Brain Controlled Robotic Arm

Arunagiri Adhithian R
New York University
Tandon School of Engineering
New York City, USA
ar7404@nyu.edu

Tzipora Ziegler
New York University
Tandon School of Engineering
New York City, USA
tzipora@nyu.edu

Sudharsan Ananth
New York University
Tandon School of Engineering
New York City, USA
sa6788@nyu.edu

***Abstract*—Mobility-impaired individuals are often unable to do simple physical tasks, such as raising a glass of water to their mouth, without relying on someone else. As a way to solve this problem, this project introduces a new robotic arm device that can be controlled using brain waves instead of requiring physical movement. The device gives users the ability to raise and lower a glass of water by simply increasing and decreasing their level of concentration or meditation.**

I. INTRODUCTION

A mobility impairment is a disability caused by a reduced range of motion or complete loss of motion in certain parts of the body. Such impairments often prevent individuals from completing simple tasks on their own that require physical movement, such as raising a glass of water to their mouth. As an alternative to requiring physical movement, this project proposes a simple and cost effective solution that instead uses brain waves to control the movement of physical and mechanical components.

This project introduces a robotic arm prototype that uses a brain computer interface (BCI) to control the arm movement. An EEG sensor is used to collect brain activity data, which is processed and incorporated into an algorithm that determines when the arm should move up and down.

An EEG, or electroencephalograph, is a test that measures electrical activity within the brain. Brain cells communicate via electrical impulses. Electrodes placed on specific sites on the scalp can detect and record these impulses within the brain.

Brain waves occur at different frequencies. This project uses a NeuroSky EEG chip which is able to differentiate between eight different brain waves: delta, theta, low alpha, high alpha, low beta, high beta, low gamma and high gamma. The wave data can be used to determine if a person is in a meditative or focused state. When the person's brain reaches a meditative state, a signal is sent to the arm to move up.

All components and circuits used in the prototype are chosen carefully based on the cost to ensure that this could one day become accessible to people in-need and could develop into a commercially viable product.

II. DETAILED DESIGN

A. *Mechanical Design*

1) *Arm Structure and Gear Mechanism Design*

a) Design structure of arm: The arm is primarily designed in Fusion 360 to lift a cup of water approximately 0.5 - 0.7 lbs with the usage of the power transmitted from the servo motor. The arm was designed with 100 % infill to withstand the weight of the cup with the water present in it.

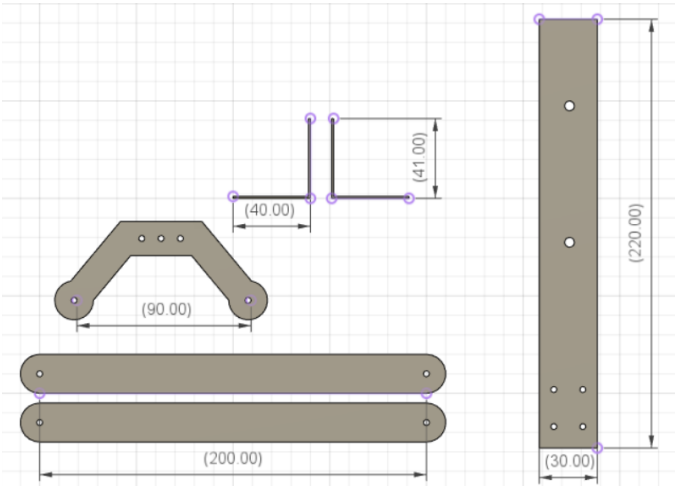


Fig. 1. Design of disassembled arm in Fusion 360 (all dimensions are in mm).

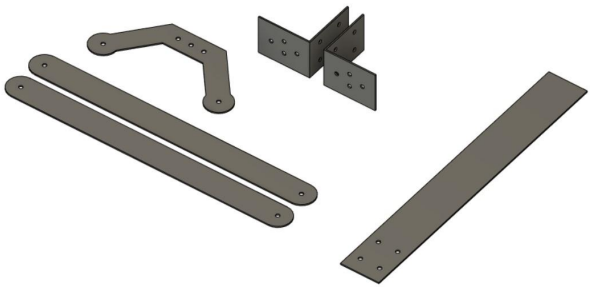


Fig. 2. 3D view of the links with 5 mm thickness.

b) Gear mechanism and design: The gear is designed considering the rotational speed between two meshing gears to lift the cup with the torque transmitted from the gear. The servo motor specification was identified as 30 kg having a rpm of 55.55. Considering the workspace of the arm, 120 degree rotation from home position is required. Thus, the motor's rpm must be reduced to ~23 rpm. So, concluding that a 1:2 gear ratio would be optimal.

