BMW (Brain-Controlled Motorized Wheelchair)

Alex Polus, Anish Sinha, Sandeep Sagoo, Yvette Weng

Abstract

For those paralyzed from the neck down (quadriplegic), simple tasks such as short-range transportation can pose a major difficulty. The goal of this research is to provide quadriplegic individuals with independent transportation. By using electroencephalogram (EEG), a noninvasive method to measure electrical activity in the brain, the group can create a brain-computer interface (BCI) that will allow the user to control a wheelchair with his or her own thoughts. Components being integrated into the project include a Raspberry Pi 3 (with Bluetooth) and a Neurosky headset, which allows the user to analyze levels of concentration, meditation, and registers blinks. Success in this research setting will be indicated by the ability of a non-paralyzed user to navigate a GoPiGo (a Raspberry Pi connected to motorized wheels) proof-of-concept wheelchair-like device from rest to an indicated ending point using only their thoughts.

Introduction

Paralysis

For many of us, simple tasks in life like using the restroom or drinking water are quite trivial. However, there are millions of people around the world afflicted by partial or complete loss of their body's proper motor function. Many with such disorders are still able to be quite independent; advancements in a wheelchair and assistive technologies are providing increasing levels of mobility to those in need of assistance. But, how can you help someone who is completely paralyzed from the neck down?

Quadriplegia is defined as partial or complete paralysis from the neck down [1]. In these situations, individuals are unable to use their arms to push the wheels of a wheelchair or use their hands to move a joystick on an electric wheelchair. Therefore, one must use some creative problem solving to allow the user to control a system with very limited I/O capability.

Existing Assistance for Immobility

Some reasonable options of control frameworks for quadripegic patients include a computer vision algorithm to analyze facial features, or even a button activation by tongue or bite like Dr. Hawking's wheelchair [2]. Sadly, these solutions are all deliverable on a per-case basis. The best generic solution on the market is the Q-Logic Advanced Drive Controller, which still requires some degree of control of the hand - at least ability to move a shoulder. This simply isn't good enough and our group sees an opportunity [9].

Analyzing Signals from the Body

Voltage values from the body can be read that indicate events or intent within the individual. ECG, EMG, and EEG signals are the three most common forms of a bodily signal. Electrocardiograms (ECG) measure electrical impulses in the circulatory system, pertaining to the surge of power associated with each heartbeat. Electromyography (EMG) measures the potential associated with the firing of muscle fibers. And finally, electroencephalograms (EEG) measure potential from neurological activity in the body. All of these values can be read from the surface of the skin at various locations [3].

Contributions

- Integrate the NeuroSky headset with Raspberry Pi, establishing Bluetooth communication
- Control the GoPiGo using signal values from the NeuroSky
- Demonstrate that a practical brain-controlled machine can be built with existing commodity hardware
- Create a method/framework that can be extrapolated from the GoPiGo to a real wheelchair

Technical Material

System Design

Our design consists of three parts: the Neurosky headset, a Raspberry Pi and a GoPiGo car kit. The Neurosky headset collects the brain signal and sends the data to Raspberry Pi via Bluetooth. Once Raspberry Pi receives the signal, it runs the movement control algorithm and drives the two motors on GoPiGo. Both the Raspberry Pi and the GoPiGo car are powered by 4 AA batteries. Figure 1 shows an overview of our design.

Figure 1 System Design Overview

MindwaveMobile Headset

The third iteration of Neurosky's line of EEG headsets, the Mindwave Mobile reads signals on the microvolt scale from the forehead. Through clear conductive contact with the skin, the Mindwave Mobile is capable of reading EEG values for attention and meditation as well as EMG values for blink intensity, measured from reference ground on the earlobe. These values are then processed into Neurosky Scores, a proprietary value calculated from 1-100, in the ThinkGear chip and broadcasted out unencrypted via Bluetooth along with the raw values. Much to the merit of our project, the MindwaveMobile is not compatible with the Raspberry Pi or Arduino via any kind of straightforward API or local application [4].

Figure 2 MindwaveMobile Headset Structure

Raspberry Pi v3

Arguably the most well-known microcontroller, the RaspberryPi is an affordable and effective tool for prototyping and agile development. Furthermore, as in this instance, it is very useful for embedded applications. The Pi was the workhorse of the project, using the v3's included WiFi and Bluetooth chips to communicate with a laptop via SSH and to the Neurosky via Bluetooth. The Raspberry Pi was booted using DexterOS from Dexter Industries, the maker of the GoPiGo. Python scripts ran on the Raspberry Pi are the glue that makes a functional solution [5].

