T.C.
BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

CAPSTONE FINAL REPORT

SEMI-AUTONOMOUS BCI BASED WHEELCHAIR

Azmi Alhaj Younes, Biomedical Engineering
Khaled Al Sibabi, Biomedical Engineering
Yahya Al Baradan, Biomedical Engineering
Ataberk Suekinci, Mechatronics Engineering
Eymen Ege Bayer, Mechatronics Engineering
Selin Demirci, Mechatronics Engineering
Remziye Maral Demirseçen, Computer Engineering
Utku Arslan, Computer Engineering

Advisors:
Assist. Prof. Hakan Solmaz, Biomedical Engineering
Assist. Prof. Amir Navidfar, Mechatronics Engineering
Assist. Prof. Ehsan Nowroozi, Computer Engineering

ISTANBUL, January 2023

STUDENT DECLARATION

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

- they have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
- they have not received unpermitted aid for the project design, construction, report or presentation;
- they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

ABSTRACT

The project is based on the concept of Brain-Computer Interface (BCI) which manifests the scholar's understanding of the interaction between the human's thoughts and the complex brain signals. In the first place, the different methods of BCI are used to overcome the physical barrier in the case of motion constraints, thus BCI methods provide a solution based on the problems the disabled people want to overcome. The applications of this technology range from the ability to write, move a cursor, and control a robot, to even control a wheelchair. Nowadays, controlling a wheelchair through BCI is a research trend since it can provide mobility to locked-in syndrome patients. The widely used methods in BCI are based on external visual stimulation such as Steady State Visual Evoked Potential (SSVEP) and P300, and internal stimulation based on imagination such as Motor Imagery (MI) and Visual Imagery. Internal stimulation not only gives control of the wheelchair to the patients, but also it showed promising results in enhancing the rehabilitation of patients' motor functions. Also BCI systems can be synchronous and asynchronous and since asynchronous systems are more accurate, it was the chosen method for this project. Through this project designing, development, and implementation of a semi-autonomous electric wheelchair will be done.

Key Words: Brain Computer Interface, wheelchair, semi-autonomous ,internal stimulation, locked-in syndrome, rehabilitation, electrical vehicle, microcontroller, Motor Imagery, Asynchronous

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANSI	American National Standards Institute
API	Application Programming Interface
ASTM	American Society for Testing and Materials
BCI	Brain Computer Interface
CSP	Common spatial pattern
CAR	Common Average Reference
EEG	Electroencephalography
EEC	European Economic Community
EMBS	Engineering in Medicine and Biology Society
ERD	Event-Related Desynchronization
ERS	Event-Related Synchronization
EU	European Union
FCC	Federal Communications Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
LED	Light Emitting Diode
LIS	Locked in Syndrome
LSL	Lab Streaming Layer
MI	Motor Imagery
SL	Surface Laplacian
SCI	Spinal Cord Injury
SNR	Signal to Noise Ratio
SSVEP	Steady State Visual Evoked Potential
HTTP	Hypertext Transfer Protocol
ICES	Institute for Clinical Evaluative Sciences
IMU	Inertial measurement unit

1. OVERVIEW

The immobility that is caused by a Spinal Cord Injury (SCI) injury results in a complete loss of physical activities in those who used to have them. Such injury turns the patients` lives upside down in a way they describe as a soul imprisoned inside a body that is unable to do any basic activity independently. To retrieve a part of the lost physical functions and more specifically the mobility function, this project intends to provide a solution: A wheelchair that is controlled by both the captured user's thoughts through Electroencephalography (EEG) and an assistive system. The proposed solution is the most possible safe to use and will regain some of the lost independence, besides being almost the only noninvasive solution. To achieve such a solution a collaboration of engineering disciplines is required, since the project is a combination of subsystems from different disciplines: Mechanical, Electrical, Software, and Physiological.

1.1. Identification of the need

A condition called Locked-in syndrome (LIS) is caused by damage in the brainstem, and a common symptom is a quadriplegia. Many individuals still have cognitive abilities and upright eye movements. Those who survive this disorder frequently have a debilitating motor loss, although early diagnosis, supportive care, and rehabilitation have demonstrated to improve people's life [1].
Because of this syndrome, the patient`s movement has been restricted. A solution to this movement-restricted problem is using a system that does not depends on any physical activity. And one of the promising solutions is utilizing a Brain-Computer Interface (BCI) method based on Electroencephalography (EEG) to control a wheelchair. However, the aim of this project is to enhance and ease the life of people who are diagnosed with Locked-in syndrome by not only building a wheelchair based on BCI but also by providing a system that gives the patients the highest independent control with the minimum cognitive workload.

In mechatronics subsystem, The need of the patient is identified as an electric wheelchair system which is controlled by patient's EEG signals. The subsystem aims to utilizes the electromechanical system for accelerating, steering, braking. The subsystem uses ultrasonic sensors for identifying near objects and takes precaution for emergency risks.

1.2. Definition of the problem

It was stated by SCI-INFO that the number of Spinal Cord injuries SCI in the United is

approximately 300,000. And about 60% of them are considered quadriplegic [2]. In the first place, the different methods of BCI are used to overcome the physical barrier in the case of motion constraints, thus BCI methods provide a solution based on the problems the disabled people want to overcome. The applications of this technology range from the ability to write, move a cursor, and control a robot, to even control a wheelchair. Nowadays, controlling a wheelchair through BCI is a research trend since it can provide mobility to locked-in syndrome patients. The widely used methods in BCI are based on external visual stimulation such as Steady State Visual Evoked Potential (SSVEP) and P300, and internal stimulation based on imagination such as Motor Imagery (MI) and Visual Imagery. Internal stimulation not only gives control of the wheelchair to the patients, but also it showed promising results in enhancing the rehabilitation of patients' motor functions. And because of that, it was the chosen method for this project**.**

1.2.1. Functional requirements

The built system must be able to acquire the brain signals, process, translate, and classify them based on the trained user's thoughts to control the electric wheelchair. An assistive system is used to reduce the user's cognitive workload besides enhancing the safety level. Since there is a relatively high level of uncertainty in using BCI techniques to control a moving device it was proposed to use an emergency system based on the user's head movement.

Besides the ability of the electric wheelchair to stop, it will perform 3 main movements which are moving forward, to the right, and to the left. And regarding the speed, it will start at a previously specified value, and then will be accelerating based on how much the user spends on giving the same command.

1.2.2. Performance requirements

In order to achieve a high-quality performance there are many parameters that should be taken into consideration. The classifier must be well trained since the accuracy will increase with more training, in order to differentiate the several commands that are forward, right, left, and brake that are generated by internal stimulation through users' thoughts and imaginations. Also, the wheelchair must be able to stop if there is an obstacle blocking its way even though there is no command generated from the patient to stop the wheelchair.

The wheelchair must be able to rotate right or left with a specified speed as the user is thinking about the same command. While motion execution the sensors must keep reading the distance around

the wheelchair to ensure patient safety.

1.2.3. Constraints

The limitations that have a significant impact on this project will be explained and categorized in the following section. The main limitations can be categorized under 4 main sections: budget, scheduling, procedure, and time constraints. Furthermore, sustainability has three pillars (economic, environmental, and social) that should be taken into consideration.

Finally, standards need to be followed even though they limit project architecture but they are used to ensure subjects' safety and final product quality since the final product will be tested and used in real life.

1.2.3.1. Budget, Scheduling, Procedure, and Time constraint

- Budget Constraint

Building an electric wheelchair is an expensive process since besides the cost of the wheelchair itself electric motors need to be used and these motors should be efficient in generating enough torque to move the user and the wheelchair. Also, to power and control these motors a battery, fuse box, and microcontroller are required which adds to the project's cost. In addition, an EEG headset is required for brain signals acquisition, however, permission was taken to use the available headset in the laboratory which reduced the project's cost significantly. Bahçeşehir University offered 500₺ to each team member as support to the proposed project and until now this is the only source of funding, thus the project should be limited to this budget as much as possible. The limited budget caused sacrifices on the quality level by changing the used BCI method.

- Scheduling Constraint

The project's team consists of 8 members from three different departments and each member has a different schedule, thus group meetings are conducted both online and face to face since it is hard to find a time that all the team members are available. Also, in some cases, the meetings had to be conducted even with one member from each department. It is hard to manage the work between 8 people but with the Gantt chart and the weekly meetings we can make sure that each task is in progress and will finish on time.

- Procedure Constraint

The operational system carries several constraints such as sensor detection failures, mixed EEG signal definitions, and controlling. The wheelchair uses ultrasonic distance sensors for detecting an obstacle, or an object with high velocity, which creates a collision risk. After getting too close, To protect the user, the chair stops moving and waits for the danger to go away, or warns the user to move away from the object to gain more space. However, due to the sensor limitations, blind spots can occur, therefore the wheelchair can fail to detect the danger. To solve this problem, more sensors can be added to the wheelchair or the position angles of the sensors can be changed. Of course, sensor detection failures aren't the only operational constraint. EEG signals have the risk of defining, mixing, or controlling. For example, the necessary signal to turn right and left can mix up, or a slowing signal can't be defined due to the nature of EEG signals. To prevent this from happening in real life, higher quality electrodes can be used, sensing software can be optimized or safety precautions such as a steering lock can be applied.

- Time Constraint

The EEG signal is a complicated signal since even with the same subject and same procedure the measured signal is different between sessions. Also, choosing the most suitable BCI method in terms of cost efficiency and accuracy is a time-consuming process besides the fact that the accuracy depends on the subject and the chosen preprocessing, feature extraction, and classification methods. Modeling the wheelchair, controlling the motors, integrating the systems, and component assembly is not only a challenging process but also time-consuming.

1.2.3.2. Environmental, Ethical, Standards

Environmental

In the project, the wheelchair part is completed with the method of converting the regular wheelchair to an electrical one, which is considered an example of upcycling. The main computer in the project is taken from a used laptop ecosystem, therefore the project recycles a computer for a specific purpose. Also, even though greenhouse gas emissions are produced in different phases of production and these emissions could come from a variety of sources, such as the production of raw materials, the transportation of materials and finished products, and the charging of batteries, we are working on reducing these emissions. To reduce greenhouse gas emissions, we are attempting to use

recyclable materials besides modeling our designs before implementing them to reduce the wasted materials.

Ethical

The project aims to help people who can't move any part of their body. While helping a disabled community, a new controlling system is being tested. The chair mainly uses sidewalks of roads, streets, and avenues. In the world, most sidewalks are not ready for the disabled community to use. It is not wide enough, pedestrians often use paths made for disabled people, and motorcycles usually go on sideways to avoid traffic. In some cases, even cars are parked sideways. It is clear that handicapped community, around the world, needs support and help to integrate into society. In order to increase the integration of disabled people into the outside world, new policies and laws which protect the rights and responsibilities of the handicapped society must be enacted and roads and sideways must be inspected regularly for any code violations.

Standards

A real electric wheelchair will be built in this project as a prototype. Also, an EEG headset is in direct contact with the subject's head and it is used to acquire EEG signals for BCI application. Thus, the standards should be taken under consideration from almost every sub-system to ensure the final product's quality and safety. The used standards are; IEEE-1726-2015 as it was used in [3] to ensure the electrical safety of the built wheelchair, also ISO 10993-10:2010, Biological evaluation of medical devices - Part 10 should be taken under confederation for the biocompatibility [4]. ISO 7176-19:2008 and ANSI/RESNA WC/19 - Wheelchairs - Part 19: Electrically Powered Wheelchair and Scooter Safety Requirements are the standards used for the emergency braking system. There are different standards for the biocompatibility of the headset, and ISO, ANSI, and ASTM standards are for mechanical safety, but having access to these standards is challenging.

Biomedical Engineering:

An EEG headset will be used to acquire the brain signals from the user, thus Emotiv Epoc Flex will be used. To make sure the headset is safe to use since it will be in direct contact with the user's skin and more specifically head, thus Epoc Flex should be built based on reliable parameters which are specified in standards.

Epoc Flex is not a medical device as defined in EU directive 93/42/EEC and it was made to be used in research. Still, EMOTIV products have been evaluated by independent organizations for safety

and have received CE (CE marking) and other regulatory approvals [5]. The products meet all international safety standards related to radio frequency emissions, electrical safety, and potential toxicity or allergic reactions to components, also the products have been deemed safe and harmless for users [5].

Also, Flex has been designed to meet the requirements of the radio equipment directive (2014/53/EU) and complies with part 15 of the FCC Rules [6]. It is intended to operate without causing harmful interference, and it must be able to withstand any interference it receives, including interference that may cause it to operate in an unintended way [6].

Flex has been designed and tested to meet the emission limits and other requirements for Class B digital devices as established by regulatory agencies and ICES-003. [6]. This means that it is intended for use in residential environments and is unlikely to cause interference with other electronic equipment in the home. It may also be required to have a certain level of immunity to RF interference.

Section	Standards Tested
EMC and Telecom: Class B	ETSI EN 301 489-1 & 489-17
	ETSI EN 301 328 v2.1.1
	AS/NZS CISPR22 :2009
	AS/NZS 4268 :2012, BTLE 4.0
USA	FCC CFR 47 Part 15B & 15C
Canada	ISED RSS-247: Issue 2, IC RSS-102: Issue 5

Figure 1. Epoc Flex Followed Standards.

Mechatronics Engineering:

To ensure a safe and comfortable experience for the wheelchair user, several standards are used. For process management and ensuring quality for the resultant structure, ISO9001 standards are applied. For the structural properties and technical details, part 15 of ISO7176 is used. Also, TS 9111 is a Turkish standard that explains the building properties specialized for wheelchair movement and steering.

1.3. Conceptual solutions

There are different BCI paradigms for noninvasive or surface EEG each has a specific use based on the source of stimulation. Exogenous-based BCI methods require the usage of an external source of stimulation like a monitor SSVEP, and P300 are famous examples of an exogenous-based BCI methods. Endogenous-based BCI methods are based on the user's thoughts which will cause a reduction or increase in the mu-beta rhythm through a phenomenon known as ERD/ERS Event-related (De)synchronization.

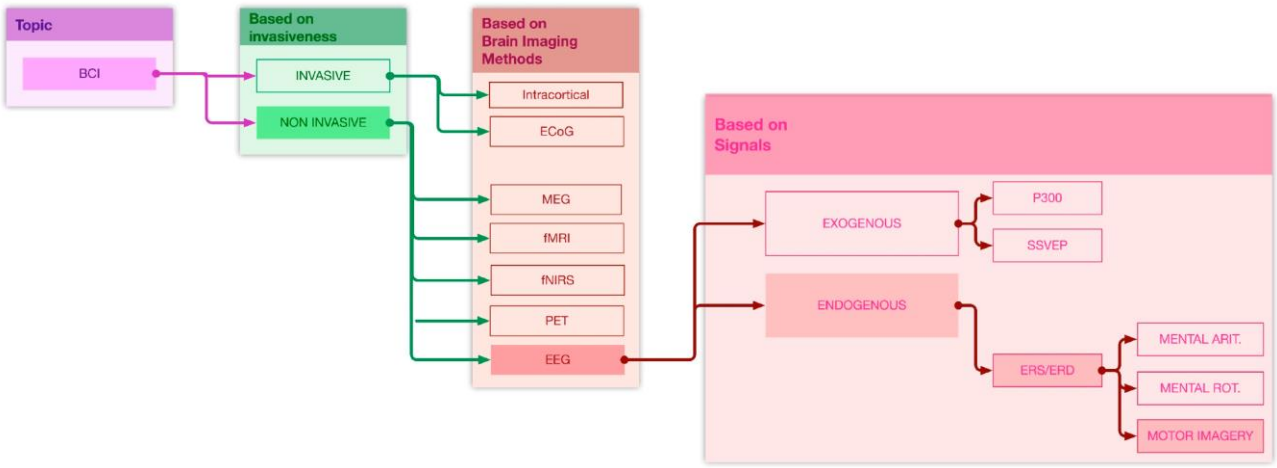


Figure 2. BCI Paradigms for EEG [7].

1.3.1. Literature Review

Choosing the most suitable BCI paradigm is a challenging task since different perspectives need to be considered, such as ‘How suitable is the chosen BCI method for the specified application?’, ‘Does the EEG headset has electrodes at the required brain locations?’, and ‘What is the cost to implement such a system?’. To answer these questions many research papers have been studied and reviewed to make sure the decisions will be taken based on a trusted source.

Steady State Visual Evoked Potential (SSVEP) is a concept widely used in literature to control a wheelchair effectively. Fred Achic, Jhon Montero, Christian Penaloza, and Francisco Cuellar (2016), used a screen and LEDs with 4 different colors and frequencies to control a wheelchair with a robotic arm [8]. Also, many research papers and projects used this method to control an electric wheelchair such as[9][10].

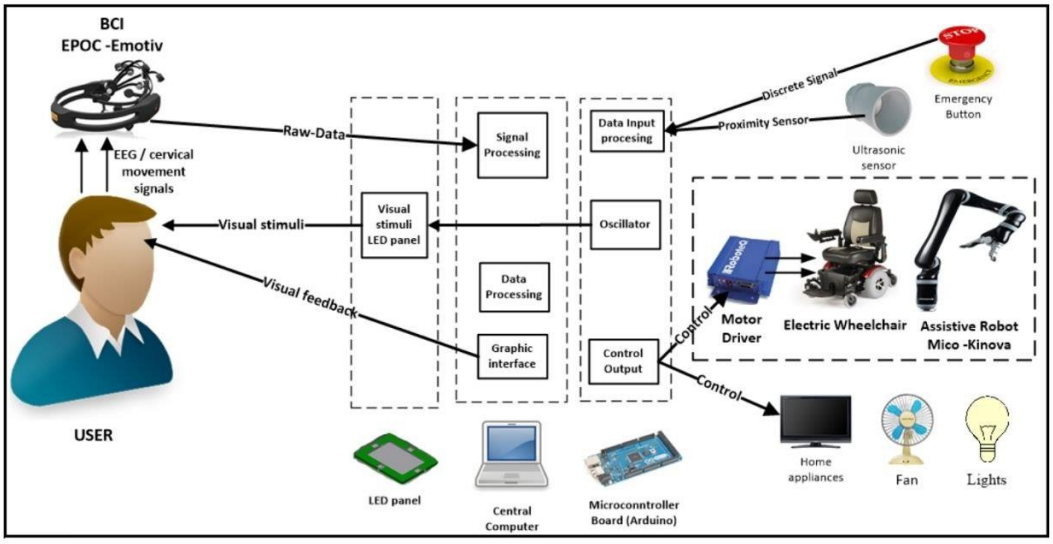


Figure 3. The interaction between the user and the systems [8].

P300 is another method that can be used to control an electric wheelchair. P300 is a positive change in voltage that occurs in the brain approximately 300 milliseconds after a stimulus is presented to a person, but as SSVEP it depends on an external source of stimuli. *Lopez et al* used P300 and a shared control system to make an electric wheelchair that can be controlled either by the user directly, or by the user can choose his/her desired destination and the wheelchair will autonomously drive to that location[11]. This allows the user to have flexibility in how they choose to operate the wheelchair and can be particularly helpful for those with mobility impairments [11]. The wheelchair in navigation mode will follow a map from the system's database which is assisted by a shared control system [11].

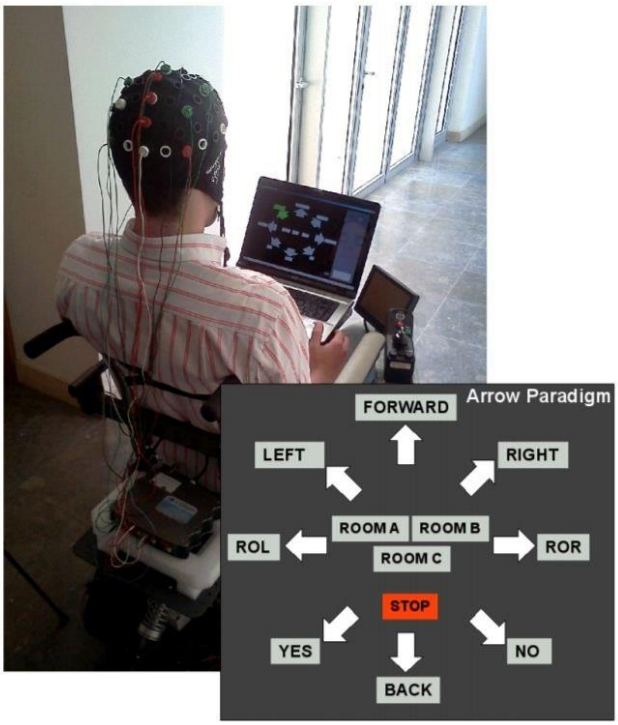


Figure 4. The possible commands [11].

Motor Imagery (MI), which involves mentally rehearsing movements without physically executing them, is widely used in the control of electric wheelchairs as well. The work of *J. Philips et al* suggests the usage of an adaptive shared control system with a wheelchair based on motor imagery and provides 3 possible mental steering commands (Forward, Left, Right) [12]. Also, *Bobrov et al* emphasize the usage of motor imagery when the controlled device changes its direction as a result of the sent command then the associated mental states can be thought of as imaginary movements of different body parts[13].

Figure 5. The usage of a wheelchair based on MI for an indoor environment [12].

Second part of our design is Machine Learning (ML), and there are various algorithms to choose from which will effect the accuracy and type of the final output. Different than regression algorithm which will return a definite numerical value, Classification is an ML algorithm widely used for Brain Computer Interface data in the literature which categorizes based on patterns and returns a class as a output.

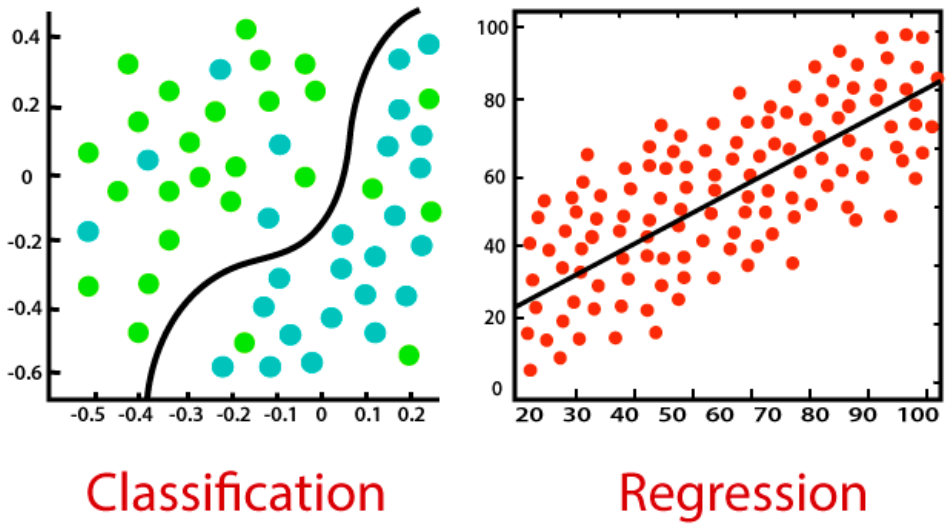


Figure 6. Classifcation vs Regression [25]

There are various types for classification algorithms. Unfortunately, there isn't any perfect "one-fit-all" algorithm for BCI data. Based on WEKA's implementations and literature review, we choosed several classification algorithms and will try them on our processed data [27, 29]. In result, we will find our own algorithm.

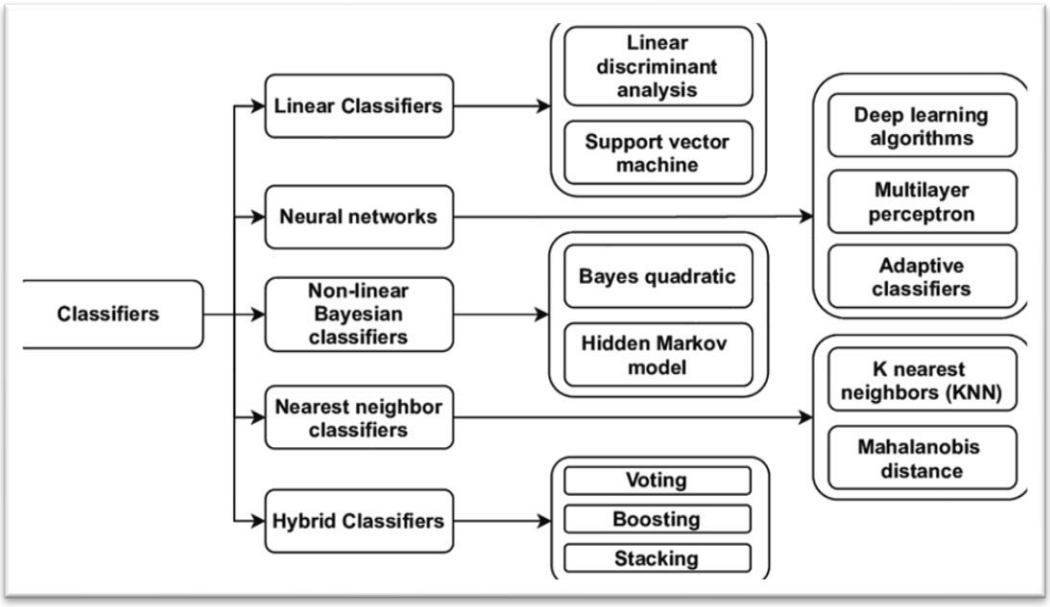


Figure 7. Different Classifiers Types [26]

As it was mentioned in the previous section and according to *Bobrov et al*, If different commands sent to an external device result in different movements, the associated mental states can be thought of as imaginary movements of different body parts[13]. For example, when a person is controlling a vehicle or wheelchair, they may associate a right-hand movement with a right turn of the device. Additionally, mental states related to imaginary movements of body parts can be identified by specific patterns in the electroencephalogram (EEG) [13]. However, we can receive much more data

per second in the SSVEP method than the MI and this is a feature referred to in the literature as information transform rate (ITR). *Wolpaw et al* studied the difference between SSVEP and MI In the study where the participants were asked to perform various tasks using a brain-computer interface system, including controlling a cursor on a computer screen and selecting items from a menu and they found that the ITR of SSVEP was significantly higher than that of MI [15]. However, SSVEP is expensive since the source of stimulation is an external source, also from the mechanical perspective it's hard to build an external screen on the wheelchair since it requires a lot of effort and time, also it may affect the movement space of the wheelchair and its balance. Since BCI is based on the activation of the sensorimotor cortex it has shown a promising positive effect on the rehabilitation processes as it was illustrated by the research done by *Tong et al* [16].

Table 1. Comparison of the three conceptual solutions.

	SSVEP	MI	P300
Cost	High	Low	High
Complexity	Moderate	Low	High
Performance	High	Moderate	Moderate
Features	Moderate	High	Low

1.4. Physical architecture

The top-level physical architecture is based on the usage of Epoc + or Eopc Flex as an EEG headset where the signals will be sent to and from Cortex, which is an application programming interface (API). Cortex can be used by python to record the data and add markers to the records which will make the analysis part easier and more efficient. Also, throughout the subject's training phase, a topographical mapping for his/her brain activity will be shown to the subject in a semi real time manner to make him/her see how the activity of the different parts in the brain change based on the subject's thoughts and to do so MATLAB is used. To stream the data from the EEG headset to MATLAB Lab Streaming Layer (LSL) will be used. Also, EEGLAB is a strong tool in MATLAB that can be used for data analysis and processing. Python will be used for commands classification since there are plenty of useful libraries. Through a serial port, the microcontroller will receive the data from the classifier and then start the action based on that command.

Lithium batteries will be utilized in the power distribution system to supply specified components in the wheelchair with the necessary power. Also, two microcontrollers (Arduino Uno) will be connected to the main computer (laptop) to manage the received data. One of the microcontrollers in this system is responsible for controlling the motors, and the other's job is to manage the data received from the distance-sensors. The data transmitted from the EEG headset to the main computer after being classified will be transmitted to the first microcontroller which is responsible for controlling the motors to provide movement. In this way, factors such as obstacle avoidance, and unconsciousness of the patient will be taken into consideration in the movement of the chair. Steering will be achieved by using two motors which are connected to the back of the wheelchair. The steering system is working as an electronic motor differential system, so when the user wants to turn the wheelchair in a specific direction, the rotational speed and rotational direction of each motor will be different causing the wheelchair to rotate.

Also, a fuse box will be designed to ensure the user and components' safety. Not only does the subject control the wheelchair but also distance sensors are used to ensure that the motors will stop and the braking system will be activated in case there is a moving body or any object that has a high potential of colliding with the wheelchair. There are two types of braking systems: Manual and emergency brakes. The manual brake system is controlled by the user and uses its inputs. In manual brakes, power to the motors is shut down and the back wheels are squeezed with a mechanical lock, which is activated with the user's wish to slow down. The emergency brakes, on the other hand, are activated by the control system that uses ultrasonic sensors and estimates a possible collision. The wheels get locked by computing both forward and backward commands at the same time.

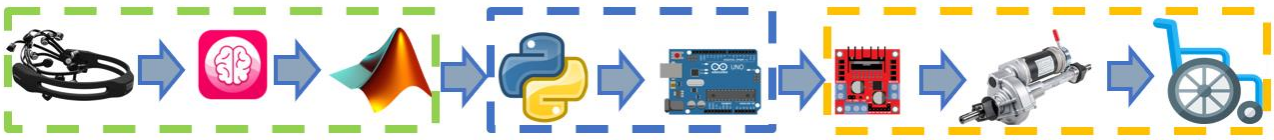


Figure 8. Summary of the sub-systems.

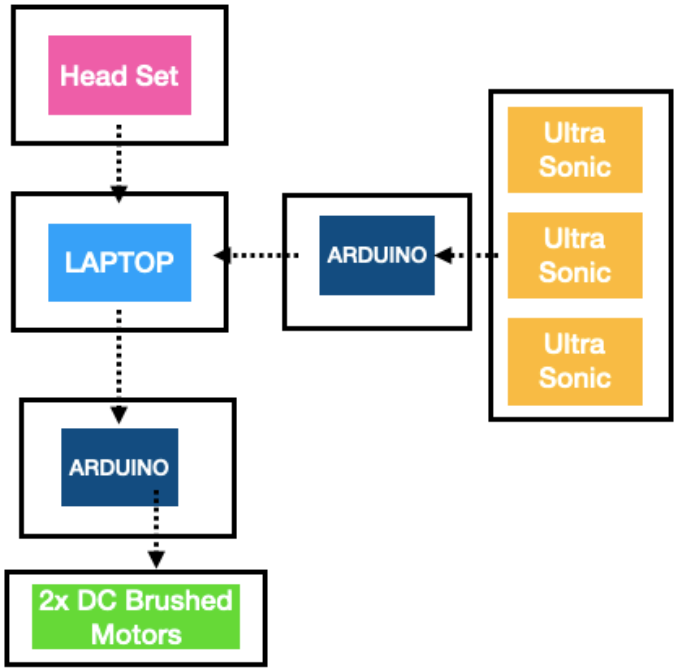


Figure 9 System Interface Diagram

2. WORK PLAN

2.1. Work Breakdown Structure (WBS)

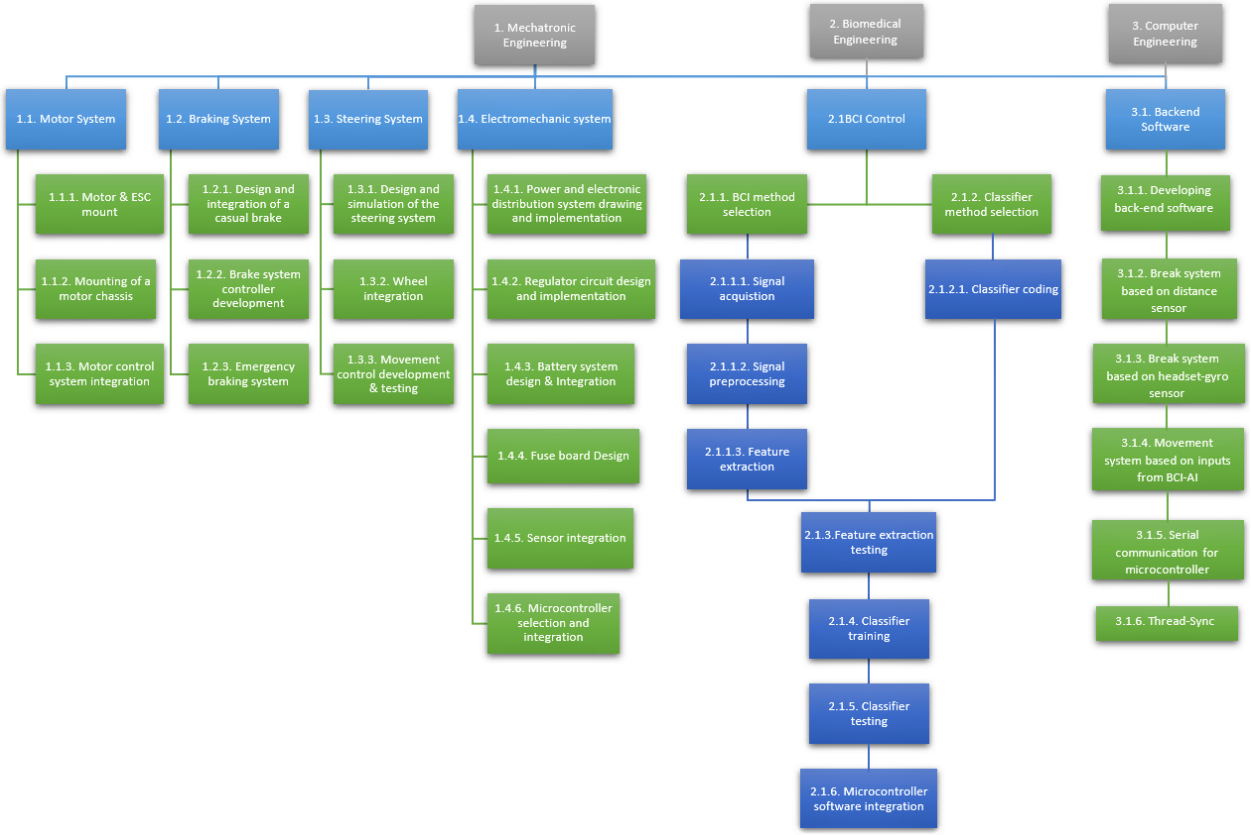


Figure 10. Work breakdown structure for the project.