The large gear and small gear have diameters of 24 mm (16 teeths) and 48 mm (32 teeths) respectively according to the ratio matching the holes of the main frame of the arm. A removable shaft was designed to connect the servo motors end effector to small gear.

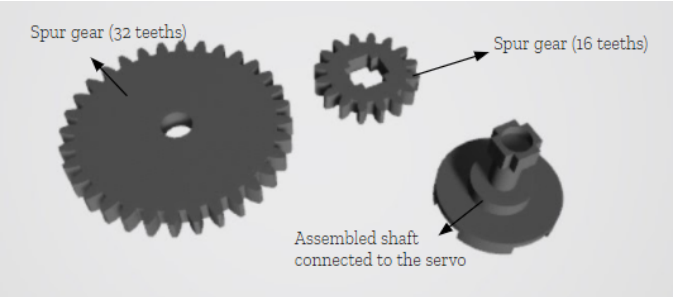
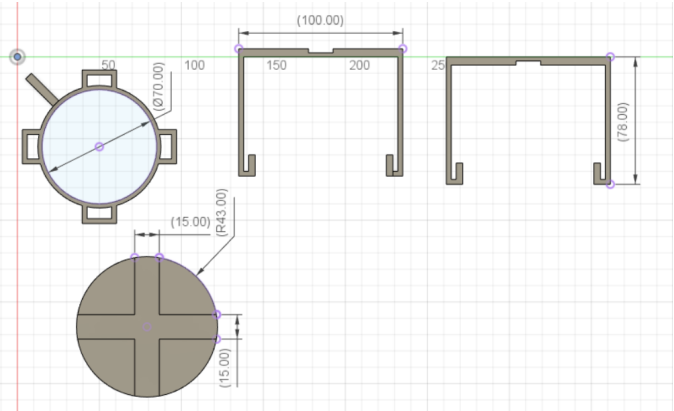
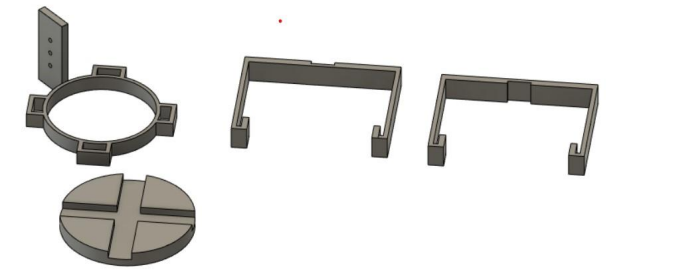


Fig. 3

c) Cup holder design: The cup holder is designed to hold cup diameter ranges upto 10cm. It is designed in a way that it can easily be 3D printed and assembled together without any fixing elements.



(a)



(b)

Fig. 4. (a) and (b) Unassembled parts of the cup holder.

d) Assembly of arm and gear mechanism: The parts are attached together using m4 and m6 screw sets based on the joints. The arm and servo stand are mounted on a wooden base.

