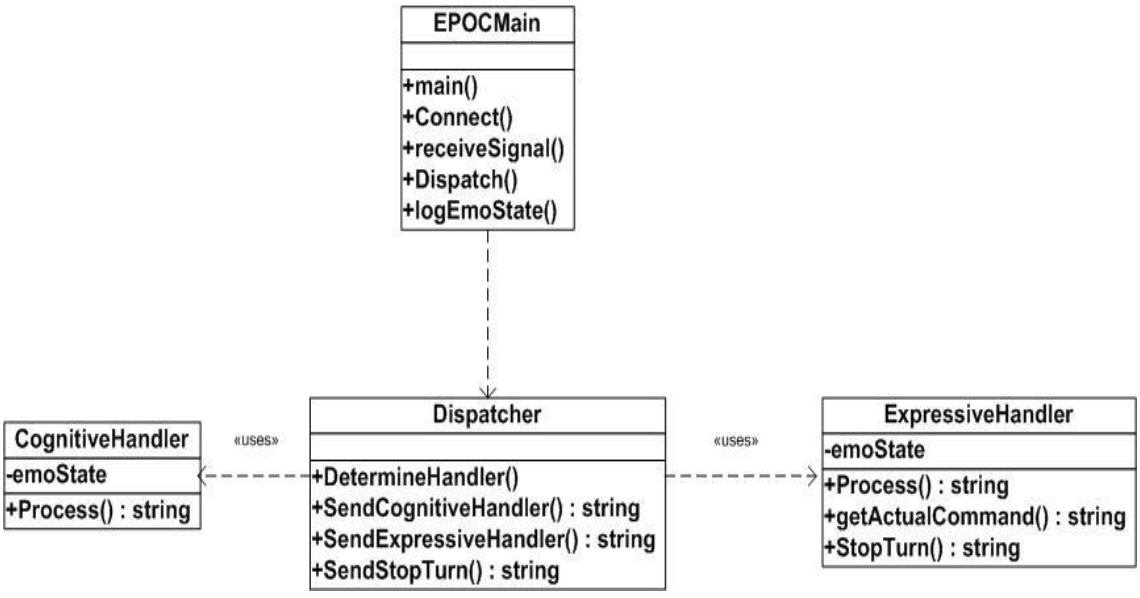
9 Design Summary of Software

9.1 Emotiv Software

The software developed using the Emotiv EPOC SDK is a vital part of this system. The functionality of this software results in the retrieval of signals from the headset as well as interpretation and processing of these signals using the Emotiv API. This software was written in C++. This language was chosen due to its object-oriented design which made it easier to organize the code. Also, much of the software included with the Emotiv package is written in C++, therefore developing the code for this project in C++ facilitated in merging the code into one project. It was also decided that Visual Studio would be the selected IDE for this project. Visual Studio provides very easy-to-use project organization as well as much C++ development support.

One of the main advantages of using C++ as the programming language for this project was the fact that it is object-oriented. This allowed the code to be organized into classes and reusable code. The class diagram representing all the classes as well as their connections is shown below in Figure 9.1:

Figure 9.1.1: Class Diagram



The Emotiv Research Edition SDK comes with several GUIs as well as its own API that proved to be very helpful for this portion of the software. Along with all these features, Emotiv also included a user manual with example code that allowed the group to become more familiar with the API. The first task that was completed was to write code that established a connection between the motherboard and the headset. This was important because the headset is already programmed to send signals and display them on the Emotiv GUI. A connection between the motherboard and the headset allows for extraction of this data which made it possible to manipulate it and use it however necessary.

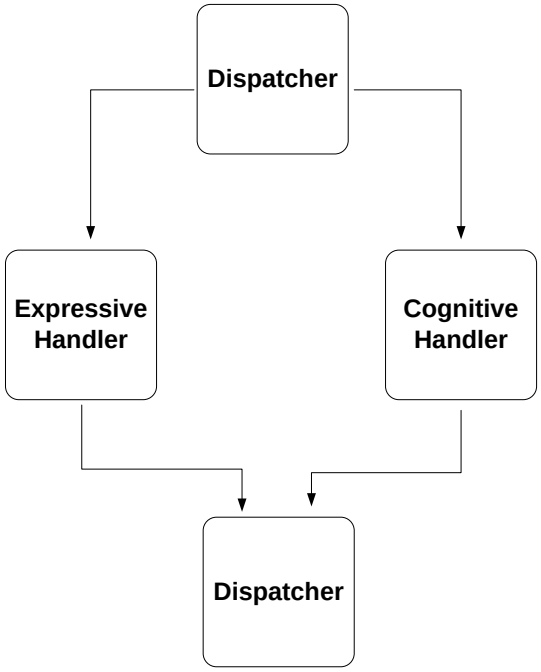
Once a connection is confirmed, the next step was to begin receiving signals from the headset. This needs to be a continuous process to make sure all signals are received until the user decides to disconnect the headset from the computer. In order to do this, the block of code that receives and handles all signals from the headset is be placed in a while loop that continues to run until the code breaks out of it.

During each iteration of the loop, the code receives a signal from the headset. If the signal represents a valid state, then a new event is made which is then used to determine whether the received signal represents a new command. According to the Emotiv User's Manual, "For near real-time responsiveness, most applications should poll for new EmoStates at least 10-15 times per second." (User's Manual). This meant it was important to make sure the loop ran enough times to record real-time readings from the headset.

Once a signal was determined to be a valid signal, the next step was to verify that the new signal received was a command. This was essential because commands are not the only pieces of information that can be sent in the EmoStates. If a new user is added to the Control Panel or if a new profile is logged into the Control Panel, this also triggers a new event. It was due to this that we added this additional step to the process of receiving each signal.

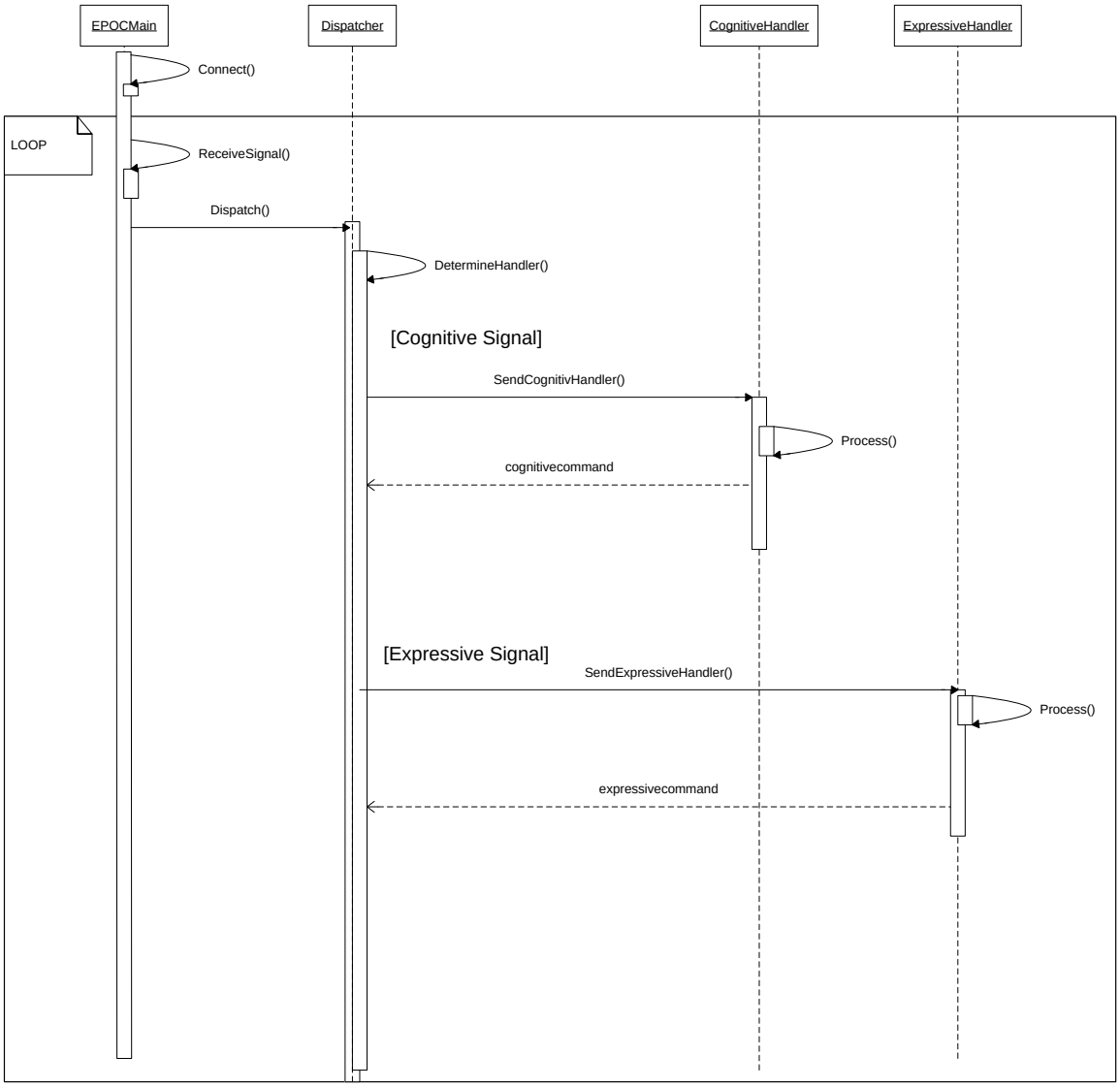
For this project, signals are taken from both the Cognitiv Suite and the Expressiv Suite. The Cognitiv Suite receives the EEG waves sent by the headset and applies them to commands such as "pull" and "rotate" which can lead to physical movements. The Expressiv Suite uses facial expressions to send commands. Both of these suites together are used to control the RC car. The signals are then filtered to each handler. Each handler knows how to process the signal and extracts the necessary information from it. Figure 9.2 below shows the data flow for this segment:

Figure 9.1.2: Data Flow for Signal Filtering to Handlers



When the process returns to the Dispatcher from each handler, the processed command is returned to the Dispatcher as a string. A Converter class and XbeeSerial class are the last classes in this code, which are used to convert the results from the handlers to a 6 character signal and finally send the signal to the PCB on the car. The overall sequence for this software is shown below and a breakdown of the different possible sequences is included in the software design content:

Figure 9.1.3: Sequence Diagram of Developed Emotiv Software



In order to send signals between the motherboard and the microcontroller, it is necessary to communicate between them wirelessly. Once this process is complete, the software then waits for the next signal to be received from the headset. Once a new event is received, the process restarts until the user disconnects the headset.

9.2 Design Summary of Software

Part of this project was to communicate between the motherboard and the car. The motherboard processes the information received from the Emotiv EPOC EEG neuro - headset and translates this information into a stream of signal pair character array binary sequence using the Emotiv software and C++ classes. This signal can be recognized by the vehicle to coordinate its movement control. As mentioned in earlier only the Atmega328 microprocessor of the Arduino Deumilanove microcontroller PCB development board is only used because our project only requires the use of the micro processing power and functions of the AtMega328 microprocessor. With the Atmega328 microprocessor on the PCB board attached to the car to communication between the motherboard processing and interpreting the signals into characters with the Emotiv software and then sending these signals to the AtMega328 microprocessor connected to peripheral devices attached, thus allowing control of the motors of the car. Essentially this is creating software for the Arduino for communicating with an external real world peripheral device. The goal was to create an Arduino language sketch code program to read in a continuous signal pair of binary character sequence using digital Receiving (RX) pins 0 and output each individual character read into specific digital pins 14-17 used. Having the microprocessor preprogrammed with the Deumilanove bootlader Arduino software we easily program, write, upload and compile/execute our microprocessor with these instructions to follow using the Arduino Software Environment.

As explained earlier in the section the Brodmann areas outlines the different areas of the brain and their specific functions the Emotiv EPOC EEG neuro – headset which reads the voltage differences in each of these areas on the scalp of the head using the 14 FELT electrode EEG sensors which are mapped on each of the Brodmann areas of the scalp of the head. To prevent eradict behavior and movement of the car one of the group members heavily conditioned himself to isolate ones thoughts and expressions in order to initiate an electrical impulse in distinct areas of the brain at a time to provide, thus eliminating any potential overlapping in thoughts and expressions which exists since – various electrical impulses initiates an unlimited amount of brain activities in different areas of the brain. By limiting only to Cognitive thoughts and Facial

Expressions we were able to activate and send electrical impulses to the Primary Motor Cortex, Primary Somatosensory Cortex and the Somatosensory Association Cortex Brodmann areas that are respectively used by the Cognitive and Facial expressions. The Emotiv EPOC EEG reading headset capabilities of transmitting these Radio Frequency (RF) signals from the user and extract information such as cognitive thoughts and facial expressions and converting them into digital form to be processed is a major part of this project that must be absolutely achieved. This neuro-sensing apparatus coupled with its FELT electrode EEG sensors which to be properly placed on the users head as detailed in the Headset section of the Hardware Overview, processes the brainwaves results once converted to digital form via wireless USB receivers connections to the motherboard. On the motherboard is the Emotiv Application Programming Interface (API) post processing Emotiv EmoEngine that sends the brainwave results to the API functions in each of their respective suites. This process is shown in the figure below.

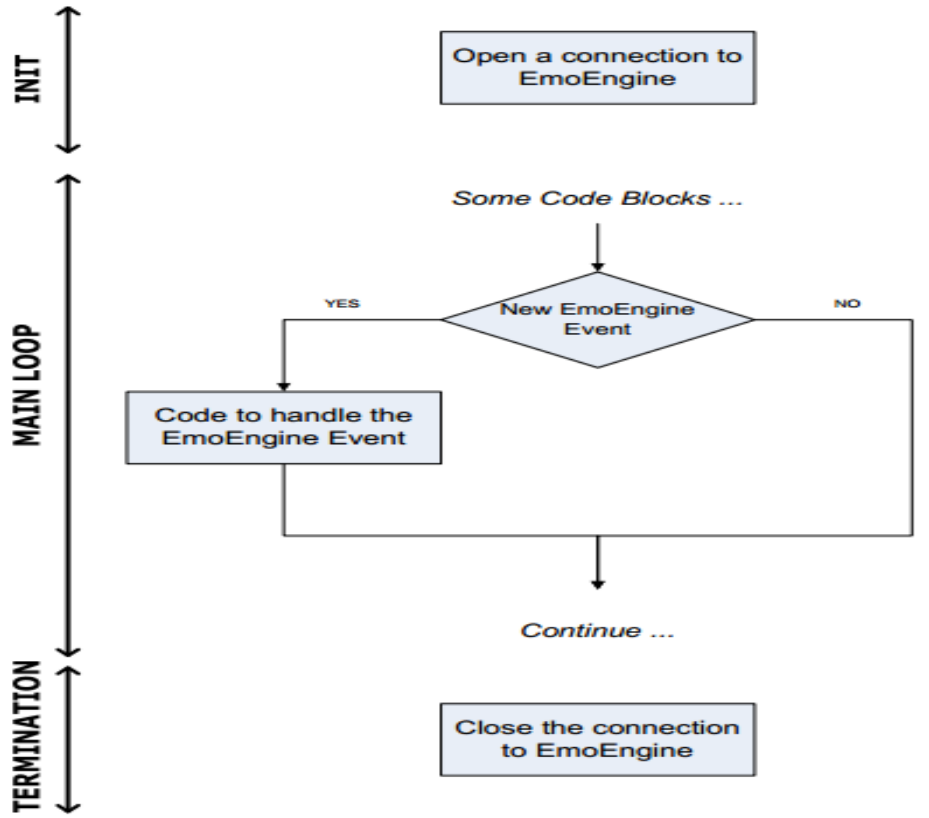


Figure 21 ***Using the API to communicate with the EmoEngine***

Figure 9.2.1 Flow diagram of Emotiv's API

They were a number of events to be expected and handled as outlined. The Emotiv API is configured by each event being grouped to their respective suites. The Cognitiv detection group which houses the methods and functions to handle the cognitive type events from the headset are grouped within the EMOTIV API's Cognitiv Suite. This is similar to how the Expressiv detection group holds the functions to handle all facial expressive type events received which are in EMOTIV API's Expressiv Suite. Thus the Cognitive detection group handles the events such as commands to think push or pull for moving the car forward and backward respectively or neutral to not move/stop car at all. Similarly, The Expressive detection group handles commands relevant to facial expressions such as winking right and winking left to shift and turn the wheels of the car right or left respectively. Blinking once again is used to straighten the wheels.