2.2. Responsibility Matrix (RM)

The following is the Responsibility matrix which represents the tasks that need to be done and who is responsible for doing each, also who is assisting in each task. R means Responsable and S means Support.

Table 2. Responsibility Matrix

Tasks	Azmi	Ataberk	Ege	Khaled	Remziye	Selin	Utku	Yahya
1.1.1. Motor & ESC mount		S	R					
1.1.2. Mounting of a motor chassis			R			S		
1.1.3. Motor control system integration			S			R		
1.2.1. Design & integration of a casual brake			R			S		
1.2.2. Brake system controller development	S					R	S	
1.2.3. Emergency braking system	R					S	S	
1.3.1. Design and simulation of the steering system			R					
1.3.2. Wheel integration			R					
1.3.3. Movement control development & testing	S					R	S	
1.4.1. Power & electronic distribution system drawing and implementation		R	S					
1.4.2. Regulator circuit implementation		R				S		
1.4.3. Battery system design & Integration		R				S		

1.4.4. Fuse board Design		R						
1.4.5. Sensor integration		R				S		
1.4.6. Microcontroller selection & integration		S				R		
1.5 Mechatronic Reporting						S		
1.6 Mechatronic Project Management		R						
2.1.1 BCI Method selection	S			R				S
2.1.1.2.B Subject Training				R				
2.1.1.A Cortex API Study	R							
2.1.1.1 Signal Acquisition	R							
2.1.12 Signal Preprocessing	S				S			R
2.1.1.3.A Signal Quality Control				S				R
2.1.13 Feature Extraction	R				S			S
2.1.1.2.A Topographical Mapping	R			S				
2.1.2. Classifier Method Selection					R			
2.1.2.1 Classification Coding					R			
2.1.3 Feature Extraction Testing	S							R
2.1.4 Classifier Training					R			

2.1.5 Classifier Testing					R			
2.1.6 Microcontroller Software Integration	S						R	
3.1.1 Developing Back-end Software							R	
3.1.2 Break System Based on Sensor							R	
3.1.3 Break System Based on Headset-Gyro Sensor							R	
3.1.4 Movement System Based on Inputs from BCI AI							R	
3.1.5 Serial Communication for Microcontroller							R	
3.1.6 Thread-Sync							R	
Documentation	R		S			S		S
Management				R				

2.3. Project Network (PN)

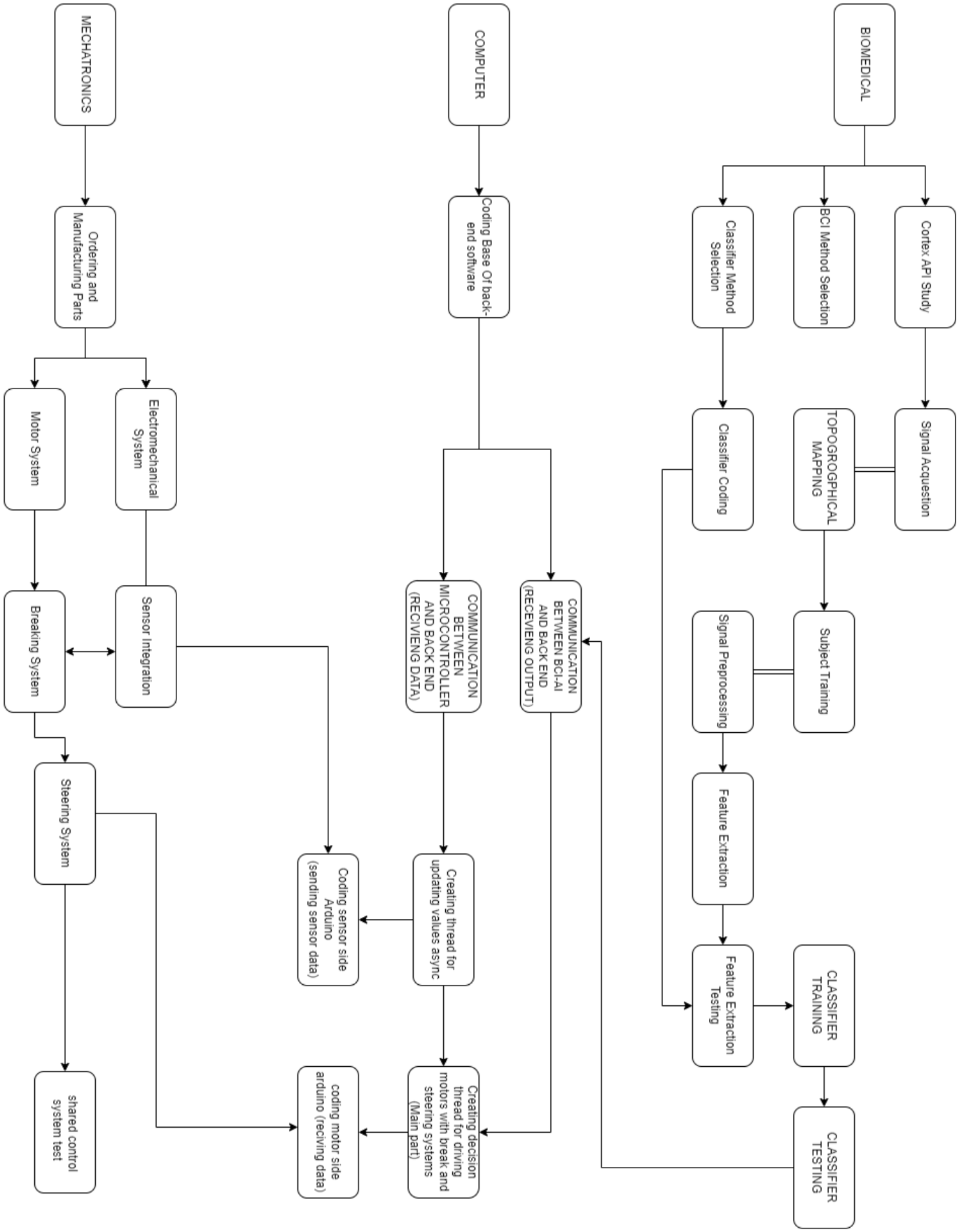


Figure 11. The project network structure.

2.4. Gantt chart

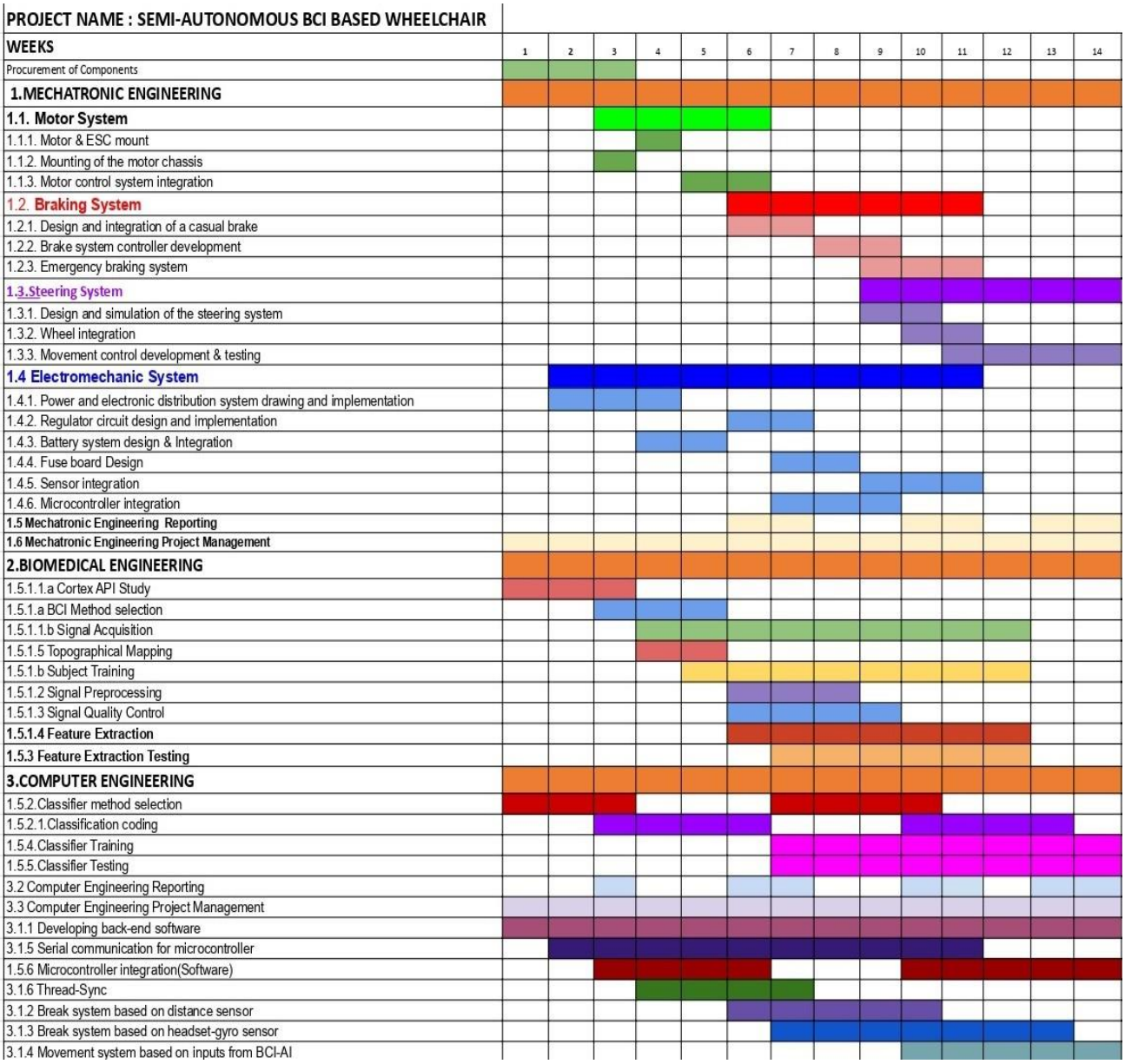


Figure 12. Gantt structure for the project.

2.5. Costs

Electro-mechanical	
DC Motor 6V x2	136.63 ₺
Ultrasonic distance sensor x3	91.12 ₺
Arduino Nano Microcontroller Board x2	350.75 ₺
Battery x2	1,395₺
Voltage Regulator	70 ₺
Kable	84.12 ₺
Jumper Kable	51 ₺
Rasbperi pi	1950 ₺
Total	**4,128₺**

Table 3. Costs

2.6. Risk assessment

Risk factors need to be considered and effective actions should be taken to reduce their effect and likelihood. The risk factors can be categorized into two major sections: Risk factors driven by brain signals and their classification and Risk factors driven by the electromechanical system.

- **Risk factors driven by brain signals and their classification**:

1. Systems incompatibility: Since there are different systems that need to be connected with each other there is a high risk that these systems do not work effectively with each other. The EEG headset should send the data to a computer to filter and extract the features of the signal then it will be sent to a classifier to determine the command which will be transmitted to the microcontroller. So, the EEG headset sends and receives commands by WebSocket which is a protocol like HTTP to provide full-duplex communication channels. For effective communication between the headset and the user, Emotiv recommends the usage of Cortex which is an application programming interface (API). Matlab is also compatible to some degree with the headset but it sends the data through Lab Streaming Layer (LSL), through working with Matlab some problems were countered in addition to its low performance with the chosen microcontroller. Thus, Python was chosen since it is compatible with all the systems

that will be integrated.

2. Electromagnetic and signal interference: The electromagnetic effect causes a reduction in the signal-to-noise ratio (SNR) which will affect the classification process, thus preprocessing methods are used to eliminate the electromagnetic noise. Also, blinking was shown to have a negative effect on the signal as well which necessitates the usage of algorithms such as Common spatial pattern (CSP) to eliminate that effect.

3. Environment interference: Motor imagery is based on the subject's thoughts and this requires a high level of focus from the subject and quietness of the environment that the recording session is being held on. Otherwise, the subject will not be able to generate the thought that will activate specific regions in the brain. To reduce the effect of the environment the recordings will be taken in the lab when it is empty, by scheduling the recording sessions.

4. Command repeatability: EEG is a dynamic signal which means that it naturally changes with time and even with exactly the same conditions the acquired signal will not be the same. Thus, controlling a wheelchair by a signal that is not highly repeatable has a high risk and high probability of failure. To solve this issue an algorithm will be built to produce decisions based on how similar the signal is to how it suppose to be.

5. Low classification accuracy: If the classifier is not able to distinguish between the different commands the wheelchair will neither be safe to use nor useful and fulfilling its purpose. Thus, the accuracy of the classifier is critical in our case. To increase the accuracy of the classifier the system should be trained as much as possible, so the number of training sessions should be enough to get sufficient system accuracy.

6. Subject’s cognitive overload: Cognitive overload is a state of being overwhelmed by too much information, which can lead to reduced performance and an increase in mistakes. In the context of BCI training, this risk can be particularly high if the training involves tasks that are too complex or require too much information processing at once, especially for people who are not familiar with BCI technology or who have cognitive impairments. To mitigate the risk of cognitive overload during BCI training, it is important to use appropriate training methods and closely monitor the person's progress, potentially including starting with simpler tasks, providing breaks, and providing feedback to help the person understand how to improve. It may also be beneficial to use training methods that are tailored to the person's individual needs and abilities.

- **Risk factors driven by the electromechanical system:**

1. Short circuit: A short circuit may occur as a result of loose connection of wires, wrong

wiring and touching of plus and minus sides of cables. To avoid the consequences of short circuit, a fuse box with separate connection for each component will be designed and manufactured. When any component is short-circuited, only that component's fuse will blow, and by changing the fuse the system and component will be protected.

2.Insufficient data from sensors: Due to insufficient data from the sensors, the distance between the obstacle and the wheelchair can not be obtained properly. This will result in not being able to stop at the right time and collision. Increasing the number of sensors or changing their location can be a solution to this problem.

3. Over Current and Heating: Since motors consume high power, the cables should be selected with appropriate length and diameter by making calculations.

4. Mechanical failure: Due to calculation errors, the wheelchair may fail to carry its user with maximum efficiency. The overweight system may cause failure with the braking and accelerating system. Loose motors can cause vibration, which affects the user's comfort in a negative way.

Table 4. Risk Assessment Table

Failure event	Probability	Severity	Risk level	Plan of action
Systems incompatibility	Possible	Major	High	Check the manufacturer's and follow their recommendations to ensure compatibility of the used software with other components.
Electromagnetic & signal interference	Likely	Moderate	Medium	Signal pre-processing and asking the user to prevent any unnecessary movement such as eye blinks.
Environment interference	Medium	Minor	Medium	Reduce the number of people in the lab beside, through scheduling the recording sessions.

Commands repeatability	Likely	Major	High	Feature extraction and utilizing a classification method will be done to estimate the command.
Low classification accuracy	Possible	Major	High	Train the classifier side using the most appropriate classification methods based on literature.
Subject's cognitive overload	Unlikely	Moderate	Medium	The recording sessions are divided into blocks and there is rest time between sessions as recommended in the literature.
Short circuit	Possible	Major	High	A fuse box is designed to make sure the user and the components will be safe
Insufficient data from sensors	Likely	Moderate	Medium	Increasing the number of sensors or changing their location can be a solution to this problem.
Over Current and Heating	Medium	Major	High	With the use of appropriate sized cables we can avoid over current and heating
Mechanical failure	Possible	Major	Medium	Revising the motor system's design and integration

3. SUB-SYSTEMS

3.1. Biomedical Engineering

This sub-team consist of 3 biomedical engineering students, who will be responsible for different tasks, the main function of this sub-system is to find the most appropriate BCI method, acquire the EEG signal, pre-process, extract the main features of the signal, and test the used method in the feature extraction.

3.1.1. Requirements

To control the electric wheelchair, the developed system must be able to gather brain signals, process, translate, and classify them based on the trained user's thoughts. An assistive system is utilized to lower the user's cognitive workload while also increasing safety. Because there is a relatively high level of uncertainty in employing BCI techniques to control a moving device, an emergency system based on the user's head movement was developed. Aside from the ability to stop, the electric wheelchair will accomplish three major movements: forward, right, and left. Also, there are various parameters that must be considered in order to obtain a high-quality performance. In order to differentiate the different commands that are generated by internal stimulation through users' thoughts and imaginations, the classifier must be effectively trained because accuracy increases with more training. Furthermore, the wheelchair must be able to stop if an object blocks its path, even if no command is issued by the patient to do so. As the user considers the same command, the wheelchair must be able to rotate right or left at a predetermined speed. To maintain user safety, the sensors must constantly read the distance around the wheelchair while in motion.

3.1.2. Technologies and methods

Method (Motor Imagery): According to a review article published in the journal Frontiers in Human Neuroscience, Motor imagery (MI) is defined as "the mental simulation of a movement or action, without physically performing it. This technique has been widely used in electroencephalography (EEG) research to study brain activity related to MI and to comprehend the neural basis of movement [20]. During MI, the user concentrates on the movement they want to perform, and the brain activates the same neural pathways as if the movement were physically performed. EEG, which records the electrical activity of the brain through electrodes placed on the scalp, can detect and measure this activation.

Technologies:

1-Software: The main communication channel between the human brain and a computer interface is a Brain-Computer Interface (BCI). The data collected by the selected headset will be sent here to be processed, analyzed, and transformed into something usable. The BCI relays the EEG signal from all electrodes, and the signals can be viewed in real-time in coupling apps depending on the software used [18]. However, mathematical analysis is more complicated, and not all systems can perform it. This section examines various EEG systems that could aid in the project's success.

a. Cortex (API): The EEG headset sends and receives data through JSON and WebSockets, so the brain signals can be recorded and their features can be observed, such as the band power and the signal in the frequency domain, through EmotivPRO. However, to be able to control the data flow to and from the headset an Application Programming Interface API is suggested to be used on python. Cortex is an API built by Emotiv based on JSON and WebSockets, also it allows the users to inject markers during a recording session which will simplify the analysis and feature extraction part. However, some modifications have been done on Cortex such as modifying the markers injection code to show images to the subject for a specified period before starting the motor imagery as suggested by [14]. Also, the used headset has a 3-axis Accelerometer, 3-axis Gyroscope, and 3-axis Magnetometer that work at a sampling frequency of 64Hz [17]. So, to be able to able acquire the signal from the headset Cortex will be used.

```
{'streamName': 'eeg', 'labels': ['COUNTER', 'INTERPOLATED', 'AF3', 'F7', 'F3', 'FC5', 'T7', 'P7', 'O1', 'O2', 'P8', 'T8', 'FC6', 'F4', 'F8', 'AF4', 'RAW_CQ', 'MARKER_HARDWARE']}
eeg labels are : ['COUNTER', 'INTERPOLATED', 'AF3', 'F7', 'F3', 'FC5', 'T7', 'P7', 'O1', 'O2', 'P8', 'T8', 'FC6', 'F4', 'F8', 'AF4', 'RAW_CQ', 'MARKER_HARDWARE']
The data stream mot is subscribed successfully.
{'streamName': 'mot', 'labels': ['COUNTER_MEMS', 'INTERPOLATED_MEMS', 'Q0', 'Q1', 'Q2', 'Q3', 'ACCX', 'ACCY', 'ACCZ', 'MAGX', 'MAGY', 'MAGZ']}
mot labels are : ['COUNTER_MEMS', 'INTERPOLATED_MEMS', 'Q0', 'Q1', 'Q2', 'Q3', 'ACCX', 'ACCY', 'ACCZ', 'MAGX', 'MAGY', 'MAGZ']
[46, 0, 4357.564, 4349.615, 4358.846, 4362.436, 4415.256, 4343.718, 4337.564, 4340.513, 4325.0, 4316.41, 4512.82, 4445.897, 4416.667, 4461.282, 0.0, 0]
[47, 0, 4357.564, 4347.692, 4356.539, 4361.282, 4411.667, 4340.897, 4337.692, 4337.308, 4315.256, 4307.82, 4499.231, 4441.026, 4411.539, 4455.256, 0.0, 0]
[48, 0, 4357.564, 4348.461, 4355.128, 4360.641, 4410.513, 4342.308, 4334.615, 4329.872, 4316.282, 4305.641, 4495.513, 4438.59, 4412.051, 4457.051, 0.0, 0]
[49, 0, 4360.0, 4352.436, 4358.718, 4365.0, 4415.513, 4346.667, 4331.026, 4328.59, 4324.231, 4311.795, 4507.18, 4442.051, 4413.077, 4462.692, 0.0, 0]
motion data: {'mot': [16, 0, 0.475844, 0.363525, -0.630371, 0.494019, 0.939079, -0.279819, -0.063973, -32.494488, -7.45825, 75.765503], 'time': 1672161844.332}
```

Figure 13. Sending and receiving data from the EEG headset through Cortex

b. EEGLAB/Matlab: MATLAB is excellent for numerical computations, algorithm applications, and user interface design on its own. The EEGLAB Plug-in, on the other hand, exists to facilitate this process rather than to build a BCI interface from the ground up. EEGLAB is an open-source toolbox and graphical user interface for MATLAB. It supports the import of raw EEG data, as

well as its easy visualization, preprocessing, component analysis, and time/frequency decomposition. The toolbox enables users to interact with data without using MATLAB. Customizable settings and functions can be used to quickly process data. Users can integrate the toolbox with MATLAB codes with some training, enabling an extra layer of communication to other devices [19].

c. Emotiva Pro: Emotiv Pro is a software application created by Emotiv Company. Emotiv Pro is a software suite that allows users to view, analyze, and present real-time brain activity data. Emotiv Pro is a comprehensive and user-friendly software application designed to be used in conjunction with the Emotiv EEG headset. It is intended for a variety of applications, including research, education and others. Emotiv Pro is available for purchase from the Emotiv website and is compatible with Windows, Mac, and Linux operating systems.

d. OpenVibe/Python: These are some programming languages that, with the help of additional plug-ins, can be used as a BCI, with Open Vibe being a platform for designing, testing, and using brain-computer interfaces (BCIs). It is a free and open source software suite that includes tools for real-time data acquisition, signal processing, visualization, and BCI experimentation.

2. Hardware: The main hardware component for the EEG side of the project is the data collection mechanism. While the mechanism could be as simple as a couple of sensor electrodes attached to the skull, this solution is too impractical to consider due to inconsistency, a low signal-to-noise ratio, and general inaccuracy. Instead, we'll look into the "Emotiv EPOC+" and "Emotiv EPOC FLEX" headsets that are available in the BAU BME laboratory. Emotiv Systems is a company that specializes in EEG headsets and BCI applications, combining both in a single convenient unit.

a. EPOC+: It is an EEG headset with 14 electrodes and 2 others as comforters, each of which can capture an EEG electric signal, which is then wirelessly transmitted to a PC via Bluetooth using a Universal USB Receiver (Dongle). The data is displayed in real-time by the Emotiv BCI applications, just like any other high-quality EEG. The headset also includes saline solution, which is rubbed on each electrode beforehand to improve conductivity and accuracy.

Figure 14. Emotiv EPOC+.

b. EPOC FLIX: It is Emotiv's more advanced, heavy EEG headset. It combines the EPOC+ headset's basic features with the accuracy and robustness of other high-quality EEG caps. The number of electrodes is increased to 32, and they are now gel electrodes, which are more accurate. Unlike the EPOC+ headset, the electrode positioning is adjustable here, allowing for the highest level of data acquisition.

Figure 15. Emotiv EPOC FLIX Headset.

3.1.3. Conceptualization

As was mentioned in section 1.3.2 and section 1.3.1 there are two BCI paradigms that are widely used in literature for electrical wheelchair control SSVEP and MI. Starting with Motor Imagery, the main idea is based on the fact that there is a relation between moving a body part and

imagining its movement as it was illustrated by the research done by *Mokienko et al* [21]. Also, it was noticed that when the EEG signals are acquired from the motor cortex there is power modulation in the mu/beta rhythms when different movements are imagined [23]. The mu/beta rhythms have a range of 8-12 Hz for mu and 18-25 Hz for beta and both should be taken into consideration, since it was demonstrated that some of the beta rhythms are harmonics of mu [22]. Signal power modulation is a phenomenon that in literature is referred to as event-related (de)synchronization based on whether the power increases or decreases when the movement is imagined. As shown in figure 13 each part of the motor cortex is associated with a specific part of the body, for example, the area in the motor cortex that is responsible for hand movement is more lateral from the central sulcus than that for leg movement. Also, the contralateral control phenomenon which means that the right hemisphere controls the movement of the left side of the body and vice versa should be taken into account.

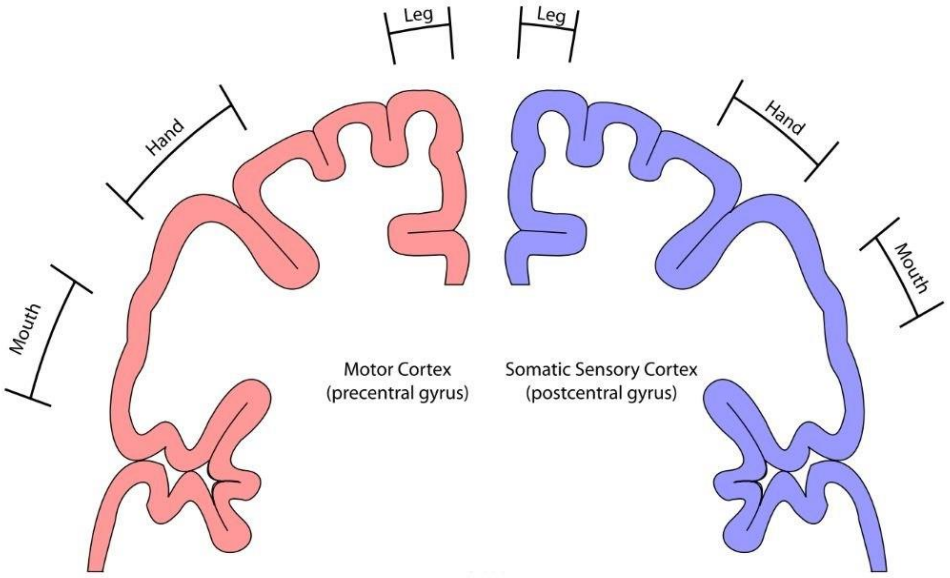


Figure 16. Motor homunculus (Left) and sensory homunculus (Right) [22].

As it was illustrated by Schalk & Mellinger, (2010) and after using a temporal filter the topographical mapping showed that when the subject imagined left-hand movement a decrease in the power of the signal acquired from the right side of the brain which is responsable for controlling the left hand [22]. The decrease in power is relative to the nutral state which can be seen the figure 14 as a blue line and the red line is the power of the signal taken from the righ side of the brain while the movement is imagened [22].

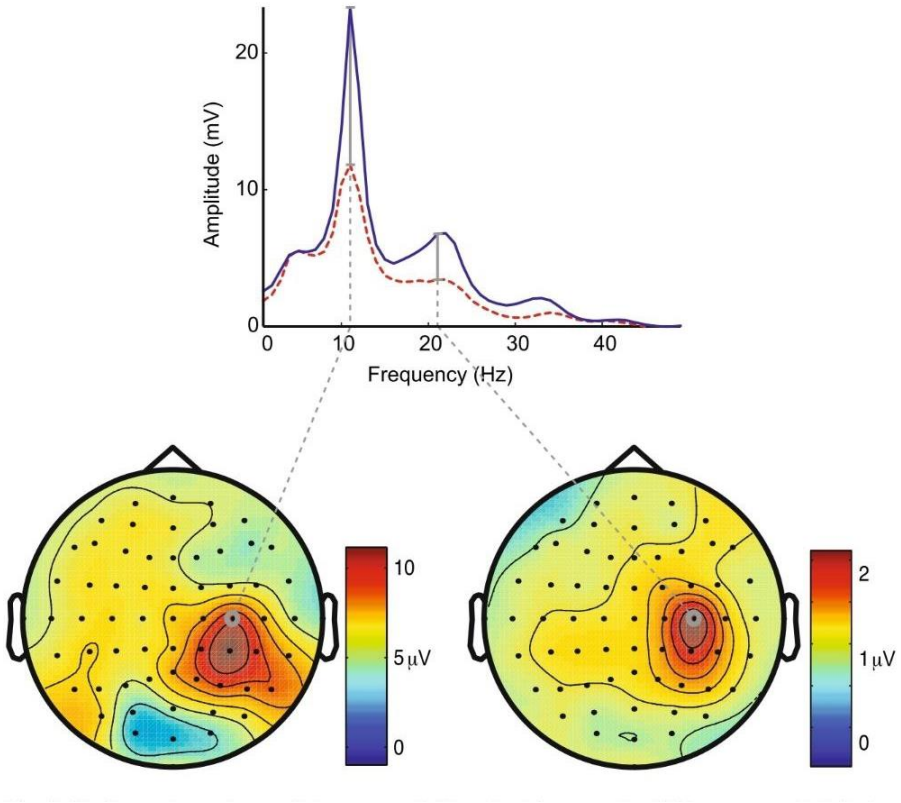


Figure 17. Topographical mapping of left-hand movement [22].

SSVEP on the other hand requires an external source of stimuli to activate the occipital lobe in a specific repeated way which is effective as a BCI method. Also, SSVEP is more comfortable for the user since it does not require a high cognitive workload. The user will only focus on a flickering object. Still, the blinking frequency should be specified and consistent because the EEG signal will have features strongly related to this frequency. The optimal stimulus frequency was found to be in the medium frequency range (30 - 45 Hz) since a decrease in the frequency was found to have a negative effect on the user and could cause discomfort or even epilepsy [24]. Also, it was found that high flickering frequencies did not have acceptable classification accuracy when tested on different subjects [24]. The flickering object could be based on LEDs or any other object like a monitor, but the screen's refresh rate should be considered. As it was illustrated by Ng, Soh, and Goh (2014), by using 12 Hz stimuli and with Fast Fourier Transform (FFT) a peak at almost 12 Hz can be seen in the frequency domain. Figure 15 shows the signal in the frequency and an obvious peak at 12 Hz can be seen clearly which indicates that the user is focusing on an object that has 12 Hz as its flickering frequency. Motor imagery was chosen because it is less complex from the programming prespective compared to SSVEP also the cost of building a wheelchair based on SSVEP exceeds the project’s budget.

