Implementation of a Mind-Controlled Wheelchair

Garrett Stoyell, Anthony Seybolt, Thomas Griebel, Siddesh Sood, Md Abdul Baset Sarker, Abul Khondker, Masudul Imtiaz
Department of Electrical and Computer Engineering
Clarkson University
Potsdam, NY
stoyelgm@clarkson.edu

***Abstract*—The application of a brain-computer interface to control an electric wheelchair may enable individuals with impaired motor skills to move without the need for any physical input. The goal of this project was to develop such a wheelchair and provide some level of independence to individuals suffering from ALS or similar physical issues. The overall design and implementation are presented in this paper. The integration of the proposed system utilizes a Drive wheelchair, EMOTIV EPOC X headset, BLE 5.0 adapter, Raspberry Pi development board, Sabertooth 2x32 dual motor controller, LM2596 buck converter, and a 24V battery. The construction, as well as limitations of our final system, are discussed; alternate designs, as well as future improvements, are explored.**

I. INTRODUCTION

The aim of this study was to develop a fully automated brain controlled smart wheelchair embedded with hands-free control technology with the aim of assisting people with severe physical disabilities. This research is substantial as this is directly related to a large vulnerable population. According to the World Health Organization, 15 percent of the world's population lives with some form of physical disability [1]. The CDC estimates, in 2020, 26 percent of the US population, i.e., one in every four adults, to have a disability. Of those, 13.7 percent are considered to have a mobility disability [2]. This muscular degeneration may cause depression, significant decreases in motivation, and loss of independence for many sufferers. There are a few mobility-assisting devices available in the market, such as powered wheelchairs. However, the control systems of these devices require significant levels of skill, attention, and judgment from the user. Without adequate control over the wheelchair, the risk of accidents and collisions increases, causing damage and injury. Hence, the primary research motivation was to give those people their autonomy back, allowing them to move when and where they want without requiring assistance from others.

The initial requirement of this project was to develop a mind-controlled wheelchair utilizing the EMOTIV EPOC X headset. The EMOTIV EPOC X headset is a cost-effective 14-channel mobile EEG brainware device whose main purpose is to provide professional-grade brain data which can be used for contextual research [3]. The headset provides coverage of the frontal, prefrontal, temporal, parietal, and occipital lobes, which can be used for classification. A custom Drive wheelchair was modified such that the EMOTIV headset captures brain waves (EEG signal) from a person sitting in the wheelchair as a means of guiding navigation. The main scope of this project was to serve those who have had their motor skills severely limited in some fashion. One such audience would be those who have had their lives irreversibly impacted by ALS, a disease that attacks motor neurons and can severely hinder voluntary movements such as walking [4]. After the user is trained with the system, the proposed system promotes mobility using just brainwaves and does not require any other physical input from the user, thus providing some level of independence to these individuals. Ultimately, this research is the continuation of the research published in [5] and [6].

The following section will describe how the final implementation of the wheelchair complies with the original design requirements. Test results on the system's basic functions and performance will also be discussed.

II. METHODOLOGY

A. System Design Specifications and Implementation

The original concept document for the mind-controlled wheelchair listed multiple requirements that the client wanted the chair to adhere to. While the design does not meet all these specific requirements, alternate designs are used to achieve the same goals. These specifications and constraints are listed as follows:

- The final system may only be controlled using the EMOTIV EPOC X headset. No displays or other control methods, such as a keyboard or joystick, may be used to steer the chair.
- The system must receive brain waves and produce corresponding navigational commands.
- A robust classification model developed through extensive training must be implemented.
- The wiring of the final version of the wheelchair should be as clean and safe as possible.

In the final implementation, the wheelchair adheres to all of these except for the use of a classification model, though a substitute was used to achieve similar functionality.

1) Navigational Control System: First, the brainwave from the EPOC X headset is the only input used to control the chair. Previously-installed control methods are removed from the chair, and the user can stop the chair or steer it left, right, or forward using only mental commands sent through the headset. To achieve directional behavior, we created a script, navigationScript.py that executes on a Raspberry Pi 4 and converts directional commands received from the classification module of the project into motor control signals. The signals are sent to a Sabertooth 2x32 motor driver [7] via a UART connection, which will output voltages to each wheelchair motor to produce the desired movement [8].