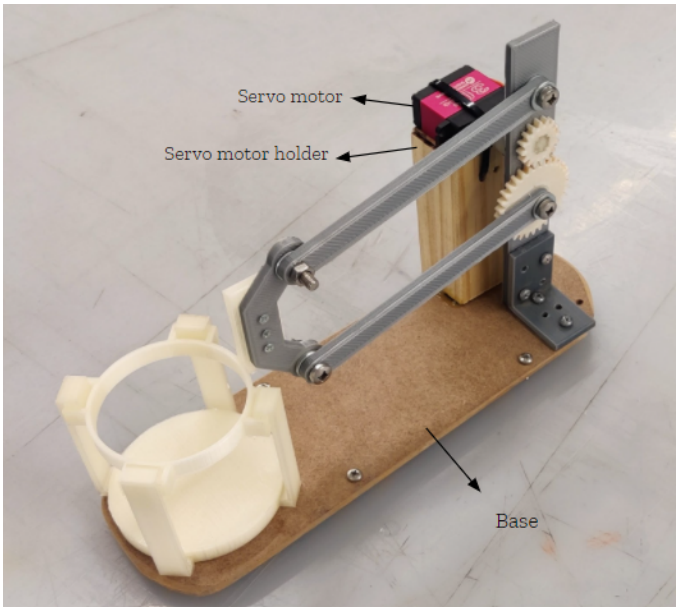


Fig. 5. Assembled arm structure.

The screws on the surface of the base are used to elevate the arm for a particular distance from the ground so that screws fixed with the arm to base do not till the structure. The servo stand is glued to the base in order to withstand the force exerted by the servo rotation. The servo is zip tagged to the stand to eliminate its movement from optimal position.

B. *Electrical Design*

1) Circuit: The system consists of two main circuits connected using radio signals.

The first circuit begins with three electrodes, one used to measure EEG activity in the brain and the other two used as reference electrodes. The electrodes send analogue signals to the NeuroSky board which amplifies and conditions the signals. The board then converts the signals into a digital serial signal (TX).

Once the data is in a format other microcontrollers can process, the TX data is routed to a 6n138 optocoupler to isolate the NeuroSky board from the Seeeduino microcontroller. In addition to improving the stability of the serial signal, this serves as a safety measure to ensure that no power enters the NeuroSky device and electrodes when the microcontroller is connected to a laptop or another external power source. The signal from the optocoupler is inverted, hence a PNP transistor is used to invert the signal back to normal PWM serial signals.

The digital signal is then fed to the RX port of the Seeeduino XIAO - SAMD21 Cortex M0 microcontroller. The microcontroller is connected to a nRF24L01 (near Radio Field) module, which is a wireless radio frequency transmitter operating at 24Ghz. There is also a red LED connected to pin 3, which is programmed to indicate bad signal when electrodes are not placed properly.

The entire circuit is designed carefully to be as compact as possible, as this entire system is attached directly to the Mindflex headset. The circuit is made using a small breadboard for easy prototyping.

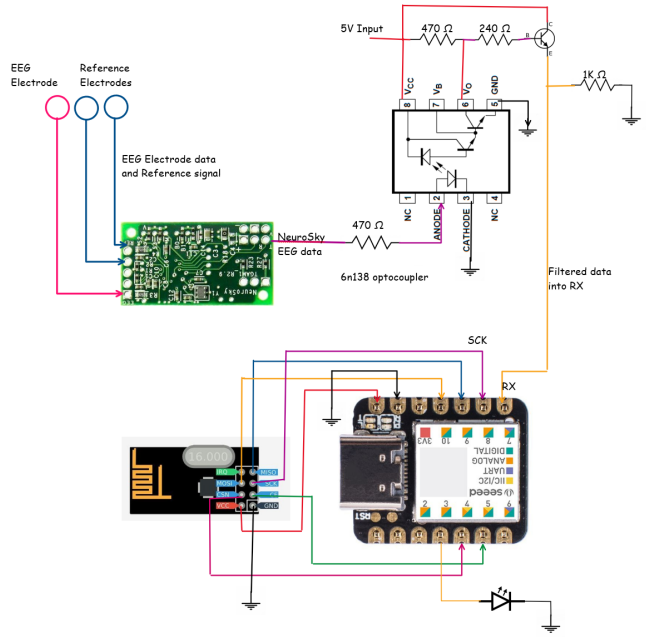


Fig. 6. Brain signal processing circuit

The second circuit, the receiver circuit, starts with a nRF24L01 module connected to an Arduino Uno microcontroller. The microcontroller receives the wireless data and then sends a signal to the attached servo, which actuates the robotic arm. We went for a 30Kg DS3230MG servo motor to ensure the arm has sufficient torque. As an extra precaution, we also used a waterproof servo since the device handles water. This circuit is powered by an external power supply or 4xAA batteries to meet the energy requirement of the servo motor.