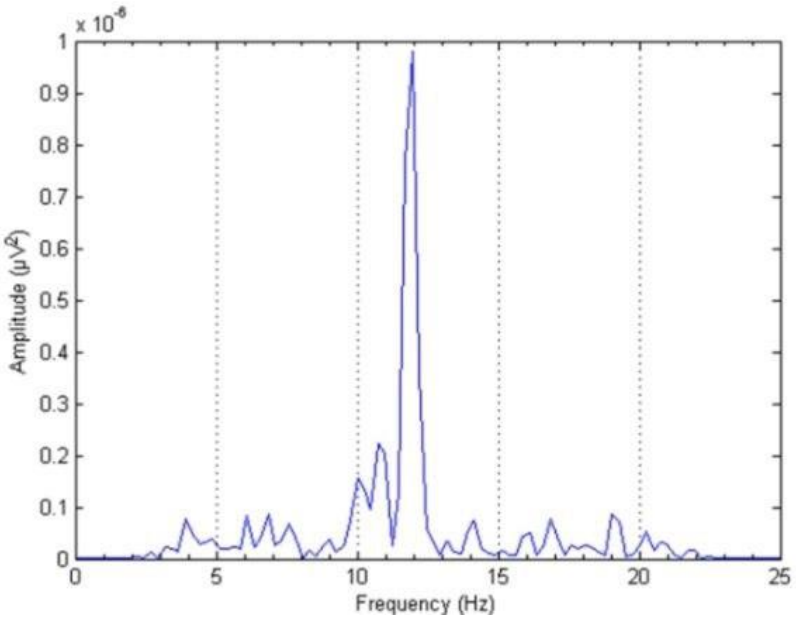


Figure 18. Visual stimuli at 12 Hz [9].

Also BCI systems can be on of two types Asynchronous and synchronous BCI systems take different approaches in deciphering and interpreting neural signals, each with their own advantages and drawbacks. Asynchronous systems offer more flexibility, allowing users to control external devices without needing to adhere to rigorous timing restrictions. However, they might be less precise and slower to respond due to the machine learning algorithms used to decode the neural signals. Conversely, synchronous systems rely on precise timing and external cues to activate specific neural responses, resulting in a faster response time and higher precision. Nevertheless, they require strict timing and external stimuli, which may not be ideal for all users. Ultimately, the selection between the two BCI systems hinges on the specific needs and capabilities of the user, as well as the intended purpose of the system.

3.1.4. Software architecture

Python was used to modify Cortex codes which will be used to record the data, inject markers in the records, live stream of the data acquired from the headset such as those from the Inertial measurement unit (IMU), and information about the headset itself such as battery level and electrodes conductivity can be measured. The modification mainly was done through OpenCV which is a library mainly used for computer vision applications, but in our case, it was used to show the user images for a specified period of time before starting recording since this method is widely used and recommended in the literature.

Also, EmotivPro was used to stream the data between the EEG headset and Matlab through LSL which gives the option to stream different types of data from the headset and the option to stream

the data as chunks or as samples as can be seen in figure 16. The data flow mainly will be between Cortex and the headset through JSON and WebSocket to send and receive data and commands such as Authentication, connecting with the headset, starting a session, data streaming, and recording. EMOTIVBCI, on the other hand, gives the chance to train a built-in artificial neural network that will differentiate between the neutral state and four other trained commands that can be trained by imagining the movement of a virtual cube and then choosing whether the trial was successful or it should be rejected. Also, there is a link between Cortex and EMOTIVBCI which will be investigated from the Cortex documentation page that is created by Emotiv.

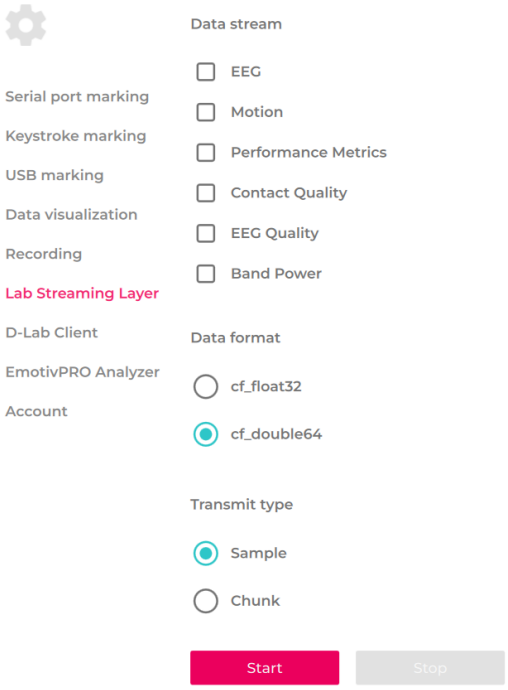


Figure 19. LSL options in EmotivPro.

3.1.5. Materialization

3.1.5.1. Emergency system parameters setup

As was mentioned early an emergency system will be built that can be triggered by the subject's head movement, but to do so in order not to activate the emergency system when not necessary an extreme, but not harmful head movement needs to be done. Different movements have been tested and evaluated based on comfort and how easy it is to be detected by the IMU sensor that is in the headset. The most suitable head movement was head tilting to the right, also the accelerometer at the z-axis was chosen to be the most suitable to detect this motion.

From Figure 38 it can be seen that after the calibration period which ends at sample 6000 each

positive peak indicate a head tilt to the right, and since the peak always exceeds 0.5 after a while 0.2 was chosen as a threshold to make the system response faster and with the same accuracy.

Figure 39 shows the IMU sensor values when the recording started with a head tilt to examine the robustness of the chosen parameters and how the calibration will affect the values of the accelerometer. It is obvious that IMU sensor calibration does not have an effect on the accelerometer readings which can not be said about the others

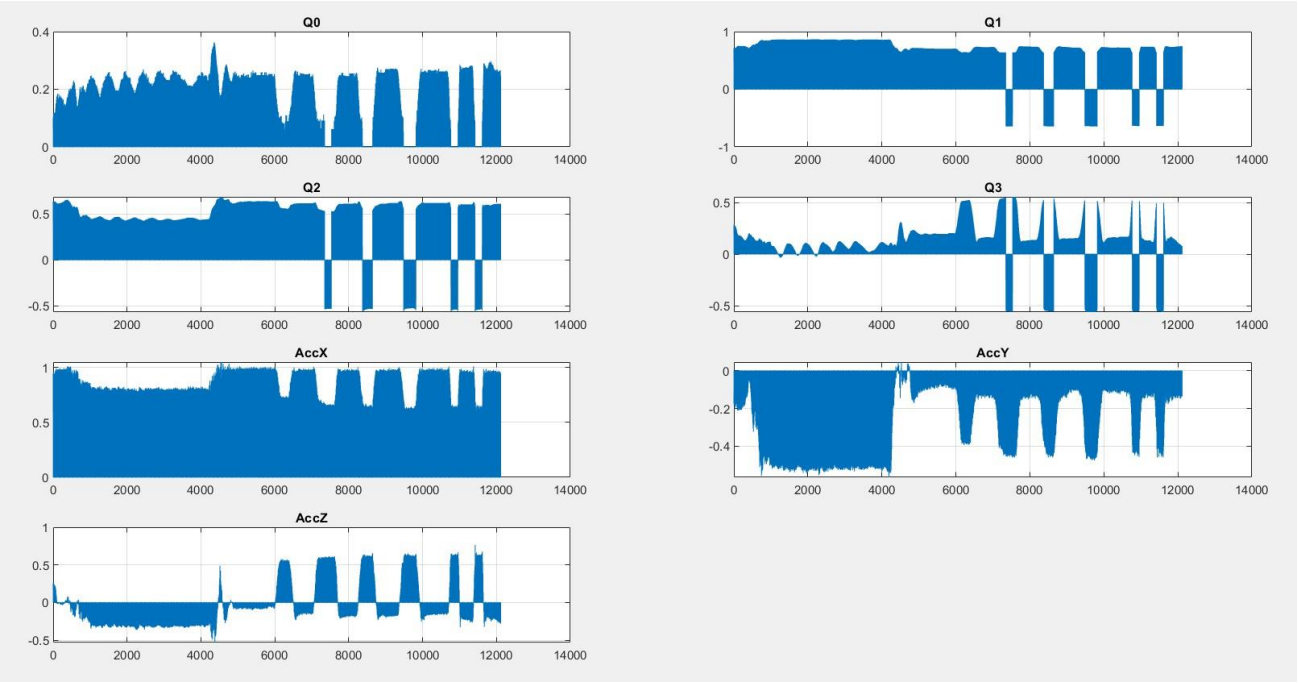


Figure 38. IMU sensor values when recording starts from rest

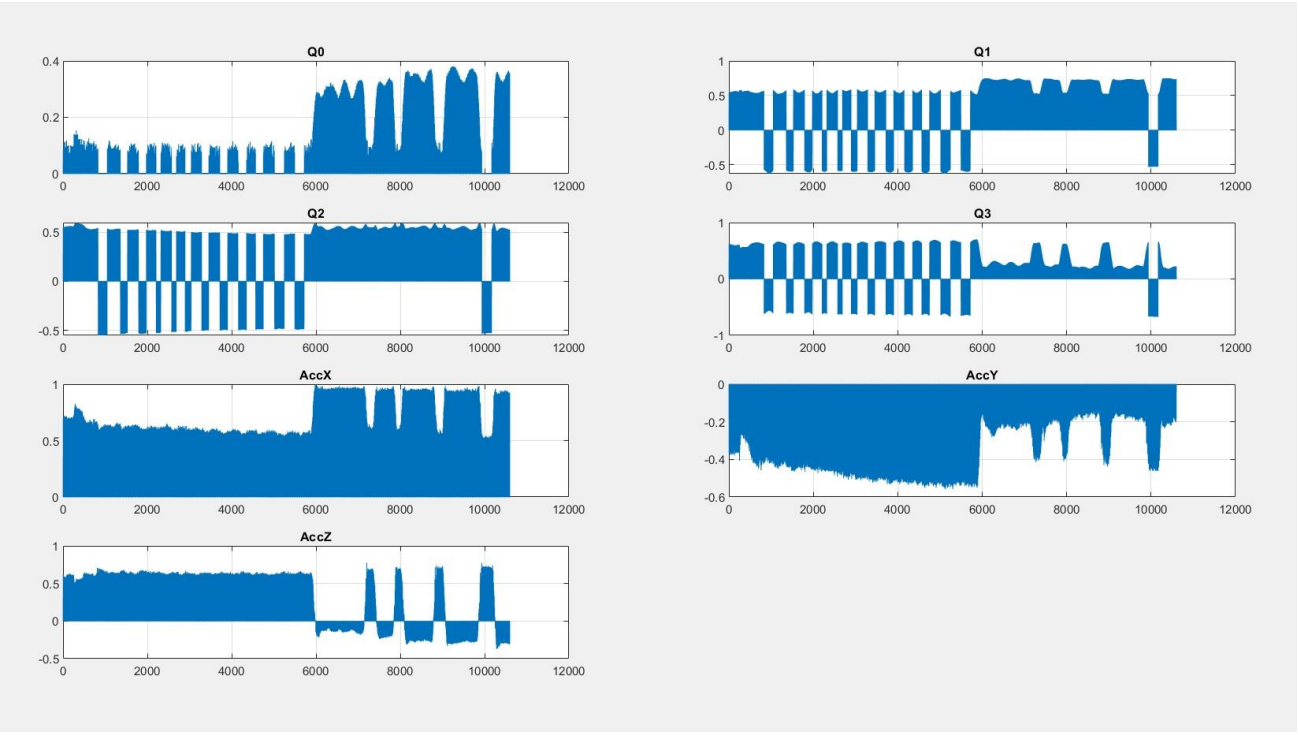


Figure 39. IMU sensor values when recording starts from tilted head

3.1.5.2. Testing the pre-processing & feature extraction methods

The methods were tested on BCI competition IV dataset 2A where the EEG signal was recorded at a rate of 250 Hz and filtered between 1 and 50 Hz using a Notch filter and the participant was seated in a comfortable chair with armrests and instructed to imagine movements of their left hand, right hand, foot, or tongue in response to a cue[35]. The experiment consisted of multiple runs, with each run containing 40 trials at the start of each trial, there was a 2-second period of quiet, followed by an acoustic signal at t=2s to indicate the start of the trial [35]. A cross "+" was then displayed on the screen, and at t=3s, an arrow pointing left, right, up, or down was displayed for 1 second at the same time, the subject was asked to imagine the corresponding movement until the cross disappeared at t=7s [35]. Each of the four cues was presented 10 times in a random order within each run.

a. Signal Preprocessing:

An 8th-order Butterworth bandpass filter with a cutoff frequency of 8-30Hz since as mentioned in the literature it contains both beta and mu rhythms since they contain the change that occurs due to motor imagery [36]. Also, different spatial filtering technics were tested as preprocessing methods such as Surface Laplacian (SL), Common Average Reference (CAR), Bipolar, and mastoid reference. Surface laplacian is mainly about subtracting the mean signal of the four surrounding channels from the signal of the middle channel. Common average reference is done by subtracting the mean signal

of all the channels from the signal that we want to filter. Bipolar subtracts the signal from each pair of channels and mastoid reference subtracts the signal from each channel from a reference channel.

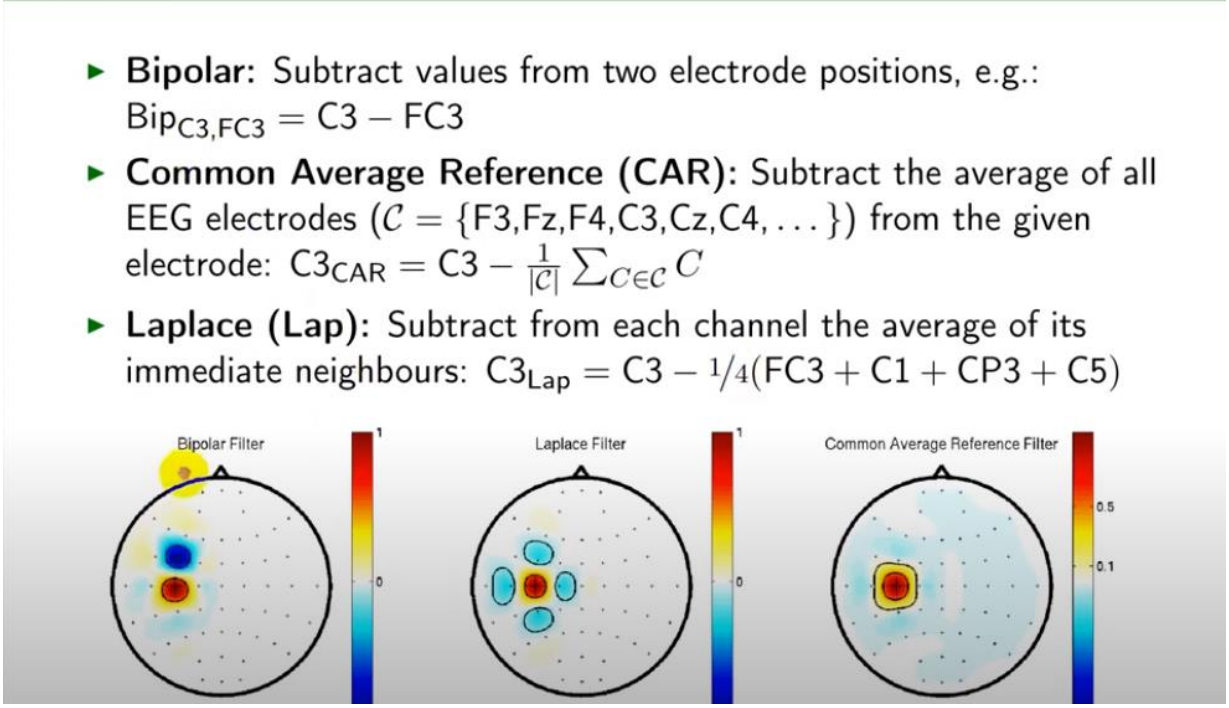


Figure 39. Three Examples of Spatial Filters [39].

Independent component analysis is another famous preprocessing method, however, it requires visual inspection of the decomposed signal components to select the components that are believed to be from noise sources such as muscle, line, and eye artifacts, and that requires years of experience [40]. Also, ICA can not be applied in real-time, thus it was eliminated from being a preprocessing step for the acquired data in real time. X is an N by T matrix where N is the number of channels and T is the number of samples. A weight matrix W will be multiplied by X and this will give components matrix U where each row is a single component of the signal. Each row of the component matrix should be visually inspected to decide if the component is related to the EEG signal or to a noise source. The row that is believed to be from a noise source will be zeroed and the multiplication of the mixing matrix which is the inverse of W times the modified components matrix will give the new filtered signal X.

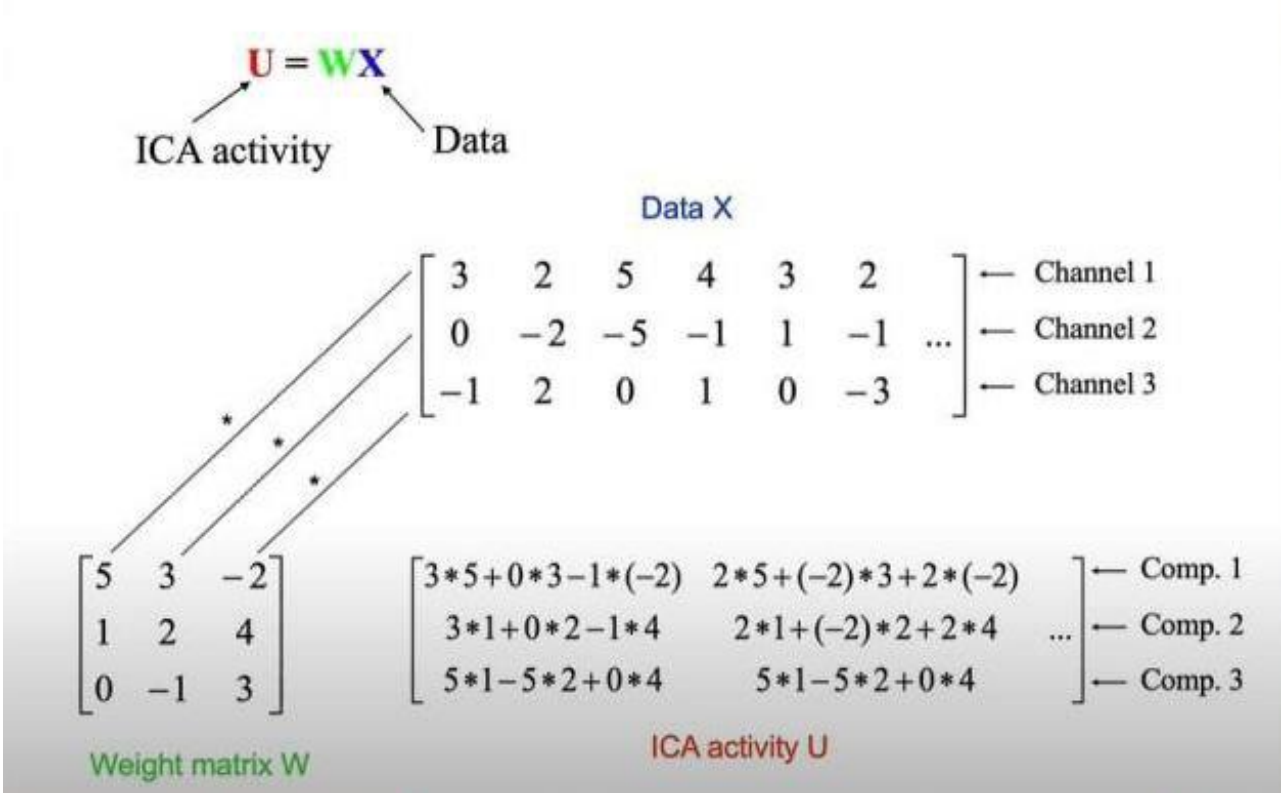


Figure 40. Applying ICA Process[41].

b. Feature Extraction:

I. ERD/ERS equation:-

The change in the signal power acquired from the motor cortex can be observed after applying the bandpass filter as a preprocessing step then the ERD% equation (1) can be applyed [36].

$$\mathbf{ERD\% = 100 *(A-R) / R \quad (1)}$$

A is the signal power estimation for a specified channel near the motor cortex, and many methods can be done as an estimation of the signal's power such as FFT, Welch, or even simply squaring each sample of the signal. R is the mean signal power at the reference period where the subject is not thinking about anything.

Welch's method as a power estimation is preferable to Fourier and just taking the square of the signal's samples. Welch's method is a technique developed by Peter Welch for estimating the power spectral density (PSD) of a signal using periodogram averaging. The PSD represents the distribution of power over different frequencies in the signal, while the periodogram calculates the squared magnitude of the Fourier transform of a signal. However, the periodogram is very sensitive to noise and variations in the signal, leading to inaccurate PSD estimates. To overcome this, Welch's method divides the signal into overlapping segments, calculates a periodogram for each segment, and then averages the periodograms to obtain a smoother estimate of the PSD. This approach is commonly used

in signal processing and spectral analysis applications, such as speech recognition and image processing.

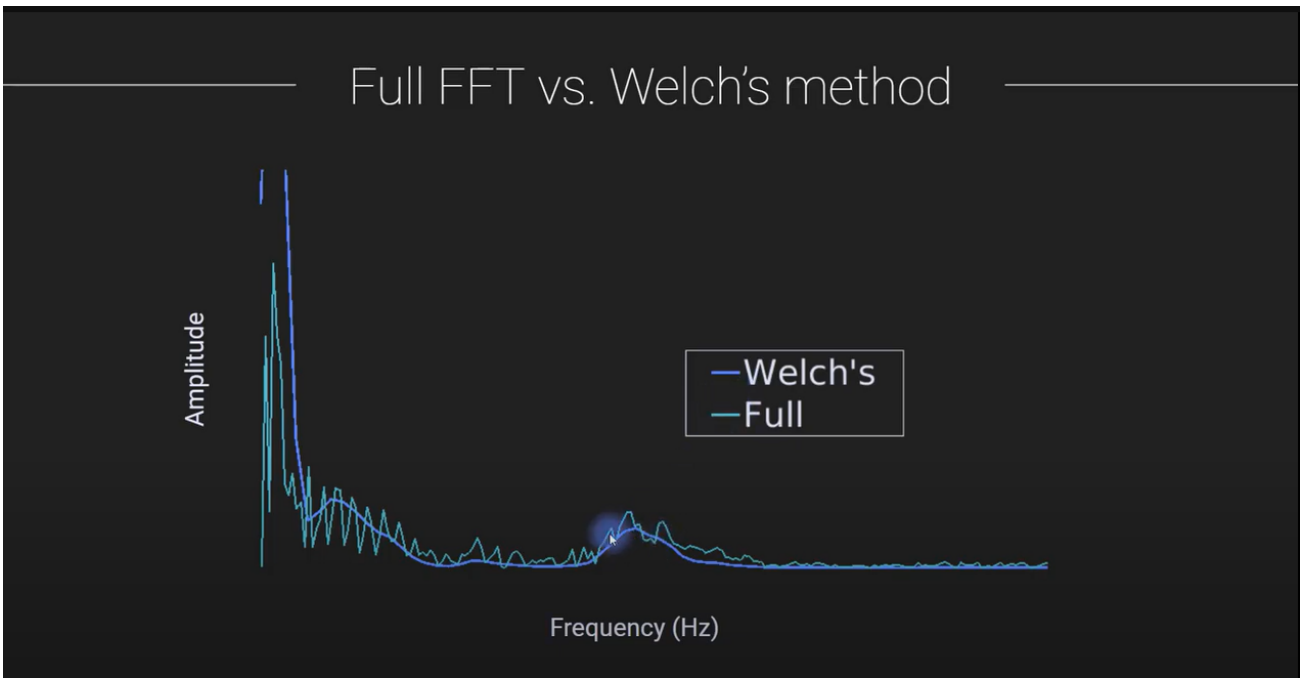


Figure 41. Fourier Trasform vs Welche's method[42].

Applying the mentioned method on BCI competition IV dataset 2A gave the predicted result with an accuracy of 82% for right-hand movement and 57% for left-hand movement. The accuracy was calculated as sensitivity where the number of trials that succeeded in showing the correct difference was divided by the total number of labeled trials, for instance, 82% means 82*45/100 = 37 trials that showed right-hand movement. Figure 42 shows the window that is used to segment the signal by using an overlapped 200-Hamming window that looks like a Gaussian distribution with 50% overlap. So the signal segment will be multiplied by the window and Fourier transform will be applied and the mean power will be calculated. The sides of the window will be zeroed, so there will be no effect coming from the signal on the sides. This is why Welch's method depends on the overlapping window.

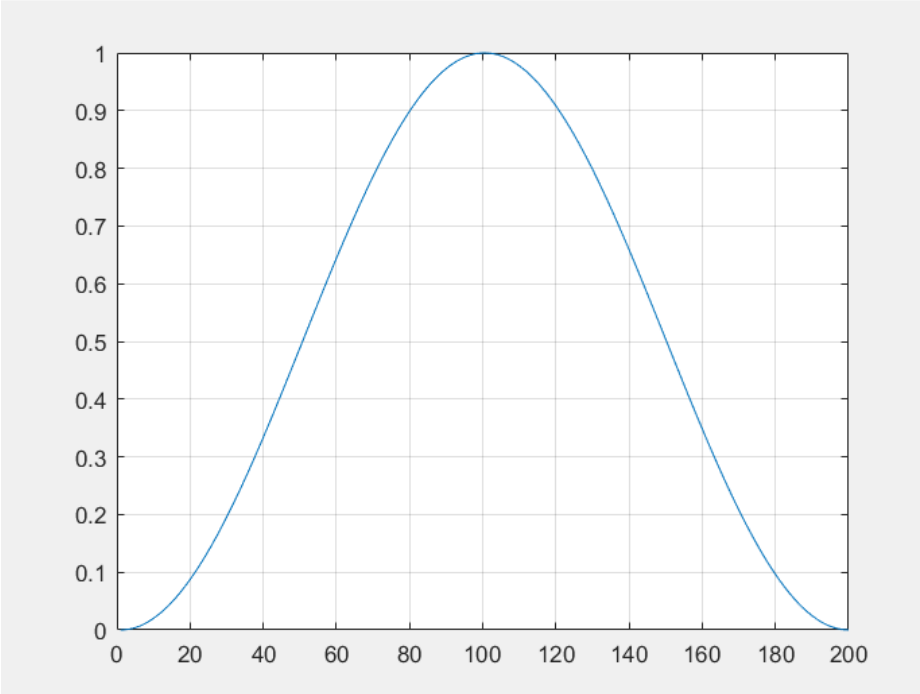


Figure 42. 200 Length Hamming Window.

Figure 43 shows left-hand movement it is obvious that there is a power depression in the signal acquired from C4 which is a channel located at the right side of the motor cortex and this is called event-related desynchronization. C3 is dominating which is located on the left side and this is called event-related synchronization. Also, there is another peak at a frequency that is twice the frequency where the biggest peak occurred and this is called a harmonic. Also, normalization can be done to show the ERD or ERS presence. Since the method is based on comparing the signal from two channels, normalization is done by dividing both signals by the highest value that exists from the first or second channel.

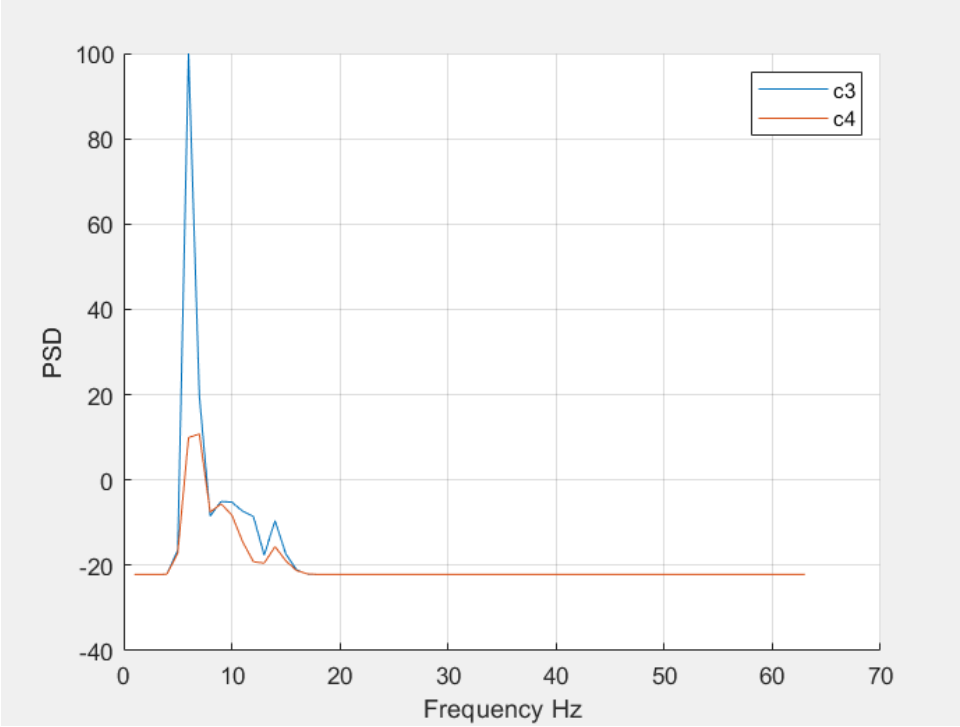


Figure 43. ERD% of Left Hand Movement.

Figure 44 shows the ERD% of right-hand movement and it shows completely the opposite with the harmonics. Common average reference and surface laplacian that are spatial filters did not have that much of a positive effect on the signal, so they were eliminated.

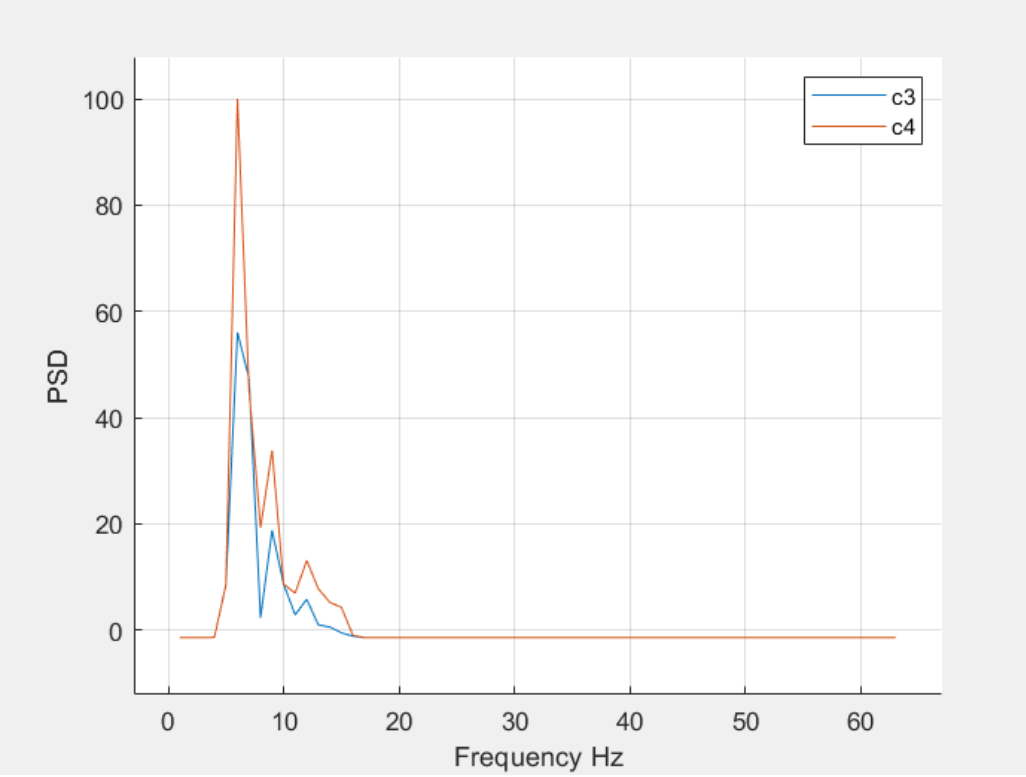


Figure 44. ERD% of Right Hand Movement.