2) Brainwave Classification: The wheelchair can also translate brain waves into navigational commands. Originally, this was going to be achieved using the classification model. But, because it was not reasonably possible to read data directly from the headset without needing to use EMOTIV's software, the classification model was forgone in favor of using the EMOTIV Cortex API [3] since it serves the same basic function. In the system, a desktop with the EPOC X headset's USB 3.0 Bluetooth receiver inserted and EMOTIV software installed runs the trainNew.py classification script. This script uses EMOTIV's libraries to convert headset signals into a new classified directional command every half-second interval. Directional commands are sent to the Raspberry Pi using a UDP (User Datagram Protocol) server connection.

3) Clean and Safe Wiring: The last major system requirement was also met in that wiring is both clean and safe. As shown in the top-level system design in Figure 1, there is minimal wiring involved to make the chair function. Communications between the headset and the desktop and between the desktop and the Raspberry Pi are entirely wireless. Additionally, only one physical signal (being the Raspberry Pi's UART Tx signal) is being sent between the Pi and the Sabertooth since the Pi does not need to receive feedback from the Sabertooth [9]. Thus, all that remains of wiring on the chair are the 24V power connections to the Sabertooth main power input, the 5V power connections to the Raspberry Pi, and the motor connections to the Sabertooth. The 24V connection to the main power input allows the motors to receive voltages ranging from -24V to 24V from the Sabertooth depending on the motor control values outputted by navigationScript.py. To ensure the Pi receives the 5V 3A required for it to operate, an LM2596 buck converter capable of providing 3A is tuned to output 5V.

To ensure all electrical components are safe, all exposed solder points and wirings have been covered using heat shrink wraps and protective cases for the Raspberry Pi and LM2596 buck converter. These protective cases have proven to be effective in protecting and cooling the two components.

B. Additional System Capabilities

While not outlined in the design specifications, certain features and behaviors exist in the system to enhance the user experience and benefit system performance.

1) Rotational and Turning Movements: For example, navigationScript.py uses a boolean variable called isMovingForward to toggle wheelchair movement between two states: forward movement and stationary. If the wheelchair command is detected as "push", the chair will continuously move forward and can make forward left or forward right movements if "left" or "right" commands are received, respectively. If "pull" is received, the system will stop all movement. If "left" or "right" are received in this stopped state, the wheelchair will rotate left or right, respectively. By introducing the isMovingForward

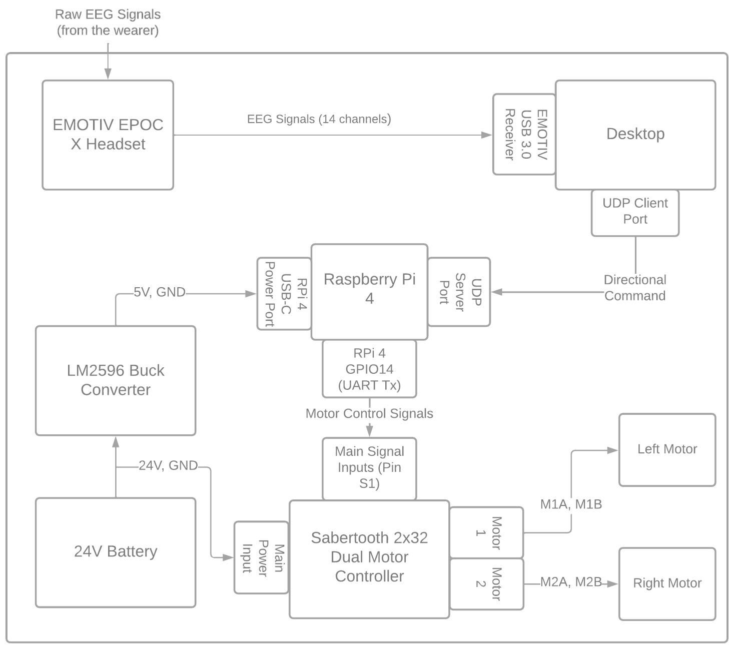


Fig. 1. Top-Level System Design

variable, the wheelchair can perform six types of wheelchair movements under the limitation of four commands, which gives the chair more maneuverability.

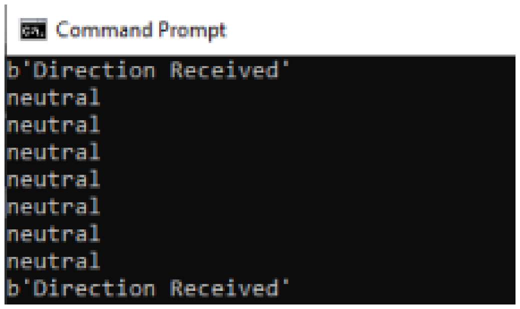
2) Steady Movement Changes: The wheelchair also has the ability to ease directional changes into each other. Because the system only has "push" and "pull" commands corresponding to moving at a hard-coded speed and stopping, there are no commands to adjust the speed of the chair. As such, abruptly stopping the chair after moving could be dangerous to the user. Thus, in the time between receiving directional commands, the chair will steadily ramp up or ramp down its original speed to the new one. Thus, the user will have a smoother experience while operating the chair.