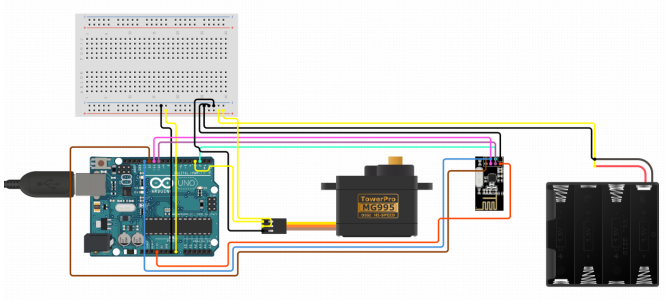


Fig. 7. Receiver circuit

2) Headset Hack: One of the first questions we had to figure out was how to get EEG data. After doing extensive research, we discovered a toy called Mindflex [1]. The Mindflex headset was developed as part of a game where users can control a ball by changing their levels of concentration. The toy contains the same board that NeuroSky uses in some of their more advanced EEG headsets, and it collects proper EEG data. Using a blogpost [2] we found as a base, we were able to hack into the toy and successfully extract usable EEG data.

The first step was to disassemble the Mindflex headset to access the NeuroSky board located inside. The electrodes from the headset connect to the EEG pins on the board. The EEG chip then processes the raw data, so all we needed to do was establish a connection to the chip's TX pin to extract the processed EEG data. As a final step, we added a connection to the headset's ground to create a common ground between the NeuroSky and Seeeduino boards.

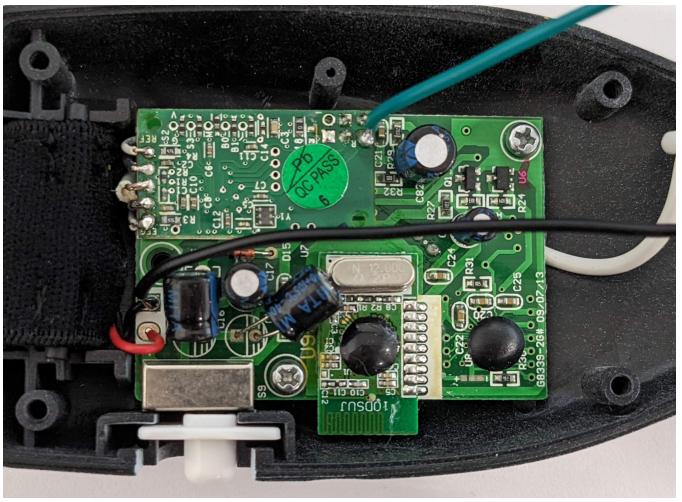
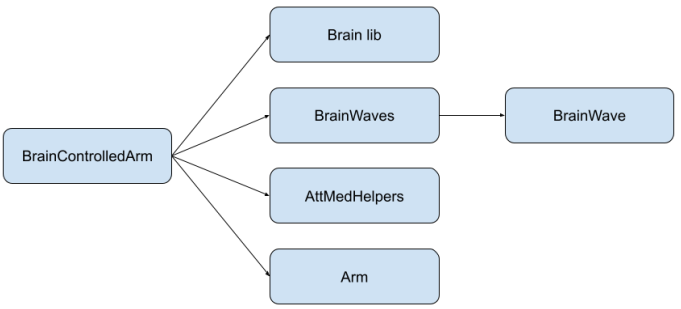


Fig. 8. View of the NeuroSky board with wires soldered to the TX pin (teal wire) and ground (black wire).

C. *Software Design*

All source code used in this project can be found at https://github.com/tziporaziegler/brain-controlled-arm. (The code is currently closed-source and viewers need to be directly added as contributors, yet we are working on making it open-source.)

1) Code Architecture



a) BrainControlledArm: The program starts with the BrainControlledArm class, which contains the main Arduino code. This class serves as a controller that coordinates between the many electrical and software components.

As a first step, a hardware as well as software serial connection is established in order to receive data from the headset as well as be able to send data to the graph visualizer (see 'Visualization' section below) and Arduino serial monitor.