The API consists of Emotiv data structures such as the EmoState which contains the current state of the Emotiv Detections. The EmoState is the result of communication between the Emotiv EEG (electroencephalography) Neuroheadset and the EmoEngine which receives the preprocessed EEG and gyroscope data, and then performs post processing – translating the Emotiv detection results. The information contained inside the EmoState can be used for in both the Emotiv detection suites (e.g. the Expressiv Suite and Cognitive Suite). The users' facial expressions are monitored through Expressiv Suite as the Cognitiv Suite measures and interprets conscious thoughts all in real time.

In order to establish connection between the EmoEngine – EE_EngineConnect or EE_EngineRemoteConnect is called depending on the headset used and the connection is closed by calling EE_EngineDisconnnect. EE_EngineDisconnect must be called before the end of the running application. The EmoState are triggered through events such as EmoStateHandle and EmoEngineEventHandle. EE_EmoEngineEventCreate allocates the corresponding Emotiv API functions and EmoEngineEventFree frees and dealocates memory from an allocated and newly created event. EE_EmoEngineEventGetEmoState and EE_EmoStateUpdated retrieve the changes in states of users. EE_UserAdded monitors input devices and EE_CognitivEvent is for the Cognitiv Suite. EE_EngineGetNextEvent() retrieves events that are called by the EmoEngine in order to communicate with the running application.

Since we only used a combination of Cognitive thought and Facial expressions we only used their respective two suites Cognitiv suite and Expressiv suite in the Emotiv API software.

In the Expressiv Suite the following Emotiv API functions will be used to acquire the task of reading the users' facial expression. ES_ExpressivGetUpperFaceAction(eState) reads upper facial action , ES_ExpressivGetLowerFaceAction(eState) reads lower face action. The ES_ExpressivGetUpperFaceActionPower(eState) and the ES_ExpressivGetLowerFaceActionPower(eState) functions measures the respective strengths of each facial expression. We will also record eyelid movement with ES_ExpressivIsBlink, ES_ExpressivIsLeftWink, ES_ExpressivIsLookingRight and etc. These Emotiv API functions returns an integer value or EDK_OK for a successful API function or Error codes. A class diagram is shown in the figure below to show each of the Expressive suits functions relations with to the EmoState that generated this suite.

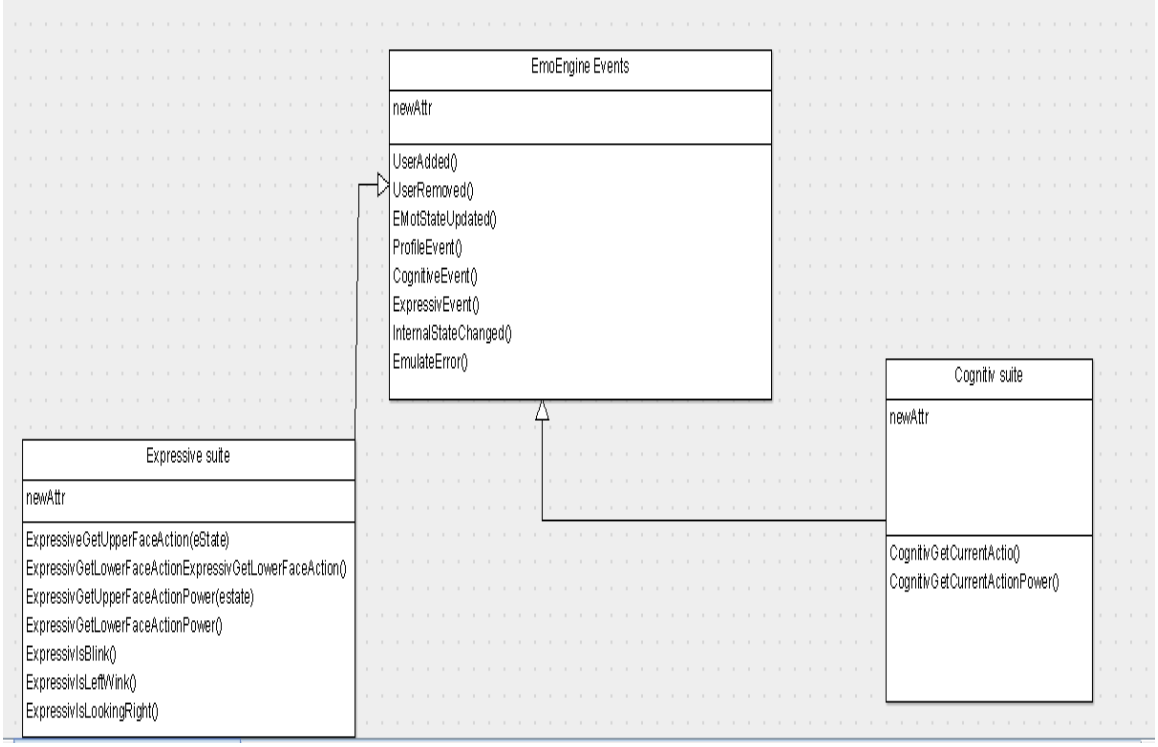


Figure 9.2.2 API functions associated with their respective suites.

Expressiv™ action type	Corresponding ASCII Text (case sensitive)	Amplitude value
Blink	B	n/a
Wink left	l	n/a
Wink right	r	n/a
Look left	L	n/a
Look right	R	n/a
Eyebrow	b	0 to 100 integer
Smile	S	0 to 100 integer
Clench	G	0 to 100 integer

Figure 9.2.2 Syntax for expressions in Emotive language.(Used with permission from Emotiv)

The Cognitiv Suite uses API functions such as ES_CognitivGetCurrentAction, and ES_CognitivGetCurrentActionPower similar to Expressiv Suite commands but instead monitors conscious mental thoughts. The figure above shows the events according to each of their respective detection groups. In the figure below are a more detailed view relationships with each API function and each of their respective suites.

The Suite's to be used are the Expressiv suite to accept and handle electrical impulse response signal events stimulated by cognitive thoughts detections/ or electrical impulses which stimulate brain activity in the Primary Somatosensory area which is activated by facial expressions. To stimulate electrical impulses in this region facial expressions such as Winking Right and Winking left are associated with lateral movements. Winking Right results in the car wheels shifting to the right enabling the car to steer and turn right. Winking left results in the car wheels shifting to the left enabling the car to steer and turn left. Blinking once again will straighten the wheels of the car. Also, the Cognitiv Suite will handle cognitive thoughts to control the motion of the car's backward and forward movements. Electrical impulses from this area stimulated with cognitive thoughts such as imagining pushing an object results in the car's wheels to go forward resulting in the car steering in the forward direction. This is just as imagining pulling an object will results the car wheels moving in the backward reverse direction

thus moving the car backward. For no neutral activity or registered neutral activity the car stops.

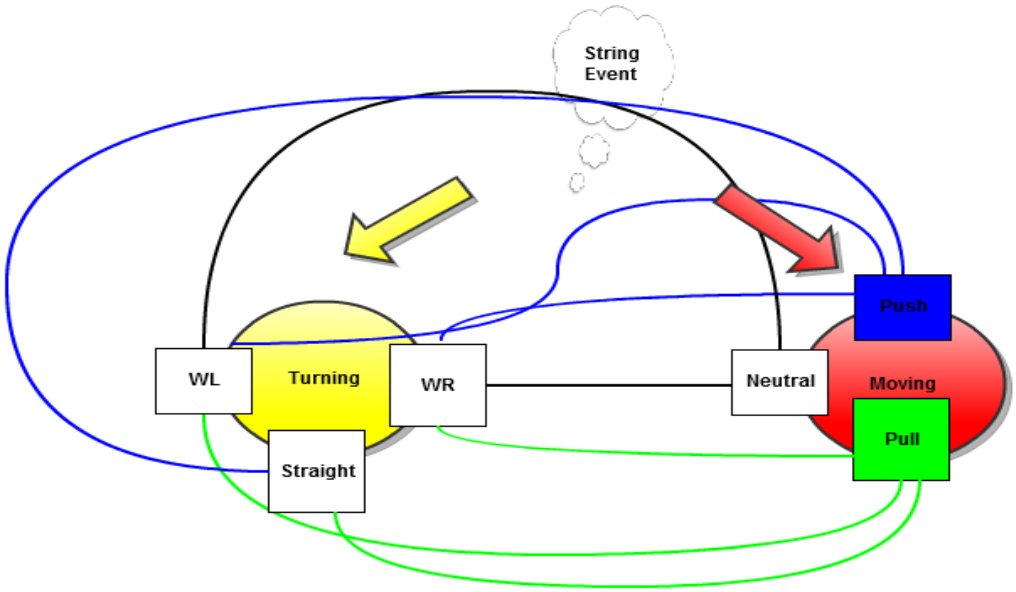


Figure 9.2.3 Overall Interaction based on pairing to determine the final Commands

The resulting car movement associated with an individual command that is analogous to the Cognitve or Facial Expression event which stimulated the electrical impulse for that distinct brain region. This was achieved by using the Emokey to map and translate the car to the each of their corresponding commands.

The state diagram below shows the flow of activities depending on the event and actions to be taken for this event, which also depends on the processed data of the headset by the EmoEngine.

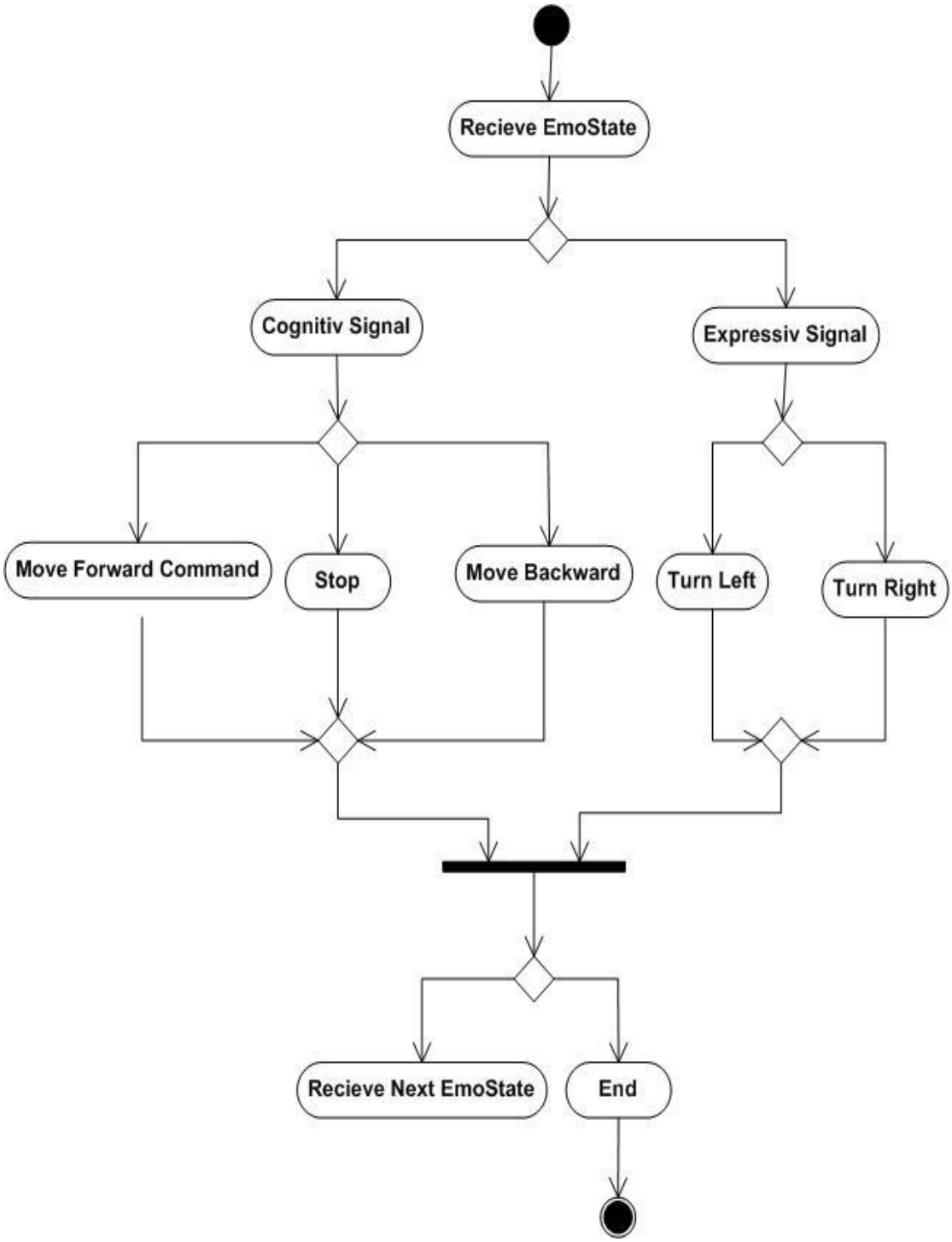


Figure 9.2.4 Activity Diagram of the event driven information to be extracted from the EmoState

It is during these steps that the EmoStates are evoked. The EEG signals after being post processed by its corresponding Emotiv API functions that are built in, uses these methods and the information extracted from these methods such as the EPOCMain, Dispatcher, CognitiveHandler , ExpressiveHandler , Converter and the XbeeSerial classes to create a signal of a binary sequence of a combination six 0's and 1's that are stored in the form of a character array of size six .

As mentioned an Xbee Series 1Transmitter/Receiver module pair was used for wirelessing communicating the signal character array between the software on the motherboard and the PCB on the car so that the car can be manipulated wirelessly.

The Dispatcher class of the Emotiv software makes a call to the output_BinarySignal() method Converter class which in turn makes a call to its internal getBinarySignal() method. These methods in the converter class uses the extracted information from the EmoState. The information contains a string value . These string values are either "Neutral", "Push", "Pull", "WR" for Wink Right, and "WL" for Wink Left. These strings commands are analogous with the Cognitive thought or facial expression that triggered the Electrical Impulse Response read from the headset. These string values are converted and placed into an character array of size six bin_command which is returned to the calling Dispatcher class. Then the Dispatcher class calls the XbeeSerial class sending the character array to the XBee transmitter connected to the USB COM 7 port of the motherboard.

Detection Group	Electrical Impulse Response read	Triggered Command	String Event	Character value assigned	Result
Cognitiv: **thought**	Neutral	Stop	Neutral	“100”	Car stops: Stops the L298 driver motor of the dual H-bridge device
	Push	Forward	Push	“100”	Moves car forward: Shifts wheels straight, and turns on drive motor
	Pull	Backward	Pull	“110”	Moves car backward: Shifts wheels straight, and turns on drive motor in reverse
Expressiv: **Facial express**	Wink Left	Right	WR	“101”	Turns car to the Right: Wheels are shifted to the Right
	Wink Right	Left	WL	“110”	Turns car left: Wheels is shifted to left,

Table 9.2.1 list of events and corresponding commands

The Neutral, Push and Pull events initialize the indexes 0-2 of the character array bin_command[6]. The WR and WL commands character values are assigned to the indexes 3-5 of bin_command[6].

The XbeeSerial class uses the Serial class library. This library is an open sourced library which contains implementations to directly communicate with an Xbee module. The Serial class library sets the baud rate for communicating with the device and opens the ports for connecting the Xbee devices.

XbeeSerial class will receive the data to transmit to and communicate directly with the Xbee transmitter connected to the USB port. XbeeSerial creates an object of Serial class so to use its IsConnected() and WriteData() the methods invoked. The IsConnected( ) method checks and see if the port with the Xbee is available for communication. The WriteData( ) method uses takes in the character array and it's length to be written and received into the Xbee Reiciever.

The Arduino Software on the Atmega328 chip will receive the data sent by the XbeeSerial class with the Xbee receiver DOUT line connected to pins 0 of the AtMega 328. The read( ) method of the SoftwareSerial library is used by using the xbeeSerial object which is an instantiation of the SoftwareSerial class. Read into the Arduino AtMega328 microprocessor is a character at a time for each of command read each time a brain activity occurred by Electrical Impulses Responses that is triggered by Cognitive thoughts or Facial expressions.