3) The Mode of Commands Over a Half-Second Interval: The EMOTIV software outputs new directional commands at approximately 18 samples per second, which is a number that was derived from observations during testing. If the training for the mental commands is weak, it is possible that multiple directions may be sent to the chair and individually processed within the span of a second. Since this would cause the chair to have shaky movements, it was decided that the most frequent command detected by the headset over a period of time would reduce the likelihood of undesired chair movement and give the user a smoother experience. In the final build, a half-second interval was chosen since it gives the user a reasonable reaction time.

To implement this half-second interval, trainNew.py uses Python's time function to test if a half-second has passed since the last command was sent to the Raspberry Pi. If not, directional commands are continuously appended to a list. If a half-second has passed, the script will determine the most frequent command in the list, send that command to the Raspberry Pi, and clear the list for the next interval. Then,

TABLE I
TABLE OF CODES USED WITHIN THIS PROJECT

Code Name	Purpose
navigationScript.py	Converts directional commands into motor control signals
trainNew.py	Collects live directional commands from the user and sends them to navigationScript.py



```
b'Direction Received'
neutral
neutral
neutral
neutral
neutral
neutral
neutral
b'Direction Received'
```

Fig. 2. Sample Output of trainNew.py in a Terminal

navigationScript.py uses Python's sleep function to receive and execute new commands every half second. All of this can be seen in Figure 2, where a number of commands are classified over a half-second, but only one command is received from navigationScript.py during this time. Table I lists each of the two codes used in the implementation of this project and their respective purpose. The project is open-source and can be accessed at https://github.com/gstoy7/mind_controlled_wheelchair

C. Unique Design Innovations

1) UDP Server: In order to establish communication between the PC that is receiving EEG data and the Raspberry Pi onboard the wheelchair, a UDP server was hosted on the Raspberry Pi. UDP allows for the transfer of messages between devices over the internet. In this case, the Raspberry Pi received strings from the desktop PC [10]. A UDP server was chosen over a TCP protocol because it is much faster and more efficient [11]. This will allow the system to respond faster to input and make the experience feel more fluid to the user. The speed and efficiency of UDP does result in some packet loss which will result in some lost transmissions, but this is believed to be unimportant for this design. [12]

2) Cortex API: Cortex is EMOTIV's API for integration of the EMOTIV EPOC X Headset with custom applications. The basis around Cortex is training profiles that can be used to train and store mental actions using the EPOC X [13]. This was used by the group as an achievable alternative to building a new classification model for EEG signals. The Cortex API provided a simple solution to this problem and allowed easy integration of the EPOC X Headset into the necessary applications. Using the API, the group developed an application for training mental commands and a live mode app which used the trained profile to decipher incoming mental commands. An example profile is shown in Figure 3. The more spread out each command is trained in the graphic, the more unique they are, meaning the API will not misclassify them for each other as often. Both applications were run on the computer because Cortex integration on the Raspberry Pi was still in beta and proved unreliable.

D. Design Limitations

1) EMOTIV EPOC X Weaknesses: All things considered, the EMOTIV EPOC X headset along with its necessary software was one of the major limitation weaknesses of our project. We found that the headset had prominent consistency issues, as it produced imprecise and unreliable commands. Throughout the project, we tried several different methods of training the headset to differentiate the mental commands; however, it was very difficult to get the commands to any degree of reliability. In the end, we found that associating a directional command with a physical action helped increase the degree of consistency. For example, we found that if a user were to clench their teeth, the headset would be more capable of identifying the action. This admittedly is not an ideal solution since this project's target is users with some level of physical disability.

Another limitation of using the EMOTIV headset for this project was that it essentially anchored us to the laboratory location due to its necessary software not being able to work on a Raspberry Pi. EMOTIV's online documentation suggested that the applications were able to be downloaded to a Raspberry Pi, but we found that not to be the case after multiple failed attempts to install the software. Due to this, we had to keep the wheelchair's UDP server in close proximity to our desktop client being used to send navigation commands.

2) UDP Server Weaknesses: Our UDP server implementation also came with one prominent drawback, being the IP address refreshing on a regular basis. We attempted on several different occasions to assign our Raspberry Pi a static IP address, but were unable to do so. As such, the IP address of our server refreshes regularly, and because of this, the user needs to update the IP addresses in 'navigationScript.py' and 'trainNew.py' whenever a new IP is automatically generated.