II. QSA:-

Quaternion-Based Signal Analysis (QSA) is a suggested method for feature extraction in motor imagery. According to Batres-Mendoza et al most of the feature extraction methods that are used to analyze the EEG signal by transforming the signal to the frequency domain this causes the signal to lose important signal features [14]. With the already low signal-to-noise ratio, the lost signal features are of a high cost. For this reason, QSA is based on the time domain and it does not require filters to eliminate the noise or identify a specific frequency band. QSA is an offline method that can be applied in real-time, but with a high reduction in the accuracy of the system as it was shown in [14]. A quaternion has two components a scaler and three imaginary numbers. When the scaler is equal to zero it is classed as a pure quaternion. First, according to Batres-Mendoza et al, the channels closest to the motor cortex will be used to create the quaternion. 1200 samples should be used to represent a trial of the right-hand or left-hand movement imagination and 600 samples for waiting time [43]. FC5, FC6, P7, and P8 are the channels that were used by [43], so 1200 quaternions will be generated from each trial; FC5 will be a scaler and the rest channels will be imaginary numbers, so a matrix **q** with 1200 rows and 4 columns for each trial will be created for right or left-hand imagination [43]. **q** is used to define the rotation of a vector **r** which is a pure quaternion. **r** is a time-shifted vector by **dt** and according to [43], 4 multiples of 7.8ms as a time delay (**dt**) is suggested to be used.

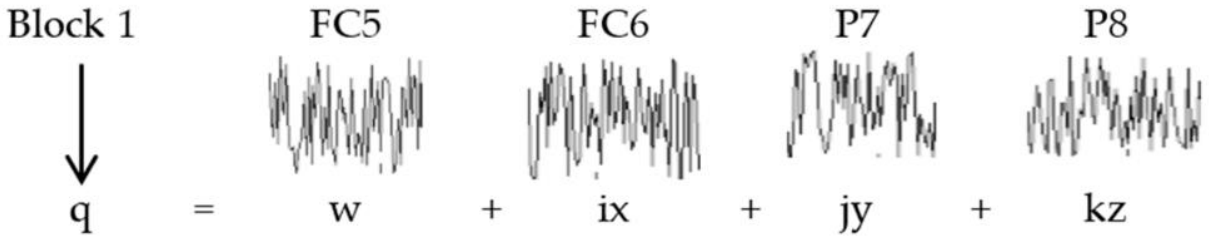


Figure 45. Generating a quaternion from an EEG signal [43].

4 features will be used to describe each trial from each class, the norm should be calculated for each quaternion by taking the square root of the quaternion components squared. Figure 46 shows the algorithm that should be built to apply QSA. Figure 47 shows the equations that will be used to create the 4 features, and feed them to the classifier to predict the class of the given signals.

1. Inputs: dt, signals, nblocks, pr
2. $y(t) \leftarrow$ segments of signals
3. quat $\leftarrow$ signals (nblocks)
4. For each $y_i(t)$ do

a $q(t) \leftarrow quat(t)$

b $r(t) \leftarrow quat(t\text{-}dt)$

c $q_{rot}(t) \leftarrow n_{rot}\ (q(t), r(t))$

d $q_{mod}(t) \leftarrow mod(q_{rot}(t))$

e $M_{i,j} \leftarrow f_j\ (q_{mod}(t))\ \{j = 1, \ldots, m\}$

f $c_i \leftarrow \{\ c=(1,2,3,\ldots,n) \mid y_i(t) \in c\}$

Figure 46. QSA Algorythm [43].

Statistical Features	Equation
Mean (μ)	$= \frac{\sum (q_{mod})}{N}$
Variance (σ^2)	$= \frac{\left(\sum (q_{mod})^2 - \mu\right)^2 + \sum (q_{mod})^2}{2N}$
Contrast (con)	$= \frac{\sum (q_{mod})^2}{N}$
Homogeneity (H)	$= \sum \frac{1}{1 + (q_{mod})^2}$
Cluster Shade (cs)	$= \sum (q_{mod} - \mu)^3$
Cluster prominence (cp)	$= \sum (q_{mod} - \mu)^4$

Figure 47. QSA Features [43].

III. IQSA:-

Improved Quaternion based Signal Analysis mainly differse than QSA in the classification part and the single trial is segmented by a sliding window. Figure 48, shows the sliding window in IQSA and how the vector r is rotated based on q and the norm is calculated.

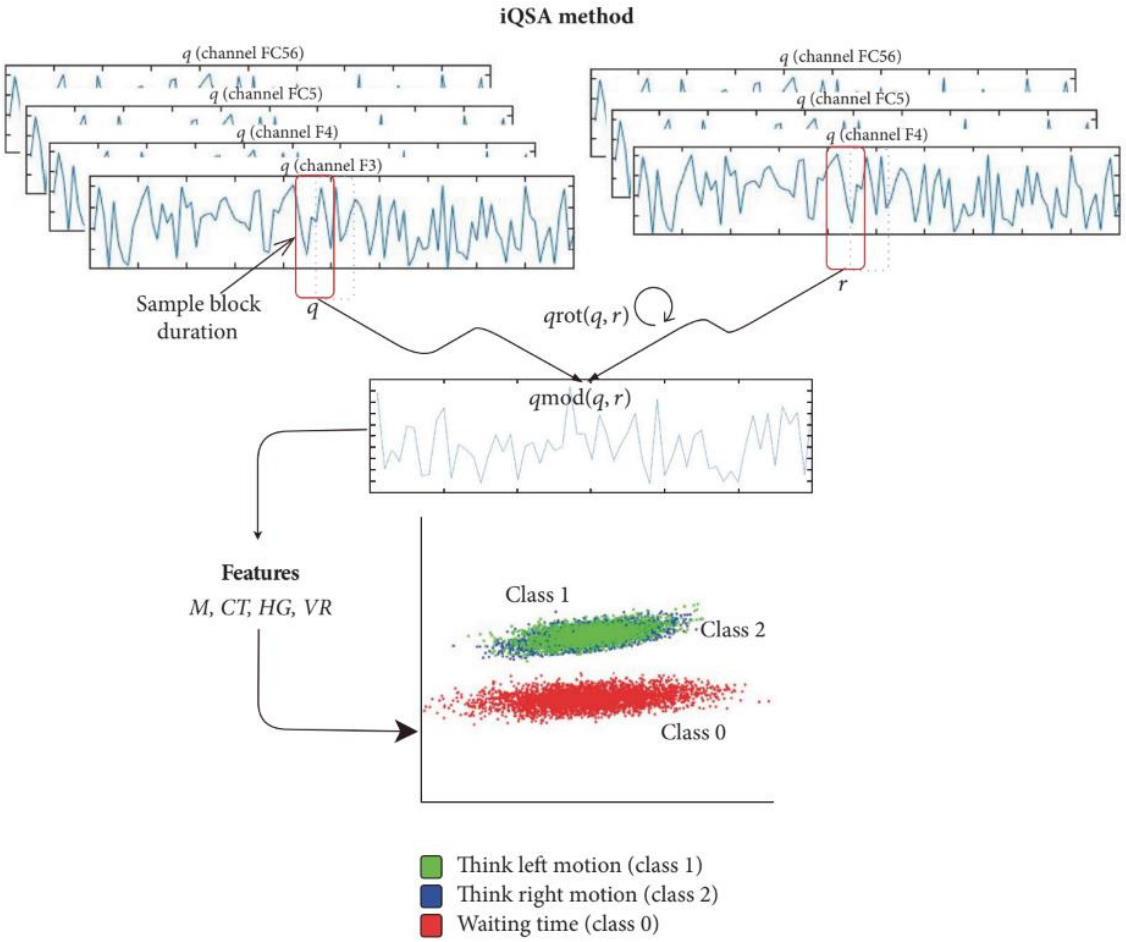


Figure 48. IQSA Feature Extraction [14].

IV. CSP:-

Common Spatial Pattern is a feature extraction method that can diffrentiate between two classes only, however additionsl steps can be taken to make it useable as a multiclass method. CSP and its variants is the method that won most of the BCI competitions [44]. The main idea behind CSP is to make the variance of one class maximized in one direction and minimize it in the other direction and completely the opposite for the other class. Figure 49 shows how CSP changes the spread of the data that belong to two different classes and it makes it easier to differentiate between these two classes. To apply CSP first a band-pass filter should be used since CSP is highly affected by the signals noise, and as it was motioned that the most important changes in the signal occurs in the range between 8 and 30 Hz. After filtering the data, the covariance matrix from each class and from each trail will be calculated, then one covariance matrix will be calculated for each class, and will bone through averaging the covariance matrices across all the trials for each class.

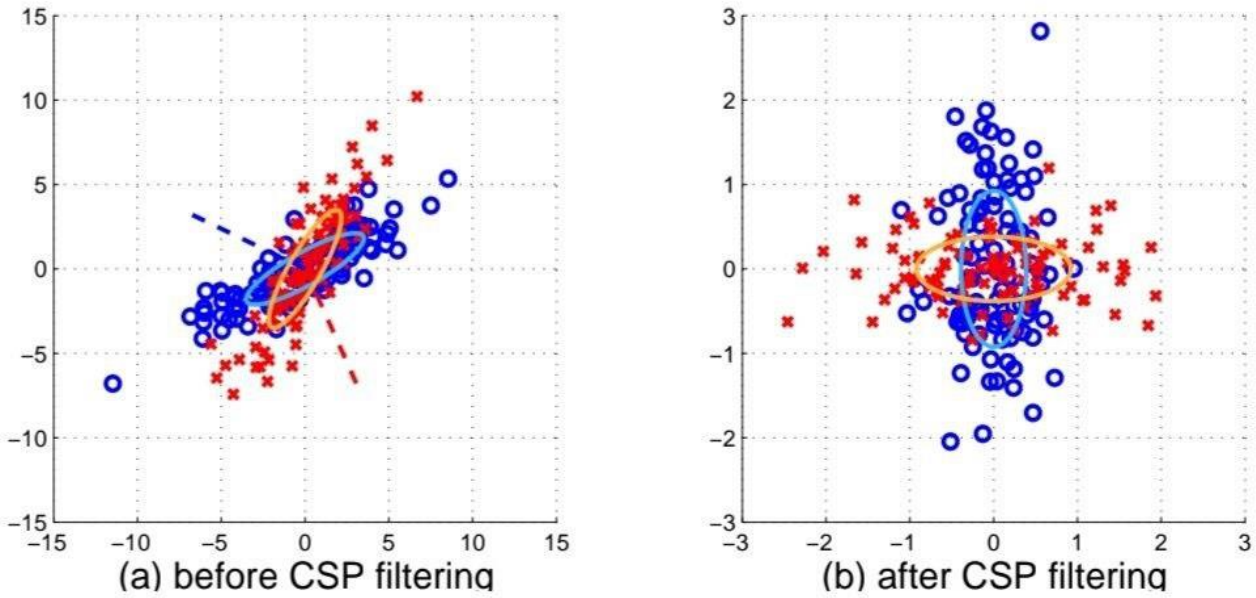


Figure 49. Before and After Applying CSP [45].

3.1.5.3. Training sessions design

In order to start taking EEG records an experiment need to be designed 5 different letreture reviwes where taken into account during the process of desighning the experiment. In the experiment desighned by *Choi & Cichocki* the subject sat in front of a monitor and a 5 second duration trial was recorded [33], also BCI competition IV dataset 2A suggested the applied similar procedure [35]. The trial was devided into two segments; two seconds of a blank page and then an arrow indicating the direction, so if the arrow was pointing to the right the subject had to imagen right hand movement [33] however the time between the trial was not mentioned. Also *Ramoser et al* used the same idea of showing arrows to the subjet as an indication of the direction, but with different segments duration, also it was suggested a nine second interval between classes[34]. It was illestrated that the trial segments durations has an impact on the systems accuracy [36] tested two different segments durations and observed an increase in the systems accuracy when the the recording duration was nine seconds with six seconds of imagenation.

Cortex API code was modefied to record for 2 classes beside a reference class for each trial and that was done by injecting a marker at the onset of each class and then showing an arrow or an image showing the onset of the reference class just as it was done by the mentioned literetures. As shown in figure 36 the trial starts at t=0 with an arrow pointing to the right which means that the subject should start imagining right hand movement, then a count from 1 to 3 will start to prepare the subject for imagining after that at t=5 a marker will be injected and the subject will imagen right hand

movement for 5 seconds until t=10 an image indicating the end of the first class. At t=15 another marker will be injected then for 9 seconds the subject will not imagen anything until t=24 an arrow indincating the onset of class 2 and at t=29 a marker will be injected, and finally the trial will end at t=34. Figure 37, also show that all the trial have the durations.

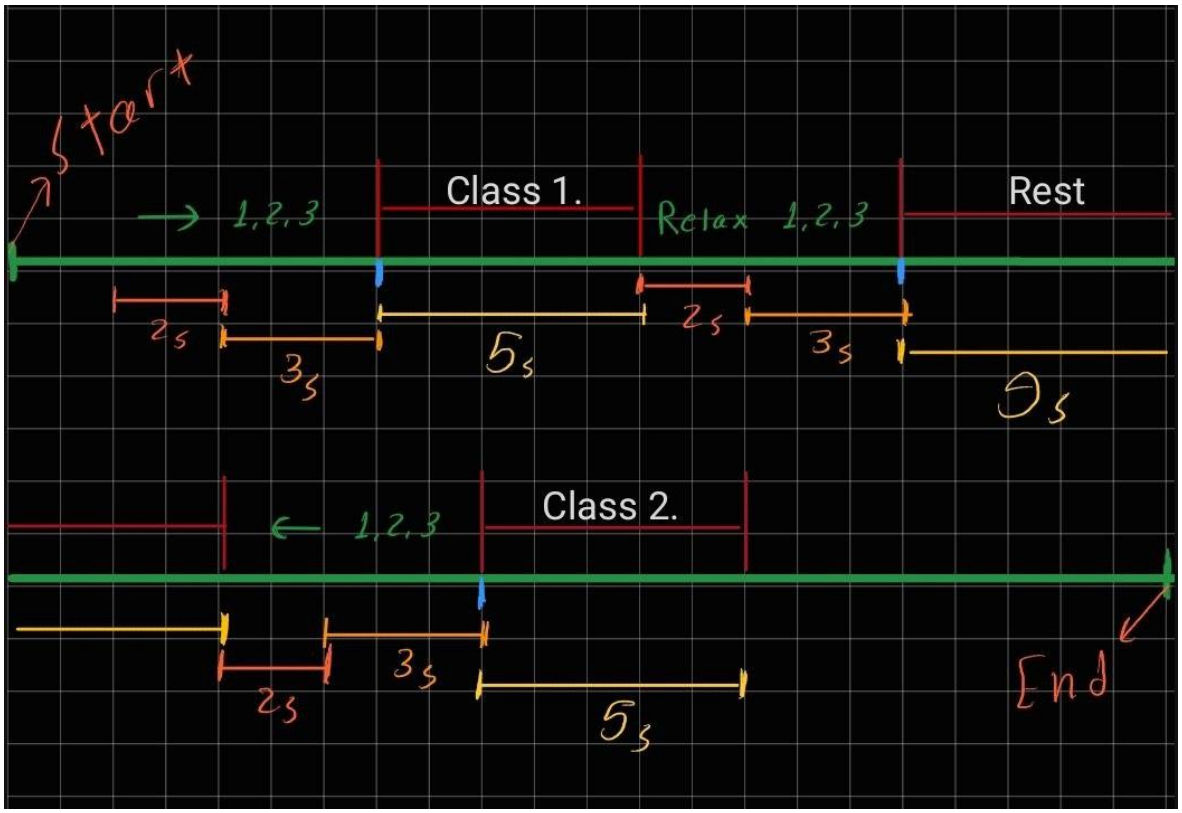


Figure 36. Trial timeline

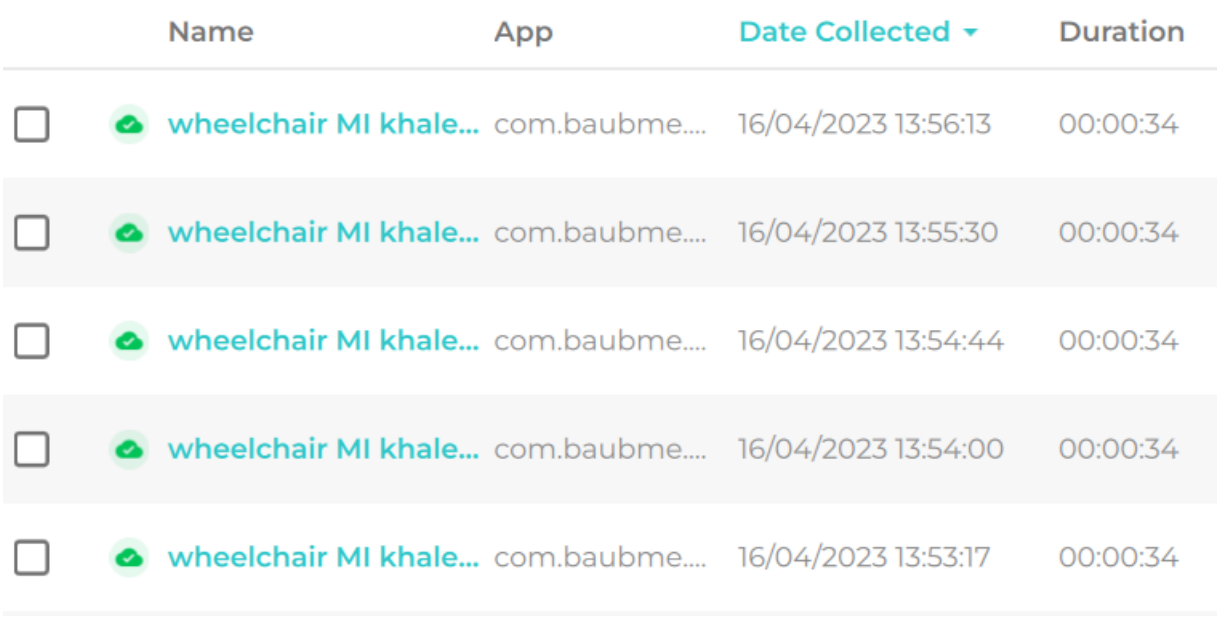


Figure 37. Trials duration

3.1.5.4. Realtime Topographical mapping

Since EmotivPro is compatible with LSL streaming the data in realtime from the headset to matlab is possible by using the matlab scripts that are published at [37] and with the appropriate modifications it can be used for the topographical mapping. Also the matlab script that will create the topographical map is created by [38]. The data will be captured for two seconds and saved creating a matrix with 265 samples and 32 channels an 8th order Butterworth bandpass filter is used to filter the data from 8 to 13Hz since this is the range that ERD/ERS will occure at. Welch's method was used with a 100 Hasnning window length and 75% overlap to estimate the signals power spectrum density. To create a topographical mapping a single value for each electride need to be used so the mean power was calculated across all the samples for each channel. Figure 40 shows the generated topographical mapping and the graph will be updated automatically every 4 seconds.

The subject was asked to close his eyes to see if the power will be dominant at the occepital lobe to make sure that the chosen parameters will give a good prediction to the power spectrum density which exactly what happend. However the applied method did not succeed in showing the ERD/ERS effect in a semi-Realtime way. Also different methods were applied on the recorded data to observe the ERD/ERS effect and all the tested methods failed to do so without the application of additional feature extraction methods such as CSP.

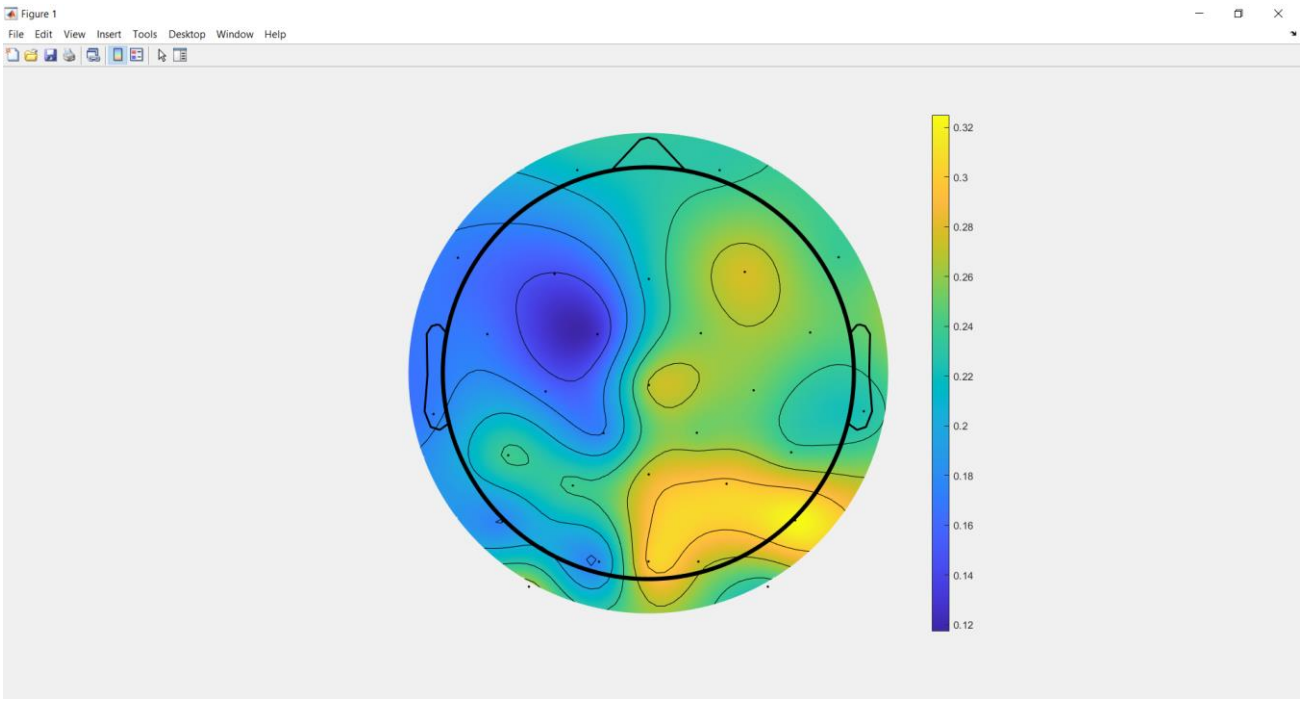


Figure 40. Semi-Realtime topographical mapping

3.1.6. Evaluation

To evaluate each of the suggested methods both visual inspection and built in tools in Matlab are used to evaluate the feature extraction methods. Starting with the ERD% equation after applying it on the recorded data we saw that most of the time there the power of the left side of the barain is lower than the right side. This means the subject is always imagening his right hand is moving. Figure 50 shows the ERD% of both right hand and left hand movement imagination, and it can be seen both can be predected as right hand which is wrong and for that reason the ERD% equation was elemenated. IQSA was tested by using the classification learner which is a toolbox in Matlab, and the accuracy on average did not exceed 55% which is not effecient to control a wheelchair directed by the subjects brain.

CSP was also tested and it can be seen from figure 51, that when the subject imagened right and left hand movement through imagening pictures and the validation accuracy increased by almost 10%. CSP when tested on new data the classification accuracy droped which may indicate overfitting of the system. And this problem according to Lotte et al, 2010 CSP is very sensitive to signals noise and overfitting is possible to occure if the data was not filtered appropriately.

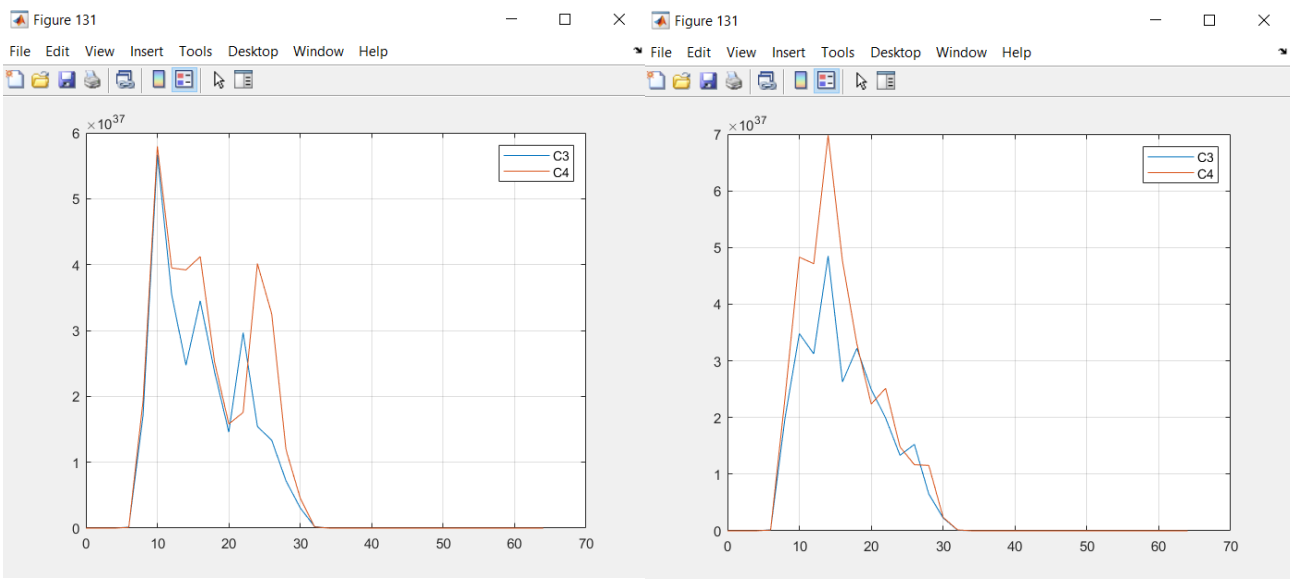


Figure 50. ERD% of Left Hand Movement (Left) and Right Hand Movement (Right).

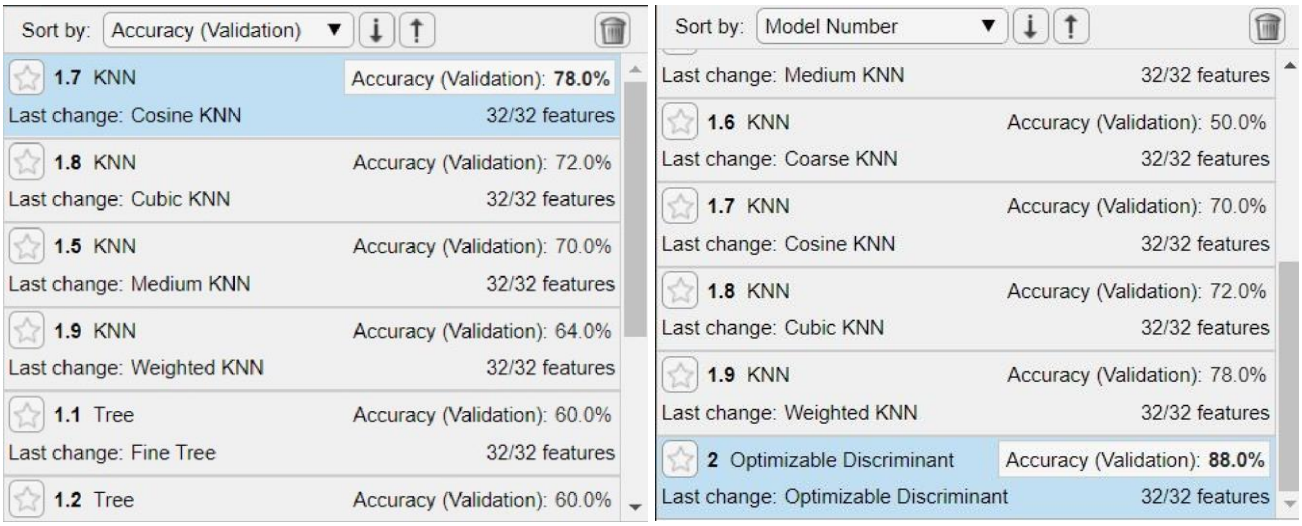


Figure 51. Validation accuracy without pictures (Left) Validation accuracy with pictures (Right).

Figure 52 shows the validation and testing accuracy of right vs left hand movement, and the drop in acuracy is obveious, so ICA was used to filter the data to reduce the noise and then test the systems accuracy. Figure 53 show a comparison between the signal before and after filtering the data. EEGLAB is a toolbox in Matlab and it was used to apply ICA and automatically reject the noise components. And even that did not solve the overfitting problem and it is still happening, maybe the signal does not has enough features or the classifier parameters should be modefied and this will be discussed in the computer engineering part.

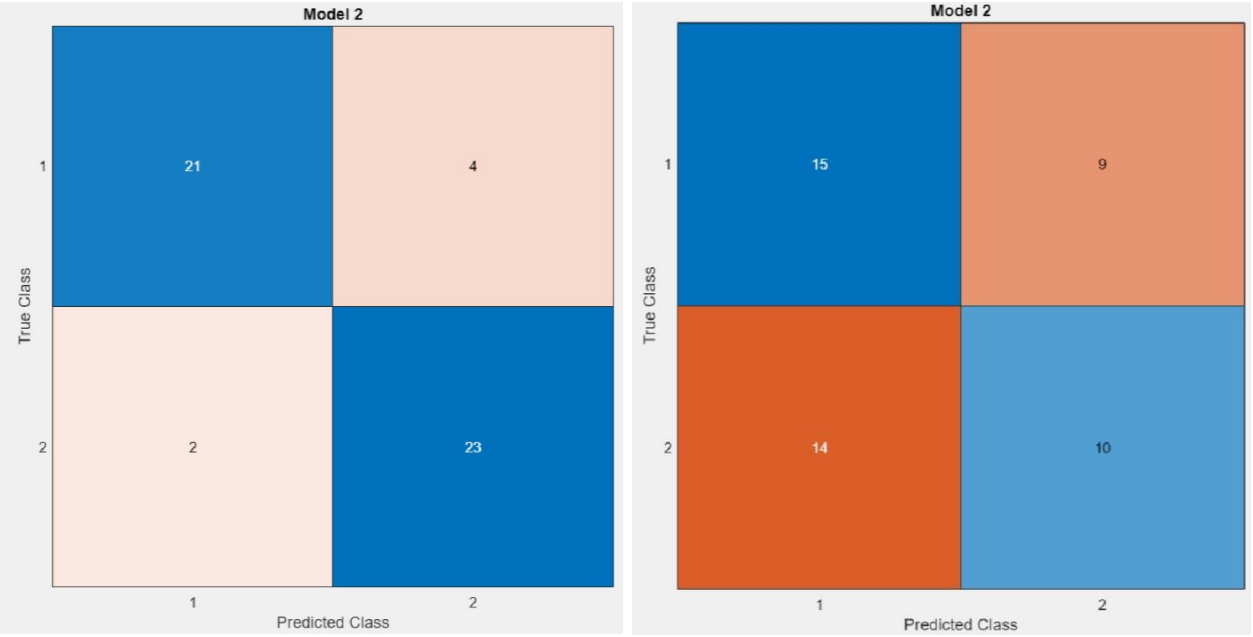


Figure 52. Validation accuracy 88% (Left) Testing accuracy 52% (Right).