The Xbee Reciever is directly wired to the AtMega328 Receiving(RX) pin 0 line. The SoftwareSerial library for serial communication on the digital pins used in the project i.e. digital pins Receiving (RX) pins 0. It reads the data 1-byte or 1- character at a time read through inputs Receiving (RX) line – pin 0 of the AtMega328 processor.

To provide control to the motors of the car the L298 motor driver was used to make our car move in forward, backward, turn right and left. The L298 is a transistor dual H-bridge circuit. The L298 motor driver chip is used and is located on the PCB on the car with the AtMega328 microprocessor. Since the Arduino AtMega328 interacts with external peripheral devices like the L298 motor driver they are directly connected with wires. Digital pins 14 – 19 are tied to Enable drive motor Input, Inputs 1 – 4 and Enable turn motor Input. The AtMega328 acts as a microcontroller to control the motors of the car with the PCB on it.

Since each individual character is needed to be outputted to pins 14 -19 specifically and using the xbeeSerial object of the SoftwareSerial library to read one character at a time sequentially storing each individual character read into character variables c1 through c6 until the six characters are read each time a command is to be interpreted once a

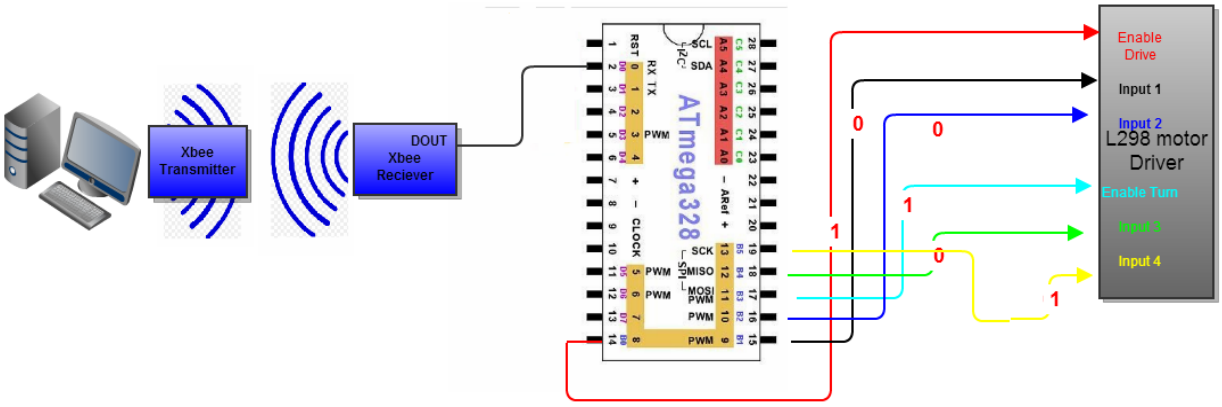
brain activity has occurred by Electrical Impulses Response. Using Pin mode setting PortB as high/low sets Digital pins 14-19 which are mapped to Port B Registers as output for writing data. Hex values equivalent to the six characters read is written to PortB which subsequently writes the equivalent binary to pins 14-19.

The example code below illustrates a value being written to output pins using manipulation of port B register and setting digital pins 14,17 and 19 to high.

```
// neutral
if(c1 == '1' && c2 == '0' && c3 == '0')
{
  // forward
  if(c4 == '0' && c5 == '0' && c6 == '0')
  {
    PORTB = 0x01;
  }
```

Figure 9.2.5 Code Snippet of Arduino

Depending on the outputs of pins 14-19 Having the outputs of the Arduino AtMega328 microprocessor as Enable Drive motor/turn Inputs and Inputs 1 -4 of the L298 Motor Driver chip device will enable the different inputs of the L298 driver motor will vary the result based on the interpreted signal sent from the Emotiv software which is what was read as inputs into the AtMega328 microprocessor. For example setting pins 14, 17 and 19 as high meant that a Neutral Cognitive and WR Expressive command was interpreted by the headset. Next it was processed and processed using the Emotiv API and C++ software on the motherboard. Then the information was extracted to translate these signals into a character signal which in this case is 100 101 and is sent to be read a character at a time via receiving (RX) pins of the AtMega328 chip on the PCB. This example is illustrated in the figure below. The following figure 9. 1.6 shows an example of Turning left procedure



The output of the AtMega328 will result in the car turning right based on the pins 14, 17 and 19 being enabled. A table is shown below to illustrate the various cased that could also occur.

cogCommand	expCommand	FinalCommand	PORTB Ass. Hex	Result
“Neutral”	“WR” “WL” “Straight”	“100 101” “100 110” “ 100 000”	0x29 0x19 0x011	Turn right Turn left Move Forward
“Push”	“WR” “WL” “Straight”	“101 101” “101 110” “101 000”	0x2D 0x1D 0x05	Move forward turn Right Move forward turn left Move forward
“Pull”	“WR” “WL” “Straight”	“110 101” “110 110” “110 000”	0x2B 0x1B 0x03	Move backward turn Right Move backward turn left Move backward

Table 9.2.2 Showing the forming of the Final Command from the EmoState

As the difference are shown by specific character array of binary sequences of 0's and 1's f being scommunicated between the EmotivSoftware and the Arduino Atmel AtMega328 microprocessor via XBee Transmitter connected directly to the motherboard computer and the XBee Reciever DOUT line connected to the Receiving (RX) pin 0 of the AtMega328 microprocessor and using digital pin 14-17 to output the resulting signal to the inputs of the L298 Motor Driver . These connections and an alternative example is shown below for where sequence 101 101 is read for a turning right command.

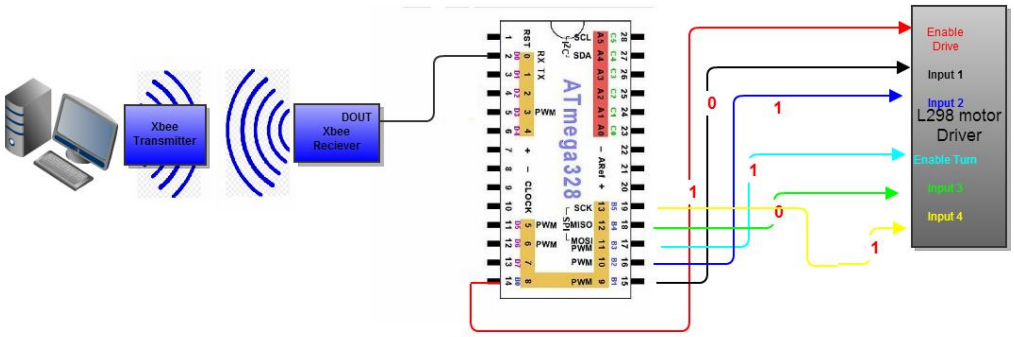


Figure 9.2.2 Example of Turning left procedure

This example illustrates Inputs 16, 19 and the Drive and Turn motors both being high and enabled thus moving the driver motor forward and shifting the car wheels to the right for the turn right command.

10 Design Summary of Hardware

10.1: Design Summary of Hardware

10.1.1: RC Car Hardware Design Overview

One of the most important aspects of this project was the electrical design of the car. It was first necessary to identify the basic structure and how every component will need to work together to perform the given task. The information starts with the headset and that will be sent to the processor on the motherboard via a Bluetooth dongle usb. The motherboard was housed on top of a small glass platform so to provide adequate cooling and space for all the wires and components that needed to be connected. The headset range is only 5 feet and needs to be within eyes view in order to work properly. The motherboard then transmits the signals to the Atmega p-pu via Xbee transmitter/receiver. The Atmega microcontroller then translates the given information. The custom pcb houses the Atmega p-pu microcontroller and the L298N H-Bridge that then tells the DC drive motor to drive forward or backwards and the DC turning motor to turn left or right.

The motherboard accepts input from the usb transmitter, which also received information from the EMOTIV headset. That input is then interpreted on the processor and sent to the Atmega p-pu microcontroller, via Xbee transmitter/receiver, to be used to control the various functions of the vehicle. Once the Atmega interprets these values and converts them into a six bit number. From there, the microcontroller sends the values out to the L298N H-Bridge on the custom made pcb located on the vehicle. The microcontroller will output a six bit value which will allow the DC drive motor to drive forward, backwards, or stop. It was also be used to control the car by turning left or right by utilizing the turning motor. Specifically, we used pins 14 to 19 on the Atmega p-pu microcontroller. The first three pins, 14-16, control the drive motor. When pin 14 is a 1, that means that the drive motor is on. Then depending on what the next two values are, determine whether the motor spun clockwise or counterclockwise. If the values of pins 15 and 16 were the same, 00 or 11, then the motor would be off. When we wanted the vehicle to turn at the same time while driving, we would active pin 14 and pin 17. This would ensure that the wheels didn't just turn while the car was stopped. Depending on what pins 18 and 19 were, would determine if the vehicles tires turned to the left or the right. Just as with the dive motor, if pins 18 and 19 were the same values, 00 or 11, the wheels would not turn at all. The following block diagram depicts the basic process of the vehicle.

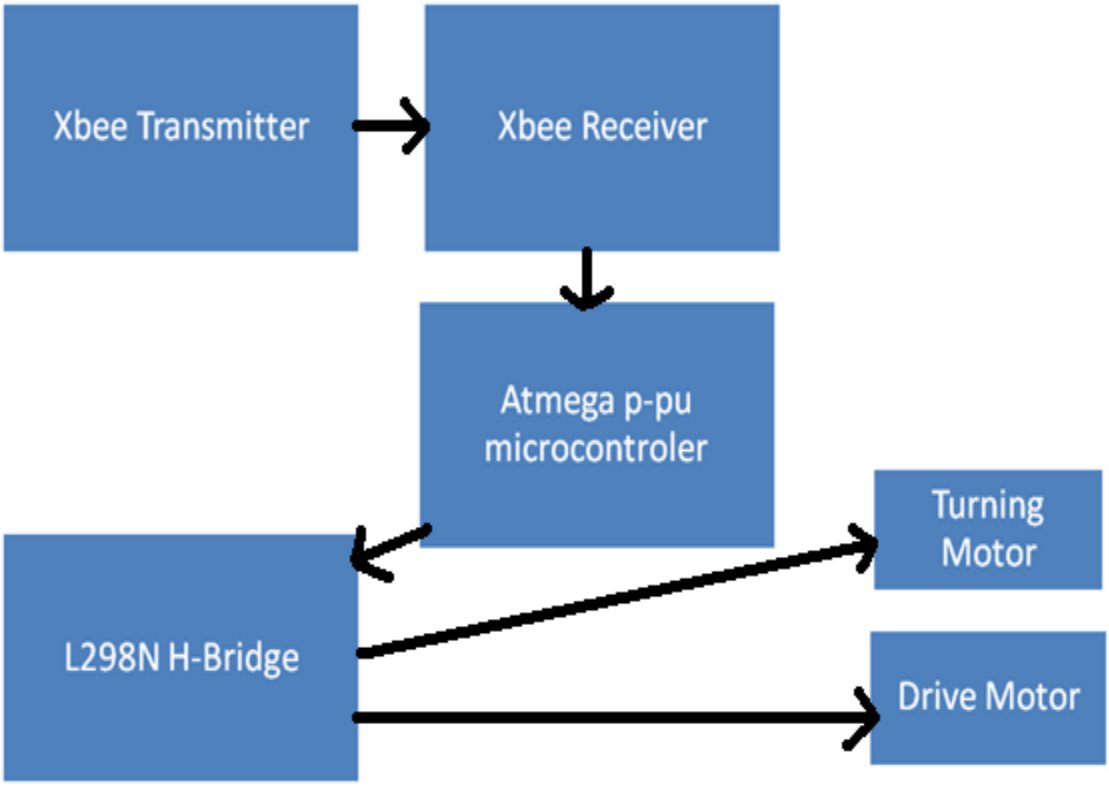


Figure 10.1.1 Vehicle Electrical Overview

Specifically, coming from the batteries, there are three voltage regulators with each one connected to a separate component. The first voltage regulator will simply be connected to the Atmega p-pu microcontroller and will step down the voltage to a value that can be used. The second voltage regulator will go from the batteries to the DC drive motor and will need to vary in voltage depending on a specific input given from the processor. The third and final voltage regulator will control the Xbee transmitter/receiver power. The block diagram in figure 10.1.2 offers a more detailed and clear view.

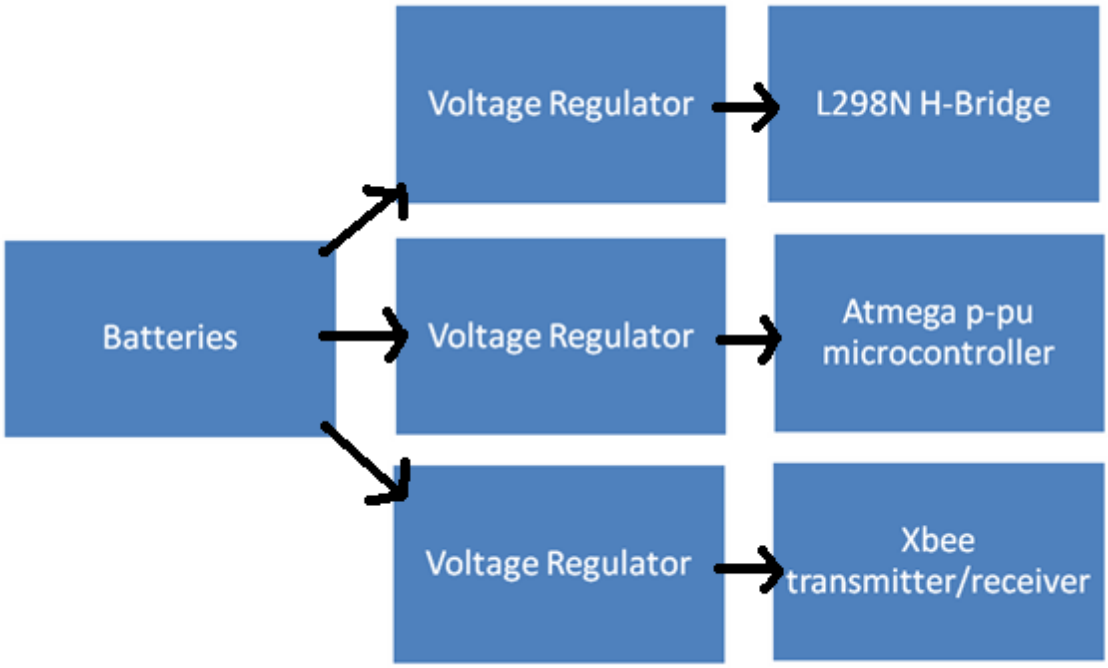


Figure 10.1.2

10.1.2: RC Car Processor Overview

The basic structure of the Processor is to collect data from the headset, through the transceiver. Following this collection of data, it is important to analyze the data and determine whether the information is significant. If the information received is significant and causes a change, the processor then sends that new signal to each of the affected components. For instance, if an object is detected in the specified range of the sensor, that signal will then be sent to the processor.

The processor will then use that received information and send thus another signal to the serial to parallel converter. That then sends out an eight bit signal of 00000000, which tells the DC drive motor to stop running. The processor will also receive the input from the transceiver which receives input from the headset. That information will then be sent to a program in the processor to be decoded. Depending on the given input, the processor will either send a signal to the DC servo motor to turn left or right a certain amount of degrees. Or, the DC drive motor will simply go forward or stop altogether.

The last step that the processor will do is to update and clear the necessary information to make way for the new incoming information. The following figure 10.1.4 offers a basic flowchart for a more clear understanding of how the processor works from a hardware point of view.