The main job of the BrainControlledArm class is to listen for new brain wave data and react whenever a new packet is detected. It accomplishes this using the Brain library (see 'External libraries' section below). The program has a built-in 3 second delay to allow for the EEG signal to settle before data processing begins. Each packet contains a connection quality indicator, and we chose to ignore any packets that came in with weak signals in order to reduce the amount of incorrect or misleading data.

In order to determine when the arm should move up or down, a meditation and attention threshold variable is calibrated to each user. As new data rolls in, it gets passed to a BrainWaves object which updates the individual waves as well as recalculates new averages of recent data. While the library we built supports numerous ways of calculating and determining if the user is in a meditative or focused state, we ended up using the NeuroSky board's processed mediation and attention values in the final prototype since those numbers were more stable and easier to control in environments such as a full classroom.

It's quite common for different external factors to trigger brain activity spikes. In order to reduce noise caused by such spikes and allow for a better user experience, the program always looks at multiple packets before making any decisions. If two consecutive packets are above the threshold, the Arm object's function to move up is invoked. If the packets are both below the threshold, the Arm object's move down function is called. If one packet is above or below the threshold, yet the following one is not, the packets are discarded and the system does nothing except wait for the next set of data.

Once the arm moves up to drinking height, a 5 second delay is added to ensure the user has enough time to drink before the arm possibly moves again. The delay length is something that will need to be calibrated to individual users, since some users will require extra time to take a proper drink.

b) BrainWaves

```
class BrainWaves {
  private:
    BrainWave waves[8] = {("delta"), ("theta"),
("low alpha"), ("high alpha"), ("low beta"), ("high
beta"), ("low gamma"), ("high gamma")};

    double recentPercent1 = 100;
    double recentPercent2 = 100;
    double recentPercent3 = 100;

    double percentThreshold;

    void updateRecentNumWavesBelowThreshold(int
newVal);
    void updateRecentPercentVals(double newVal);
    double getRecentPercentAvg() ;
  public:
    double recentNumElementsBelowThreshold1 = 0;
    double recentNumElementsBelowThreshold2 = 0;
    double recentNumElementsBelowThreshold3 = 0;

    BrainWaves(double percentThreshold);

    void update(unsigned long* powerArray);
    double getNumElementsBelowThresholdAvg();
    bool percentAvgAboveThreshold();

    bool recentPercent1BelowThreshold();
    bool recentPercent2BelowThreshold();
};
```

The BrainWaves class holds an array of the eight BrainWave objects. When a new packet is detected, the update function is called, which updates the individual waves as well as calculates new recent averages and number of items above or below the threshold.

c) BrainWave

```
class BrainWave {
  private:
    const char* label;
    unsigned long latestVal;
    unsigned long maxVal = 0;
  public:
    BrainWave(const char* label);

    void update(unsigned long newVal);

    const char* getLabel();

    double getPercent();
    void printPercentStr();
};
```

One way to determine if a user is in a meditative or focused state is by looking at recent wave values and determining their percentages by comparing them to the corresponding max values detected for each specific wave. Every individual has different max wave values, so each BrainWave object keeps track of the max value detected during the particular session, and continues to compare and update the value as new values are processed. The current system uses an ongoing calculation process, yet it may be possible to make this more efficient in the future by personalizing the max values once for each user during a one-time longer calibration session.

d) AttMedHelpers

```
class AttMedHelpers {
  private:
    int recentAttentionVal1 = 0;
    int recentAttentionVal2 = 0;
    int recentAttentionVal3 = 0;
    int recentMeditationVal1 = 0;
    int recentMeditationVal2 = 0;
    int recentMeditationVal3 = 0;
  public:
    AttMedHelpers();

    void updateRecentAttentionVals(int newVal);
    void updateRecentMeditationVals(int newVal);

    double calculateRecentAttentionAvg();
    double calculateRecentMeditationAvg();
};
```

AttMedHelpers is a simple helper class used for calculating different averages based on the NeuroSky board's attention and meditation values. The idea is to use averages of data from multiple packets instead of a single packet to reduce the amount of noise caused by brain activity spikes.

e) Arm

```
class Arm {
  private:
    RF24 radio;
    int currentServoAngle = 0;
    const int MIN_HEIGHT_STEP = 0;
    const int MAX_HEIGHT_STEP = 10;
    int currentHeightStep = 0;]
    bool stopped = false;
  public:
    Arm();

    void moveUp();
    void moveDown();
    void stay();

    void start();
    void stop();
    bool isStopped();
};
```

The Arm class is responsible for sending signals to the arm to move it up or down, and keeping track of the arm's current position. When the functions to move the arm up or down are evoked, the system uses a RF24 radio connection to send different angles to the servo motor.