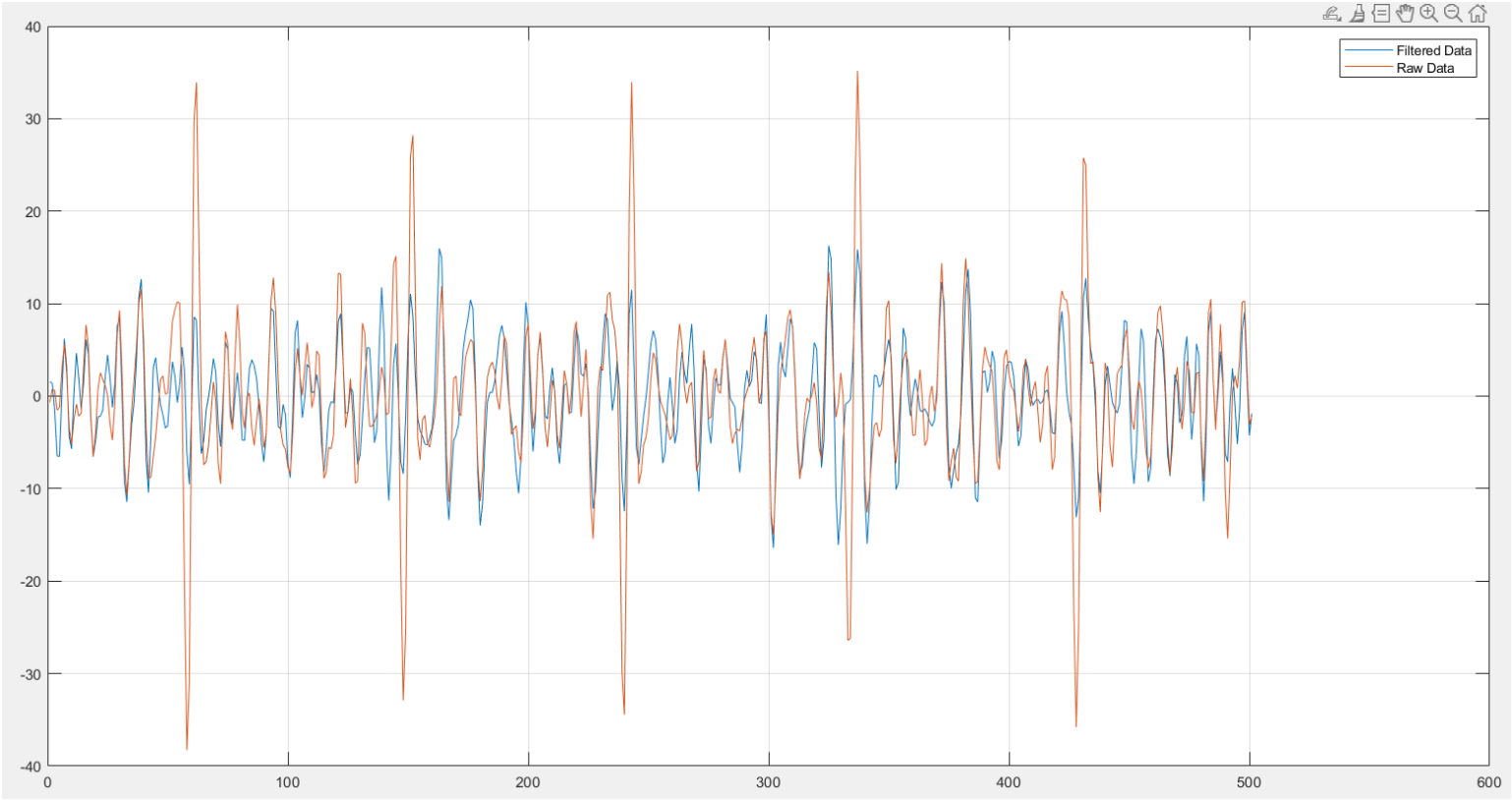


Figure 53. The signal before and after filtering by using ICA .

3.2. Mechatronics Engineering

This sub-team consists of 3 mechatronics engineering students, who will be responsible for design, analysis and construction of the wheelchair. Construction of the wheelchair consists of integration of the motor, braking, steering and electromechanical systems. These systems include design and integration of the battery, fuse board, sensor, wheel and microcontroller components.

3.2.1. Requirements

To set up an EEG-controlled wheelchair environment, proper preparation and fine analysis are required. To be fully prepared, the assignment of tasks is vital. Every member must know their responsibilities, and what they can support in the project. To prepare a good project, analysis and simulations are also required to scope the failure and success conditions of the project. A qualified team has to have access to modeling, testing, and assessing the risks and boundaries.

Communication between departments is extremely important for creating a healthy plan. Interaction between Mechatronics, Computer, and Biomedical Engineering decreases the possibility of failure in intersecting work packages. Also, different departments contain different approaches which can benefit every aspect of the project. The system requirements are designed to be tolerant, measurable and well described and clear with the limitations.

3.2.1.1 Functional

The system needs to have a double DC Motor. With the double DC motor, the vehicle must turn left and right, accelerate and stop. Also, the wheelchair requires a 12V battery system for power distribution. There have to be at least three ultrasonic distance-sensing sensors to detect obstacles. The energy obtained from batteries must be regulated by a circuit to provide the required voltage of 5V for components.

3.2.1.2 Safety

For safety, correct wiring in dimensions that can give the required current is required A fuse box must be designed and built to protect the system from situations that may be caused by a short circuit.

3.2.1. Performance

Also, the wheelchair must not exceed the 14-15 km/h speed limit to maintain safety while moving.

The sensors must identify an object from, at least, 30 centimeters. When turning, the motors rotate in opposite directions. The voltage regulator must convert the incoming voltages to 5V.

3.2.2. Technologies and methods

In this experimental design, technologies such as modeling software (Solidworks) and simulation tool (Solidworks Simulation Environment) and circuit design tools (Altium), Ms projects are used. The microcontroller program is written with C and C++ language using Arduino IDE.

In the context of semi-autonomous BCI (brain-computer interface) based wheelchairs, mechatronics engineering plays a crucial role in the design and implementation of the various technologies and methods used to enable the chair to respond to the user's brain activity.

This may involve the development of specialized hardware and software systems for measuring brain activity using techniques such as electroencephalography (EEG) and electromyography (EMG), as well as the integration of machine learning algorithms and haptic feedback mechanisms to enable the chair to respond to the user's intentions in a natural and intuitive manner. Haptic feedback mechanism provide sensory feedback to the user, such as vibrations or pressure, to inform them of the status of the wheelchair or to alert them to virtual obstacles. The "Brain-Controlled Wheelchair for Paraplegic Patients" developed by researchers at the University of Maryland used a tactile display on the armrest of the chair to provide haptic feedback to the user, allowing them to feel virtual obstacles as they navigated through a virtual environment (Shih et al., 2011).

One of the key technologies used in electronic wheelchairs is the motor system. These devices use electric motors to power the wheels and propel the chair forward. The motors are typically brushless DC motors, which are known for their high efficiency, low maintenance, and long lifespan [30].

Another important technology used in electronic wheelchairs is the control system. This system allows the user to control the movement of the chair and access its various features. There are several methods for controlling an electronic wheelchair, including joystick, touchpad, and voice control [31] but we preferred to use BCI based control which involves the measurement of the electrical activity of the brain. The system is designed to control the wheelchair with five commands: move forward, move backward, stop, turn left and turn right in real conditions.

In addition to the motor and control systems, electronic wheelchairs also have safety features like the emergency braking system and fuse box to ensure the safety of the user.

Overall, the integration of these technologies and methods is essential for the development of effective and user-friendly semi-autonomous BCI based wheelchairs. These devices have the potential to greatly improve the independence and quality of life of individuals with mobility impairments, and mechatronics engineering plays a critical role in their design and development.

3.2.3. Conceptualization

The optimal prototype is a full-sized platform that carries a human and functions as a real system. However, due to budget limitations, a half–sized model is constructed for demonstrating purposes.

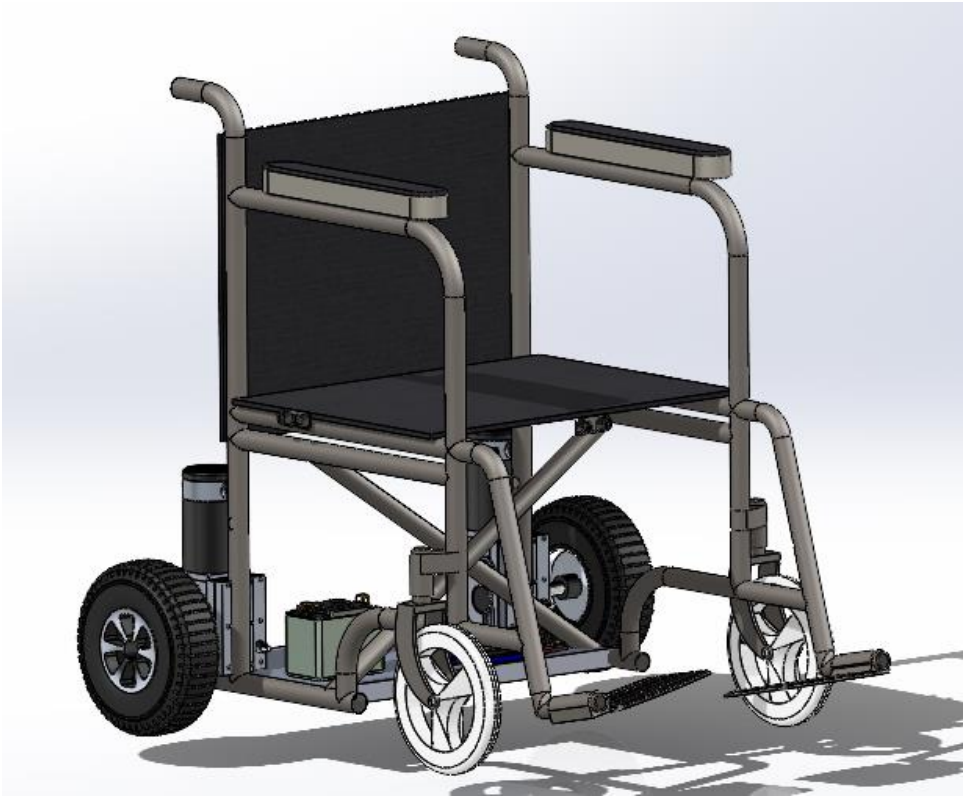


Figure 54. CAD drawings of the full-scaled system.

BAU TTO (Technology Transfer Office) did not support the project with the budget, so a small scaled prototype is constructed. The design has been created as a small-scale design, reduced by 35% of its original size, in order to protect the concept. The small design is produced with 3D printers, due to the advantages such as accessibility, mobility, and cost management. Regarding electronics, the smaller system uses cheaper motors, cheaper sensors, clone microcontrollers, budget-friend components, and

a smaller battery.

3.2.3.1 Plan A (Full-Sized Model)

Concept A will be considered if we have the BAU TTO (Technology Transfer Office) Scientific research project's budget and a full-sized prototype of the wheelchair will be constructed as a real system.

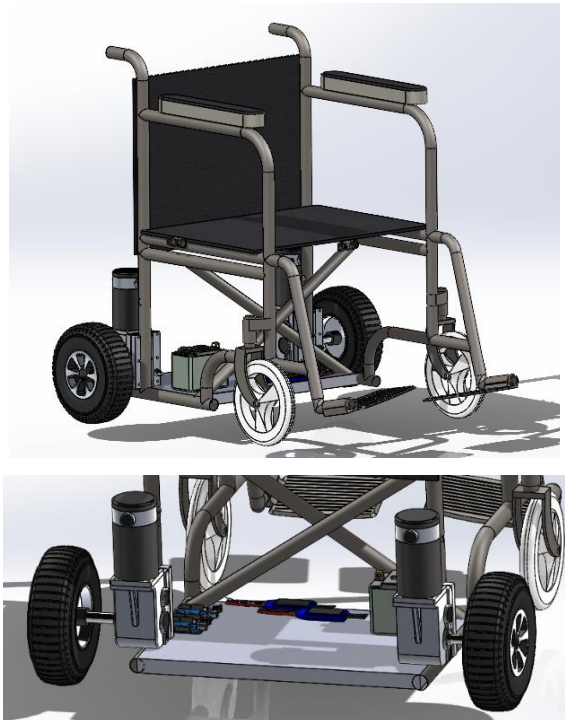


Figure 71. CAD designs of the system, with the part alignments.

3.2.3.1.1 Main Computer

For the main computer, a casual laptop with high processing technology (At least Intel I5 processor required) and high RAM capacity (at least 8 Gb) is required. A Raspberry Pi fails to operate with the necessary processing power, and PCs are too heavy, big, and expensive for assembling.

3.2.3.1.2 Microcontroller

The Arduino board is a microcontroller board based on the ATmega328 microcontroller. It has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. It is used for sensing inputs from sensors, controlling motors and other actuators, and communicating with other devices. It is also used for data logging, display,

and control interfaces.

Arduino Mega, Arduino Uno, and STM32 Nucleo were compared, and based on the values shown in the table below, it was decided to use the Arduino Uno.

Table 6 . Microcontroller comparison

		Accuracy	Cost	Complexity
	Arduino Mega	Medium	Medium	Medium
	Arduino Uno	Medium	Low	Low
	STM32 Nucleo	High	High	High

3.2.3.1.3 Distance Sensors

Since the infrared sensors cover less area, and ultrasonic distance sensors are more economical, different models of ultrasonic sensors are considered for the project. HC-SR04 is a quite cheap model that has a range of 3-400 cm, and URM37 is more expensive compared to HC-SR04, however, URM37 has superior range parameters (2-800cm), so URM37 is preferred for ultimate safety.

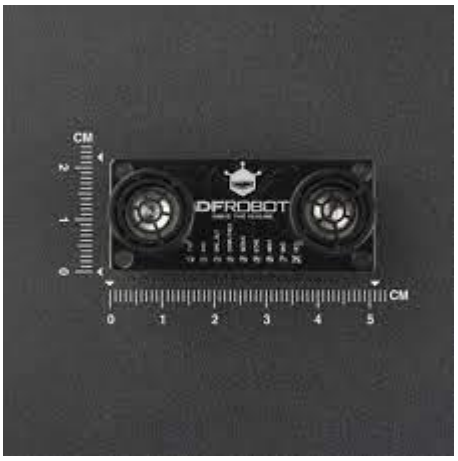


Figure 72. URM37 Ultrasonic Distance Sensor

3.2.3.1.4 Motors & Electronic Speed Controller

Motors are chosen from electric chair motors, scooter motors, and regular brushed DC motors. Electric chair motors have a wide range of Watt options in the market, and they are mechanically more compatible with the project, so in the regular-sized version electric chair motors are used.

Figure 73. EDEC82L2 Electric motors.

Electronic Speed Controllers (ESC)

While choosing ESC, one controller for both motors or two controllers for two motors are available as options. It is easier for wiring and cheaper for using to use one ESC which has two motor control capacities.

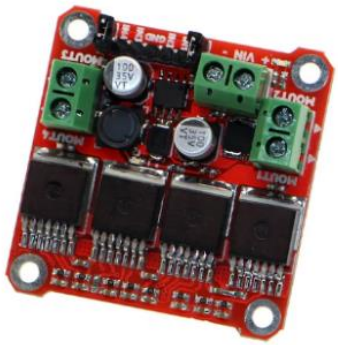


Figure 74. ESC

3.2.3.1.5 Battery System

The wheelchair requires a 12V or 24V battery system for power distribution. We can obtain a 24V battery by connecting two 12V batteries in series. Since availability is easier, the 12V Lithium

single battery has been preferred over the 24V battery.

Table 7. Battery comparison

	Capacity	Cost	Compatibility	Discharge Current
Lithium Battery 24V	High	Moderate	Moderate	Moderate
Lithium Battery 24V	Moderate	High	Moderate	Moderate
Lithium Battery 2X12V	Moderate	Low	High	Low

Figure 75. PLAN A, 12V Lithium Battery

3.2.3.2 Plan B (Small-Scaled Prototype)

Concept B will be considered if we do not have the BAU TTO (Technology Transfer Office) Scientific research project's budget and a small-scaled prototype of the wheelchair will be constructed which protects all the necessary data and shows how the real system would be worked.

For the smaller budget model, HC-SR04 sensors are used. This sensor has less range gap (3-400cm), however the component is quite cheap, making it a budget-friend part. Two Arduino nano were used for small scaled prototype. Motors in this system are much weaker and smaller due to the reduced size and weight parameters. Brushed DC motors are chosen due to reducing the cost. Raspberry Pi is used for main computer instead of a laptop. One ESC is used for motors. Instead of purchasing a separate voltage regulator, we integrated it directly into the fuse circuit. 12V Lithium Battery is used for power distrubution.

Table 9. Comparison of concepts

Alternatives	Full-Scale System	Small-Scaled Prototype
Electronics	Regular-sized electric chair components.	Minified electric components.
Mechanics	Regular sized structure.	%35 of the original size.
Cost	Bigger, stronger parts,expensive system.	Smaller and weaker mechanics and electronics, cheaper parts.

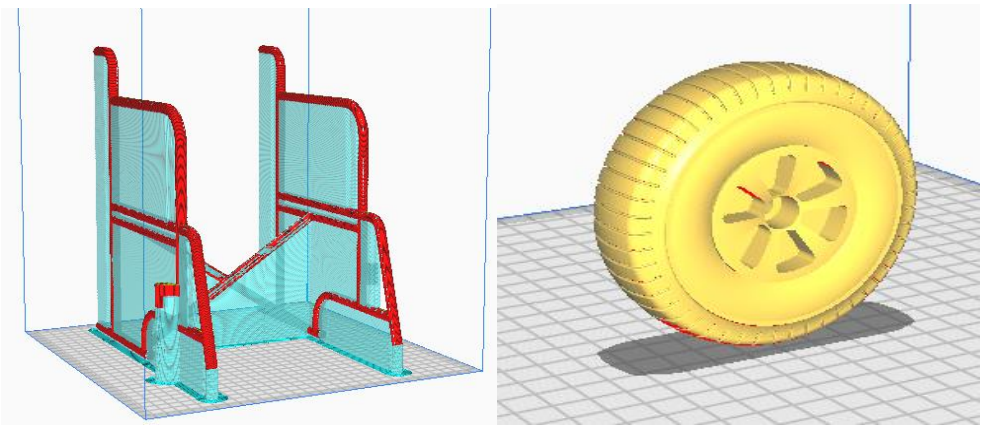


Figure 76. Minimized models in the 3D printing interface.

3.2.4. Physical architecture

An electromechanical system refers to the physical connection between electrical and mechanical components. The physical connection of the electromechanical system includes power and electronic distribution management, distance-sensor suite and battery system integrations, etc. Lithium batteries will be utilized in the power distribution system to supply specified components of the system and two DC motors. The energy obtained from batteries will be regulated by a circuit to provide the required voltage of 5V for components. Also, two microcontrollers (Arduino Uno) will be connected to the laptop or Raspberry Pi. One of the microcontrollers in this system is responsible for controlling the motors, and the other's task is to manage the data received from the distance sensors.

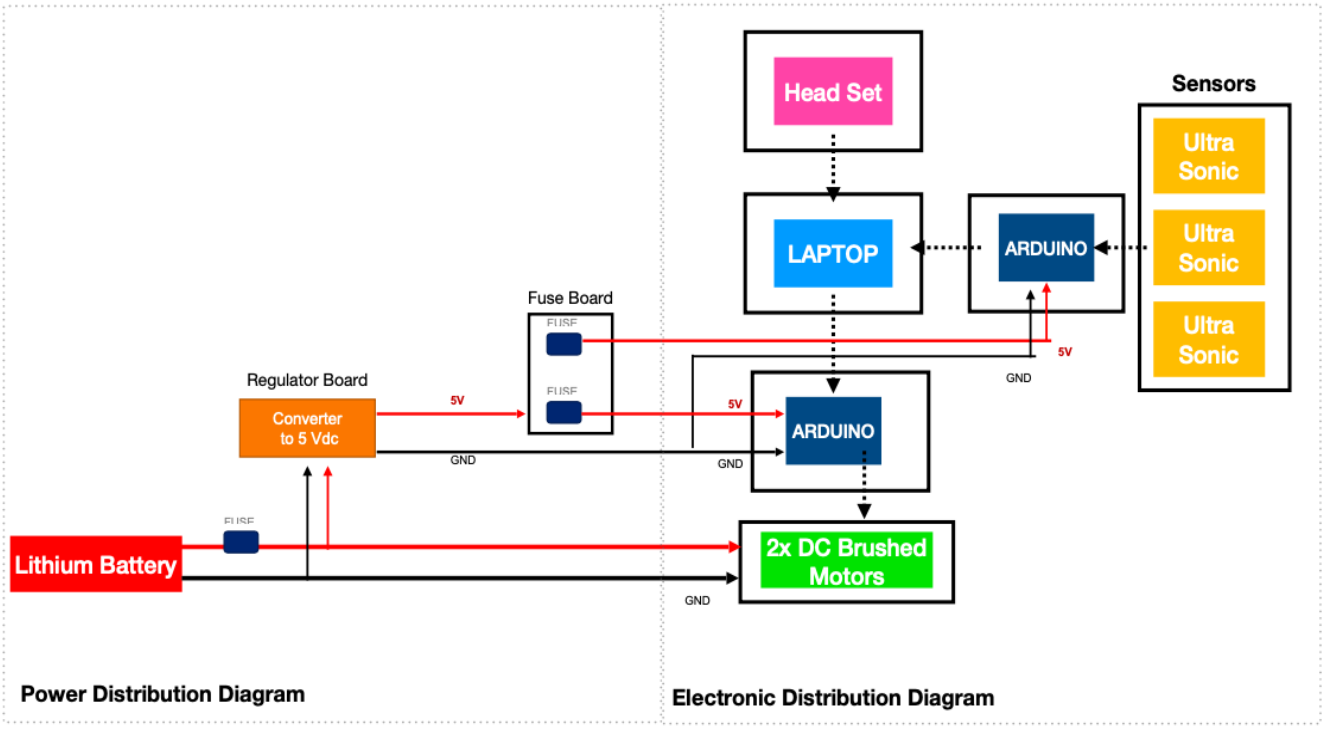


Figure 55. Power And Elecronic Distribution Diagram of system

3.2.4.1 Fuse Board Design & Manufacture

The Fuse box is designed to protect the system from situations that may be caused by a short circuit. The diagram below shows the detailed connection of three ultrasonic sensors—one on each side and in the front—and two microcontrollers with the fuse board in detail.

Thanks to this fuse board, if one of the components fails or short-circuits, the entire system does not shut down, burn out, or become dangerous. Only the fuse responsible for that component will burn and the system will continue to operate after the fuse is changed.

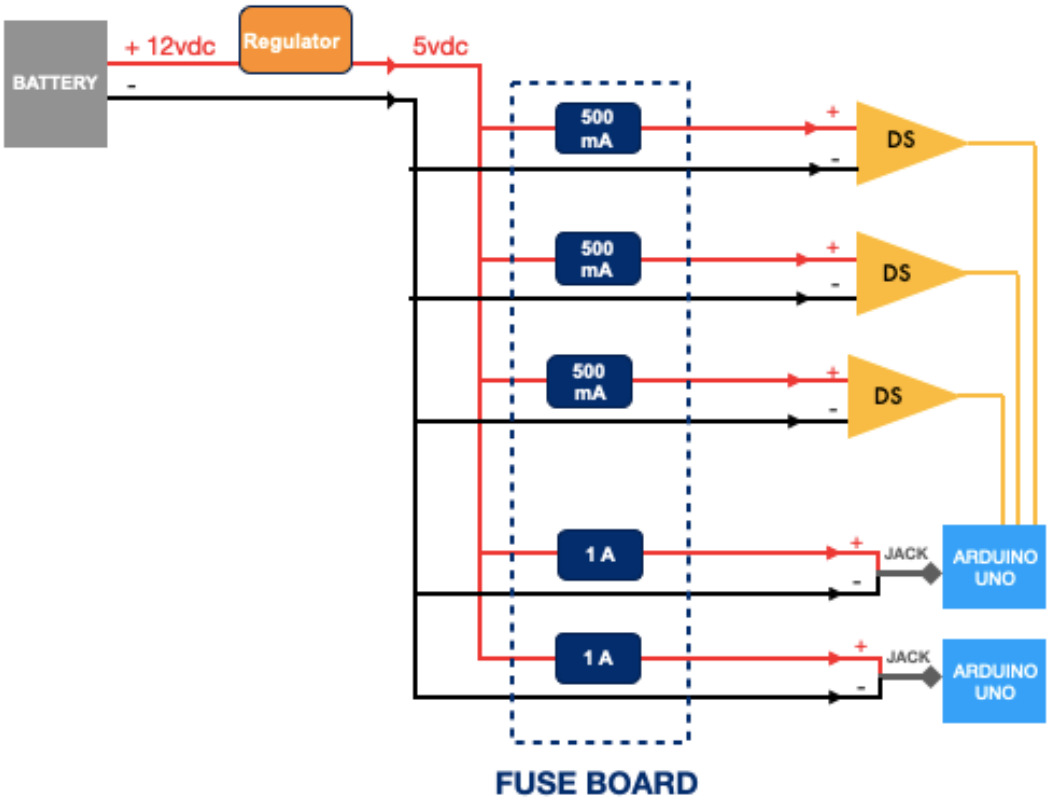


Figure 56. Fuse Board And Electronic Distrubition Schematic

The current drawn by the Arduino Uno and the distance sensors were calculated, and fuses with appropriate values were inserted on the fuse board. If one of the components faces a short circuit condition, the current used by the component will go to infinity and just the fuse that corresponds to it has to be changed.

3.2.4.2 Voltage Regulator Implementation

The energy obtained from batteries will be regulated by a voltage regulator to provide the required voltage of 5V for microcontrollers and distance sensors.

3.2.5. Materialization

BAU TTO (Technology Transfer Office) scientific research project's budget is not used. So, small-scaled version of the vehicle is built and the components in concept B as mentioned above is

used. BAU biomedical laboratory is used to make the motor system, wheel system, brake system, and electromechanical integrations of the wheelchair. We supplied some of the materials and produced some of them ourselves by printing them on a 3D printer. While waiting for the cargo, we started building the chair.

Figure 57: Chair printed on a 3D printer.

Figure 58: Front wheels printed on a 3D printer.

After the arrival of necessary materials, checks are made to ensure the integrity of the components. After the controls, the system was ready to be built.

Figure 59: Materials were procured and checks were made.

The building process started with the integration of back motors and motor driver. The 12 AWG supply cables are soldered to the motor and connections are made to the motor driver. After the adjustments on the motor driver are completed, Arduino Nano board was ready to be connected to the system for transferring data.

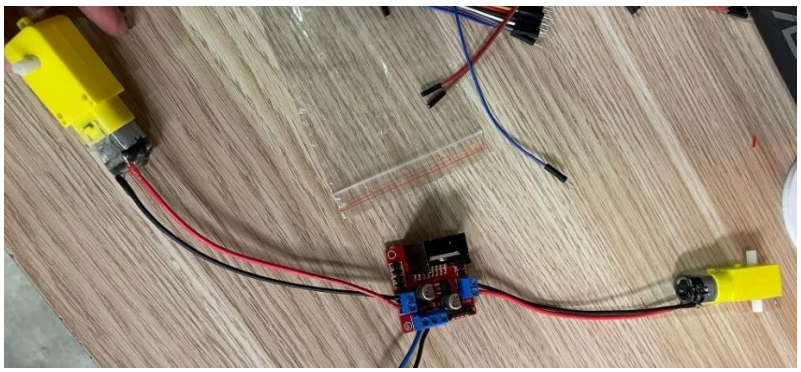


Figure 60: The supply cables soldered and connected to the motors and motor driver.

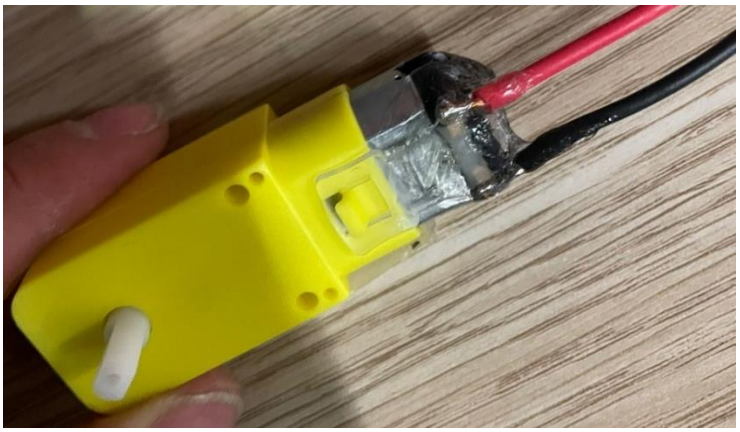


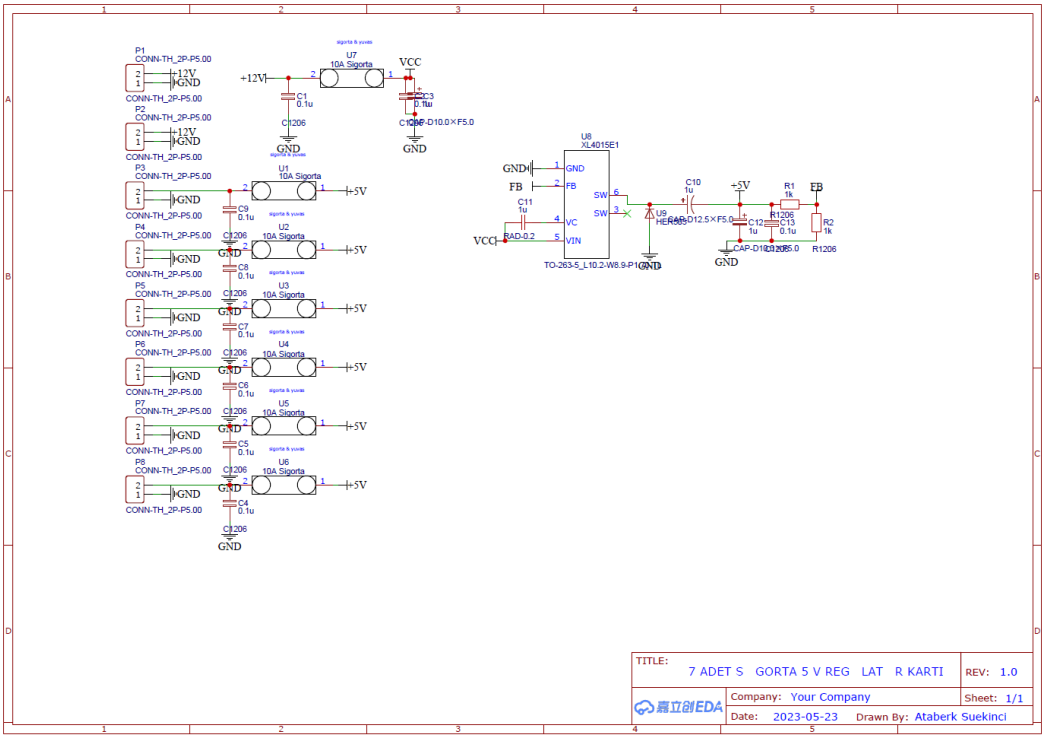
Figure 61: The supply cables of the motors were coated with spray epoxy, thus preventing the cables from breaking due to external factors such as bending. Both durability and safety were increased.

We detaily calculated ampers and powers that compenents needed.

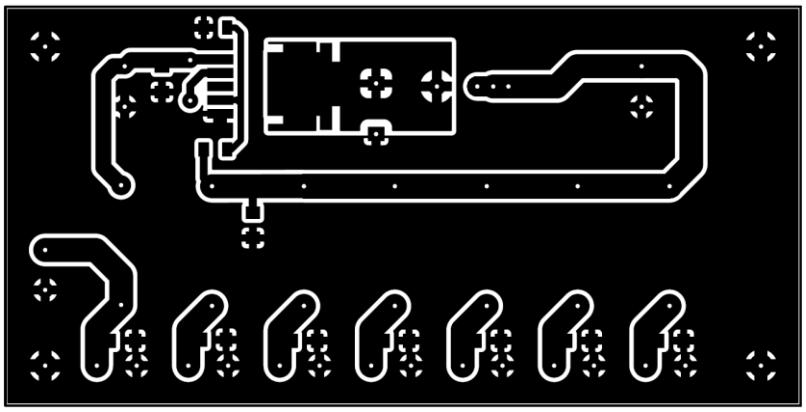
Piece	Component	Ampere	Voltage	Sum	Power(W)
2	Arduino Nano	160 mA	5 V	320 mA	1,6W
2	Motor	160 mA	3-12 V	320 mA	0,96-3,84W
3	Ultrasonic Sensor	15 mA	5 V	45 mA	0,225 W
1	Raspberry Pi	1 A	5 V	Max 2 A	5W (opp) 10 W(Max)
Total				2,85 A	

Table 8: Power calculation table of components

We designed the circuit diagram of the fuse board and voltage regulator ourselves. You can see the diagram in the figure below.