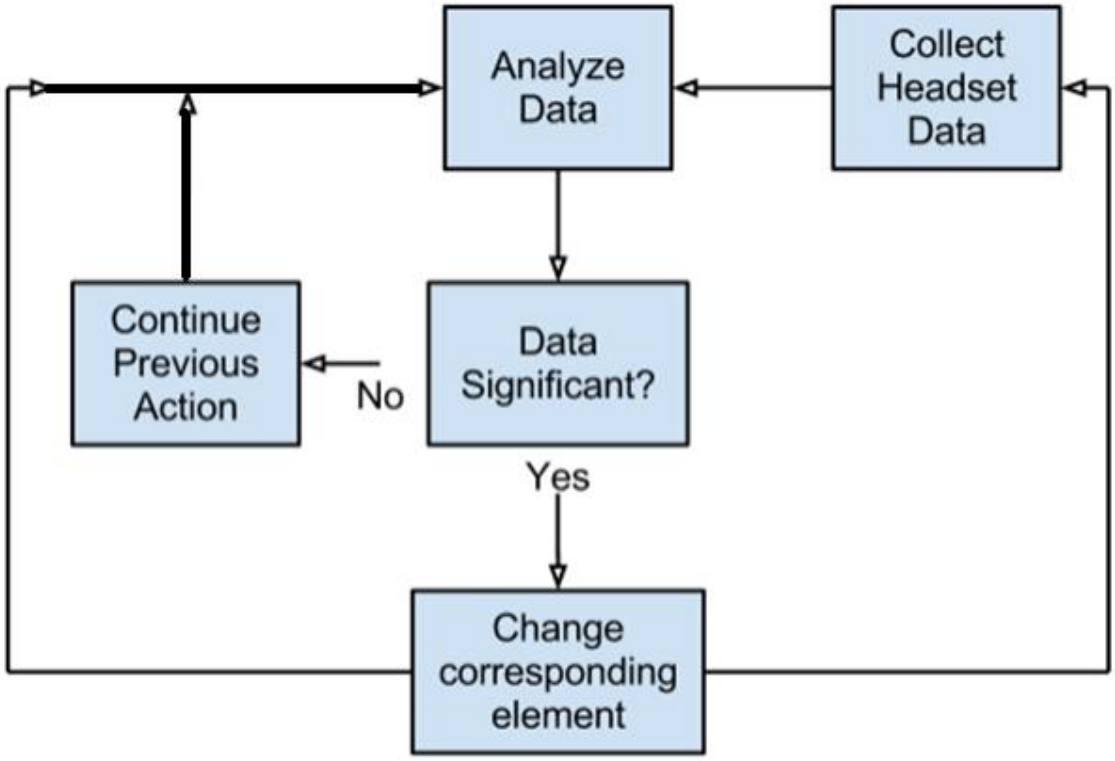


Figure 10.1.4
Processor Hardware Overview

In order to begin, there first was the need to have a location in the memory where all of the data received is stored. An array, each of length three, will be used to store the data from the DC drive motor and the DC turning motor. In addition to this, there was the need to have two more arrays to hold the information for the control of the vehicle. Specifically, the arrays will hold two separate elements being the new information and the old information.

Data communication between the EMOTIV headset and the remote controlled car will take place through the USB dongle receiver and the processor located on the motherboard. The given USB received the headset data and then transmitted that data in a serial fashion to the processor. From there, the remote controlled car transmitted 3 bytes for a voltage reading from across the DC drive motor and another 3 bytes corresponding to the voltage reading for the battery power of the power source operating the controls of the remote controlled car. Three bytes is more than sufficient enough for each individual voltage reading because then there can be 512 different unique values represented.

In addition to these memory locations, there was the need to have a set of functions that are called along the way in order to break this seemingly daunting task into smaller, more approachable tasks. A function will be created to read the information received from the headset. It will also compare the new data that is collected with the previous data. This function will then return an answer as to whether the data is different enough to change a movement on the car.

After each of the individual component readings had been taken and the new information had been stored in each of the individual memory locations. Then it was necessary to analyze this given data. This analysis took these new values and compared them with the old averages to see if there was a difference. All of the new averages were to be used to go to the output buffer. In the case that the values are not different enough from the previous values, there was to be no change to the vehicle and the previous task would continue.

10.1.3: RC Car RF Interference Overview

There are many environmental issues in RF communication which include many uncontrollable concerns. Among these are the weather such as rain, snow, heat, signal obstruction, etc. These should not be of concern for the aspect of this project though. There is no snow in Florida, so this would not ever be a problem. The remote controlled vehicle would never be driven in the rain and heat should not be of any concern so long as the car is properly insulated. Given all of these precautions, it is impossible to completely eliminate noise or avoid complete obstruction of the signal between the transceiver and the headset and the vehicle. This being said, it is important to use the headset and the vehicle within the specified range given with the EMOTIV headset in order to reduce as much interference as possible. To design our own transceiver would be impractical, it would cost much more than the budget allows and would most likely sacrifice efficiency, range, and size. Therefore, the USB receiver that came with the headset will be used.

Synchronization must take place to ensure that communication between each of the devices is effective. Each component must be awaiting the moment when it receives data, to submit said data, and know exactly what to do with all of the data. All of this will be accounted for under the integration process with its respective programming platform. Synchronization cannot take place when there are significant radio frequency interferences, a lack of battery power, or physically blocked transmissions. Thus a connection would not be able to be established and the corresponding control operations could not be executed.

One of the problems that we ran into was the fact that the Xbee transmitter/receivers operated on 2.4 GHZ. This just so happened to be the exact same frequency that the headset operated on. This caused some interference between the two respective parts, and in some cases it caused the program that was running on the motherboard to lag. This was especially frustrating when the EMOTIV software would freeze up in the middle of a demonstration. The way that we would fix this in the future would be to change the frequency that all of the parts operate on so that there is no cross interference.

10.1.4: RC Car Structural Modification Overview

Since the remote controlled car was being purchased instead of being made from scratch, some basic structural modifications will need to take place in order for each of the necessary components to be properly and safely fastened. In addition to this, in order to properly mount the PCB, the Xbee transmitter/receiver, and the batteries, the entire plastic body of the vehicle will need to be detached. Since the vehicle was to be purchased already manufactured, the positions of the DC drive motor and the DC servo motor should were not to be changed in any way. The actual DC drive motor and servo motor parts and connecting wires were however not to be changed either. The PCB for the DC drive and servo motors was mounted a few inches away from the actual motors. This way it will prevent as much damped feedback as possible. The microcontroller was to be placed as far as possible from the DC drive and servo motors in order to minimize interference. It was also to be placed close to the edge of the vehicle so that the attached Xbee transmitter/receiver would have an adequate amount of room and would not interfere with any of the other components. Another necessary structural modification was to create some more space for the added battery inside the vehicle. The current space was not large enough to accommodate both of the required batteries to run the vehicle and all the necessary added components.

10.1.4.1 Structural Modification Process

In order to properly secure the various components to the vehicle, proper precautions must be taken. Two large zip ties were to be used in order to fasten the PCB to the frame of the vehicle to ensure stability as well as having the option to take off the printed circuit board easily if necessary. In addition to this, the batteries used to power everything needed to be securely fastened so that the vehicle will always run properly and not have the possibility of the wires coming out if the vehicle hit a wall or some sort of object. The four double A batteries were to be stacked into the cavity that we made in the vehicle and sealed in place. The Xbee transmitter/receiver also needed to be securely mounted onto the vehicle by use of the Xbee explorer regulate mounting board. This was to be located in the bed of the truck that so not to interfere with any other components, as well as giving more to the allure of the vehicle moving around on its own without too many wires hanging out or showing. This will allow the circuit board to be out of the way of any of the parts that allow the vehicle to move such as the front and back wheels.

11. Project Testing

Inputs and Outputs are evaluated based upon the expectations of each API function's return statements and values and consistent arguments for inputs. Also to be made in the software testing activity is test performance, bugs reports and design. The test demonstrates a working and completed software by repeated test running with various test cases which include different input condition arguments depending upon the environment. The Software is demonstrated as working software because each user command corresponds to the desired response. This software functions as an extension of the user. The software provided the key to control a car that responds to the users' brain activity. The project conforms to specifications required of this project. The project testing was based upon the following principles of the overall interaction between the Hardware and the Software discussed.

11.1 Software Testing

In order to test every aspect of this software, it was important to include unit tests, integration testing and system testing. Before any tests were done with the physical headset, all the software was tested using the Emotiv EmoComposer. This is an emulator of the EmoEngine, which sends signals to the computer as if it were the actual headset. This allowed the software to be tested without any loss of signal or interference. Once the software was proven to work with the EmoComposer, then all the tests were repeated using the physical headset.

The process of testing this project was intended to accomplish the following tasks:

a) Verify that all methods work as stated

b) Confirm that all signals were being received correctly

c) Assure that there were no bugs or unhandled exceptions in the code

d) Make certain that signals were being processed correctly

e) Confirm that the processor was sending the signals correctly to the car

Although it is very important to ensure that the software works correctly, these tests did not prove that any individual can use the prototype. This is due to the fact that the Emotiv EPOC headset requires much training to master, and therefore it would not be possible to pick someone at random and have them test our project. However, it would

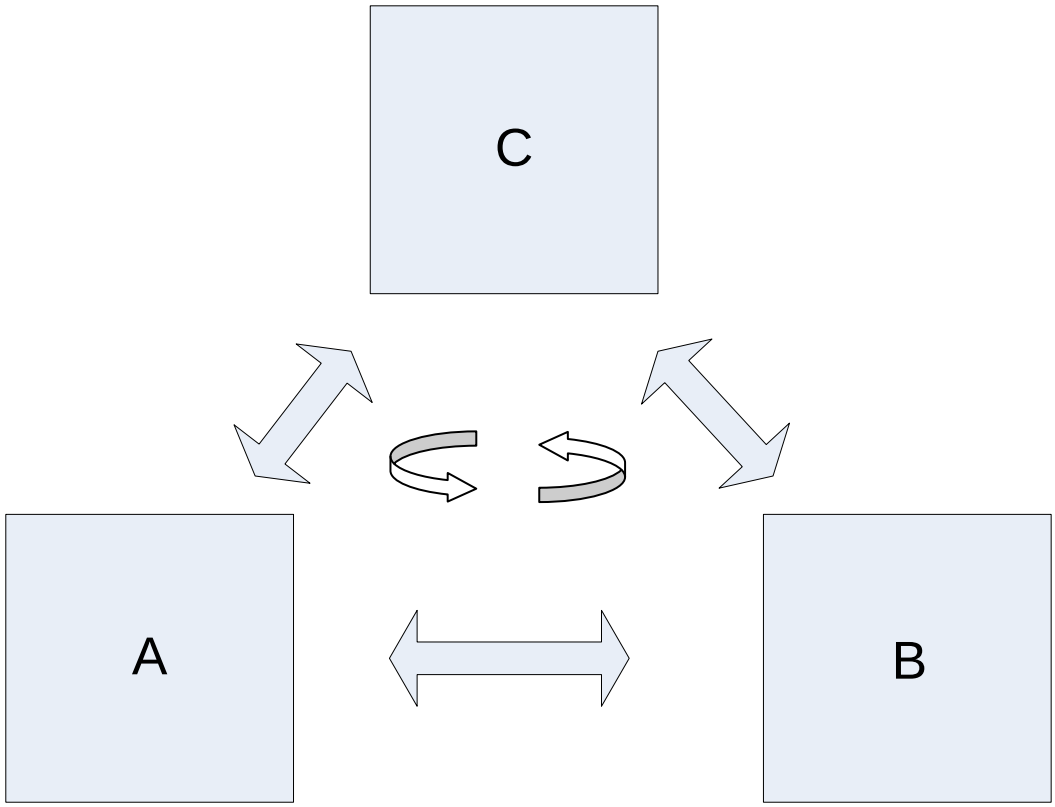
be possible to verify that the software works as written by using the EmoComposer because then the tests will not require the tester to use the headset during the testing process.

The first tests that were carried out were the unit tests. These were the earliest tests because they consisted of testing specific methods in the classes of this project. The unit tests consisted mostly of verifying that the results received by a certain method were equivalent to the expected results.

In order to implement integration testing, verified parameters were used in various classes to make sure that they are working together correctly. For example, verified signals from the Dispatcher were sent to the Converter class in order to make sure that the commands were being translated correctly.

The final test phase was the system testing. This was the most important series of tests in which the headset made a connection with the processor and signals travelled through the entire translation process, beginning with the headset and finishing with the processor sending the correct command. These tests verified that the project not only performs normally, but can also handle unusual and unexpected scenarios. The process of system testing is shown visually below in Figure 11.7:

Figure 11.7: System Testing



This figure shows that all modules were tested and unit tests were also completed. This is necessary because it minimizes the difficulty in determining any sources of error if all modules have already been proven to work correctly. Each arrow signifies the testing of more than one module and the circular arrows in the middle signify a complete system test, involving all the modules put together.

11.2 Overall Hardware and Software Interaction and Testing

Some overviews of the hardware software are knowledge of programming languages such as Arduino Programming Language and C++ in order to provide correct feedback controls. A 2.4 GHz Intel Pentium 4 processor (or equivalent), Microsoft Windows XP with Service Pack 2, Windows Vista or Windows 7, 1 GB of RAM, 50MB disk space available and One or Two unused USB 2.0 ports are some of the minimum System Requirements to run the Emotiv EPOC API software SDK. Also EDK.dll must be installed in order for an Application created by Emotiv EmoEngine and Microsoft Visual Studio 2010 (VC 8.0.CRT version 8.0.50727.762 or later) SP1. Please see user's manual from emotiv.com for details. With the DH61AG core i3 mini-ITX motherboard running at 2.5 GHz has 1 GB of RAM and 160 GB laptop hard drive added for storing the Software along with the extra USB COM ports for connecting external devices like the Emotiv Epoc Bluetooth dongle at 2.4 GHz to transmit the data from the headset to be processed by the Emotiv Software running on the motherboard and the XBee transmitter on USB COM port 7 for transmitting the character signal that was created by the Emotiv Software to XBee Receiver connected to the AtMega328 microprocessor on the PCB of the car.

The software produced was tested based on its capability to acquire the EEG data using the Emotiv EPOC EEG neuro – headsets reading of the voltage differences and brain activity that occurred by Electrical Impulses Responses in each of Brodmann areas on the scalp of the head using the 14 FELT electrode EEG sensors that is triggered by Cognitive thoughts or Facial expressions . Coordinating the car movement control which results from isolating specific brain regions by imaginative cognitive thoughts with thinking of pushing and pulling an object or neutral thought(not thinking- default) to move the car forward, backward or making the car stop respectively or alternatively using facial expressions like winking right or left, are extensively analogous to their respective Cognitiv or Facial suite detection group handlers of the Emotiv software API which is determined based on where their respective electrical impulse response command that stimulated the brain activity originated in that particular Brodmann brain area. Then having this information processed and translated into a stream of signal pair of binary character sequence to be sent each time an electrical impulse response command is read by the headset using the Emotiv software API methods such as the EPOCMain, Dispatcher, CognitiveHandler , ExpressiveHandler , Converter and the XbeeSerial classes.

The character is sent via XBee series 1 transmitter. The XBee Explorer Dongle USB adaptor interface board will mount with the XBee Series 1 Explorer Module which is

then connected to USB COM 7 port of the motherboard. This XBee Series 1 serves as the Transmitter part of the XBee series 1 module pair. Also, the XBee Explorer Regulated adaptor interface board will mount with the other XBee Series 1 Explorer module This Xbee Series 1 will act as the Receiver part of the XBee series 1 module pair. The DOUT line of this XBee series 1 module is wired to the Receiving (RX) pins 0 of the Arduino Atmel AtMega328 micro processing chip on our PCB board attached to the car. The Arduino Atmel AtMega328p-pu microprocessor with an operating voltage of 5V, 32 KB of Flash Memory and EEPROM of 2KB for storing the program, 14 digital I/O pins that are configured as either input or output. Microprocessor preloaded with the Arduino Duemilanove bootloader so that the microcontroller to be programmed with code from Arduino programming language and the sketches to be stored, loaded, verified and compiled from the microprocessor . The software on the Arduino must read in the data sent by the Emotiv software on the motherboard onto the Atmega328 microprocessor using digital Receiving (RX) pins 0 to read one character at a time sequentially storing each individual character read into character variables c1 through c6 until the six characters are read and outputting each individual character read. Then, writing the equivalent Hex to the six characters to PortB which subsequently outputs the equivalent binary to pins 14-19 tied to Enable drive motor Input, Inputs 1 – 4 and Enable turn motor Input of the L298 Motor Driver dual H-bridge device to control the motors of the car.