E. Future Improvements

Ultimately, we were able to meet all of our design specifications for the mind-controlled wheelchair project. That being said, the findings of this project provide a strong proof of concept which can be used for future improvements. For example, the existing classification system is capable of working with one trained user at any given time; future studies could look to make a more representative model so that training profiles have a higher degree of reusability. In order to realize this idea, future research would likely need to move away from the Cortex API and use machine learning algorithms to derive a classification model that has the capabilities of being used with multiple users. Other future optimizations to the system would most likely consist of addressing the weaknesses outlined in the previous section.

TABLE II
SOFTWARE TESTS AND RESULTS

Test Name	Test Description	Test Result
Navigational Command Script Logic Test	Checks the logic sequence of the navigational Python script. This test must verify that the navigational script produces the expected output based on each possible directional input.	PASS
UDP Server Connection Test	Check that the UDP server established on the Raspberry Pi can send and receive data to/from the desktop client.	PASS

III. TEST RESULTS

A. Software Testing

Table II details the procedures used in the testing of our software system. All aspects of the testing which involved the EMOTIV EPOC X headset were completed across 3 trials. Each trial consisted of a training period, as well as a test using EMOTIV's built-in live mode functionality. Our tests of the software system broke down into two phases: the navigational script testing and the UDP server testing. In order to test the navigational script, we utilized a counter and a while loop in order to test every possible directional command to ensure the voltages output across UART Tx were correct. We were successfully able to demonstrate that our navigational script produced the expected output based on each possible directional output. From there, we looked to test our ability to send and receive data across the UDP server. In order to test this, we established the server on our Raspberry Pi, connected to it with our desktop client and attempted to send a sequence of commands. Ultimately, this test was successful as well as the Pi was able to send and receive data reliably.

B. Hardware Testing

Table III details the procedures used in order to test the components of our hardware system. We first performed a test on our Sabertooth 2x32 dual motor driver as a means of ensuring that it could successfully provide motor signals to each of the two DC motors. In order to perform this test, a series of scripted commands were sent across UART Tx from the Raspberry Pi to the Sabertooth's UART Rx port. We found that the Sabertooth was able to reliably receive these directional commands and produced the desired output. From there, we performed electrical tests on our LM2596 Buck Converter in order to ensure that it was providing the correct amount of voltage to our Raspberry Pi. The buck converter used in this project needed to be able to derive 24V of voltage from our batteries and in turn supply our Raspberry Pi with 5V of voltage and 3A of current. In order to achieve this, we needed to use the buck converter's built-in potentiometer to tune the output voltage down to 5V. We then tested the output with a digital multimeter and found that the buck converter was able to reliably provide our Raspberry Pi with the desired voltage and current.

TABLE III
HARDWARE TESTS AND RESULTS

Test Name	Test Description	Test Result
Sabertooth 2x32 Dual Motor Driver Integration Test	Checks functionality of Sabertooth 2x32 dual motor driver to ensure that it can successfully provide motor signals to each of the two DC motors.	PASS
LM2596 Buck Converter Unit Test	Checks that the buck converter is providing the Raspberry Pis with the correct voltage/current	PASS

TABLE IV
SPECIFICATION TESTS AND RESULTS

Test Name	Test Description	Test Result
Full System Test	Checks functionality of entire system to ensure that wheelchair displays the desired behavior when receiving brainwaves from the the EMOTIV EPOC X headset	PASS

C. Specification Testing

Table IV provides a brief overview of the procedure undergone in order to test our final system. This final test looked to check overall functionality of the system in order to ensure that the wheelchair displayed the desired behavior when receiving live commands from our EMOTIV EPOC X headset. In order to perform this test, we had to first make sure the EMOTIV headset was turned on and connected its desktop receiver. From there, we plugged in the Raspberry Pi and powered on the system. Upon startup, the Raspberry Pi would establish the UDP server, so the next step was to connect the desktop client and begin sending directional commands to the wheelchair. We found that the wheelchair was able to display the desired behavior, and ultimately, the system meets full specification.