2) External Libraries

a) Brain: Brain [3] is an Arduino Library for parsing data from Neurosky-based EEG headsets. It's designed to make it simple to access processed brain wave information. It converts the NeuroSky board's EEG data into something any C++ or Arduino class can understand.

The library continuously listens for updates from the headset. Whenever a new packet is detected, the brain wave data is broken down into different variables that are easy to access and use.

The library has a useful function that returns a comma-separated ASCII string of all the values. This string can easily be printed to the Arduino serial monitor for debugging purposes.

Example:

```
[0,63,98,321806,30780,145895,21233,13970,
15215,4623,7774]
```

The first number represents the signal strength. The signal ranges from 0 - 200, with a lower number indicating a stronger signal. Ideally, the signal should always be 0. In order to ensure that we were only reading proper wave data, we chose to ignore any packets that didn't have a 0 signal strength.

The next two numbers represent attention and meditation values calculated on the NeuroSky board. The values range from 0 - 100, with higher numbers representing higher levels of attention or meditation. These numbers are calculated using a combination of the eight brain waves, although NeuroSky has chosen to keep the algorithm proprietary.

The final eight values correspond to the eight brain waves being detected. Each number represents the amount of electrical activity detected for the wave's corresponding frequency.

3) Visualization: In order to make it easier to test and experiment with the headset brain data, we used the BrainGrapher [4] visualizer. BrainGrapher is a Processing [5] application designed to graph changes in brain waves over time. The library was last updated in 2016, so we needed to modify it a bit to get the visualizations working again.

The graph shown below in Fig. 9 shows actual data from one of our experiments. You can see the connectivity quality, packet number, attention and meditation values, as well as the eight brain wave values.

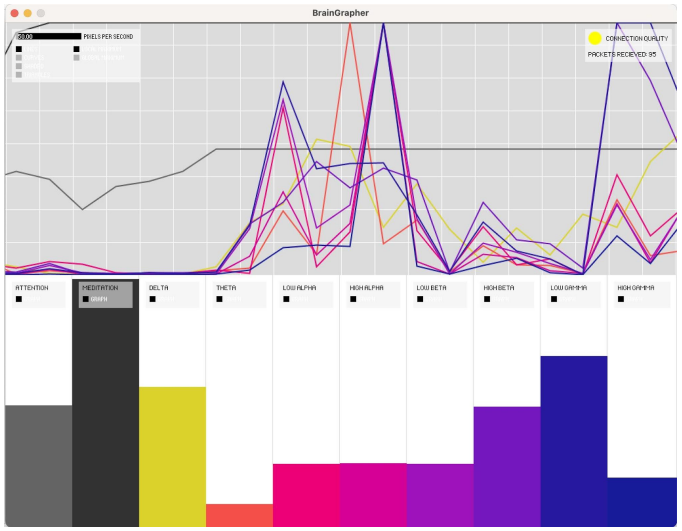


Fig. 9 . Visualization of live brain wave data.

4) Experimentation: One of the biggest challenges we faced was getting steady EEG data that clearly indicated when a user was in a meditative or focused state. Spikes in brain activity can easily occur, and without proper training, it can be quite difficult to relax or focus your mind on command. When working with brain-controlled interface devices, users often spend months training to gain better mind control.

In order to get as accurate a signal as possible, we experimented with many different ideas for the algorithm including:

- Moving the arm up when the user was focused instead of when they were meditating. While we were able to determine some activities that consistently increased the level of brain activity (such as drumming multiple complex sequences), we determined that most users had an easier time controlling their meditation state.
- Calculating the average of all 8 brain waves and seeing if that number was below a specified threshold.
- Checking to see if a certain number of waves were below the threshold. We experimented using 4 - 7 waves.
- Calculating averages by combining data from 2 - 4 packets.
- Using the attention and meditation values provided by the NeuroSky board in combination with the raw wave data.