The ability of the L298N- dual H-Bridge Motor Driver device to control the drive and turn motors of the car with a 5V operating voltage located on the PCB on the car with the AtMega328 microprocessor Digital pins 14 – 19 tied to Enable drive motor Input, Inputs 1 – 4 and Enable turn motor Input of the L298 device. Thus, to control the motors of the car with the PCB on it and move the car forward, backward, turn right and left. Thus, the objective of controlling an RC car using only the mind is accomplished.

Before any tests were done with the physical headset, all the software was tested using the Emotiv EmoComposer. This is an emulator of the EmoEngine, which sends signals such as an EmoState to the computer and handled by the Emotiv Software API as if it were the actual headset sending the information. The information extracted from the EmoState is then used and sent to the software on the Arduino Atmel AtMega328 microprocessor to use accordingly. This allowed the software to be tested without any loss of signal or interference. The software is proven to work with the EmoComposer, and all the tests were repeated using the physical headset providing the same result

The process of testing this project is intended to accomplish the following tasks:

a) Verify that all methods work as stated

b) Confirm that all signals are being received correctly

c) Assure that there are no bugs or unhandled exceptions in the code

d) Make certain that signals are being processed correctly

e) Confirm that the processor is sending the signals correctly to the car

Requirement 1

Emotiv Epoc SW API EPOCMain, Dispatcher, CognitiveHandler , ExpressiveHandler , Converter and the XbeeSerial classes acquires the EEG data from the Emotiv EPOC EEG neuro headsets

Dending on Cognitiv or Facial suite detection group handlers determined based on where the respective electrical impulse response command that stimulated the brain activity originated in that particular Brodmann brain area. Then having this formation processed into an EmoState

Connections with the headset were some conflicts that occurred during setup.

Evaluation method is based on the EmoState interpreted

Requirement 2

Emotiv Epoc SW API and methods extracts information from EmoState into a character array of size six and send to Atmega328 microprocessor

Depends on the EmoState which is created using the methods from the Emotiv API such as the EPOCMain, Dispatcher, CognitiveHandler , ExpressiveHandler , Converter and the XbeeSerial classes

Conflicts in communicating the data across the XBee modules occured

Is evaluated by the input arguments i.e. (Neutral, Push, Pull, WR, and WL) output i.e. (100 000, 101 000 ,110 000, 101 101, 101 110)

Requirement 3

The Software on the Arduino Atmel AtMega328p-pu Microcontroller receives a character at a time as input

This is sent from the XBee Reciever connected, by the XbeeSerial procedure of the Emotiv Epoc software being called

This depends on the particular character sequence sent from the Emotiv software and read by the AtMega328 processor

Conflicting data types occurred since the Arduino AtMega328 only reds in a character at a time

The Evaluation Method used is a message to be outputted to the Serial Monitor of the Arduino development Environment saying a successful transmission message then the data has been sent and received successfully

Requirement 4

The Software on the Arduino Atmel AtMega328p-pu Microcontroller outputs to Digital pins 14-19 binary values

Depends on the unique HEX values assigned to each of the individual characters read

No conflicts

Evaluated based output and control of motors of the L298 motor driver

Requirement 5

XBee Series 1 Module and L298 dual H-Bridge Motor Driver transitor device chip configured to communicates the characters across the Xee Transmitter module on the motherboard and the XBee Reciever connected to the AtMega328 microprocessor on the PCB of the car using the XCT-U software from Digi and the L298 driver motor with its inputs connected to the AtMga328 microprocessor controls the drive and turn motors of the car

No Source

Depends on connection and communication between XBee Series 1 modules, the AtMega328 microprocessor and the L298 motor driver

Some conflicts arises in this area

This is evaluated by ensuring the car moves in the direction that was desired.

In order to isolate possible errors and to verify the correct output and response as desired is acquired individual tests cases are based off of each of the Emotiv API functions as well as the functions used in Arduino software and they are extensively tested with various test cases. This ensures the correct communication and data being communicated between each of the functions thus determining if the desired input and output is communicated. Test cases are described by who ran the test, what function specifically was tested. All these testing procedures were last updated, verified and tested everyday until April 26, 2013. Such informative can be obtained from the Emotiv API functions as each function has a signal to notify user if it's corresponding task has been achieved successfully with errors or EDK_OK signals type. This is shown in the table below.

Emotiv API function Library and Arduino Library function (expected functions to be used) and also **Function and what was tested**	**Specific Suit Detection or Particular Software**	**Output or Evaluation method**	**Who Tested**

(based on its ability to do)			
EmoEngine (Communcitation with Emotiv neuroheadset, translates Emotiv Detection into an EmoState)	Emotiv Software API	EDK_OK	Chris
EmoState(A datastructure containing info of current state of all activated Emotiv Detections)	Emotiv Software API	String values	Chris
EE_EngineConnect (Connect App to EmoEngine) **EE_EngineDisconnect (disconnects connect between App and EmoEngine)** **EmoStateHandle (Alocates appropriate Emotiv API)** **EE_EmoEngineEventCreate (Creates a buffer for EmoEventHandler)** **EmoEngineEventFree(frees an allocated memory for emostate)** **EE_EmoStateUpdated(gets updates users facial expression**	Emotiv Engine not suite specific	String values to confirm success of failure EDK_OK for a successful API function or Error codes	Chris

and records the changes) **EE_EngineGetNextEvent(asks the EMoEngine to get the current published EmoEngine event)**			
ES_ExpressivGetUpperFaceAction(eState)- **(reads upper facial action)** **ES_ExpressivGetLowerFaceAction(eState – (reads lower face action.)** **ES_ExpressivGetUpperFaceActionPower(eState)** **-(measures the intensity of each facial expression)** **ES_ExpressivGetLowerFaceActionPower(eState)** **ES_ExpressivIsLeftWink (records blink expression)** **ES_ExpressivIsLookingRight (records look right expression**	Expressiv suite	EDK_OK for a successful API function or Error codes	Chris

and action event)			
ES_CognitivGetCurrentAction(), and ES_CognitivGetCurrentActionPower() (monitors users conscious mental thoughts)	Emotiv API software	EDK_OK for a successful API function or Error codes	Chris
EpocMain(makes calls to the procedures below and pulls all the functions and methods of each class and the classes themselves together – running the whole program of the Emotiv EPoc Software) **The Dispatcher class (call to the output_BinarySignal() method Converter class- is returned a character array of the string values from the extracted information from the EmoState data structure)** **CognitiveHandler(determine whether the user is directing the car to move forward, backward or stop.)** **SendCognitiveHandler()(send the information from the current EmoState in order to extract the necessary information)**	Emotiv Software Cognitiv	Dependent upon particular method	Chris

ExpressiveHandler (used to determine if the user has winked or not) **SendExpressiveHandler()(sends the information from the current EmoState to extract the necessary information from the Expressive signal)**	Expressiv		
Converter class(makes a call getBinarySignal( ) method uses the extracted information from the EmoState : extracts information from EmoState into String values) **XbeeSerial class (sends the character array to the XBee transmitter connected to the USB COM 7 port of the motherboard)** **Xbee_SoftwareSerial_Reciever** **(Softwar on the Arduino Atmel AtMega328p-pu microprocessor**	Emotiv SoftwareSoftware Arduino Software	Character array of size 6 of a binary sequence of 0's and 1's dependent upon particular function	Lee

receives the character array sent from the XbeeSerial class of the Emotiv software and outputs binary values of the Hex values that are equivalent to the character read using SoftwareSerial library and PinMode i.e. read(), print(), and PORTB)			

Table 11.2 Emotive commands and corresponding suites and functions.

12.Operating Procedure

This section contains a step by step set of instructions to properly set up and operate the EPOC-alypse mind controlled car.

Step 1: Headset setup.

Step 1.1: Remove the Emotiv neuro-headset from the case as shown in figure 12.1.

Step 1.2: Remove the sensor box on the left and apply approximately 2 to 3 drops of saline solution to each of the 14 individual sensor pads, as demonstrated in figure 12.2.

Figure 12.1 Emotiv neuro-headset in case.

Figure 12.2 Applying saline solution to headset sensor pads

Figure 12.3 Attaching the sensor pads to the headset

Step 1.3 : After the saline solution has been applied to the sensor pads, remove each sensor pad from the case by gently griping and twisting the plastic base of the pad to the left until it unlocks from the base of the case. Then take the sensor pad and place it in one of the sensor holders on the headset, twist right until you can feel the sensor snap into place as shown in figure 12.3. Repeat this until all 14 sensors are in place.

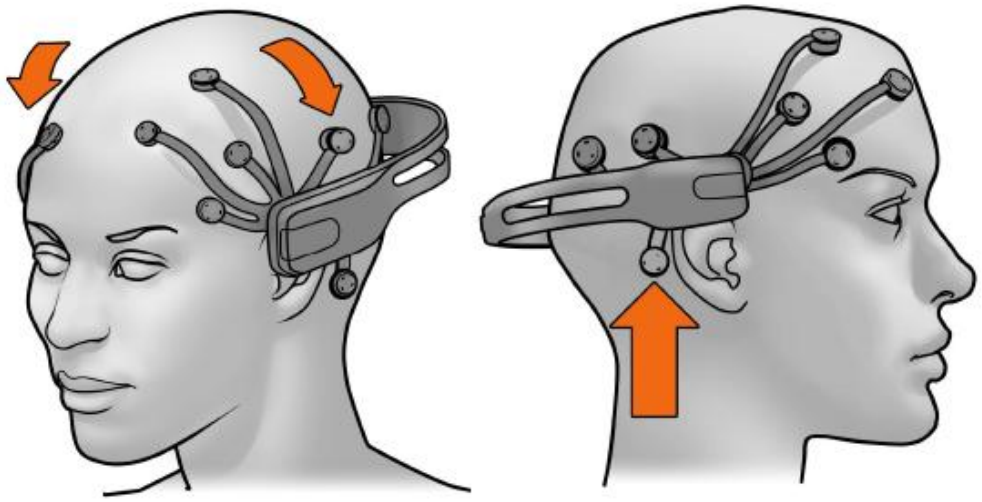


Figure 12.4 Shows the correct placement of the headset with respect to the rubber reference sensors.

Step 1.4: After all sensor pads are locked securely into the headset, take the headset and slowly slide it down over the head. Figure 12.4 shows the proper locations on the skull for each sensor on the headset. Make sure to check that the felt tip of each sensor is making a flush connection with the users skull (Try to remove as much hair out from under the sensor as possible to get better contact to the skin).

Step 2: Motherboard/Hardware setup

Now that the headset is properly set up, the motherboard and relating hardware must be set up.

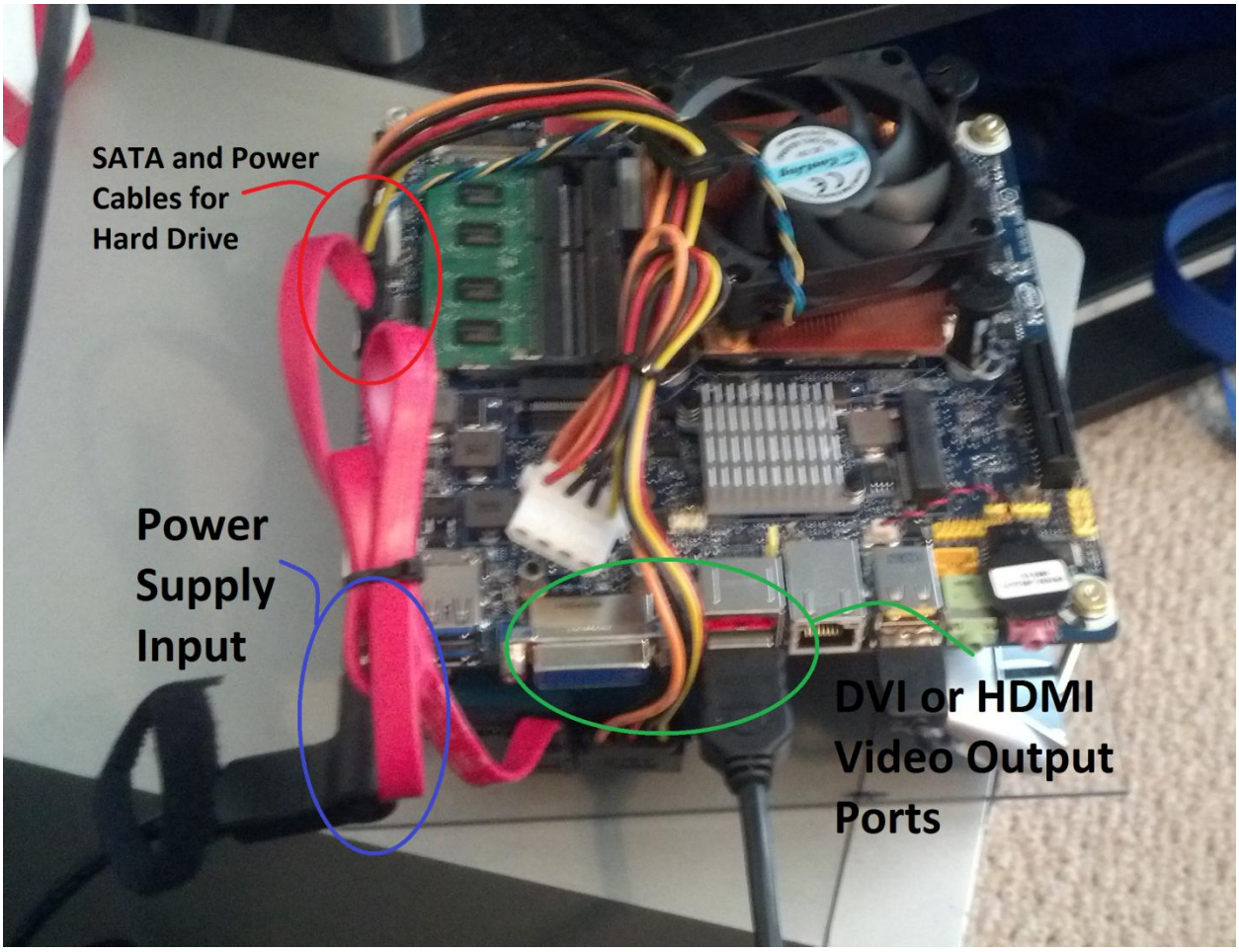


Figure 12.5 Intel DH61AG board with hard drive and HDMI video output connected.

Step 2.1: Place the Intel DH61AG mini-ITX motherboard on a flat stable surface. Connect the SATA and power cables for the laptop hard drive into the appropriate slots on the far left side of the board, these ports are circled in red in figure 12.5.

Step 2.2: The DH61AG board has the option for DVI video output or HDMI output connect which ever option is preferable for the user. Circled in green in figure 12.5 shows the location of both video out ports (HDMI video is being used in the figure).

Step 2.3: Lastly connect the power cord to the power port, circled in blue.

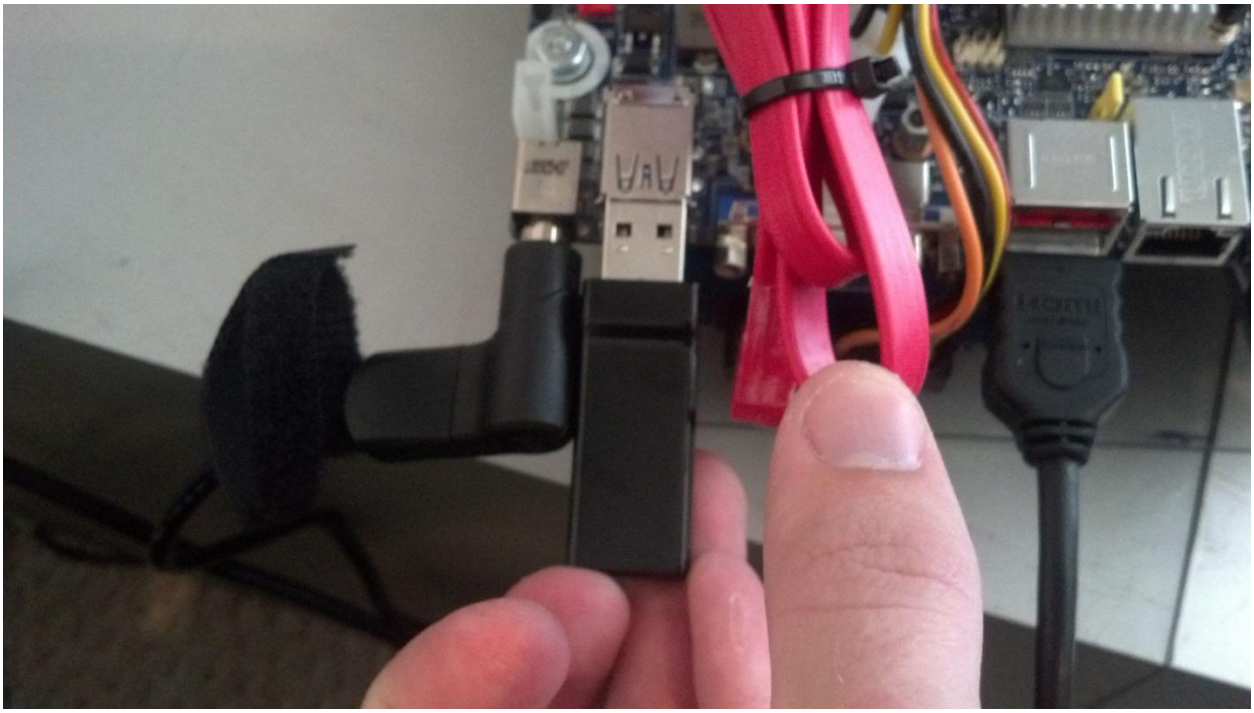


Figure 12.6 Bluetooth Dongle for Emotive Neuro-headset plugging into USB 3.0 port.