We have designed the layout for the PCB production of the fuse board and voltage regulator as shown in the figure below.



According to our design and diagrams, PCB is produced by etching process.

Etching process is a method used in PCB (Printed Circuit Board) manufacturing, where a copper-clad substrate is treated with chemicals to selectively remove copper and create circuit traces and patterns.

Figure 62: PCB produced by etching process

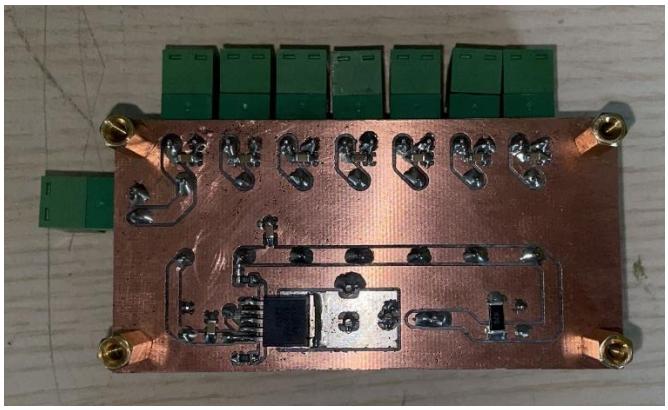


Figure 63: Back view of PCB

Due tou the the battery has not reached us yet, we used a power supply to provide power to the motors.

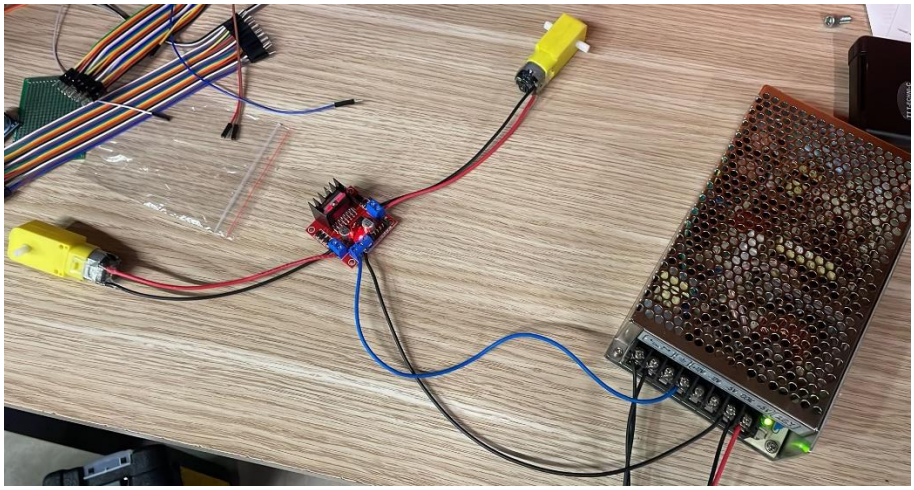


Figure 64:Motor Test

The motors and motor driver were connected to the power supply for testing. The motors were powered by 5V and the motor driver was powered by 12V.

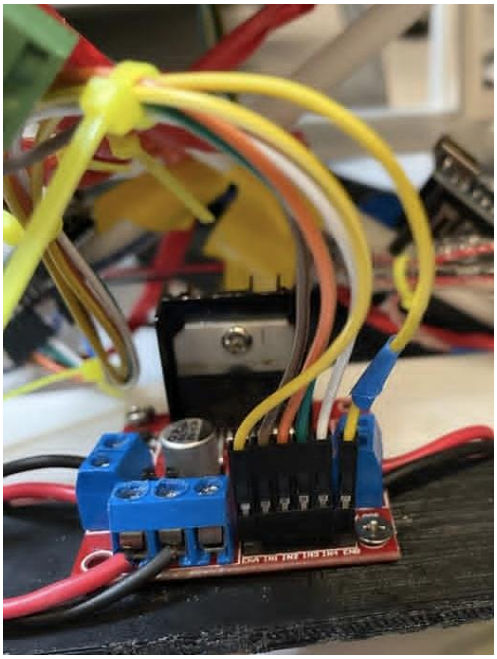
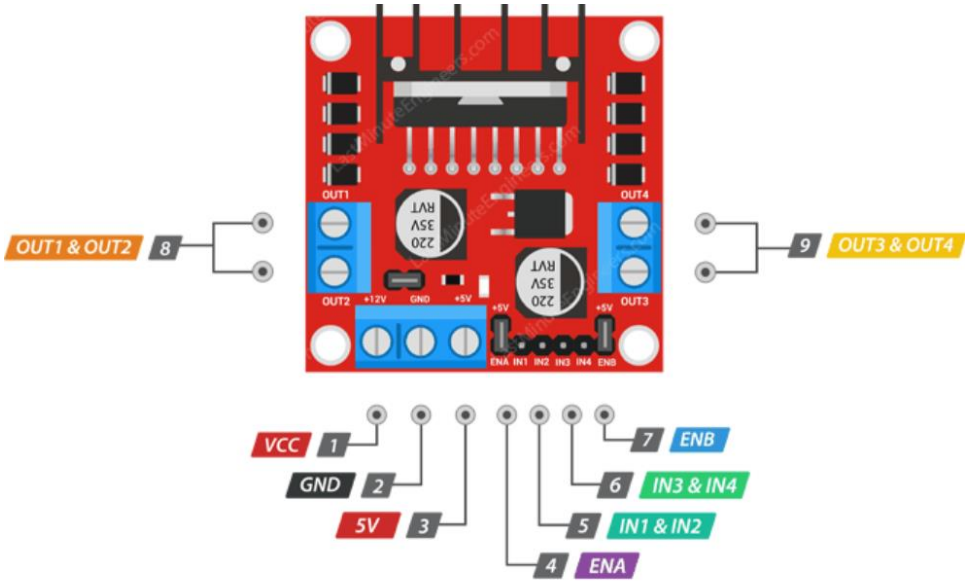


Figure 65: The pin connections of the motor driver were made with jumper cables.

To connect the Nano board to the motor driver, data and power connections are vital to be known.



In the figure, pins numbered 1,2 and 3 functions as the power input and output parts of the system. 12V and 5V power inputs are avaliable for feeding the driver board. Pins numbered 5 and 6 are used to control the direction of the motors. Pins numbered 4 and 7 are used for feeding power to

the motors. Finally pins numbered 8,9 are the motor connection pins.

Input1	Input2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

In the figure it is seen that two inputs are used to control the spinning direction of one motor.

While number 8 and number 9 pins feed power to the motors, the speed of them can be controlled via Pulse Width Modulation (PWM). Pulse Width Modulation is a technique which includes controlling power by adjusting duty cycle of the signal, or power in the system.

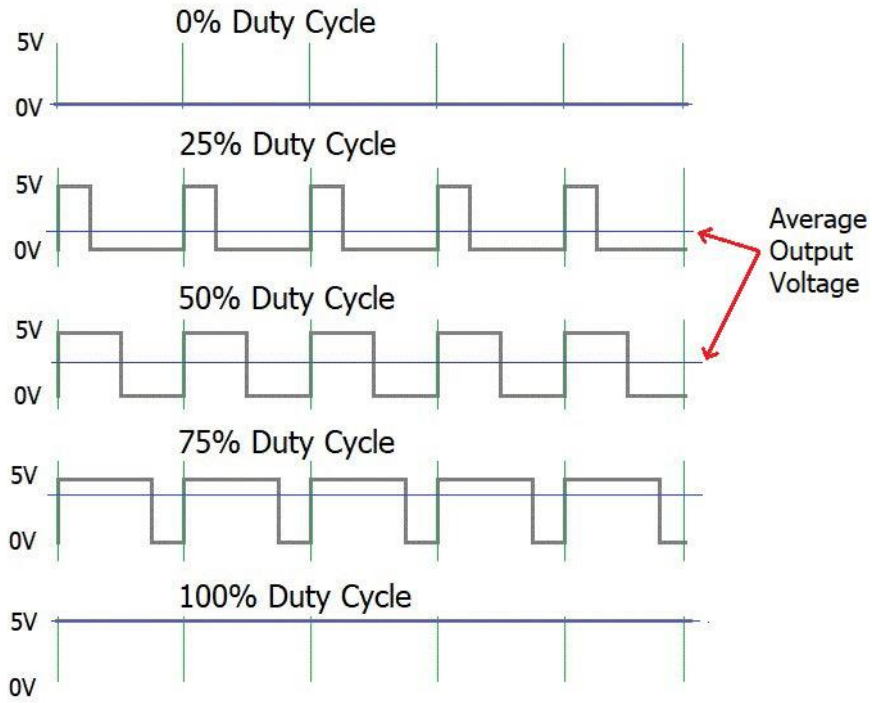


Figure 66: Pulse Width Modulation (PWM) logic.

Since the pin logic of the motor driver is known, the required code can be written easily.
Code for Arduino nano:

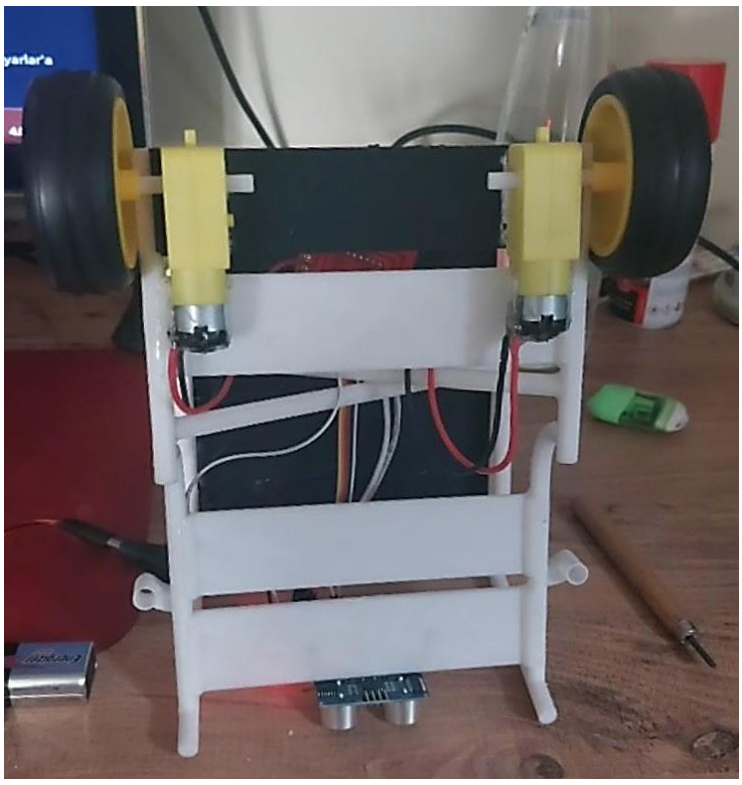


Figure 67: The wheels were mounted onto the chair. The source code was uploaded to the Arduino nano and the forward-backward motion of the wheels was enabled.

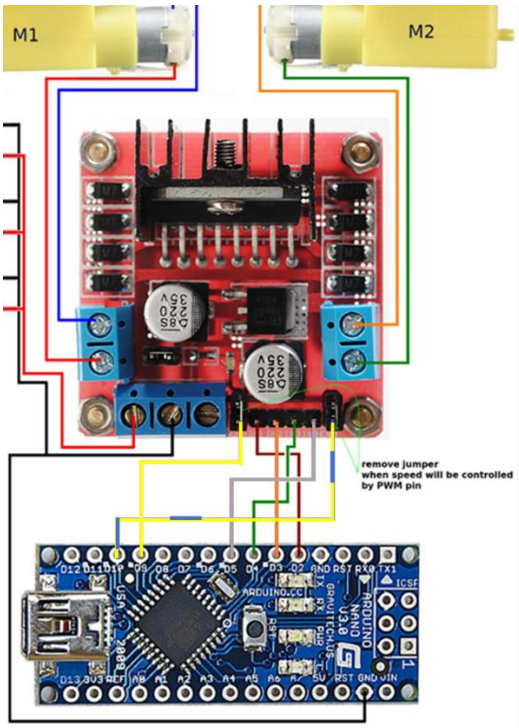


Figure 68: The connections between the Arduino nano and the motor driver were made according to this diagram.

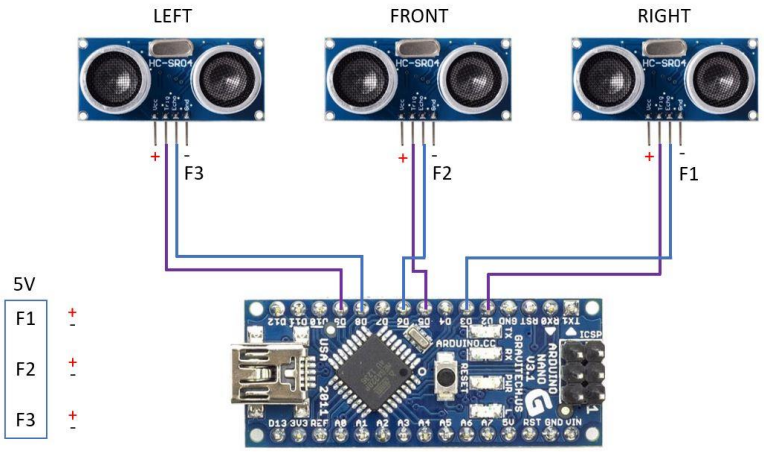
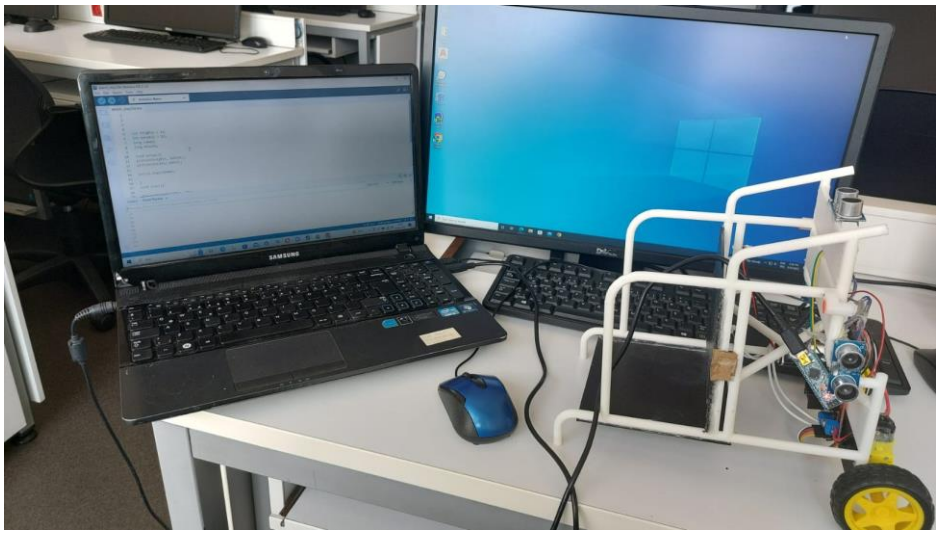


Figure 69: Ultrasonic sensor wiring diagram

We integrated the sensors into the wheelchair. Then performed the necessary controls.



Finally, we performed the integration of the PCB into the wheelchair in the laboratory.

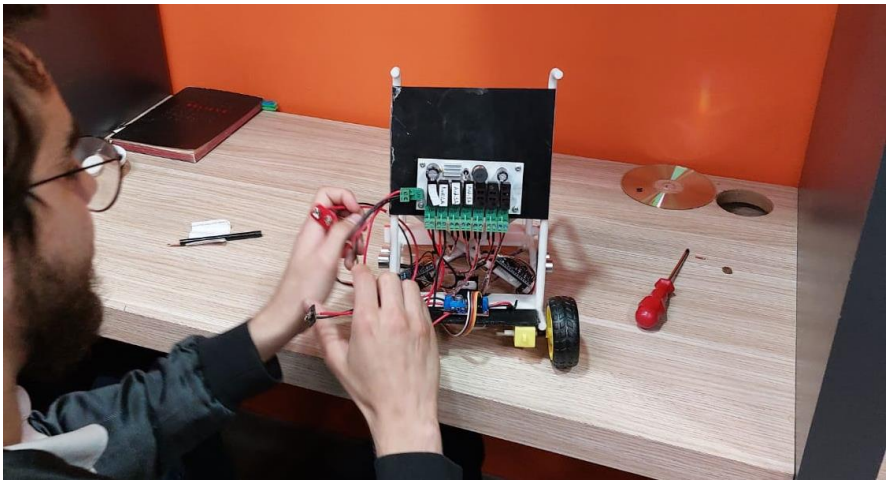


Figure 70: The Final Version of Wheelchair

3.2.6. Evaluation

The four primary techniques for verification are inspection, demonstration, test, and analysis. Each of the four ways tests a product's or system's requirements [32]. Demonstration and analysis verification methods will be used for the validation of mechatronics subsystems.

A demonstration is the verification method of the system to ensure that the outcomes are as expected or planned, the system should be used as intended. An analysis is the verification of the system employing calculations, and testing tools.

The electromechanical system includes the physical integrations of a fuse board, voltage regulator circuit, battery system, distance sensor, microcontrollers, and electronic speed controller. When all the electrical connection of the electronic and power distribution system is completed, the electromechanical system will be done. For the validate the electromechanical system, analysis and demonstration will be used. For example, analyze data by using a multimeter that the voltage regulator gives appropriate 5V.

We provide 5V power to Raspberry Pi, Arduinos, and ultrasonic sensors. To ensure safety against short circuits and excessive current draw, we also pass each component through a fuse.

Figure 77: It has been confirmed that the voltage we provide is approximately 5V.

System current test: https://www.youtube.com/shorts/5RvJvC3DmUc

When the motors achieve the ability to go both forward and backward the motor system will be completed. For the validation of the motor system, a demonstration verification method will be used.

Motor system check: https://www.youtube.com/shorts/gKgc_NJI7c0

If the wheelchair can be stopped while it is moving, the braking-sensor system will be completed. For the validation of the braking-sensor system, a demonstration verification method will be used.
Brake-sensor work test: https://www.youtube.com/watch?v=Xm9drvkGkCg

When the wheelchair can turn in left and right directions from any type of data signal, the steering system will be completed. For the validation of the steering system, a demonstration verification method will be used.
Steering/Wheel system check: https://www.youtube.com/shorts/BTF3u-cPOe0

3.3 Computer Engineering

This sub-team consist of 2 Computer engineering students, who will be responsible for different tasks, the main function of this sub-system are training AI based on BCI waves and then developing back-end software and Arduinos software to apply AI's decision.

First we will take the gathered signal data from BCI, help getting it ready for machine learning. Then we will use the data to train our machine learning algorithm that we wrote which is a classification method. The result of this classification will give the decision made by our patient's mind for movement (such as the direction or the stop command).

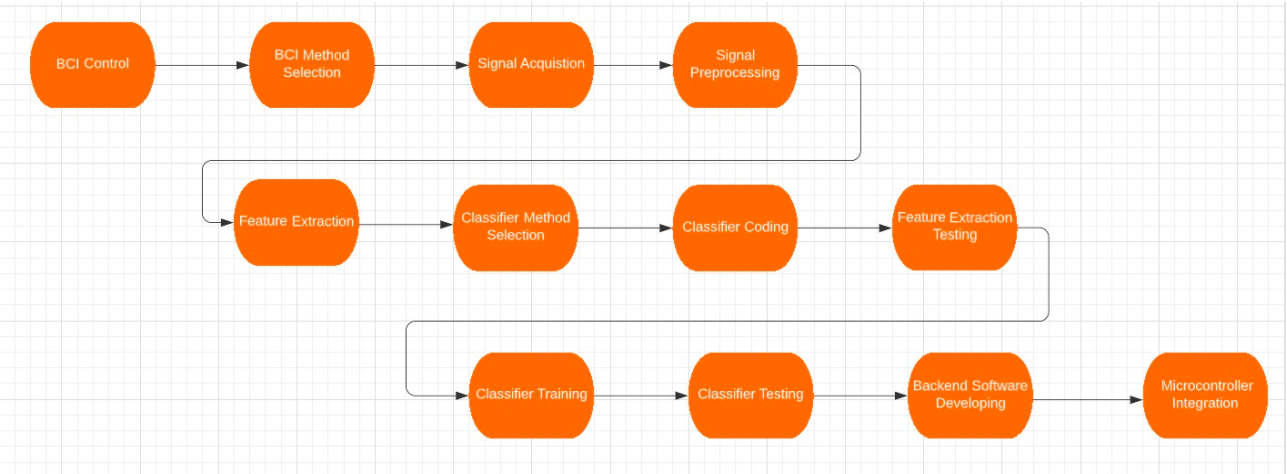


Figure 30. Machine Learning System

Second we will create back-end software. This software will control sensor data and Motor's behave with ai's decision. Therefore sofware will be master of 2 slave arduinos. First arduino's software will be responsible from controling distance sensors and then sending it to the back-end via serial communication. Second arduino's software will be responsible from creating proper PWM wave for driving motors with receiving the command by back-end software.

3.3.1. Requirements

Controlling the direction the patient wants to go using the distance sensor, applying or canceling the decision in line with the sensor data.

For functional metrics, we have movements: stop command, turning left, turning right, and going forward. As for non-functional performance metric: we are aiming for Machine Learning to perform with more than %65 accuracy.

The classifier must be well trained since the accuracy will increase by more training, in order to differentiate the different commands that are forward, right, left and brake that are generated by internal stimulation through users thoughts and imaginations. Also the wheelchair must be able to stop if there is an obstacle blocking its way even though there is no command generated from the patient to stop the wheelchair.

A 'backend' software is needed to provide motor controls and create an 'obstacle avoidance' system with sensor data. Other topics that backend software will provide are: The wheelchair should stop when it getting close to an object. Wheelchair must have a speed limit for safety. Changing the position should be avoided during the rotation of the wheelchair.

3.3.2. Methods

3.3.2.1. Master-Slave

While the backend software is running on the main computer, embedded systems are needed to receive data from the sensors and run the engines properly. For this, 2 Arduino uno will be connected to the main computer with the software in it. In this way, while the main computer is in the master position, 2 arduinos with serial communication will be in the slave role, which will provide the necessary data flow and motor control to the system.

3.3.2.2. Multi-Thread

The systems should work independently of each other in order to slow down the information flow from the arduino in the slave role and not interfere with the decision algorithm. In this case, system reqire 2 different type of the thread. First is the main thread which will only work on main decision algorithms for obstacle avoidence. Second thread should be sub thread of the main and the goal is updating sensor values without effecting main process.

3.3.2.3. Communication with AI

There are 2 alternatives that we can apply at this stage. As a first plan, the whole system can be included in the backend software, in this case the whole system will work from a single source and communication will be provided without the need for any protocol. As an alternative and Plan B, the local network of the main computer can be used to communicate between the AI and the Backend software. In this case, the appropriate communication protocol between them can be provided with TCP, which is an internet protocol.

3.3.2.4. Obstacle Avoidance

Raspberry PI receives the data by communicating with the arduino responsible for the sensors. According to the data received from the Arduino, the software stops the motors if it is closer than 30cm to the obstacle in front. Due to the inadequacy of the vehicle hardware, only the vehicle engines can be stopped as a software measure.

Sensor Codes test video: https://youtu.be/Xm9drvkGkCg

Sensor stops the vehicle in the test: https://youtu.be/3t57x_fIWto

3.3.2.5. Error Handling

For receiving any kind of error which is related with backend software, system and motors will be shutted down programaticly with an displaying cause of error.

3.3.2.6. AI Training

Like we talked in 1.3.1 part, we will use classification algorithm based on our literature review. Classification is a type of Machine Learning algorithm that automatically categorizes data into "classes" based on a set of features, which means classes will be created based on patterns. Since the frequencies of the "thoughts" are in a constant change and has ups and downs, they all create their specific patterns. Thus, we required a classification algorithm and we will differentiate our data based on the pattern they create. As a result, we will have 4 classes: Left, right, forward, stop. There isn't any perfect "one-fit-all" algorithm for BCI data. Based on WEKA's implementations and literature review, we choosed several classification algorithms and will try them on our processed data to find our perfect algorithm [27, 29].

	classifier	Description and parameters
1	ZeroR	Predicts the majority class in the training data; used as a baseline.
2	1R	A rule based on the values of 1 attribute i.e. one level decision tree, (Holte 1993).
3	Decision Tree (DT)	A classical divide and conquer learning algorithm (Quinlan 1986). We used J 48.
4-5	K Nearest Neighbor (k-NN)	A classical instance-based algorithm (Aha and Kibler 1991); uses normalised Euclidean distance. We used k=1 and 5.
6	Naïve Bayes (NB)	A standard probabilistic classifier.
7	Radial-bases Network (RBF)	A 2-layer network. Uses Gausssians as basis functions in the first layer (number and centers set by the k-means algorithm) and a linear second layer and learning algorithm (Moody and Darken 1989).
8	Support Vector Machine (SVM)	Finds the maximum margin hyperplane between 2 classes. We used Weka's SMO with polynomial kernel. SMO is based on Platt's optimisation algorithm (Platt 1998)
9	Logistic Regression (LogReg)	Standard linear regression. Weka's implementation is based on (Le Cessie and van Houwelingen 1992).
10	Ada Boost	An ensemble of classifiers. It produces a series of classifiers iteratively; new classifiers focus on the instances which were misclassified by the previous classifiers; uses weighed vote to combine individual decisions (Freund and Shapire1996). We used boosting of 10 decision trees (J 48).
11	Bagging	An ensemble of classifiers. Uses random sampling with replacement to generates training sets for the classifiers; decisions are combined with majority vote (Breiman 1996). We combined 10 decision trees (J 48).
12	Stacking	A 2 level ensemble of classifiers (Wolpert 1992).We used 1-NN, NB and DT (J 48) as level-1 classifiers and 1-NN as a level-2 classifier.
13	Random Forest (RF)	An ensemble of decision trees based bagging and random feature selection (Breiman 2001). We used t=10 trees.

Table 1: Classifiers used

Figure 31. Classification Algorithms used based on WEKA [27]

To find our algorithm, we will try all of the different algorithms on our data after feature extraction. We will go into the loop of training and testing. Firstly, to avoid overfitting we will split our data into two (70% and %30), and label one part only. We will split the labeled data's %70 as training set and %30 as validation set. We will test the algorithms with changing their parameters.

Then, we will evaluate and eliminate them into half based on their accurasies while being careful not to overfit.

Afterwards, we will validate the remaning algorithms on our unlabeled data for just one time to find the real-life accuracy, and will choose our final algorithm. The results of the chosen algorithm will be the choice of left, right, forward or stop command, and it will be transferred to motors via backend-software.

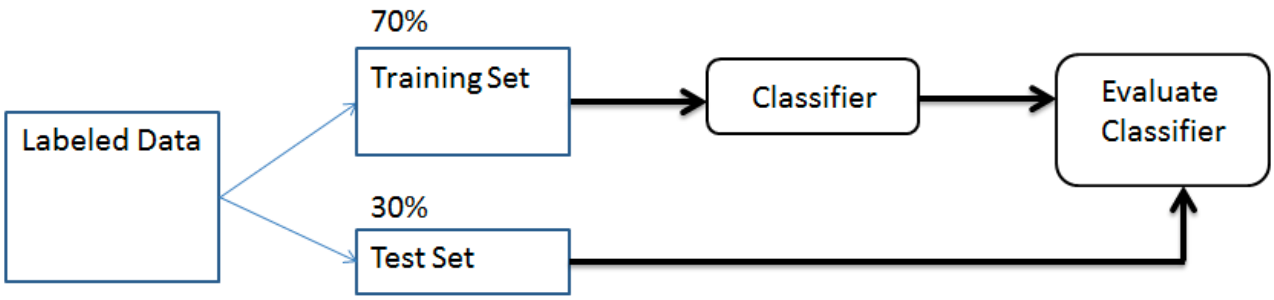


Figure 32. Loop of chosing the algorethm

3.3.2.7. Cloud Communication

There was a communication problem on the software side due to the processing of the data from the computer by the Biomedical Department, but the use of raspberry PI in the vehicle design of the mechatronics team. A dedicated server was used to overcome this communication problem. For this internet communication, the UDP protocol was preferred and thus "handshake" was avoided.

3.3.3. Conceptualization

As a general concept, in this section, it is aimed to create an AI that can make inferences from brain waves and to be able to move the wheelchair properly in line with the decisions made by this AI. In order to achieve this, another software is needed independent of the artificial intelligence which is created with help of biomedical students.

All systems will be connected to each other with the backend software.

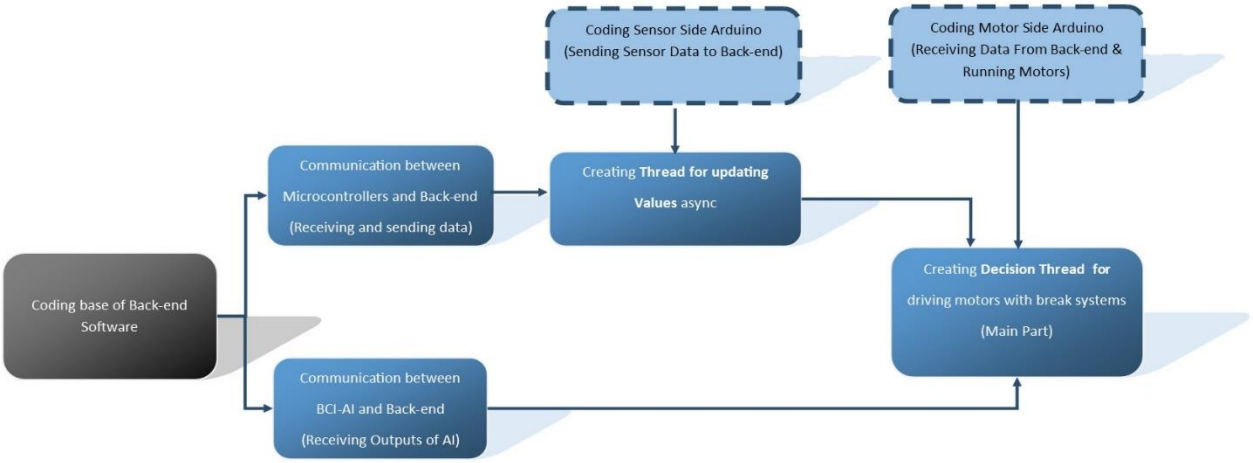


Figure 33. Developping steps of the backend software

3.3.4. Software architecture

3.3.4.1 Machine Learning architecture

We mentioned in part 1.3.1 that we can't choose our classifier method without having our processed data. Thus, we will go into train-validate loop for each of the classification algorithm with our 70% of data and we will decide on our parameters. This will give us the best perfoming 50% algorithm. The reason we are not choosing our final algorithm here, is to avoid overfitting. Then, we will test our remaining algorithms with the remaining 30% unlabeled data for just one time to find the perfect algorithm for our data. Finally, we will connect our algorithm with the microcontroller to perform real time.