GoPiGo3

Ideal for our project, the GoPiGo is a cart designed to be driven by the Raspberry Pi. The kit requires a couple hours of assembly, but is very straightforward. The chassis is made up of a ball bearing as a stabilizing third wheel, with two tank-drive wheels used to create motion. This is quite applicable to the goal of extrapolating to a wheelchair, as both use tank-drive. In addition to the chassis, the GoPiGo comes with a board that stacks on the GPIO pins of the Pi and uses 8 connected AA batteries to provide the necessary current to the two motors that attach to it [6].

Challenges of Integration

Although integrating the Raspberry Pi with the GoPiGo was fairly trivial, integrating the MindwaveMobile into the system was a massive headache. There is no straightforward API or library to use with the MindwaveMobile. Furthermore, there is no local application for Raspberry Pi or Arduino. Luckily, given the fact that the MindwaveMobile is an insecure Bluetooth device with an unencrypted signal, we were able to use the 'bluetooth' library in Python 2.7 to parse the signals sent out by the MindwaveMobile.

Before running a script with this code, the group had to use the bluetoothctl Linux command line tool to manually connect to the MindwaveMobile. Our headset's bluetooth ID is CC:78:AB:25:46:44. Once trusted and connected, we had to bind the bluetooth ID to rfcomm0 to support the necessary serial communication.

Once this connection is established, our group was able to build upon the 'NeuroPy' repository [7] with some useful lines and information drawn from 'python-mindwave-mobile' as well [8]. Our Implementation parsed the bluetooth packets based on the following opcodes (left) and acted upon the attention and meditation opcodes which are read in the following manner (right).

```
34  # Byte codes
35  CONNECT = '\xc0'
36  DISCONNECT = '\xc1'
37  AUTOCONNECT = '\xc2'
38  SYNC = '\xaa'
39  EXCODE = '\x55'
40  POOR_SIGNAL = '\x02'
41  ATTENTION = '\x04'
42  MEDITATION = '\x05'
43  BLINK = '\x16'
44  HEADSET_CONNECTED = '\xd0'
45  HEADSET_NOT_FOUND = '\xd1'
46  HEADSET_DISCONNECTED = '\xd2'
47  REQUEST_DENIED = '\xd3'
48  STANDBY_SCAN = '\xd4'
49  RAW_VALUE = '\x80'
```

```
        elif(code == '04'):  # attention
#print "attention"
            i = i + 1
            self.attention = int(payload[i], 16)
#i = i + 1
        elif(code == '05'):  # meditation
#print "meditation"
            i = i + 1
            self.meditation = int(payload[i], 16)
#i = i + 1
        elif(code == '16'):  # blink strength
#print "blink"
            i = i + 1
            self.blinkStrength = int(payload[i], 16)
```

Figure 3 Example Opcode Implementations

The data following the opcode is the value corresponding to that opcode, whether it be a raw EEG value or a NeuroSky score, or even a connect/disconnect signal. Once these values could be read consistently, the group was able to implement a one-dimensional driving mechanism by tracking attention and meditation values.

Attention/Meditation Control

With a lot of muscle movements on our forehead and one single electrode on the headset, the data collected was very noisy. In order to get a better sense of the performance of the Mindwave Mobile, we ran multiple attention and meditation control tests over different time intervals. Figure 4 is one set of the results of the tests. For attention tests, our team used math worksheet to aid the control of concentration. In the left plot of Figure 4, the state of attention was toggled in 20-second intervals. When the user was concentrated, the attention value goes up to around 70. And when the user was distracted the value was at about 30. On the other hand, the meditation test did not give us as promising results as the attention test, as shown in the right plot of Figure 4. The state of meditation was mainly controlled by slow, deep breaths versus quick, short breaths. The results showed no clear change of meditation level between states.