Step 2.4: After everything described in steps 2.1 – 2.3 has been connected, take the USB Bluetooth dongle that is in the Emotiv neuro-headset box and plug it into the top USB 3.0 port located next to the power port. (Dongle must be plugged into USB 3.0 port as a USB 2.0 port will cause a lag in data acquisition).

Step 2.5: Connect the Xbee USB explorer with an Xbee series 1 wireless transmitter chip plugged in, to the top USB 2.0 port located on the right hand side of the board as shown in figure 12.7

Figure 12.7 Xbee USB explorer with transmitter plugging into USB 2.0 port.

Step 2.6: After having plugged all the main components into the motherboard, connect a mouse and keyboard to the USB expander located underneath the board.

Step 2.7: Located on the left and side of the board towards the front is a set of exposed pins with different color bases. Locate the 2 pins with the red bases and short them quickly to turn on the motherboard as shown in figure 12.8. This will be accompanied by the CPU cooler fan kicking on as a sign it has been successfully switched on.

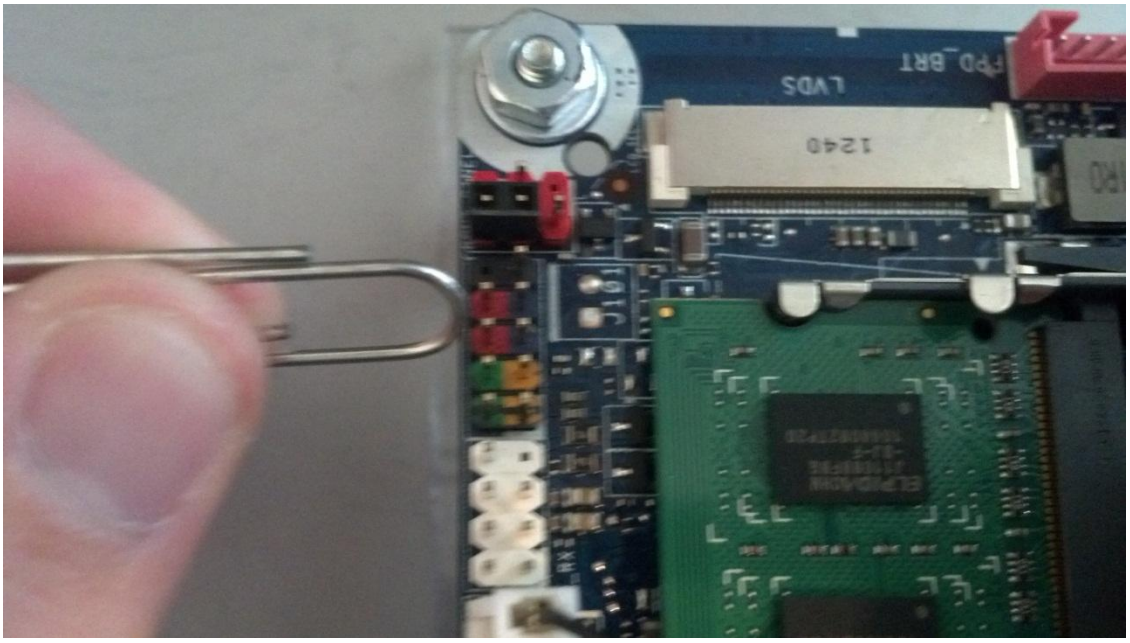


Figure 12.8 Short the red pins with a paper clip to turn on the system.

Step 3: Software Interfacing

Step 3.1: Once the Intel DH61AG has booted up. The home screen will be available. As shown in figure 12.9, find the orange Emotiv icon on the top, middle of the screen called “Control Panel” and double click it. This whill bring up the Emotiv software’s control panel.

Step 3.2: Once the control panel is open you are prompted to choose a user profile, select the proper profile and click ok as shown in figure 12.10

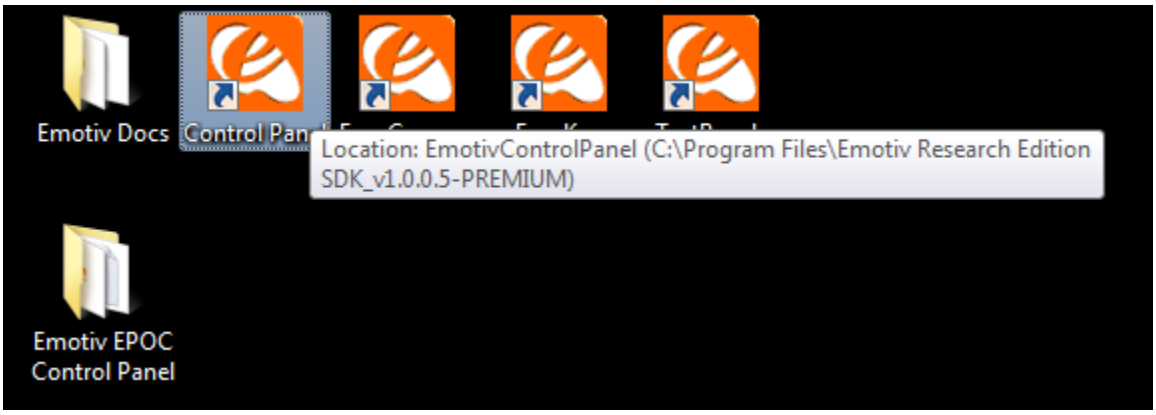


Figure 12.9 Emotiv Control Panel icon.

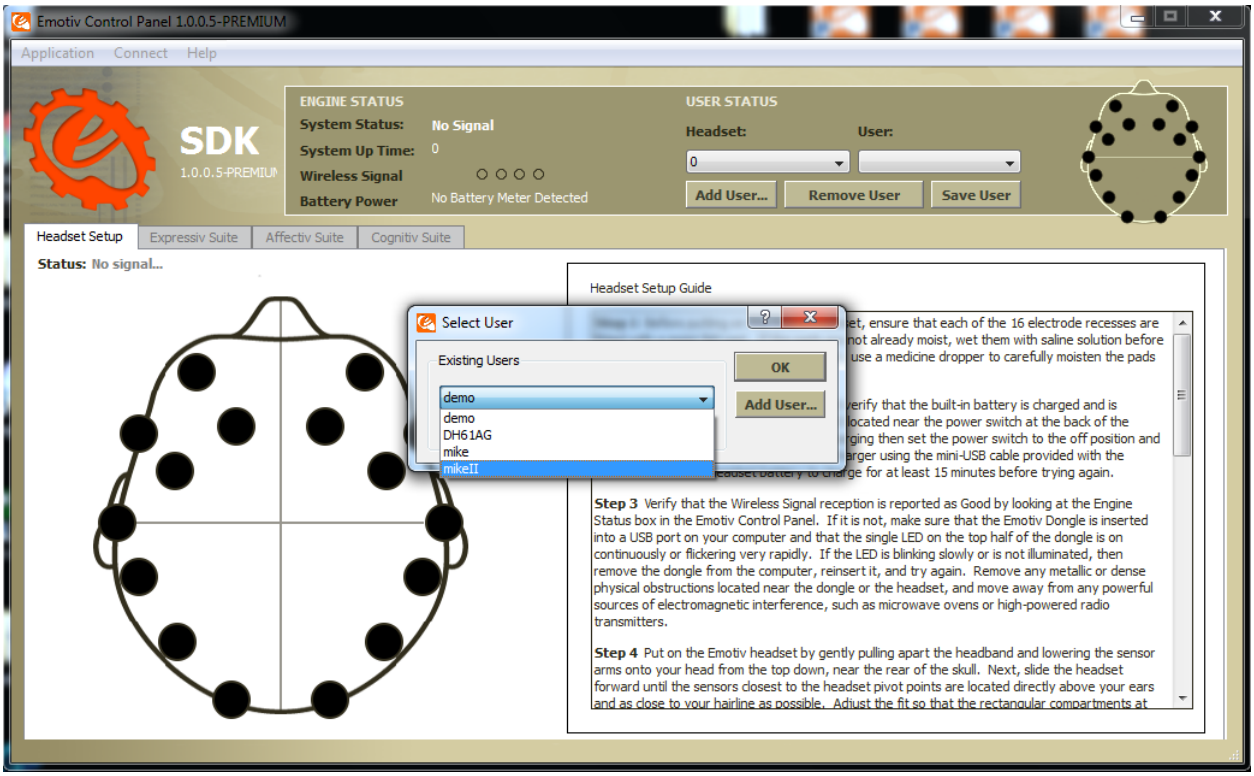


Figure 12.10 Select the proper user profile from the list given.

Step 3.3: Once the profile has been loaded, toggle the switch on the back of the headset being worn to the “on” position, this will be indicated by a blue light on the back of the headset. The black dots on the figure of the head in the control panel will be lit up in a different color, the meaning of which are detailed below;

Black – No connection

Red - Poor connection

Orange – Fair connection

Yellow – Good connection

Green – Great connection

If all connections are made properly, the head on the control panel will be marked with all green dots, as shown in figure 12.11.

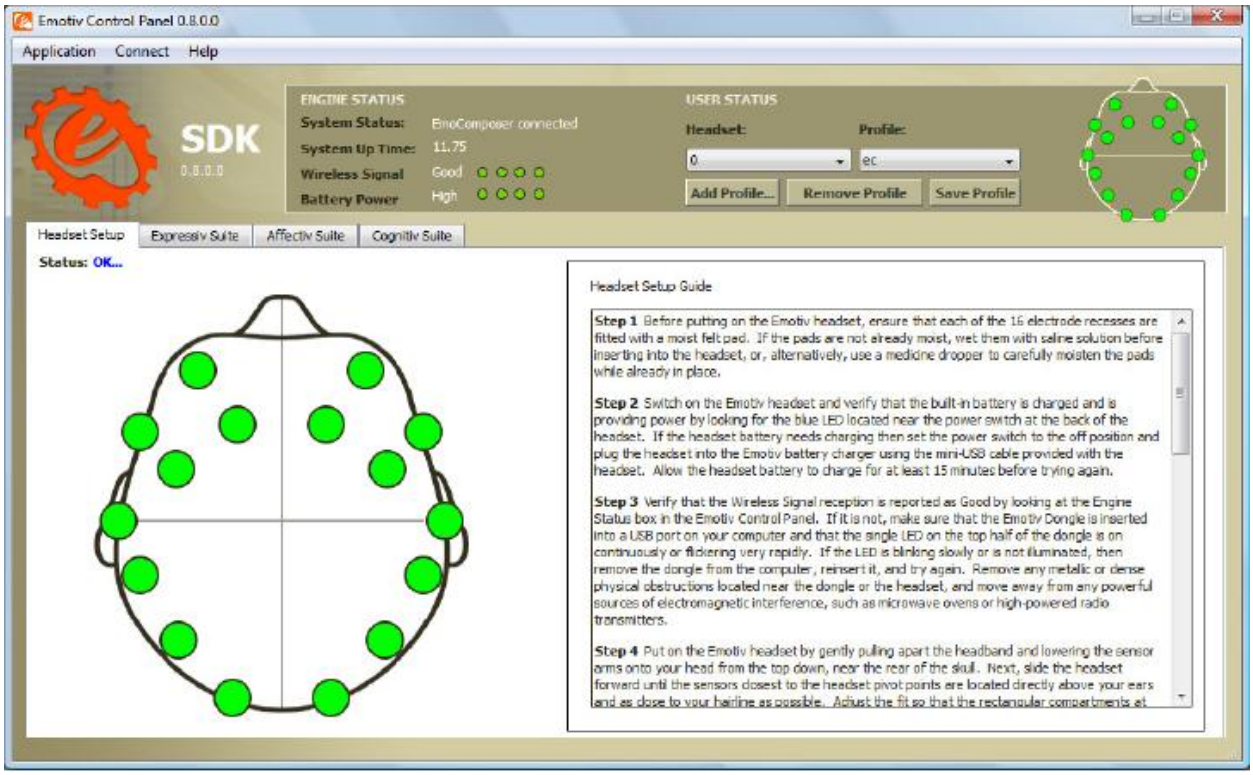


Figure 12.11 Control panel showing the ideal operating condition, all sensors marked green.

Step 3.4: Once headset is seen to be communicating properly with the board via analysis of the control panel, return to the home screen and find the shortcut icon for the

GUI specifically developed by us as a simple user interface. This icon is depicted in figure 12.12. Double click the icon and it should bring up the user interface as shown in figure 12.13.

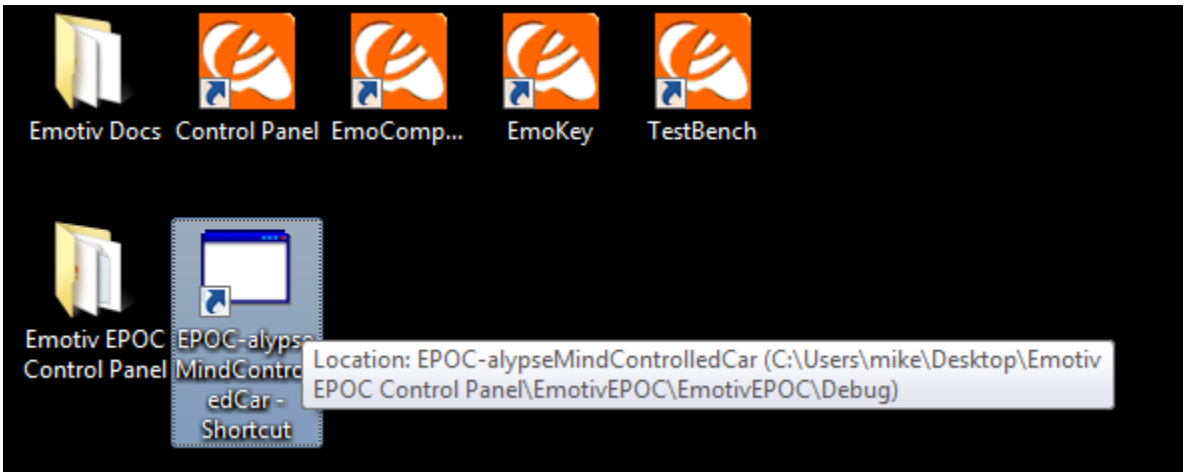


Figure 12.12 Shortcut to the control GUI program.

Figure 12.13 Prompt window for interacting with the GUI.

Step 3.5: The window brought up will prompt you to either connect with the EmoEngine with will set up a link between the headset and the car, or to connect with the EmoComposer, which is an emulator for the headset used for testing and troubleshooting. Click on the connect to EmoEngine

Step 3.6: the program will ask you to enter the name of the profile that the EmoEngine will be using, shown in figure 12.14. Enter the same profile as you selected before in step 3.2 and click the ok button.