D. Performance Testing

Table V provides a brief overview of each of the performance tests used in the optimization of the system. The

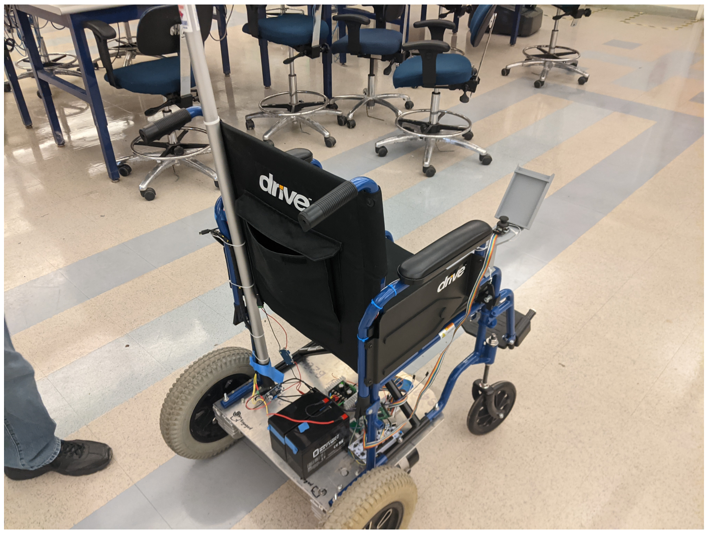


Fig. 3. The Customized Wheelchair after Motor and EMOTIV Integration

Test Name	Test Description	Test Result
Speed Ramping Tests	Ensure that the wheelchair eases between states instead of making sudden speed changes.	PASS
Cortex API Reliability Testing	Check the responsiveness of Cortex API live mode when using easily differentiable physical commands.	PASS

TABLE V
PERFORMANCE TESTS AND RESULTS

first performance based test we performed was on our ability to ramp our speeds to provide smoother transitions between states. For example, if a user transitioned from going forward to a stopped state, a ramping function would slow the motors down instead of instantly stopping the system. All things considered, this test was successful and provided significant improvements to the overall feeling of state transitions. The other major performance goal we wanted to achieve was to have a very responsive interface with the Cortex API. At an earlier stage of the project, we had an issue with compounding lag, where the API's live mode was at points able to build a 5-10 second lead on the commands being output to the Raspberry Pi. As a means of addressing this issue, within 'trainNew.py', we had to make changes in order to ensure that the system would never fall greater than half a second behind the live mode. Once we made these changes to the script, we found that the system was far more responsive and ultimately met our performance goals. The final system is shown in Figure 4.

IV. CONCLUSION

The outcome of this project was a mind-controlled wheelchair utilizing the EMOTIV EPOC X headset. A custom Drive wheelchair was modified such that the EMOTIV headset captures brain waves from a person sitting in the wheelchair as a means of guiding navigation. Future work of this project will target to fine-tune all parameters and enable the wheelchair for mass implementation.

REFERENCES

[1] "Disability and health." https://www.who.int/news-room/fact-sheets/detail/disability-and-health.

[2] CDC, "Disability impacts all of us infographic — cdc." https://www.cdc.gov/ncbddd/disabilityandhealth/infographic-disability-impacts-all.html.

[3] "Epoc x user manual." https://emotiv.gitbook.io/epoc-x-user-manual/.

[4] D. Murrell, "All about amyotrophic later sclerosis (als)." https://www.medicalnewstoday.com/articles/281472l.

[5] E. Sola-Thomas, M. A. Baser Sarker, M. V. Caracciolo, O. Casciotti, C. D. Lloyd, and M. H. Imtiaz, "Design of a low-cost, lightweight smart wheelchair," in *2021 IEEE Microelectronics Design Test Symposium (MDTS)*, pp. 1–7, 2021.

[6] M. Caracciolo, O. Casciotti, C. Lloyd, E. Sola-Thomas, M. Weaver, K. Bielby, M. A. B. Sarker, and M. H. Imtiaz, "Autonomous navigation system from simultaneous localization and mapping," *CoRR*, vol. abs/2112.07723, 2021.

[7] "Sabertooth 2x32 manual." https://www.dimensionengineering.com/datasheets/Sabertooth2x32.pdf.

[8] "Raspberry pi 4 model b datasheet." https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf., 2019.

[9] "Uart - universal asynchronous receiver/transmitter." https://pinout.xyz/pinout/uart.

[10] Gus, "How to handle raspberry pi serial reading and writing." https://pimylifeup.com/raspberry-pi-serial/.

[11] L. Williams, "Tcp vs udp: Key difference between tcp and udp protocol." https://www.guru99.com/tcp-vs-udp-understanding-the-difference.html/.

[12] D. Industries, "Five ways to run a program on your raspberry pi at startup." https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/.

[13] "Emotiv cortex api user manual." https://emotiv.gitbook.io/cortex-api/bci.