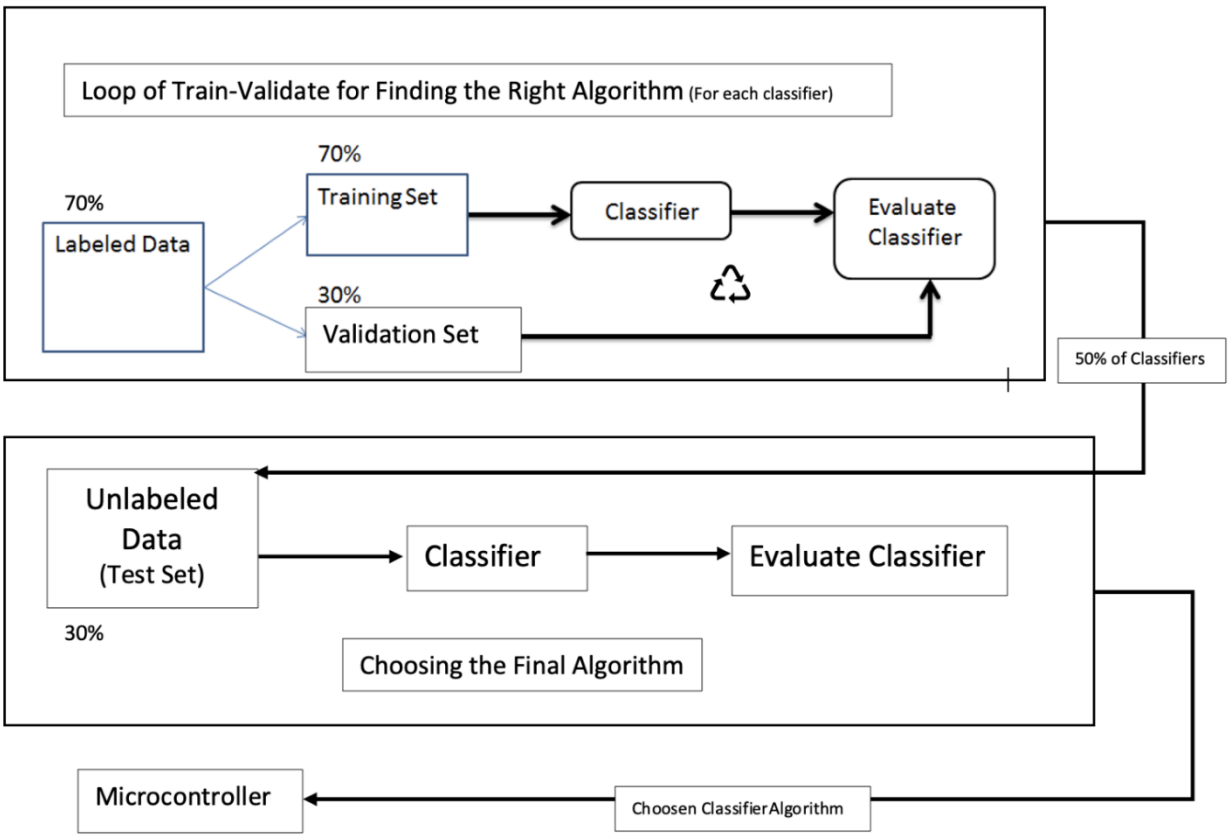


Figure 34. Machine Learning architecture

3.3.4.2 Backend architecture

The main purpose of this software is to provide 'purposeful physical movement of the chair'. In this direction, the decision from the artificial intelligence must be processed with the data from the sensors and transmitted to the arduino which is responsible for the motors. For this software has 3 main part

3.3.5. Materialization

3.3.5.1. Backend Side

For the backend software, preferably the same software language as the people responsible for the biomedical department should be preferred. This language can be 'Python' as it can easily communicate with embedded systems via 'Serial Communication' but in case of unexpected problem, considering we are using Windows OS, ‘C#’ may be preferred as an alternative to 'python'.

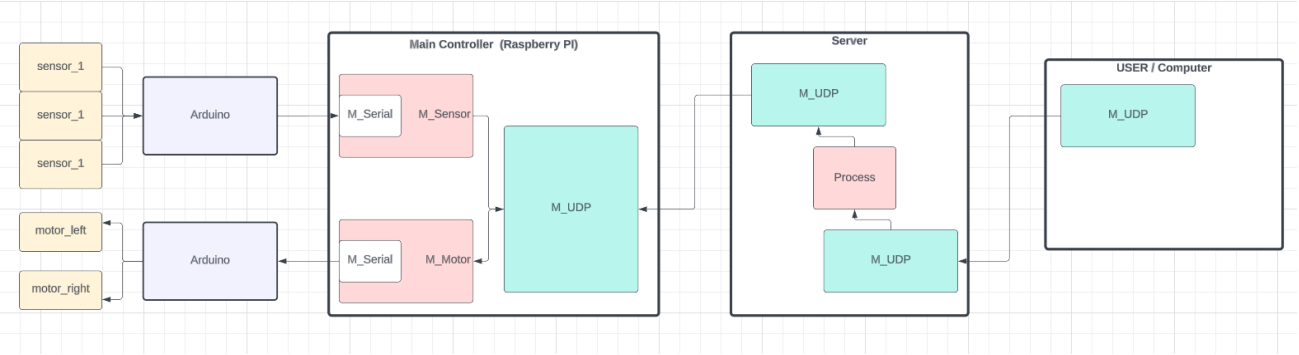


Figure 35. Backend general work flow

3.3.5.1.1. Server Side

The server allocates port 6464 to transmit users' messages to Raspberry PI. This port receives the messages sent with the UDP protocol from the user with control authority and forwards them to Raspberry PI. In case the Raspberry PI and the user send a message to the server, the ip address and port number are kept on the server, so the address of the next message is determined. Just as port 6464 is reserved for the user, port 6565 is reserved so that Raspberry PI's messages can be processed. For access the code, see the “AppendixC” part in below or related Url link:

https://github.com/TkRsln/CapstoneRemoteControl/blob/main/DedicatedServer/SERVER_Lib.py

Technical features of the server: A private cloud server with Ubuntu 18 operating system was preferred. Domain (utkuarslan.xyz) is assigned to the "IP" address of the server to facilitate communication 3.3.4.2.2. User Side

Here, the user is the person controlling it remotely. The users here are normal people who drive with a headset or computer engineers who test the codes. The software here is to forward the message to the dedicated cloud server using the UDP protocol to port 6464.

For access the simple remote control code:
https://github.com/TkRsln/CapstoneRemoteControl/blob/main/SimpleRemoteController/main.py

3.3.5.1.2. User Side

Here, the user is the person controlling it remotely. The users here are normal people who drive with a headset or computer engineers who test the codes. The software here is to forward the message to the dedicated cloud server using the UDP protocol to port 6464.

For access the code, see the “AppendixC” part in below or related Url link

https://github.com/TkRsln/CapstoneRemoteControl/blob/main/SimpleRemoteController/main.py

In order to test the operation of the system independently of the Headset, the Remote control application was developed in python, it was developed for the Headset software in a similar structure and integrated into the codes of the Biomedical team.

3.3.5.1.3. Raspberry PI & Arduino Side (Embedded System)

The main purpose of the codes written here is to make sense of the command from the dedicated server and move the vehicle accordingly. For this, the codes written to Raspberry PI were written modularly and written in accordance with "Object Oriented" and "Threads" were planned to prevent errors and "Dead Lock" problems were prevented with "Lock".

For access the code, see the “AppendixC” part in below or related Url link

https://github.com/TkRsln/CapstoneRemoteControl/blob/main/RaspberryPI/PI_Lib.py

Raspberry pi technical details: "Raspberry Pi OS Lite" was preferred as the operating system. The Debian version is "11 Bulseye". Python version 3.9 has been installed.

For the Arduino embedded system, 2 different software has been written with "C++" according to its basic purposes: "Motor Control" and "Sensor Fields". Arduinos transmit or receive data from raspberry pi via USB.

For access the Arduino Motor code, see the “AppendixC” part in below or related Url link:

https://github.com/TkRsln/CapstoneRemoteControl/blob/main/Arduinos/ArduMotor.ino

For access the Arduino Sensor code, see the “AppendixC” part in below or related Url link

https://github.com/TkRsln/CapstoneRemoteControl/blob/main/Arduinos/ArduSensor.ino

More details about modules in Raspberry PI software:

- M_UDP

This module enables data to be transmitted or received via UDP from the "dedicated server". user, server and Raspberry PI are also using the module.

- MainController

This module is the main module in Raspberry PI. Makes sense of incoming commands and decides speeds.

- M_Motor

This module is responsible for transmitting the incoming speed values to the Arduino. Connects to Arduino via the given port 'ID'

- M_Sensor

This module is one of the important modules. This module is responsible for receiving data from the sensors and also stops the motors via the "M_Motor" module to avoid the object by looking at the values of the distance sensor.

Sensor Codes test video: https://youtu.be/Xm9drvkGkCg

3.3.5.2 Machine Learning Side

3.3.5.2.1. Challenges with Previous Classifiers

For the classification task on EEG data, initially I employed three machine learning algorithms: Gaussian Naive Bayes (GNB), Random Forest, and Support Vector Machine (SVM) classifiers. Each of these models offers distinct advantages and is suitable for EEG classification.

```
# Create and train the Random Forest classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_resampled, y_resampled)

# Make predictions on the test set
y_pred = rf.predict(X_test)


# Create and train the SVM model
svm = SVC()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)


# Train a Gaussian Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train,y_train)
y_pred = gnb.predict(X_test)
```

Figure78.Previous-Classifiers

The GNB classifier is based on the assumption that features are conditionally independent given the class label. Despite its simplicity, GNB can perform well when the features are approximately independent. This classifier is computationally efficient and requires minimal tuning of hyperparameters.

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It works by constructing a multitude of decision trees during the training phase and outputs the class that is the mode of the classes predicted by individual trees. Random Forests are known for their robustness against overfitting, ability to handle high-dimensional data, and capability to capture non-linear relationships.

SVM is a powerful classifier that uses a hyperplane to separate data points of different classes in a high-dimensional feature space. It can handle both linear and non-linear classification tasks by employing different kernel functions. SVM has been widely used in various domains due to its strong generalization capabilities and effectiveness in handling complex datasets.

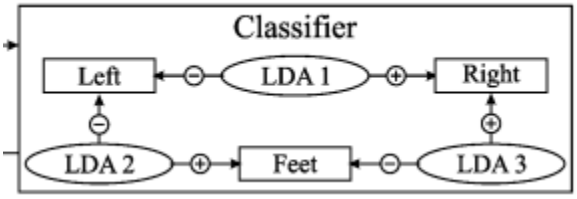
In the process of selecting a suitable classifier for the BCI data, several challenges were encountered with the previously chosen classifiers. These challenges primarily stemmed from the preprocessing method applied to the BCI data, leading to what was referred to as the "W problem." The biomedical team explained that they had used three different W multipliers to differentiate

between classes, which is a method suitable for differentiating between two classes. However, since we have three classes (four with the stop command being considered as another class), and the class would be unknown at the time of classification, it became impossible to select the correct W multiplier. Consequently, the previously chosen classifiers were not applicable in real-time scenarios, resulting in low accuracy rates

Efforts were made to overcome the W problem by attempting to apply classifiers to the BCI data without using the W multipliers. Various preprocessing techniques, such as data augmentation (e.g., SMOTE, noise addition) and outlier handling (e.g., IQR, outlier mask), were employed, but these methods proved ineffective in improving classifier performance. However, these methods did not significantly improve classifier performance, with accuracy remaining around 50%

3.3.5.2.2. Selection of Linear Discriminant Analysis (LDA)

Considering the limitations and challenges faced with previous classifiers, the Linear Discriminant Analysis (LDA) algorithm was chosen as an alternative. LDA is a supervised machine learning algorithm specifically designed for classification tasks and serves as a dimensionality reduction technique. By finding a linear combination of features that maximizes the separation between classes, LDA aims to improve classification accuracy.



LDA 1	LDA 2	LDA 3	classification result
⊖	⊖	⊖	Left hand
⊕	⊖	⊖	Not classified
⊖	⊕	⊖	Foot
⊕	⊕	⊖	Foot
⊖	⊖	⊕	Left hand
⊕	⊖	⊕	Right hand
⊖	⊕	⊕	Not classified
⊕	⊕	⊕	Right hand

Figure 79.1.-79.2. LDA Classifiers' Operation Method [45]

To implement LDA, the following code snippet was used:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis


lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
```

Figure 80. LDA Implementation

To address the W problem, three separate LDAs were trained, each with its corresponding W multiplier, for different combinations of classes: left-forward, right-forward, and left-right:

```
    # Define the class combinations
class_combinations = [(1, 2), (1, 3), (2, 3)]
#left:1 right:2 forward:3


    # Train LDA models for each class combination
ldas = []

for combination in class_combinations:
    class1, class2 = combination

    # Prepare the data for the current combination
    X = np.concatenate((X_lf[y_lf == class1], X_lf[y_lf == class2]))
    y = np.concatenate((np.zeros(np.sum(y_lf == class1)), np.ones(np.sum(y_lf == class2)))) #0

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Train the LDA model
    lda = LinearDiscriminantAnalysis()
    lda.fit(X_train, y_train)

    # Predict the class labels for the test set
    y_pred = lda.predict(X_test)
```

Figure 81. Training 3 different LDAs

The classification decision was then determined based on the combined results of these three LDAs:

```
if predictions[0] == 0 and predictions[1] == 0 and predictions[2] == 0:
    result.append('left')
elif predictions[0] == 1 and predictions[1] == 0 and predictions[2] == 0:
    result.append('not classified')
elif predictions[0] == 0 and predictions[1] == 1 and predictions[2] == 0:
    result.append('forward')
elif predictions[0] == 1 and predictions[1] == 1 and predictions[2] == 0:
    result.append('forward')
elif predictions[0] == 0 and predictions[1] == 0 and predictions[2] == 1:
    result.append('left')
elif predictions[0] == 1 and predictions[1] == 0 and predictions[2] == 1:
    result.append('right')
elif predictions[0] == 0 and predictions[1] == 1 and predictions[2] == 1:
    result.append('not classified')
elif predictions[0] == 1 and predictions[1] == 1 and predictions[2] == 1:
    result.append('right')
```

Figure 82. LDA Combined Result Method

For instance, if the result was positive-negative-positive, the corresponding class would be identified as "right."

3.3.5.2.3. Data Preprocessing

Before training the classification models, the dataset underwent several preprocessing steps. First, outliers were removed using the interquartile range (IQR) method, which ensured the removal of extreme values while preserving the overall distribution of the data. Additionally, MinMax scaling was applied to normalize the feature values between 0 and 1, ensuring that each feature contributes equally to the model training process.

```
        #preprocessing

    # IQR - removing outliers

        #removing outliers for left-forward
Q1 = np.percentile(X_lf, 25, axis=0, interpolation='midpoint')
Q3 = np.percentile(X_lf, 75, axis=0, interpolation='midpoint')
IQR = Q3 - Q1

    # Above Upper bound
upper = Q3 + 1.5 * IQR
upper_array = np.array(X_lf >= upper)

    # Below Lower bound
lower = Q1 - 1.5 * IQR
lower_array = np.array(X_lf <= lower)

    # True values (outliers) in the array
upper_indices = np.argwhere(upper_array)
lower_indices = np.argwhere(lower_array)

    # dropping the outliers
if upper_indices.size > 0:
    X_lf = np.delete(X_lf, upper_indices[:, 0], axis=0)
    y_lf = np.delete(y_lf, upper_indices[:, 0], axis=0)

if lower_indices.size > 0:
    X_lf = np.delete(X_lf, lower_indices[:, 0], axis=0)
    y_lf = np.delete(y_lf, lower_indices[:, 0], axis=0)
```

Figure 83. IQR Method

```
    # Scale the features using MinMaxScaler
scaler = MinMaxScaler()
X_lf = scaler.fit_transform(X_lf)
X_rf = scaler.fit_transform(X_rf)
X_rl = scaler.fit_transform(X_rl)
```

Figure 84. Min-Max Scaler

3.3.5.2.4. Class Imbalance Handling

The original dataset exhibited a class imbalance, with one class having a significantly smaller number of samples compared to the others. To address this issue, Synthetic Minority Over-sampling Technique (SMOTE) has been employed to artificially generate synthetic samples for the minority

class. This technique helped balance the class distribution, ensuring that the models are trained on a more representative and balanced dataset.

```
# Apply SMOTE to balance class distribution
smote = SMOTE()
X_lf, y_lf = smote.fit_resample(X_lf, y_lf)
X_rf, y_rf = smote.fit_resample(X_rf, y_rf)
X_rl, y_rl = smote.fit_resample(X_rl, y_rl)
```

Figure 85. SMOTE Method

3.3.5.2.5. Model Training and Evaluation

To train the models, we randomly split the dataset into a training set and a testing set with a ratio of 80:20. The models were trained on the training set and evaluated on the testing set using several performance metrics, including accuracy, precision, recall, F1-score, mean squared error, mean absolute percentage error, and R^2 score. These metrics provided insights into the models' overall classification performance and their ability to correctly identify each cognitive state.

```
# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall =  recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
mse=mean_squared_error(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
r2= r2_score(y_test, y_pred)
print(f"Accuracy for classes {class1}-{class2}: {accuracy}")
print(f"Precision for classes {class1}-{class2}: {precision}")
print(f"Recall for classes {class1}-{class2}: {recall}")
print(f"F1-score for classes {class1}-{class2}: {f1}")
print(f"Mean Squared Error for classes {class1}-{class2}: {mse}")
print(f"Mean Absolute Percentage Error for classes {class1}-{class2}: {mape}")
print(f"R^2 Score for classes {class1}-{class2}: {r2}")
```

Figure 86. Evaluation Metrics

3.3.5.2.6. Real-Time Prediction

One of the key advantages of the trained models is their ability to make real-time predictions on new, unseen EEG samples with the following code:

```
        #Real Time  Prediction

while True:
    # Get input from user
    user_input = input("Enter 8 values (separated by spaces): ")

    # Split the input string into a list of values
    values = user_input.split() #need to seperated by space

    # Convert the values to float and create a numpy array
    sample = np.array([float(val) for val in values])

    # Make prediction using the predict_class function
    predicted_class = predict_class(sample)

    # Print the predicted class
    print("Predicted class:", predicted_class)
```

Figure 87. Real Time Application

By providing the models with a single EEG sample, they can predict the cognitive state of the subject, along with probability estimates for each class. This capability opens up opportunities for real-time applications, such as cognitive state monitoring during tasks or interventions.

3.3.6. Evaluation

3.3.6.1. Machine Learning Evaluation

3.3.6.1.1. Results and Improvements with LDA

The application of LDA resulted in significant improvements in accuracy rates during both the training phase and real-time testing. Notably, the right and forward classes achieved accuracy rates exceeding 90%. This approach proved to be effective in handling the challenges posed by the W problem and the limitations of the previous classifiers and preprocessing methods.

3.3.6.1.2. Challenges and Limitations Encountered

It is worth mentioning that while the adoption of the LDA approach proved successful, several challenges and limitations were encountered throughout the classification process.

1. Insufficient Data: The availability of preprocessed data for this project was comparatively lower than what is typically encountered in similar endeavors, primarily due to various challenges faced during data collection by the biomedical team. The limited diversity and representativeness of the dataset had a significant impact on the classifier's performance. With a larger, more

comprehensive, and varied dataset, the classifier could have better generalized and achieved higher accuracy in its predictions. Increasing the dataset size and diversity should be considered in future efforts to improve the classifier's performance.

2. Imagining Problem: The absence of a specific image or clear representation for the stop command posed difficulties for the client in accurately interpreting the command. Without a well-defined mental imagery associated with the stop command, the classification process became more challenging.

3. Switching between Commands: The process of switching between different commands or classes introduced complexities for the client. This required the client to adapt and transition their mental focus and motor actions quickly, which could result in misclassifications. Developing strategies or techniques to facilitate smoother transitions between commands could improve the accuracy of the classification system.

4. “Left Class Problem": The classification process faced challenges in accurately distinguishing and classifying the left command. The client, being right-hand dominant, experienced difficulties in imagining themselves using their left hand. As a result, the client tended to associate the left command with the concept of right, leading to misclassifications. Addressing this issue would require finding alternative ways to capture the client's intended left command, potentially by exploring different mental imagery techniques or adjusting the classifier's training approach to accommodate individual preferences and motor dominance.

3.3.6.1.3. Future Work

- To overcome the challenges and limitations mentioned above, several avenues for future work are suggested:
- Gathering a more comprehensive and varied dataset to improve the classifier's performance.
- Exploring alternative mental imagery techniques for the left command to enhance classification accuracy.

- Developing strategies to facilitate smoother transitions between commands to improve overall accuracy.

- Considering individual preferences and motor dominance in the classifier's training approach.

3.3.6.1.4. Conclusion

In summary, the chosen classifiers initially appeared suitable for the available BCI data but proved impractical in real-time scenarios due to the W problem arising from the preprocessing method. By implementing LDA and training three separate LDAs with their respective W multipliers, notable improvements in accuracy rates were achieved for the right and forward classes. Nonetheless, further research and refinement are necessary to overcome the remaining challenges and enhance the overall accuracy and usability of the BCI system.

3.3.6.2. Backend Evaluation

Since the computer that runs the BCI codes and the computer that controls the vehicle is different, we were very interested in network programming. As a result of our research, we decided on the most suitable protocols such as TCP and UDP and the one that provides us convenience. However, we deeply learned the idea of how cloud systems work and we built a system of our own.

We've looked closely at another important piece, embedded system coding. We have brought together various modules so that the vehicle can work properly and fulfill the given commands. These were basically 2 different parts, communication via Serial port and receiving commands from the internet. While doing all these coding, we benefited from the algorithm analysis, embedded system coding and Object oriented courses. In this way, the system took into account the data coming from the sensors while properly applying the "movement command" coming from the internet, thus gaining features such as "obstacle avoidance". Such features are important for the end user. Due to the modularity of the system, an easy and clean software was realized.

Video of remotely controlling the vehicle by simulating the headset commands:

https://youtube.com/shorts/0aMiKZVlxZg

Test video of deciding the wheel directions of the vehicle with Headset:

https://youtu.be/RdQFOoEb3g4

Test video on land with the vehicle's obstacle avoidance method, taking command over the headset:

https://youtu.be/j8FBnRCBTyI

The software architecture of this project is similar to the architecture of the vehicle which was developed for the competition [28] and fully coded by me

5. SUMMARY AND CONCLUSION

Immobility caused by a Spinal Cord Injury (SCI) injury is a significant drawback for those who are affected, particularly in their daily activities. A BCI-based wheelchair is described in the scope of this project to reverse its negative impact. The identification of the problem is thoroughly discussed in order to clarify the project's needs from various departments. Several works from the literature are scanned in order to be inspired by them and to apply the best solution possible when all design constraints are taken into account. Each team member's responsibilities are distributed based on their interests and talents. Time, budget, and implementation constraints are represented in order to set a realistic goal and achieve it on time with optimal design. Also, required standards and ethical codes are and will be followed in each step of the project. An architecture between the headset and the Emotiv apps is designed, and PRO/BCI apps are used with Cortex on Python. This project shows how an individual can be taught to confidently perform a set of mental commands, which can then be used to control a wheelchair.

In conclusion, this capstone project will provide a better understanding of brain signals, their construction, and analysis. And finally, putting all of the parts and subsystems together to control the wheelchair using the appropriate and analyzed signals.

ACKNOWLEDGEMENTS

We wish to thank our advisors Assist. Prof. Hakan Solmaz from Biomedical Engineering department, Assist. Prof. Amir Navidfar from Mechatronics Engineering department and Assist. Prof. Ehsan Nowroozi from Computer Engineering department. Also a special thank to Yara Corky for her contribution in this project where she contributed in designing the system's training method & in choosing the BCI method.

APPENDIX A

The following are the codes written by the Biomedical engineering students

```
%% ERD% BCI comp2
time = 0:1/128:((1152-1)/128);

data_right = x_train(:,:,(y_train==2));
data_left = x_train(:,:,(y_train==1)); %2 right 1 left

[b,a] = butter(8, [8/(128/2) 30/(128/2)], 'bandpass');
filtered_right = filtfilt(b,a,data_right);
filtered_left = filtfilt(b,a,data_left);
for i=1:50
window = hann(200);
[pxx,f] = pwelch(filtered_left(:,1,i),window,100,128/2,128);
power_right = pxx;

[pxx1,f1] = pwelch(filtered_left(:,3,i),window,100,128/2,128);
power_left = pxx1;

rest = filtered_left(time>=0&time<=2,2,i);
[pxx2,f2] = pwelch(rest,window,100,128/2,128);
power_rest = pxx2;
A = mean(power_rest);


right_C3 = 100*((power_right-A)/A);
right_C4 = 100*((power_left-A)/A);
figure(i)
plot(f,right_C3);grid on;hold on
plot(f1,right_C4);legend('c3','c4')
end
```

```
%% csp comp2 c3 c4 for all trials generating one mixing matrix
W -> 84% right 56% left if the variance is across the trial
% Optimal Spatial Filtering By.. Herbert Ramoser
for i = 1:50
time = 0:1/128:((1152-1)/128);
data_right = x_train(:,:,(y_train==2));
data_left = x_train(:,:,(y_train==1)); %2 right 1 left
[b,a] = butter(8, [8/(128/2) 30/(128/2)], 'bandpass');
filtered_right = filtfilt(b,a,data_right);
```

```
filtered_left = filtfilt(b,a,data_left);
data_right1 = filtered_right(:,:,i)';
data_left1 = filtered_left(:,:,i)';
S1=0;
S2=0;
    for j = 1:50
    data_right1 = filtered_right(:,:,j)';
    data_left1 = filtered_left(:,:,j)';

    S1 =
S1+((data_right1*data_right1')/trace(data_right1*data_right1')
);   % S1~[C x C]
    S2 =
S2+((data_left1*data_left1')/trace(data_left1*data_left1'));
    end

Cr = S1/50;
Cl = S2/50;
Cc = Cr+Cl;
[Uc,Lc] = eig(Cc);
diagVec = diag(Lc); % extract the diagonal values
sortedDiagVec = sort(diagVec,'descend'); % sort the diagonal
values in descending order
Lc = Lc - diag(diagVec) + diag(sortedDiagVec);
Uc_new = [Uc(:,3),Uc(:,2),Uc(:,1)];
P = sqrt(inv(Lc))* Uc_new';
s1 = P*Cr*P'
s2 = P*Cl*P'

[B1,l1] = eig(s1);
diagVec = diag(l1); % extract the diagonal values
sortedDiagVec = sort(diagVec,'descend'); % sort the diagonal
values in descending order
l1 = l1 - diag(diagVec) + diag(sortedDiagVec)
B=[B1(:,1),B1(:,3),B1(:,2)]

% [B2,l2] = eig(s2);
% diagVec = diag(l2); % extract the diagonal values
% sortedDiagVec = sort(diagVec,'descend'); % sort the diagonal
values in descending order
% l2 = l2 - diag(diagVec) + diag(sortedDiagVec)
% B3=[B2(:,2),B2(:,3),B2(:,1)]

%[B,x2] = eig(s1,s1+s2);

W = (B1'*P)';

X_csp1 = W'*data_right1;
```

```
X_csp2 = W'*data_left1;

window = hann(100);
[pxx,f] = pwelch(X_csp2(1,:),window,50,128/2,128);
power_right = pxx;

[pxx1,f1] = pwelch(X_csp2(3,:),window,50,128/2,128);
power_left = pxx1;

rest = filtered_left(time>=0&time<=2,2,i);
[pxx2,f2] = pwelch(rest,window,100,128/2,128);
power_rest = pxx2;
A = mean(power_rest);

right_C3 = 100*((power_right-A)/A);
right_C4 = 100*((power_left-A)/A);
figure(i)
plot(f,right_C3);grid on;hold on
plot(f1,right_C4);legend('c3','c4')
 end
```

```
%% scatter plot and proving csp working principle
dat=zeros(2,2);
for i=1:45
[b,a] = butter(8, [8/(128/2) 30/(128/2)], 'bandpass');
filtered_right = filtfilt(b,a,data_right);
filtered_left = filtfilt(b,a,data_left);
data_right1 = filtered_right(:,:,i);
data_left1 = filtered_left(:,:,i);


%scatter(data_right1(:,1),data_right1(:,3),),grid on
x = data_right1(17,:);
y = data_right1(25,:);
x1 = data_left1(17,:);
y1 = data_left1(25,:);
figure(i);
subplot(2,1,1);
scatter(x, y, [], 'red', 'o', 'MarkerEdgeColor', 'red');
hold on;  % Keep current plot visible
scatter(x1, y1, [], 'blue', 'x', 'MarkerEdgeColor',
'blue');grid on
hold off;  % Release hold

%W=csp(data_right1',data_left1');
X_csp1 = abs(W)'*data_right1;
X_csp2 = abs(W)'*data_left1;
subplot(2,1,2);
```

```
scatter(X_csp1(17,:), X_csp1(25,:), [], 'red', 'o',
'MarkerEdgeColor', 'red');
hold on;  % Keep current plot visible
scatter(X_csp2(17,:), X_csp2(25,:), [], 'blue', 'x',
'MarkerEdgeColor', 'blue');grid on
hold off;  % Release hold
% dat(i,1)=var(X_csp1(1,:))
% dat(i,2)=var(X_csp2(1,:))
end
```

```
%% QSA on Comp3 F3, F4, Fc5, Fc6

% Fc5 = 17, Fc6 = 25 , F3 = 10, F4 = 16;%28 = c3 31=cz 34=c4
8*7.8ms =
% 62.4 = 8 samples
X_RIGHT = HDR.TRIG(HDR.Classlabel==2);
X_LEFT = HDR.TRIG(HDR.Classlabel==1);
data_right = zeros(2561,4,45);
data_left = zeros(2561,4,45);
time = 0:1/250:((2561-1)/250);
Qright = zeros(10,4);
Rright = zeros(10,3);
Qleft = zeros(1200,4);
n2 = zeros(5,1);
%filtered_right = zeros(4,2561,45);
for i = 1:45
data_right(:,:,i) =
s(X_RIGHT(i):(X_RIGHT(i)+2560),[10,16,17,25]);
data_left(:,:,i) =
s(X_LEFT(i):(X_LEFT(i)+2560),[10,16,17,25]); %2 right 1 left
end
M = zeros(2,4,2);
for j = 1:45

for i=1:1000

Qright(i,:) =
[data_left(i+750,1,j),data_left(i+750,2,j),data_left(i+750,3,j
),data_left(i+750,4,j)];
Rright(i,:) =
[data_left(i+742,2,j),data_left(i+742,3,j),data_left(i+742,4,j
)];

end

n = quatrotate(Qright, Rright);

for i = 1:1000
```

```
n2(i) = sqrt( (n(i,1)^2) + (n(i,2)^2) + (n(i,3)^2) );

end

M(j,1,2) = mean(n2);
M(j,2,2) = ( ( sum(n2.^2 -M(1) ) )^2 + sum(n2.^2))/ (2*1200);
M(j,3,2) = (sum(n2.^2))/1200;
M(j,4,2) = sum(1/(1+n2.^2));

end

X1 = M(:, :, 1);
X2 = M(:, :, 2);

% Generate labels for each class
Y1 = ones(size(X1, 1), 1); % First class is labeled as 1
Y2 = ones(size(X2, 1), 1) * -1; % Second class is labeled as -
1

% Concatenate the features and labels
X = [X1; X2];
Y = [Y1; Y2];

% Shuffle the data and labels to avoid bias during training
idx = randperm(length(Y));
X = X(idx, :);
Y = Y(idx);

% Train the SVM using the radial basis function (RBF) kernel
SVM = fitcsvm(X, Y, 'KernelFunction', 'RBF');

% Predict the labels for the training data
Ypred = predict(SVM, X);

% Evaluate the accuracy of the SVM
accuracy = sum(Y == Ypred) / length(Y);

for i = 1:45
    predict(SVM,M(i, :, 1))
end
```

```
%% Head Movement
move = csvread('Head
Movement_EPOCFLEX_87126_2023.04.04T17.27.01+03.00.md.csv',2,0)
;
for i = 1:7
figure(2)
subplot(4,2,i)
plot(move(:,114+i)),grid on;
```

```
end
subplot(4,2,1);title('Q0')
subplot(4,2,2);title('Q1')
subplot(4,2,3);title('Q2')
subplot(4,2,4);title('Q3')
subplot(4,2,5);title('AccX')
subplot(4,2,6);title('AccY')
subplot(4,2,7);title('AccZ')
```

```
%% TopoPlot
BP = zeros(33,32);
for i = 119:130

[b,a] = butter(8, [8/(128/2) 13/(128/2)], 'bandpass');
EEGSignals.x = filtfilt(b,a,EEGSignals.x);  %Ch X T



MarkersIdx = find(input_data(:,44,i) == -1000)
if(length(MarkersIdx) == 0)
MarkersIdx = find(input_data(:,43,i) == -1000)
end


if(filtered_Classes(1,4,i)==0)
    Class1 =
filtered_Classes(MarkersIdx(1):MarkersIdx(1)+581,5:36,i)';
    Class2 =
filtered_Classes(MarkersIdx(2):MarkersIdx(2)+581,5:36,i)';
    Class3 = filtered_Classes(MarkersIdx(3):end,5:36,i)';
else
    Class1 =
filtered_Classes(MarkersIdx(1):MarkersIdx(1)+581,4:35,i)';
    Class2 =
filtered_Classes(MarkersIdx(2):MarkersIdx(2)+581,4:35,i)';
    Class3 = filtered_Classes(MarkersIdx(3):end,4:35,i)';
end