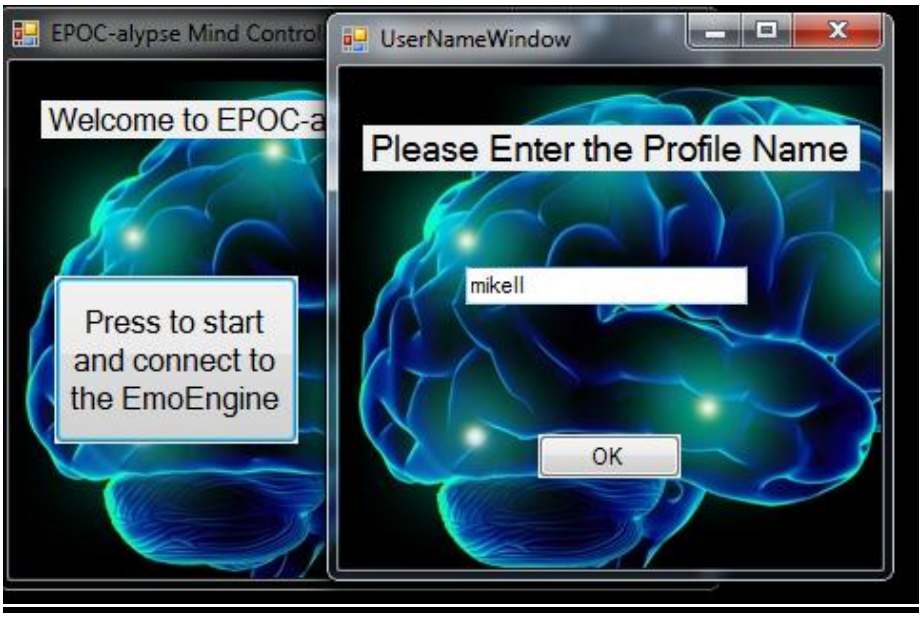


Figure 12.14 Enter the name of the profile selected in step 3.2.

Step 3.7: Once the profile name has been entered and verified a window displaying Cognitive and Expressive will appear as depicted in figure 12.15. This is the window that will display the commands that the user is sending to the car. (Do not click start untill the car has been connected!)

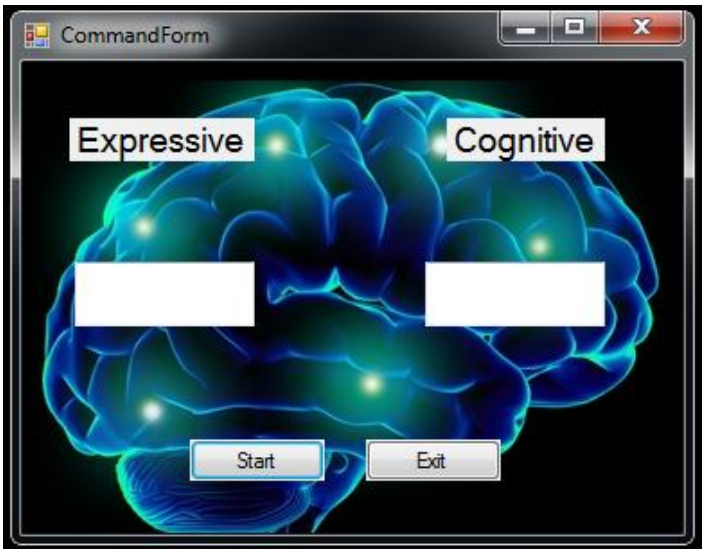


Figure 12.15 Command display window, “Expressive” commands are WL (wink left/turn left) and WR (wink right/turn right) will be displayed in the box below, will denote which way the car is being steered. “Cognitive” commands “Push” (go forward) and “Pull” (go backwards) will be displayed in the box below, denoting if the car is moving forward or backward.

Step 4: Setting up the Car

Step 4.1: Remove the plastic truck shell from the undercarriage of the car by removing 4 screws that at holding it in place.

Step 4.2: Now exposed is the custom PCB of the car. There are 5 sets of 2 input female wire connectors. As shown in figure 12.16 the pin headers circled in blue are for Xbee and circuit board power. The pin header circled on the right of figure 12.16 is the input for the power to run the board. This is to have the 6 volts supplied by the 4 AA batteries onboard plugged into it. The positive terminal is denoted by the red dot and the negative by the black.

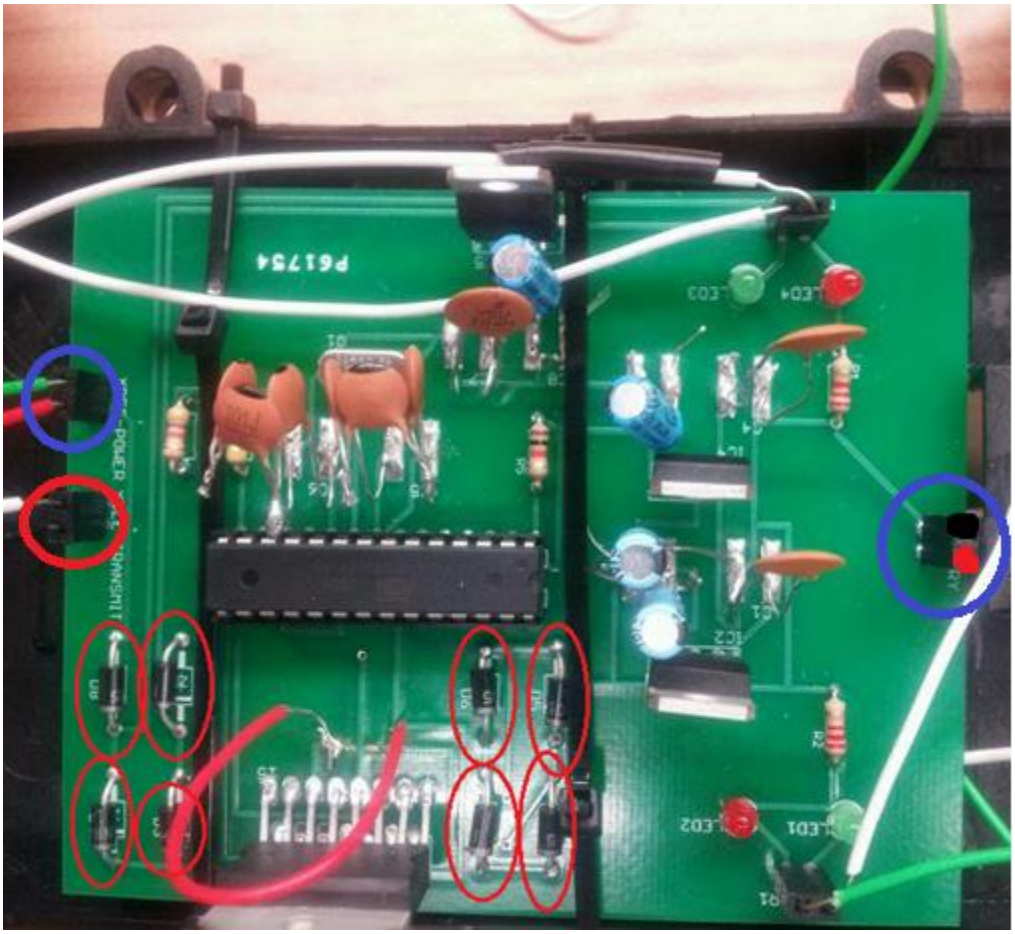


Figure 12.16 Custom PCB with female pin headers circled along with flyback diodes.

Step 4.3: Once power has been connected to the board, locate the pair of pin headers on the opposite side of the board as the circuit board power connector. Take the Xbee explorer, with the 4 wire leads already soldered on to it and plug them into the ports highlighted on the left side of the board in figure 12.16 by the blue and red circles. The blue circle connects power to the chip and the red circle is the data ports. The color order of the wires from the explorer that need to be plugged in is as follows; (from top of board to bottom) green, red, white, black. These are shown in figure 12.17 and also already plugged in in figure 12.16.

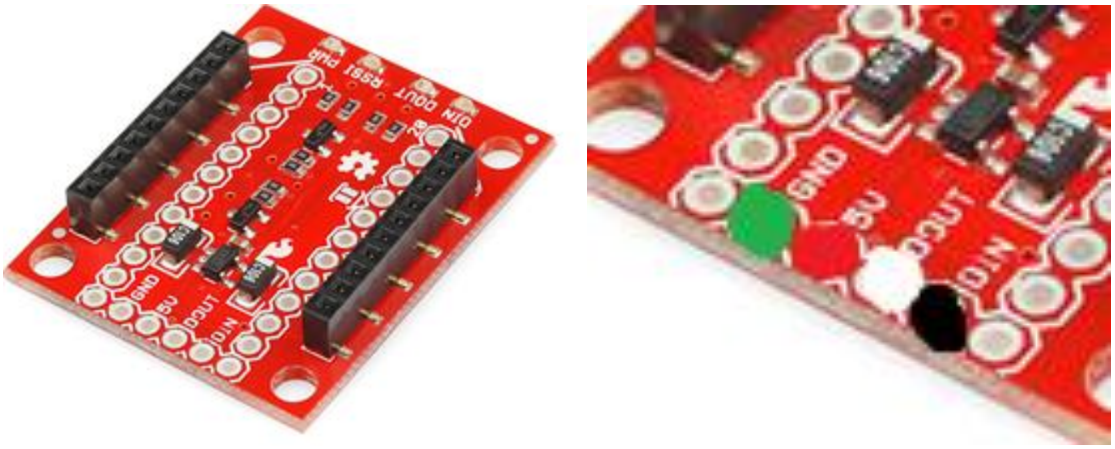


Figure 12.17 Wire color and location on Xbee explorer.

Step 4.4: Once the Xbee is connected there should be a red light that lights up on the explorer. This signifies that the chip is ready to receive commands from the DH61AG motherboard.

Step 5: Operating the Car

Step 5.1: Now that everything has been set up, The user can go head and click the start button displayed on the user interface window. This will initiate communication between the motherboard and the on board PCB. And the User is ready to operate the vehicle based on how the user programmed his profile to read the electrical activity sent from the headset.

Troubleshooting

If the user encounters any problems this section should help find a remedy to a variety of problems.

1. **I've lost signal from the headset to the motherboard.**
   Occasionally signal is lost from the headset to the motherboard. You will know this when the dots on figure of the head on the control panel all turn from a color to black. This problem is normally cause buy the user standing too far away from the Bluetooth receiver on the board or by standing to far above the Bluetooth receiver.
   To remedy this, the user must move their head close to the Bluetooth receiver on the DH61AG, to make the transmitter on the back of the headset be in close proximity. After a few moments there should be a reacquisition of signal between the headset and the motherboard, and the user may continue operating the car normally.

2. **The Emotiv and user interface software are starting to lag or not responding to commands.**
   After running for an extended period of time, the limitations of the board and software can be reached. The programs will start to lag and eventually become non responsive. This is due to the limited amount memory physically on the motherboard and memory leaks from the program.
   To solve this problem the DH61AG motherboard must be shut down and rebooted. This will restart the programs and they will return to working order as soon as the board has rebooted.
   (Note: this problem WILL occur after 30 – 40 minutes of constant operation.)

3. **The is no longer power on the on board PCB**
   Due to the fact that perfect operation of the mind controlled car is very difficult due to the amount of training required, the user is bound to make hard contact with an obstacle while operating. The jostling of the onboard PCB can cause the wires that are plugged into the female pin headers to come lose and pop out. Most susceptible to this is the power wires coming from the battery.
   All that is needed to fix this issue is remove the plastic car shell from the underbody exposing the PCB and reattach the power wires to the proper ports.

4. **I can't get all green connections on the control panel**

When the headset is first put on and the connections are viewed on the control panel, initially some connections will be bad (red or orange). To try and make better connections try the following techniques;

1. Check to see that all felt sensor are flush with scalp.
2. Try and remove as much hair as possible from underneath the sensors to try and make a more solid connection with the skin.
3. Rewet the sensor pads that are showing bad connections. They might have dried out while preparing the setup.
4. Remove sensor sending bad connection from headset and polish gold connection on back of sensor. Corrosion sometimes builds up while being store in sensor case.

13. Administration

13.1 Project Budget Estimate

The goal was to build a mind controlled car that would be modestly priced and affordable to four broke college seniors. Never factored in, were plans for any possible consumer interest or make it marketable, so when the initial cap of $1000-$1100 that had been set was smashed by the harsh reality of “things ain’t so cheap” it didn’t hinder any of the proposed ideas, just social lives. The following budget chart illustrates the cost of all materials used.

Part	Price	Part	Price
Emotiv EPOC Neuro-Headset	$750.00	**Atmega 328 Processor (2)**	$5.68
Intel Mini-ITX Motherboard	$135.95	**16 MHz Crystal Oscillator (2)**	$1.20
Intel Core i3 Processor	$189.00	**H-Bridge**	$2.95
1 GB Onboard Memory	$18.00	**1N4004 Diodes (8)**	$2.21
CPU Cooler	$39.00	**Xbee Series 1 (2)**	$64.00
160 GB Laptop Hard Drive	$95.84	**Xbee Explorer USB**	$24.95
Mounting Hardware	$15.25	**Xbee Explorer Regulated**	$28.00
90W 19V Power Source	$53.24	**PCB**	$78.34
Remote Control Car	$26.00	**Miscellaneous**	$24.44
DVI – VGA Converter	$17.03	**Total**	**$1569.66**

(Table 13.1) Budget estimate for senior design 1.

13.2 Timeline/Milestones

13.2.1 September

This moth the major goal was the final decision for our design idea. We all brainstormed to come up with the final idea so that it would meet all of the desired bullet points for the electrical engineering criteria. Once the idea was settled upon we split the research up amongst the group according to either computer engineering or electrical engineering. The research would include what kind of EEG headset would be best suited for our needs, what kind of board would meet all of our requirements and be most cost efficient, and if we were going to design and build out own car or buy one and adapt it to our needs. We also have started looking for potential sponsors.

13.2.2 October

For October we still haven't found any potential sponsors, so we have started fundraising from friends and family to ease some of the financial stress. The Emotiv EPOC headset was ordered and delivered so testing could begin on that to ensure knowledge of proper use by the time the second semester comes around. We also figured out that the original platform we wanted to run the Emotiv software on was not powerful enough and have started searching for a better replacement that can support the 2.4 GHz needed. this also reduced the work load on the computer engineers eliminating the need to learn python to write the architecture.

13.2.3 November

In November we finalized most of the designs for the project, with the exception of the motherboard we will be using. We have figured out all of the specs including the proper processor needed but still cannot decide between the Intel DH77DF or the DH61AG mini-ITX motherboards. Training with the headset has taught us that this project will not be as simple to pick up and play with for anyone. We've discovered that a minimum of 5 hours training is required to become fluid enough in thought manipulation to accomplish the tasks that are required to move the car with ease. We settled on the car that will be used in the final project along with the kind of motors, servos, and transmitter/receiver pair.

13.2.4 December

December was a crucial month finalizing everything for our project. All of the background research was completed and almost all of the components had been picked out. We decided on the DH61AG main board as the back bone for our project using the Intel i3 sandy bridge processor. Training on the headset finished and now all motions of the 3D cube can be controlled at will. The budget was also finalized giving us a goal for our fundraising.

13.2.5 January

January was the month that all of the party started to arrive in so we were able to start preliminary testing and assembly. Training on the headset was completed and finalized. We began to test the software with the headset and discovered the major errors that needed to be remedied. We also redesigned our PCB to a more efficient circuit to get rid of the serial to parallel converter we initially were going to use and replaced it by moving the Atmega 328 chip to the circuit along with the dual h-bridge driver.

13.2.6 February

We started the month out by trying to learn Eagle's software so we could start laying out our PCB, this consumed most of the month. There was also major progress with coding the translation program that turns the commands into a 6 bit binary string.

13.2.7 March

The last of out necessary parts came in. we configured the Xbees to communicate with one another while simultaneously writing the proper code to send the 6 bit binary string between the motherboard and the car. The PCB circuit was prototyped over the course of 2 weeks and then finalized and sent to manufacturing (later than we anticipated). The biggest milestone of the month was getting out entire project working in prototype form, which included rough software codes breadboard circuits and the Emotiv headset emulator.