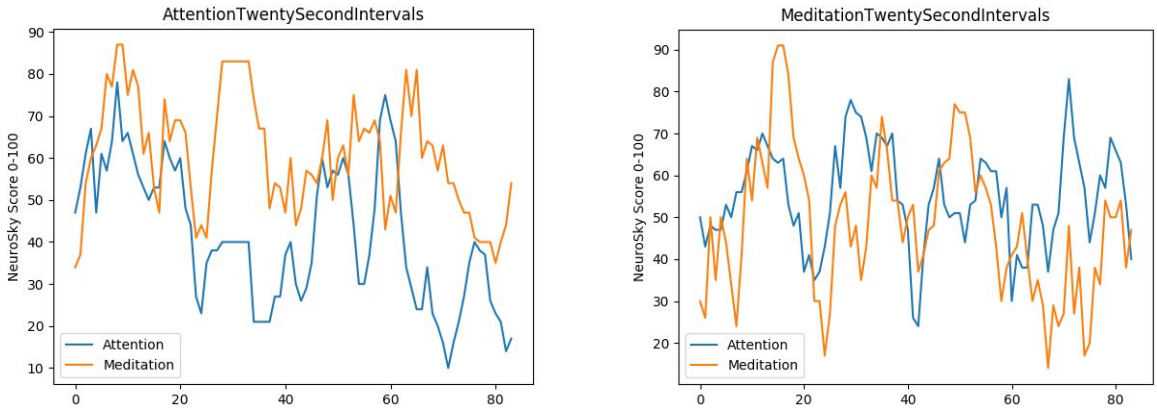

Figure 4 Data Plots of Attention and Meditation Tests

Movement Control

From the testing results, our team decided to rely more on the attention value. We have designed various movement control algorithms. The first one is a sliding window method, where we stored five attention values for the past five timestamps. If the mean for a window is above a certain threshold, the GoPiGo should start moving. However, this method caused a huge lagging in car movement. And it was difficult to stabilize the values within a window so that using the average was not accurate enough. Our second implementation was using double thresholds--both attention and meditation. The reason is that we found people tends to be less relaxed when concentrating on some work. This method did not work well either due to the large fluctuation of meditation values. Lastly, we tried the simplest method with a single attention threshold. Surprisingly, this simple algorithm gave us the best results. From the previous attention tests, we chose the threshold to be 65. When the user is moving for a long distance, the concentration sometimes dropped below 65. Although it takes several starts and stops for the car to get to the destination, we had good control of stopping the car. Our team was able to obtain a 70% accuracy within the target.

Milestones

Project Proposal Milestones

Milestone	**Status**
Drivetrain for the prototype of the wheelchair	Complete
Configure Mindwave Mobile to establish different outputs	Complete
Mapping Mindwave Mobile outputs to simple outputs: forward, backward, left, and right	Complete
Map Mindwave Mobile outputs as inputs to the wheelchair prototype	Complete
Receiving raw EEG data from the Mindwave Mobile	Complete

Signal processing of EEG data to create outputs: forward, backward, left, right	Incomplete
Mapping all four outputs to “wheelchair” prototype	Incomplete

Mid-Quarter Milestones

Milestone	**Status**
Acquire the Mindwave Mobile from CogSci	Complete
Get a Raspberry Pi with Raspbian OS installed and a USB mouse for setup	Complete
Purchase cheap USB keyboard for setup	Complete
Order/assemble GoPiGo & Integrate with Raspberry Pi	Complete
Test Mindwave Mobile Capabilities with iPhone App	Complete
Python Script to Pair Sky/Pi via Bluetooth	Complete
Resolve Python Version Error	Complete
Python Script to test GoPiGo driving functions	Complete
Start Writing Main Python Script for BCI Control Flow	Complete

Final Milestones

Milestone	**Status**
Every test outputs a csv about what happened (time, EEG values, % values)	Complete
Keep track of target % error when driving (here, error will be defined as the distance from the target we can get it to stop. I.e. if our GoPiGo stops at 5 feet and our target is at 10 feet, our error is 50%).	Complete
Get Pi move forward and stop within 10 feet of target . We still don’t know how precise we will be able to get our filter, and we feel this is a fair baseline measure, to wind up closer than we started.	Complete

Problems Faced

When we first started our project, things were moving relatively smooth. We were getting familiar with the Mindwave Mobile and GoPiGo functionalities and tested them separately to ensure that our project would work the way we desired it to. However, once each individual component was tested, we began to face problems. The biggest issue we faced was data from the Mindwave Mobile. The data was very noisy and it became hard to control the GoPiGo using just our meditation and attention values. Initially we thought that our Mindwave Mobile would support a blink function, just as in the mobile app. However, when connected to the GoPiGo, we did not receive any data from the blink function and were limited to only attention and meditation values. We began experimenting with different ideas for triggering different directions via thresholding and using the slope of our graphs. However, the Mindwave Mobile signals took about 5 seconds to register on the GoPiGo and the data was very noisy. Part of our team searched the