In addition to experimenting with the brain activity level detector algorithm, we also tried out different methods for moving the arm. We created a model that gradually moved the arm up and down in steps instead of at once. This gave the user more control, but at the same time, required more mental focus. While this would likely be an improvement for the future, we chose to go with a simpler design that moved the arm in one movement, since that was easier to control with little mind training.

III. COST ANALYSIS

A. Cost of Prototype

Part	**Cost**
Mindflex headset with NeuroSky EEG chip	$25
30kg rated Servo motor	$23
Seeeduino XIAO Microcontroller	$5.40
NRF24L01 Module (wireless RF module)	$1.30
Voltage regulator	$3
Optocoupler	$0.90
Batteries (for one headset)	$4
Breadboard	$0.90
Jumper cables	$5
Resistors	$6
3D printed parts	~$3 (estimated)
Screws	$10
Total	**$87.50**

3D printed parts pricing can fluctuate based on the material and printer used for printing. Parts from voltage regulator to screws will also fluctuate based on the consumer's usage and requirements.

1) Headset Cost Comparison: When looking for the best way to get EEG data, we explored multiple EEG sensor options. The table below lists different EEG options we looked into. We chose to move forward with the Mindflex headset because it is cheap, yet still provides EEG data with quality decent enough to work for this project.

Medical Grade EEG	Advanced system with better data quality, reliability, and depth of analysis.	$500 - $25,000+
Open EEG	EEG built from scratch.	$200+
MindWave Headset	NeuroSky's official single-channel, dry EEG headset.	$100+
Mindflex Headset	Headset designed for a game, yet has a working NeuroSky chip that can be modified to extract EEG data.	✔ $25

B. *Cost Analysis for Mass Production*

Parts	**Quantity**	**Cost/ Piece**	**Cost Reduction**
NeuroSky EEG chip + electrodes	100	$9	$14/piece
30kg rated Servo motor	100	$20	$3/piece
Seeeduino XIAO Microcontroller	100	$5.40	—
NRF24L01 Module (wireless RF module)	100	$1	$0.30/piece
Voltage regulator	100	$3	—
Optocoupler	100	$0.90	—
Batteries (for one headset)	100	$4	—
Breadboard	100	$0.90	—
Jumper cables	100	$5	—
Resistors	100	$6	—
3D printed parts	100	~$3 (estimated)	—
Screws	100	$10	—
Total		**$69.30**	**$17.30/ piece**

C. *Reducing Cost*

What can be done to reduce the cost of the mechanism?

Companies like NeuroSky do not sell single EEG chips, yet you are able to purchase them in mass, so one of the biggest ways we will be able to reduce the cost is by using open EEG chips and electrodes instead of needing a whole NeuroSky headset.

Additionally, we chose a servo that was guaranteed to support a full cup of water. 30kg is likely more powerful than what is needed though, so with additional testing, we should be able to determine how much power is actually needed. Less powerful motors are less expensive, so switching to a less powerful servo motor will allow for additional reduction of cost.

Lastly, once we are done experimenting and know what operations are required of the microcontroller, we should be able to switch to a less complex and cheaper microcontroller that can accomplish the basic operations.

IV. CONCLUSION

This project started with a mission to help mobility imparired individuals. The prototype created was able to successfully control and move a physical object using brain signals. With further research and experimentation, the arm's design can continue to evolve and improve. Some ideas include enhancing the stability of the algorithm used to control the arm, adding automatic calibration capabilities and improving the hardware components. The idea of using brain data to control physical items in the real world has endless applications and it's exciting to see a path for future progress in this field to really make a difference in people's lives.

V. REFERENCES

[1] "Mindflex", *Wikipedia*, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Mindflex.

[2] E. Mika, "How to Hack Toy EEGs", *Frontier Nerds*, 2010. [Blog]. Available: http://frontiernerds.com/brain-hack.

[3] E. Mika, (2014). Brain [Source code]. https://github.com/kitschpatrol/Brain

[4] E. Mika, (2014). BrainGrapher [Source code]. https://github.com/kitschpatrol/BrainGrapher

[5] *Processing*. The Processing Foundation, 2021. [Software]. Available: https://processing.org/.