13.2.8 April

We only had 11 days to finish assembling our project, our PCB arrived 4 day before the final presentation. It was populated and troubleshot in a single day, leads had to be severed and bridge and wires had to be connected to where leads were accidentally left out. The final project was up and running 3 days before final presentation. We got bored the day before we presented and added LED lights to the shell of the truck for added visual effect. April 11th we presented.

13.3 Areas of Assigned Responsibility

The figure below is a visual representation of the roles that were assigned based on major before actually researching into the design of the project in general.

Team Member	Emotiv EPOC Headset	Emotiv Coding	Software / other coding	Onboard PCB	Remote Controlled Car
Kathryn				**X**	**X**
Mike	**X**			**X**	
Chris		**X**	**X**		
Lee		**X**	**X**		

Table 13.2 Areas of Assigned Responsibility

The people responsible for each portion of the project had to obtain materials and conduct research independently.

With regards to the paper, parts were assigned accordingly to the area of interest that was initially assigned to each individual. Our responsibilities with the paper included;

- Keeping within the laws of ethics.
- Researching and citing sources accurately and in a orderly fashion.
- Keeping our budget within the set estimate of funding.
- Meeting in an orderly fashion as well as keeping track of what occurs during each meeting.
- Getting the paper done by the deadline set by the professor of the class.

These rules will be adhered to so that we may be able to show good effort towards the final evaluation of our project and of our peers during the next semester within Senior Design II. Once these guidelines are followed we will be able to demonstrate a mastery of skill as well as finalize our development throughout our college career.

14. Summary/Conclusion

It was the general goal of the group to learn about the detection, function, and implementation of EEG readings in a practical, instead of medical way. The EPOC-alypse mind controlled car allowed us to gain experience with dealing with raw EEG data and transforming that data into a code that could be recognized and be used with a remote control car. Researching and designing the actual device has proven to be a very exciting process.

Because there were many more aspects to this project than originally anticipated, designing such a system to incorporate everything that was called for, meant that each member of the project became specialized in a specific area of expertise, be it the headset, motherboard, or car. All the members of the group though have learned the essentials of the components and understand what is necessary to make the car run properly.

It can be noted that the learning curve in general for this project was quite steep, with none of the group members having no background in most of the subjects that this project required. Leaning about the inner workings of the brain is something most electrical and computer engineers don't have to worry about, but for proper operation of the headset it was important to know what area of the brain a sensor pair is situated over and what that specific area of the brain is responsible for.

While difficult for our budget, not having a sponsor proved to be quite beneficial for the creativity of the team. Being able to come up with loop holes around certain problems that were holding up progress without having any outside constraint from a client was to put it simply; nice. The group could be as extravagant or minimal with a certain aspect of the project as they saw fit. As an example take the motherboard. Our original platform to run our software on was a raspberry pie, due to its size and multiple functionalities it seemed like a good choice.

Come to find out later that it was highly improbable that we would be able to develop a custom architecture to interpret EEG signals and would need an extravagant 2.4 GHz to run the Emotiv software on a much larger board. If there had been a sponsor and they had put restrictions on parts or board specs it could be said that that would have made life exceedingly more difficult for the group.

Learning about the different areas of the brain was also a fascinating experience, seeing a predicted response from a certain stimuli and then being able to tell which area of the brain it came from is definitely a skill not many people can say they have. This knowledge came into play the most while testing the headset and training for the 3D cube to perform a second action other than push. This initial roadblock was caused by the brains natural inability to switch from one area of activity to another without the influence of external stimuli. So when a tack was introduced into a shoe and worn to try and train, it was predicted that a response in the somatosensory association cortex would present itself, which it did. This reaction to pain was initially what caused the cube to rotate right. But after a couple weeks of training simply having the shoe with the tack *next* to the foot it was typically on, and concentrating on the pain that it would yield, produced a readout identical to a readout if the tack shoe was being worn. The brains ability to learn this association over such a short time was incredible, and to be able to see the physical patters produced made it that much better.

The brain computer interface aspect of this project limits users to those who have proficient control over isolating thoughts by training with the headset for a period of time. So it does not accommodate our original goal of anyone just being able to put on the Emotiv EPOC headset and being able to drive the car.

The EPOC-alypse car is a design that has a very specific niche. The market for BCI technologies hasn't yet emerged as a mainstream thing. It is reserved for those who want to explore and learn about the emerging field. So there is no target audience or market that this project is directed towards, the ultimate goal of this project is to educate ourselves and others about the used for brain computer interface and possibly pass on our knowledge to those who one day might be able to bring BCI into everyday life.

15. List of Figures

16. List of Tables

17. References

- Bishop, Bryon. "Emotiv EPOC EEG Headset Hacked." *Emotiv EPOC EEG Headset Hacked*. H plus Magazine, 13 Sept. 2010. Web. 27 Nov. 2012. <http://hplusmagazine.com/2010/09/13/emotiv-epoc-eeg-headset-hacked/>.

- "Emotiv Epoc." *LabVIEW Hacker*. N.p., n.d. Web. 27 Nov. 2012. <http://labviewhacker.com/epoc.php>.

- Fairclough, Steve. "Physiological Computing." *: Emotiv EPOC and the Triple Dilemma of Early Adoption*. Physiological Computing, 13 Dec. 2010. Web. 27 Nov. 2012. <http://www.physiologicalcomputing.net/?p=1191>.

- Herman, Stephen L. *Industrial Motor Control: Workbook and Lab Manual, 6E*. 6th ed. Clifton Park, NY: Delmar Cengage Learning, 2010. Print.

- "Proximity Sensor." *Www.sensors-transducers.machinedesign.com*. Machine Design, n.d. Web. 12 Nov. 2012. <http://www.sensors-transducers.machinedesign.com/guiEdits/Content/bdeee4/bdeee4_7.aspx

- Torres, Gabriel. "Everything You Need to Know About The Motherboard." *Hardware Secrets*. N.p., 10 Feb. 2010. Web. 14 Nov. 2012. <http://www.hardwaresecrets.com/article/Everything-You-Need-to-Know-About-The-Motherboard-Voltage-Regulator-Circuit/616/1>.

- "What Is the Difference between a DC Motor and Servo Motor?" *The Handy Board*. N.p., n.d. Web. 14 Nov. 2012. <http://handyboard.com/hb/faq/hardware-faqs/dc-vs-servo/>.

- "Choosing and Using Nickel-Metal-Hydride (NiMH) Rechargeable Batteries." *Http://www.stefanv.com*. N.p., n.d. Web. 16 Nov. 2012. <http://www.stefanv.com/electronics/using_nimh.html>.

- "What's the Best Battery?" *Www.BatteryUniversity.com*. N.p., n.d. Web. 16 Nov. 2012. <http://batteryuniversity.com/learn/article/whats_the_best_battery>.

- "Sensors: Proximity." *Www.EngineersHandbook.com*. N.p., n.d. Web. 17 Nov. 2012. <http://www.engineershandbook.com/Components/proximitysensors.htm>

- Whitlock, Bill. "UNDERSTANDING and Controlling RF Interference." *Sound & Video Contractor*. N.p., 1 Feb. 1999. Web. 18 Nov. 2012. <http://svconline.com/mag/avinstall_understanding_controlling_rf/>.

- Brain, Marshall, Charles W. Bryant and Clint Pumphrey. "How Batteries Work" 01 April 2000. HowStuffWorks.com. <http://electronics.howstuffworks.com/everyday-tech/battery.htm> 18 Nov. 2012.

- Berman, Eric, Kalyn Kovac, and Bruno Umeadi. *Mind-controlled R/C Car*. Tech. N.p., n.d. Web. <http://cratel.wichita.edu/blogs/eecsfinalreportspr2012mindcontrolledrccar/>.

- Bothra, Jitendra, and Baturalp Torun. "SWARM Extreme." *SWARM Extreme*. N.p., n.d. Web. <http://www.slideshare.net/bet3/swarm-extreme>.

- Emotiv. Emotiv Software Development Kit User Manual for Release 1.0.0.5.

- Rouse, Margaret. "Event-Driven Architecture (EDA)." *SearchSOA*. N.p., n.d. Web. <http://searchsoa.techtarget.com/definition/event-driven-architecture>.

- Vourvopoulos, A., Brain-controlled NXT Robot - Tele-operating a robot through brain electrical activity, Bachelor's Thesis, Department of Engineering and Computing, Coventry University, UK, (2011).

- Bruce CJ, Goldberg ME, Bushnell MC, Stanton GB. (1985). "Primate frontal eye fields. II. Physiological and anatomical correlates of electrically evoked eye movements.". Journal of Neurophysiology 54 (3): 714–734. PMID 4045546.

- Kirchner H, Barbeau EJ, Thorpe SJ, Régis J, Liégeois-Chauvel C. (2009). "Ultra-Rapid Sensory Responses in the Human Frontal Eye Field Region".Journal of Neuroscience 29 (23): 7599–7606.doi:10.1523/JNEUROSCI.1233-09.2009.

- Philip David Zelazo and Ulrich Muller: Executive function in typical and atypical development, in: Usha Goswam (ed): Blackwell Handbook of Child Cognitive Development, 2002

- Philip David Zelazo and Ulrich Muller: Executive function in typical and atypical development, in: Usha Goswam (ed): Blackwell Handbook of Child Cognitive Development, 2002

- Brodmann K. Vergleichende Lokalisationslehre der Grosshirnrinde. Leipzig : Johann Ambrosius Bart, 1909

- Brent A. Vogt, Deepak N. Pandya, Douglas L. Rosene (August 1987). "Cingulate cortex of the rhesus monkey: I. Cytoarchitecture and thalamic afferents". The Journal of Comparative Neurology 262 (2): 256–270.

- Burgess, P.W., Dumontheil, I., & Gilbert, S.J. (2007). The gateway hypothesis of rostral prefrontal cortex (area 10) function. Trends in Cognitive Science, Vol. 11, No. 7

- Ramnani N, Owen AM. (2004). Anterior prefrontal cortex: insights into function from anatomy and neuroimaging. Nat Rev Neurosci. 5(3):184-94.

- Semendeferi K, Armstrong E, Schleicher A, Zilles K, Van Hoesen GW. (2001). Prefrontal cortex in humans and apes: a comparative study of area 10. Am J Phys Anthropol.

- Rivara CB, Sherwood CC, Bouras C, and Hof PR (2003). "Stereologic characterization and spatial distribution patterns of Betz cells in the human primary motor cortex". The anatomical record. Part A, Discoveries in molecular, cellular, and evolutionary biology 270: 137-151

- Lassek, A.M. (1941). "The pyramidal tract of the monkey". J. Comp. Neurol 74: 193-202.

- “Human brain.” www.en.wikipedia.org. Web. 12 Oct 2012 < http://en.wikipedia.org/wiki/Human_brain >.

- Busey, Tom. “Brain Structure.” www.cognitrn.psych.indiana.edu. University of Indiana, Web. 1 Nov 2012 <http://cognitrn.psych.indiana.edu/busey/Q301/BrainStructure.html>.

- “Arduino.” www.en.wikipedia.org. Web. 29 Nov 2012 < http://en.wikipedia.org/wiki/Arduino >.

- “Printed circuit board.” www.en.wikipedia.org. Web 20 Sept 2012 <http://en.wikipedia.org/wiki/Printed_circuit_board >.

- “Arduino Tutorial.” www.ladyada.net. 27 April 2012. Web. 1 Dec 2012 <http://www.ladyada.net/learn/arduino/index.html >.

- “ATmega328.” www.atmel.com. Web. 1 Dec 2012 <http://www.atmel.com/devices/atmega328.aspx?tab=overview>.

- “ARDUINO.” arduino.cc/en/. Web. 1 Dec 2012 <http://arduino.cc/en/ >.

- Ogawa S**,** Tank DW, Menon R, Ellermann JM, Kim SG, Merkle H, and Ugurbil K. *Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging. Proc Natl Acad Sci USA 89: 5951–5955, 1992.*

- Pierce, Kate. "The Brain Responds To Music The Same Way As Eating." *The Huffington Post*. TheHuffingtonPost.com, 10 Jan. 2011. Web. 05 Dec. 2012.

- Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W., LaMantia, A., McNamara, J. O., & White, L. E. (Eds.). (2008). Neuroscience (4th ed.). Sunderland, MA: Sinauer Associates.
- Brodmann K. Vergleichende Lokalisationslehre der Grosshirnrinde. Leipzig : Johann Ambrosius Bart, 1909 “Human brain.” www.en.wikipedia.org. Web. 12 Oct 2012

- < http://en.wikipedia.org/wiki/Human_brain >.
  Busey, Tom. “Brain Structure.” www.cognitrn.psych.indiana.edu. University of Indiana, Web. 1 Nov 2012

- <http://cognitrn.psych.indiana.edu/busey/Q301/BrainStructure.html>. “Arduino.” www.en.wikipedia.org. Web. 29 Nov 2012

- < http://en.wikipedia.org/wiki/Arduino >.
  “Printed circuit board.” www.en.wikipedia.org. Web 20 Sept 2012

- <http://en.wikipedia.org/wiki/Printed_circuit_board >.
  “Arduino Tutorial.” www.ladyada.net. 27 April 2012. Web. 1 Dec 2012

- <http://www.ladyada.net/learn/arduino/index.html >.
  “ATmega328.” www.atmel.com. Web. 1 Dec 2012

- <http://www.atmel.com/devices/atmega328.aspx?tab=overview>.
  hobbytronics. http://www.hobbytronics.co.uk/arduino-atmega328-pinout

- “ARDUINO.” arduino.cc/en/. Web. 1 Dec 2012 <http://arduino.cc/en/ >.
  Berman, Eric, Kalyn Kovac, and Bruno Umeadi. *Mind-controlled R/C Car*. Tech. N.p., n.d. Web. <http://cratel.wichita.edu/blogs/eecsfinalreportspr2012mindcontrolledrccar/>.

- Bothra, Jitendra, and Baturalp Torun. "SWARM Extreme." *SWARM Extreme*. N.p., n.d. Web. <http://www.slideshare.net/bet3/swarm-extreme>.

- Emotiv. Emotiv Software Development Kit User Manual for Release 1.0.0.5.

- Rouse, Margaret. "Event-Driven Architecture (EDA)." *SearchSOA*. N.p., n.d. Web. <http://searchsoa.techtarget.com/definition/event-driven-architecture>.

- Vourvopoulos, A., Brain-controlled NXT Robot - Tele-operating a robot through brain electrical activity, Bachelor's Thesis, Department of Engineering and Computing, Coventry University, UK, (2011).
- "[ARCHIVE] What Are Flyback Diodes and Why Are They Used?" *National Instruments*. National Instruments, 23 Oct. 2004. Web. 27 Apr. 2013.
- McManis, Chuck. "H-Bridge Theory & Practice." *Chuck McManis*. N.p., 23 Dec. 2006. Web. 27 Apr. 2013

18. Permission of use

[Christopher] Hello, I would like to know who I would need to talk to in order to get written permission to use and cite the User's Manual included in the LITE SDK. I was directed to the media page earlier, but that does not have anything for the User's Manual. The purpose of this is strictly for research documentation and WILL NOT be used for any other purpose. Please let me know.

[Kimberly] Hello, welcome to Emotiv! My name is Kimberly. How may I help you today?

[Kimberly] I answered your support ticket.

[Christopher] Oh, perfect. You are who I wanted to speak to

[Kimberly] The information that I gave you is the only information that is available to be used.

[Christopher] Ok, so there is no way of attaining permission to use the manual? We just want to be able to use some of the figures included, such as the enums and data structrues in order to explain how the sdk is used

[Christopher] it is for a research document for my university

[Christopher] the user's manual is included in the download that is open to the public, but we are not sure if we need permissions to use it

[Kimberly] I will be glad to ask, however, we have not allowed it in the past except for those images, etc. that I mentioned in the support ticket.

[Christopher] Ok, please let me know. This is crucial for our research and we would like to make it as thorough as possible

[Christopher] Maybe there is someone else I can talk to in regards to this?

[Kimberly] Do you have SDKLite?

[Kimberly] Or one of the other SDKs?

[Christopher] Yes, we have SDKLite as well as the research SDK

[Kimberly] Then you can use if for the purposes that you stated as long as it is correctly mentioned.

[Christopher] Thank you very much. I very much appreciate your help!

[Christopher] We will be sure to include all necessary citations and references

[Kimberly] Thank you for contacting Emotiv.