available Mindwave Mobile libraries to see if there were any repositories with the blink function working. We found several repositories with different code to get blink to work, unfortunately, we could not get any of them to work on our Raspberry Pi. We then proceeded to tinker with a library that many found useful to no avail. We could not get our Mindwave Mobile and GoPiGo to exchange blink values. This hindered our progress in terms of toggling the different directions of our GoPiGo. However, we were able to control the GoPiGo: we could make it stop and go using our attention and meditation values. With limited functionality from our Mindwave Mobile, we shifted our focus to gathering data and trying to control the single forward movement. We used our graphs to help us pinpoint a value threshold that would aid in controlling the GoPiGo. This new threshold was now set at 65% of the Concentration Level. Our new goal was to toggle stop on the GoPiGo accurately in respect to our measurements. Focusing on a single direction allowed us to understand the data more deeply and allowed us to simulate real life scenarios. We were able to figure out the delay from when the Mindwave values stabilized to fine tune our attention and meditation values. Once that was achieved, it took us a few attempts to accomplish our goal.

Related Work

When we were researching about our project in the very beginning, we were surprised to see the amount of work that has already been put in into the development of such technology. There were two ways others had implemented the brain connected part of the system: using a synchorons prompt based model like the P300, or the asynchronous model like MI which would allow the user to initiate sequences and actions rather than wait for a prompt. The most famous example of a prompt based wheelchair is Stephen Hawking's wheelchair. There was a screen on Mr. Hawking's wheelchair that would show all the options he choose from, and then a selector would be highlighting each option one at a time. Whenever the selector landed on the option that Mr. Hawking wanted to perform, he would twitch his cheek muscle to select that option. However, the biggest problem with this approach is that the user is at the mercy of the selector traversing through every option. In our development of the project, our goal was to create an intent based model that would enable the user to control the wheelchair at their own mercy. Thus, we opted for the second method as we wanted the user to be in full control of the wheelchair at any time of their interaction.

Future Work

If the group were to take this project further, it would be critical to spend more time trying to figure out the functionality for blink. One way to do this would be to write code that would exhaustively test all bytecode values and return only the valid opcodes. From there, we could use process of elimination to validate what the correct opcode for blink is. Every repository we found uses '\x16' for the blink opcode, however the code we wrote corresponding to that opcode never executes. Enabling blink recognition would allow for a whole new group of features, as blinks could be grouped into single blinks, double blinks, and triple blinks, and offer an asynchronous control functionality that the attention and meditation simply aren't responsive enough to support.

Another good option would be to move away from the more commoditized hardware, and use the OpenBCI tool to collect EEG data from the brain. This would open up possibilities for spatial and orientational recognition in the brain, as well as a variety of other brainwaves available in lobes inaccessible via the MindwaveMobile.

Conclusion

While this project can be improved in many ways, our goal of attempting to control a "wheelchair" was satisfied. We found that it is possible to control a prototype with the mind (specifically EEG & EMG) using a Mindwave Mobile headset. We were able to connect the Mindwave to the Raspberry Pi and received meditation and concentration values back. However, we were not able to receive blink values from the headset. This limited our toggling functionality for different directions. Even so, using the Mindwave, we

successfully controlled the GoPiGo in a singular forward direction. The ability to control a GoPiGo via a Mindwave serves as a proof of concept and shows that building a brain controlled wheelchair is feasible. In the future we plan to move away from using the Neurosky Mindwave, since it is very limiting in its functionality and available interfacing software. We would like to add multi directional movement through the toggling of different frequencies of blinking. Lastly, we would like to thank all of the contributors of the countless GitHub repositories that we built upon. Without their contributions and code, we would not have been able to connect the Mindwave and the Raspberry Pi together.

References

1. https://www.brainandspinalcord.org/quadriplegia/
2. https://www.howitworksdaily.com/how-stephen-hawkings-wheelchair-works/
3. Mdbelal Bin Heyat, Mohd Maroof Siddiqui, “Recording of EEG, ECG, EMG Signal”, *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 5, Issue 10, October 2015.
4. http://developer.neurosky.com
5. https://www.raspberrypi.org
6. https://www.dexterindustries.com/gopigo3/
7. https://github.com/lihas/NeuroPy
8. https://github.com/robintibor/python-mindwave-mobile
9. https://www.quantumrehab.com/quantum-electronics/q-logic-3-advanced-drive-control-system.asp
10. https://www.openbci.com