% for k = 1:32
% Class1(k,:) = Class1(k,:) -  ( 1/32 * mean(mean(Class1,2))
);
% end

% X_csp1 = W3'*Class1;
% X_csp0 = W3'*Class2;
% X_csp2 = W3'*Class3;

for j = 1:32
%  X_csp1 = W'*Class3;
```

```
window = hann(100);
[pxx,f] = pwelch(Class1(j,:),window,75,128/2,128);
power_right = pxx; % C3
BP(:,j) = pxx;
end

% gg = W'

%for l = 1:32
figure(i)
topoplotIndie(mean(Class1,1),EEG.chanlocs);colorbar
%end

end
```

```
%% Recorded data Preprocessing & Feature extraction
xl = zeros(3,1);
res = zeros(1,8,2);

S1=0;
S2=0;
for j = 1:40
% MarkersIdx = find(input_data(:,44,j) == -1000)
% if(length(MarkersIdx) == 0)
% MarkersIdx = find(input_data(:,43,j) == -1000)
% end
%
% % Class1 =
filtered_Classes(MarkersIdx(1):MarkersIdx(1)+581,5:36,j)';
% % Class3 = filtered_Classes(MarkersIdx(3):end,5:36,j)';
%
% if(filtered_Classes(1,4,j)==0)
%     Class1 =
filtered_Classes(MarkersIdx(1):MarkersIdx(1)+500,5:36,j)';
%     Class2 =
filtered_Classes(MarkersIdx(2):MarkersIdx(2)+500,5:36,j)';
%     Class3=
filtered_Classes(MarkersIdx(3):MarkersIdx(3)+500,5:36,j)';
% else
%     Class1 =
filtered_Classes(MarkersIdx(1):MarkersIdx(1)+500,4:35,j)';
%     Class2 =
filtered_Classes(MarkersIdx(2):MarkersIdx(2)+500,4:35,j)';
%     Class3 =
filtered_Classes(MarkersIdx(3):MarkersIdx(3)+500,4:35,j)';
% end
%
% % Class1 =
[Class1(7,:);Class1(8,:);Class1(27,:);Class1(28,:)];
```

```
% % Class2 =
[Class2(7,:);Class2(8,:);Class2(27,:);Class2(28,:)];
% % Class3 =
[Class3(7,:);Class3(8,:);Class3(27,:);Class3(28,:)];


S1 =
S1+((Right(:,:,j+100)*Right(:,:,j+100)')/trace(Right(:,:,j+100
)*Right(:,:,j+100)'));   % S1~[C x C]
S2 =
S2+((Foot(:,:,j)*Foot(:,:,j)')/trace(Foot(:,:,j)*Foot(:,:,j)')
);
end
Cr = S1/40;
Cl = S2/40;
Cc = Cr+Cl;
SigmaComps=Cc;
[W,lmds] = eig(Cl,Cl+Cr);
[lmds,Idxs] = sort(diag(lmds),'descend');
W = W(:,Idxs)

xl = zeros(3,1);
res = zeros(1,8,2);
for i = 1:148

[b,a] = butter(8, [8/(128/2) 30/(128/2)], 'bandpass');
filtered_Classes = filtfilt(b,a,input_data);  %Ch X T

MarkersIdx = find(input_data(:,44,i) == -1000)
if(length(MarkersIdx) == 0)
MarkersIdx = find(input_data(:,43,i) == -1000)
end


if(filtered_Classes(1,4,i)==0)
    Class1 =
filtered_Classes(MarkersIdx(1):MarkersIdx(1)+500,5:36,i)';
    Class2 =
filtered_Classes(MarkersIdx(2):MarkersIdx(2)+500,5:36,i)';
    Class3 =
filtered_Classes(MarkersIdx(3):MarkersIdx(3)+500,5:36,i)';
else
    Class1 =
filtered_Classes(MarkersIdx(1):MarkersIdx(1)+500,4:35,i)';
    Class2 =
filtered_Classes(MarkersIdx(2):MarkersIdx(2)+500,4:35,i)';
    Class3 =
filtered_Classes(MarkersIdx(3):MarkersIdx(3)+500,4:35,i)';
```

```
end


X_csp1 = Left(:,:,i); % Right
X_csp2 = W2'*Right(:,:,i+100); % Left

for j = 1:32
res(1,j,i) = log( var( X_csp1(j,:)    ) / sum( var(X_csp1,0,2)
) );
res(2,j,i) = log( var( X_csp2(j,:)    ) / sum( var(X_csp2,0,2)
) );
end
end

M1 = res(1,:,1)

    for j = 1:148
    M1(j,:) = res(1,:,j);
    end

Features = [M1(:,1), M1(:,5), M1(:,7), M1(:,8), M1(:,27),
M1(:,28), M1(:,30), M1(:,32)]; % 1x8 vector
Features2 = [res2(1), res2(5), res2(7), res2(8), res2(27),
res2(28), res2(30), res2(32)];
Features3 = [res3(1), res3(5), res3(7), res3(8), res3(27),
res3(28), res3(30), res3(32)];
```

Python Biomedical Code For Feature extraction & Preprocessing

```
import numpy as np
from scipy.signal import butter, filtfilt
import scipy.io

def feature(data):
        signal = np.array(data)
        print(len(signal.T),len(signal.T[0]))
        mat_data_LR = scipy.io.loadmat('LeftVSRight(W1).mat')
        mat_data_RF = scipy.io.loadmat('RightVSFoot(W2).mat')
        mat_data_FL = scipy.io.loadmat('FootVSLeft(W3).mat')

        W1 = mat_data_LR['W1']
        W2 = mat_data_RF['W2']
        W3 = mat_data_FL['W3']

        # Generate a random signal with shape (32, 64)
        signal = signal.T

        # Pre-processing
        fs = 128
        lowcut = 8 / (fs / 2)
        highcut = 30 / (fs / 2)
        b, a = butter(8, [lowcut, highcut], 'bandpass')
        filtered_signal = filtfilt(b, a, signal)
```

```
        # Feature extraction
        X_csp1 = W1.T @ filtered_signal
        X_csp2 = W2.T @ filtered_signal
        X_csp3 = W3.T @ filtered_signal

        res1 = []
        res2 = []
        res3 = []

        for j in range(32):
                res1.append(np.log(np.var(X_csp1[j, :]) / np.sum(np.var(X_csp1,
axis=1))))
                res2.append(np.log(np.var(X_csp2[j, :]) / np.sum(np.var(X_csp2,
axis=1))))
                res3.append(np.log(np.var(X_csp3[j, :]) / np.sum(np.var(X_csp3,
axis=1))))

        # Select CSP features
        Features1 = [res1[0], res1[4], res1[6], res1[7], res1[26], res1[27],
res1[29], res1[31]]  # 1*8 vector
        Features2 = [res2[0], res2[4], res2[6], res2[7], res2[26], res2[27],
res2[29], res2[31]]
        Features3 = [res3[0], res3[4], res3[6], res3[7], res3[26], res3[27],
res3[29], res3[31]]
        print(Features1)
```

Emotiv Script Modified by the Biomedical engineering students to capture the streamed data and use it to control the wheelchair.

```
from cortex import Cortex
import time
import numpy as np
from scipy.signal import butter, filtfilt
import scipy.io
import pandas as pd
import Features
data_list = []

class Subcribe():
    """
    A class to subscribe data stream.

    Attributes
    ----------
    c : Cortex
        Cortex communicate with Emotiv Cortex Service

    Methods
    -------
    start():
        start data subscribing process.
    sub(streams):
        To subscribe to one or more data streams.
    on_new_data_labels(*args, **kwargs):
        To handle data labels of subscribed data
    on_new_eeg_data(*args, **kwargs):
        To handle eeg data emitted from Cortex
    on_new_mot_data(*args, **kwargs):
```

```
        To handle motion data emitted from Cortex
    on_new_dev_data(*args, **kwargs):
        To handle device information data emitted from Cortex
    on_new_met_data(*args, **kwargs):
        To handle performance metrics data emitted from Cortex
    on_new_pow_data(*args, **kwargs):
        To handle band power data emitted from Cortex
    """
    def __init__(self, app_client_id, app_client_secret, **kwargs):
        """
        Constructs cortex client and bind a function to handle subscribed data
streams
        If you do not want to log request and response message , set debug_mode
= False. The default is True
        """
        print("Subscribe __init__")
        self.c = Cortex(app_client_id, app_client_secret, debug_mode=True,
**kwargs)
        self.c.bind(create_session_done=self.on_create_session_done)
        self.c.bind(new_data_labels=self.on_new_data_labels)
        self.c.bind(new_eeg_data=self.on_new_eeg_data)
        self.c.bind(new_mot_data=self.on_new_mot_data)
        self.c.bind(new_dev_data=self.on_new_dev_data)
        self.c.bind(new_met_data=self.on_new_met_data)
        self.c.bind(new_pow_data=self.on_new_pow_data)
        self.c.bind(inform_error=self.on_inform_error)


    def start(self, streams, headsetId=''):
        """
        To start data subscribing process as below workflow
        (1)check access right -> authorize -> connect headset->create session
        (2) subscribe streams data
        'eeg': EEG
        'mot' : Motion
        'dev' : Device information
        'met' : Performance metric
        'pow' : Band power
        'eq' : EEQ Quality

        Parameters
        ----------
        streams : list, required
            list of streams. For example, ['eeg', 'mot']
        headsetId: string , optional
             id of wanted headet which you want to work with it.
             If the headsetId is empty, the first headset in list will be set as
wanted headset
        Returns
        -------
        None
        """
        self.streams = streams

        if headsetId != '':
            self.c.set_wanted_headset(headsetId)

        self.c.open()

    def sub(self, streams):
        """
        To subscribe to one or more data streams
```

```
        'eeg': EEG
        'mot' : Motion
        'dev' : Device information
        'met' : Performance metric
        'pow' : Band power

        Parameters
        ----------
        streams : list, required
            list of streams. For example, ['eeg', 'mot']

        Returns
        -------
        None
        """
        self.c.sub_request(streams)

    def unsub(self, streams):
        """
        To unsubscribe to one or more data streams
        'eeg': EEG
        'mot' : Motion
        'dev' : Device information
        'met' : Performance metric
        'pow' : Band power

        Parameters
        ----------
        streams : list, required
            list of streams. For example, ['eeg', 'mot']

        Returns
        -------
        None
        """
        self.c.unsub_request(streams)

    def on_new_data_labels(self, *args, **kwargs):
        """
        To handle data labels of subscribed data
        Returns
        -------
        data: list
              array of data labels
        name: stream name
        For example:
            eeg: ["COUNTER","INTERPOLATED", "AF3", "T7", "Pz", "T8", "AF4",
"RAW_CQ", "MARKER_HARDWARE"]
            motion: ['COUNTER_MEMS', 'INTERPOLATED_MEMS', 'Q0', 'Q1', 'Q2',
'Q3', 'ACCX', 'ACCY', 'ACCZ', 'MAGX', 'MAGY', 'MAGZ']
            dev: ['AF3', 'T7', 'Pz', 'T8', 'AF4', 'OVERALL']
            met : ['eng.isActive', 'eng', 'exc.isActive', 'exc', 'lex',
'str.isActive', 'str', 'rel.isActive', 'rel', 'int.isActive', 'int',
'foc.isActive', 'foc']
            pow: ['AF3/theta', 'AF3/alpha', 'AF3/betaL', 'AF3/betaH',
'AF3/gamma', 'T7/theta', 'T7/alpha', 'T7/betaL', 'T7/betaH', 'T7/gamma',
'Pz/theta', 'Pz/alpha', 'Pz/betaL', 'Pz/betaH', 'Pz/gamma', 'T8/theta',
'T8/alpha', 'T8/betaL', 'T8/betaH', 'T8/gamma', 'AF4/theta', 'AF4/alpha',
'AF4/betaL', 'AF4/betaH', 'AF4/gamma']
        """
        data = kwargs.get('data')
        stream_name = data['streamName']
```

```
        stream_labels = data['labels']
        print('{} labels are : {}'.format(stream_name, stream_labels))

    def on_new_eeg_data(self, *args, **kwargs):
        """
        To handle eeg data emitted from Cortex
        Returns
        -------
        data: dictionary
             The values in the array eeg match the labels in the array labels
return at on_new_data_labels
        For example:
           {'eeg': [99, 0, 4291.795, 4371.795, 4078.461, 4036.41, 4231.795, 0.0,
0], 'time': 1627457774.5166}
        """
        global data_list
        data = kwargs.get('data')
        #print('eeg data: {}'.format(data))
        print(data['eeg'][2:34])

        data_analysis = data['eeg'][2:34]
        data_list.append(data_analysis)
        print(len(data_list),len(data_list[0]))
        if len(data_list) == 64:
            print('-----------------------------------------------')
            Features.feature(data_list)
            data_list = []




    def on_new_mot_data(self, *args, **kwargs):
        """
        To handle motion data emitted from Cortex

        Returns
        -------
        data: dictionary
             The values in the array motion match the labels in the array labels
return at on_new_data_labels
        For example: {'mot': [33, 0, 0.493859, 0.40625, 0.46875, -0.609375,
0.968765, 0.187503, -0.250004, -76.563667, -19.584995, 38.281834], 'time':
1627457508.2588}
        """
        data = kwargs.get('data')
        print(data['mot'][8])
        #print('motion data: {}'.format(data))
        #time.sleep(0.7)
        if(data['mot'][8] > 0.2) :
            print('----------------------------------------------------------------
-----')
        else :
            print('Normal')

    def on_new_dev_data(self, *args, **kwargs):
        """
        To handle dev data emitted from Cortex

        Returns
        -------
        data: dictionary
```

```
            The values in the array dev match the labels in the array labels
return at on_new_data_labels
        For example:  {'signal': 1.0, 'dev': [4, 4, 4, 4, 4, 100],
'batteryPercent': 80, 'time': 1627459265.4463}
        """
        data = kwargs.get('data')
        print('dev data: {}'.format(data))



    def on_new_met_data(self, *args, **kwargs):
        """
        To handle performance metrics data emitted from Cortex

        Returns
        -------
        data: dictionary
            The values in the array met match the labels in the array labels
return at on_new_data_labels
        For example: {'met': [True, 0.5, True, 0.5, 0.0, True, 0.5, True, 0.5,
True, 0.5, True, 0.5], 'time': 1627459390.4229}
        """
        data = kwargs.get('data')

        print('pm data: {}'.format(data))

    def on_new_pow_data(self, *args, **kwargs):
        """
        To handle band power data emitted from Cortex

        Returns
        -------
        data: dictionary
            The values in the array pow match the labels in the array labels
return at on_new_data_labels
        For example: {'pow': [5.251, 4.691, 3.195, 1.193, 0.282, 0.636, 0.929,
0.833, 0.347, 0.337, 7.863, 3.122, 2.243, 0.787, 0.496, 5.723, 2.87, 3.099,
0.91, 0.516, 5.783, 4.818, 2.393, 1.278, 0.213], 'time': 1627459390.1729}
        """
        data = kwargs.get('data')
        print('pow data: {}'.format(data))

    # callbacks functions
    def on_create_session_done(self, *args, **kwargs):
        print('on_create_session_done')

        # subribe data
        self.sub(self.streams)

    def on_inform_error(self, *args, **kwargs):
        error_data = kwargs.get('error_data')
        print(error_data)

# -----------------------------------------------------------
#
# GETTING STARTED
#   - Please reference to https://emotiv.gitbook.io/cortex-api/ first.
#   - Connect your headset with dongle or bluetooth. You can see the headset via
Emotiv Launcher
#   - Please make sure the your_app_client_id and your_app_client_secret are set
before starting running.
#   - In the case you borrow license from others, you need to add license =
```

```
"xxx-yyy-zzz" as init parameter
# RESULT
#    - the data labels will be retrieved at on_new_data_labels
#    - the data will be retreived at on_new_[dataStream]_data
#
# ----------------------------------------------------------------



def main():


    # Please fill your application clientId and clientSecret before running
script
    your_app_client_id = 'Vqfs127oS3c3Mspaofqb0h0Xe6naZrzRbEBau1o0'
    your_app_client_secret =
'UpOHsrrugPEItxhkIAstnJGSztSd3cSsufUG2Ymq1GcEbkk6CfgIhIfvtputrHv7QN4KHtkTYpqY9h8
mPbdt8NHDN9wnFk0Oj3h3Wfl9gQHKtpSYF8LNiwunFkvroxGI'

    s = Subcribe(your_app_client_id, your_app_client_secret)

    # list data streams
    streams = ['eeg','mot']
    s.start(streams)

if __name__ =='__main__':
    main()

# ----------------------------------------------------------------
```

APPENDIX B

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun  4 01:06:56 2023

@author: maral.demirsecen

3 LDAs

"""

import numpy as np
from scipy.io import loadmat
#ml
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
from sklearn.model_selection import train_test_split
#accuracies
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error, r2_score
#preprocessing
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE
# to read real time data
import scipy.io as sio

# Load the data
data_left_forward = loadmat(r'C:\Users\Dell\Downloads\eeg-data-w-combinations\LeftVSFoot_8Features(M2).mat')
data_right_forward = loadmat(r'C:\Users\Dell\Downloads\eeg-data-w-combinations\RightVSFoot_8Features(M1).mat')
data_right_left = loadmat(r'C:\Users\Dell\Downloads\eeg-data-w-combinations\LeftVSRight_8Features(M3).mat')

# Extract data
data_lf = data_left_forward['M2']
data_rf = data_right_forward['M1']
data_rl = data_right_left['M3']

# Extract features and labels for each class combination
X_lf = data_lf[:, :-1]
y_lf = data_lf[:, -1]

X_rf = data_rf[:, :-1]
y_rf = data_rf[:, -1]

X_rl = data_rl[:, :-1]
y_rl = data_rl[:, -1]
```

```
        #preprocessing

    # IQR - removing outliers

        #removing outliers for left-forward
Q1 = np.percentile(X_lf, 25, axis=0, interpolation='midpoint')
Q3 = np.percentile(X_lf, 75, axis=0, interpolation='midpoint')
IQR = Q3 - Q1

    # Above Upper bound
upper = Q3 + 1.5 * IQR
upper_array = np.array(X_lf >= upper)

    # Below Lower bound
lower = Q1 - 1.5 * IQR
lower_array = np.array(X_lf <= lower)

    # True values (outliers) in the array
upper_indices = np.argwhere(upper_array)
lower_indices = np.argwhere(lower_array)

    # dropping the outliers
if upper_indices.size > 0:
    X_lf = np.delete(X_lf, upper_indices[:, 0], axis=0)
    y_lf = np.delete(y_lf, upper_indices[:, 0], axis=0)

if lower_indices.size > 0:
    X_lf = np.delete(X_lf, lower_indices[:, 0], axis=0)
    y_lf = np.delete(y_lf, lower_indices[:, 0], axis=0)

        #removing outliers for right-forward
Q1_rf = np.percentile(X_rf, 25, axis=0, interpolation='midpoint')
Q3_rf = np.percentile(X_rf, 75, axis=0, interpolation='midpoint')
IQR_rf = Q3_rf - Q1_rf
```

```
    # Above Upper bound
upper_rf = Q3_rf + 1.5 * IQR_rf
upper_array_rf = np.array(X_rf >= upper_rf)

    # Below Lower bound
lower_rf = Q1_rf - 1.5 * IQR_rf
lower_array_rf = np.array(X_rf <= lower_rf)

    # True values (outliers) in the array
upper_indices_rf = np.argwhere(upper_array_rf)
lower_indices_rf = np.argwhere(lower_array_rf)

    # dropping the outliers
if upper_indices_rf.size > 0:
    X_rf = np.delete(X_rf, upper_indices_rf[:, 0], axis=0)
    y_rf = np.delete(y_rf, upper_indices_rf[:, 0], axis=0)

if lower_indices_rf.size > 0:
    X_rf = np.delete(X_rf, lower_indices_rf[:, 0], axis=0)
    y_rf = np.delete(y_rf, lower_indices_rf[:, 0], axis=0)




        #removing outliers for right-forward
Q1_rf = np.percentile(X_rf, 25, axis=0, interpolation='midpoint')
Q3_rf = np.percentile(X_rf, 75, axis=0, interpolation='midpoint')
IQR_rf = Q3_rf - Q1_rf

    # Above Upper bound
upper_rf = Q3_rf + 1.5 * IQR_rf
upper_array_rf = np.array(X_rf >= upper_rf)

    # Below Lower bound
```

```
lower_rf = Q1_rf - 1.5 * IQR_rf
lower_array_rf = np.array(X_rf <= lower_rf)

    # True values (outliers) in the array
upper_indices_rf = np.argwhere(upper_array_rf)
lower_indices_rf = np.argwhere(lower_array_rf)

    # dropping the outliers
if upper_indices_rf.size > 0:
    X_rf = np.delete(X_rf, upper_indices_rf[:, 0], axis=0)
    y_rf = np.delete(y_rf, upper_indices_rf[:, 0], axis=0)

if lower_indices_rf.size > 0:
    X_rf = np.delete(X_rf, lower_indices_rf[:, 0], axis=0)
    y_rf = np.delete(y_rf, lower_indices_rf[:, 0], axis=0)

        # removing outliers for right-left
Q1_rl = np.percentile(X_rl, 25, axis=0, interpolation='midpoint')
Q3_rl = np.percentile(X_rl, 75, axis=0, interpolation='midpoint')
IQR_rl = Q3_rl - Q1_rl

    # Above Upper bound
upper_rl = Q3_rl + 1.5 * IQR_rl
upper_array_rl = np.array(X_rl >= upper_rl)

    # Below Lower bound
lower_rl = Q1_rl - 1.5 * IQR_rl
lower_array_rl = np.array(X_rl <= lower_rl)

    # True values (outliers) in the array
upper_indices_rl = np.argwhere(upper_array_rl)
lower_indices_rl = np.argwhere(lower_array_rl)

    # dropping the outliers
```

```
if upper_indices_rl.size > 0:
    X_rl = np.delete(X_rl, upper_indices_rl[:, 0], axis=0)
    y_rl = np.delete(y_rl, upper_indices_rl[:, 0], axis=0)

if lower_indices_rl.size > 0:
    X_rl = np.delete(X_rl, lower_indices_rl[:, 0], axis=0)
    y_rl = np.delete(y_rl, lower_indices_rl[:, 0], axis=0)



    # Scale the features using MinMaxScaler
scaler = MinMaxScaler()
X_lf = scaler.fit_transform(X_lf)
X_rf = scaler.fit_transform(X_rf)
X_rl = scaler.fit_transform(X_rl)



# Apply SMOTE to balance class distribution
smote = SMOTE()
X_lf, y_lf = smote.fit_resample(X_lf, y_lf)
X_rf, y_rf = smote.fit_resample(X_rf, y_rf)
X_rl, y_rl = smote.fit_resample(X_rl, y_rl)



    # Define the class combinations
class_combinations = [(1, 2), (1, 3), (2, 3)]
#left:1 right:2 forward:3


    # Train LDA models for each class combination
ldas = []
```

```
    for combination in class_combinations:
        class1, class2 = combination

        # Prepare the data for the current combination
        X = np.concatenate((X_lf[y_lf == class1], X_lf[y_lf == class2]))
        y = np.concatenate((np.zeros(np.sum(y_lf == class1)), np.ones(np.sum(y_lf == class2))))
#0-1 for differenting between 2 classes

        # Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

        # Train the LDA model
        lda = LinearDiscriminantAnalysis()
        lda.fit(X_train, y_train)

        # Predict the class labels for the test set
        y_pred = lda.predict(X_test)

        # Add the trained LDA model to the list
        ldas.append(lda)

        # Evaluate the accuracy of the model
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average='macro')
        recall =  recall_score(y_test, y_pred, average='macro')
        f1 = f1_score(y_test, y_pred, average='macro')
        mse=mean_squared_error(y_test, y_pred)
        mape = mean_absolute_percentage_error(y_test, y_pred)
        r2= r2_score(y_test, y_pred)
        print(f"Accuracy for classes {class1}-{class2}: {accuracy}")
        print(f"Precision for classes {class1}-{class2}: {precision}")
        print(f"Recall for classes {class1}-{class2}: {recall}")
        print(f"F1-score for classes {class1}-{class2}: {f1}")
        print(f"Mean Squared Error for classes {class1}-{class2}: {mse}")
```

```
        print(f"Mean Absolute Percentage Error for classes {class1}-{class2}: {mape}")
        print(f"R^2 Score for classes {class1}-{class2}: {r2}")




        # Make predictions on a single sample
    #example sample points
    sample = np.array([-1.43589, -3.31068, -3.14798, -4.06314, -5.37112, -4.65965, -4.94756, -5.71435]) #3
    sample= np.array([-3.09556, -3.64001, -3.27239, -4.28112, -5.19645, -4.50978, -4.30835, -5.82496]) #3
    sample= np.array([-3.49385, -4.22158, -3.72729, -4.74264, -2.90173, -4.41674, -2.75319, -3.0339]) #2
    sample = np.array([-3.6087, -4.35362, -4.06211, -4.84057, -3.82706, -3.89551, -3.31436, -3.33825]) #1


    # Function to predict the class of a sample
    def predict_class(sample):
        predictions = []

        for lda in ldas:
            prediction = lda.predict([sample])
            predictions.append(prediction[0])

        # Map the predictions to the corresponding classes
        result = []

        if predictions[0] == 0 and predictions[1] == 0 and predictions[2] == 0:
            result.append('left')
        elif predictions[0] == 1 and predictions[1] == 0 and predictions[2] == 0:
            result.append('not classified')
```

```
    elif predictions[0] == 0 and predictions[1] == 1 and predictions[2] == 0:
        result.append('forward')
    elif predictions[0] == 1 and predictions[1] == 1 and predictions[2] == 0:
        result.append('forward')
    elif predictions[0] == 0 and predictions[1] == 0 and predictions[2] == 1:
        result.append('left')
    elif predictions[0] == 1 and predictions[1] == 0 and predictions[2] == 1:
        result.append('right')
    elif predictions[0] == 0 and predictions[1] == 1 and predictions[2] == 1:
        result.append('not classified')
    elif predictions[0] == 1 and predictions[1] == 1 and predictions[2] == 1:
        result.append('right')

    return result

print("Predicted class:", predict_class(sample))
#accuracy is really goof but without overfitting - tried on validation sample

"""
if the result is - then class is:
    null-0-0 : left ;
    1-0-0 not classified;
    0-1-0 forward;
    1-1-0 forward;
    0-0-1 left;
    1-0-1 right;
    0-1-1 not classified;
    1-1-1 right
    #not classified = stop
"""


        #Real Time  Prediction
```

```
while True:
    # Get input from user
    user_input = input("Enter 8 values (separated by spaces): ")

    # Split the input string into a list of values
    values = user_input.split() #need to seperated by space

    # Convert the values to float and create a numpy array
    sample = np.array([float(val) for val in values])

    # Make prediction using the predict_class function
    predicted_class = predict_class(sample)

    # Print the predicted class
    print("Predicted class:", predicted_class)

#-1.43589 -3.31068 -3.14798 -4.06314 -5.37112 -4.65965 -4.94756 -5.71435
```

APPENDIX C

1_ Rasberry PI Codes:

```
# -*- coding: utf-8 -*-
"""
Created on Sun May 28 16:48:17 2023


@author: Utku ARSLAN

This code is responsible from receiving data and appliying proper movement to vehicle
with its sensor
"""
import socket
from threading import Lock,Thread
import time
import serial
import traceback
from datetime import datetime
```

```
###################################
stop_distance=20
motor_speed=80
motor_speed_base=1500
###################################



class M_Broadcast:
    def __init__(self,port=7070,connection=None):
        self.port=port
        self.tag='[UDP-Broadcast]'
        print(f'{self.tag} initializing.')
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        ip = socket.gethostname()
        self.sock.bind(( ip,self.port))
        self.start_reader()
        self.connection=connection
    def start_reader(self):
        Thread(target=self.reader_thread).start()

    def reader_thread(self):
        print(f'{self.tag} listening port {self.port}')
        while True:
            data, address = self.sock.recvfrom(1024)
            print(self.tag,data)
            #self.sock.sendto('Im here!'.encode('utf-8'), address)
            if self.connection !=None:
              self.connection.sender("Im here!".encode('utf-8'),address)


class M_UDP:

    """


    """

    udp_target="utkuarslan.xyz"
    udp_port=6565

    udp_self_port=6060

    def __init__(self,aware_ping=True,aware_time=60,reader=True):
        self.tag='[UDP]'
        self.reader=reader
```

```
        self.receive_listeners=[]
        self.aware=aware_ping
        self.aware_time=aware_time

        print(f'{self.tag} initializing.')
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        #name = socket.gethostname()
        #ip=socket.gethostbyname(name)
        #dt = self.sock.bind()

        self.lock = Lock()
        self.aware_lock=Lock()
        self.thread_start_loop()
        if reader:
            self.reader_lock=Lock()
            self.thread_start_reader()

        #self.sender('Raspi Available!')

    #////////////////////[Sender]
    #msg sender, takes argument as a str
    def sender(self,msg,adress= 'utkuarslan.xyz', port=6565):
        msg=msg.encode('utf-8')
        with self.lock:
            try:
                print('[Sender]',msg,adress,port)
                self.sock.sendto(msg, (adress, port))
            except Exception:
                print(f'{self.tag}[ERR] couldnt send the data to target')
                print(traceback.format_exc())


    #////////////////////[Aware-Functions]
    def stop_aware(self):
        with self.aware_lock:
            self.aware=False

    def start_aware(self):
        if not self.aware:
            self.aware=True
            self.thread_start_loop()

    def thread_start_loop(self):
        print(f'{self.tag} thread starting.')
        Thread(target=self.aware_loop).start()

    def aware_loop(self):
        while True:
            with self.aware_lock:
```

```
                    if not self.aware:
                        #print(f'{self.tag} interupting loop.')
                        break
                self.sender('Raspi Available!')
                time.sleep(self.aware_time)

        #////////////////////[Reader]
        def thread_start_reader(self):
            Thread(target=self.reader_loop).start()


        def reader_loop(self):
            while True:
                with self.reader_lock:
                    if not self.reader:
                        break
                data, address = self.sock.recvfrom(1024)
                print(data)
                data=data.decode("utf-8")
                self.l_send(data, address)


        def l_send(self,data,address):
            print(f'{self.tag}[Receive] {data}')
            for f in self.receive_listeners:
                try:
                    f(data,address)
                except Exception:
                    print(f'{self.tag}[OnListener]   error   accured   on   running
listener function')
                    print(traceback.format_exc())

        #////////////////////[Reader]
        def add_listener(self,listener_function):
            self.receive_listeners.append(listener_function)

        def remove_listener(self,listener_f):
            self.receive_listeners.remove(listener_f)

        #/////////////////////[Shutdown]
        def shutdown(self):
            self.reader=False
            self.stop_aware()
            try:
                self.sock.shutdown(socket.SHUT_RDWR)
            except:
                pass
            print(f'{self.tag}[Shutdown] Socket closed')
```

```
class M_Serial_2:


    def onSerialMSGReceived(self, msg, port):
        return


    def writeSerial(self, data):
        self.arduino.write(data.encode())


    def   __init__(self,   _port='/dev/ttyUSB0',   _baudrate=9600,sleep=0.09,
loop=True,debug=False,listener=None):
        self.debug=debug

        self.serial_port = _port
        self.serial_sleep=sleep
        self.serial_lock=Lock()
        self.loop_read=loop
        self.listener = listener
        self.init()
        if not debug:
            #self.init()
            return
        else:
            print("[Serial_2][DEBUG MODE] Serial ports not ready...")

    def init(self):
        self.thread_loop = Thread(target=self.serial_starting)
        self.thread_loop.start()
        print(f"[Serial_2] initialized on port: {self.serial_port} ")


    # Tries to connect Arduino via Serial
    def serial_setup(self):

        try:
            self.arduino = serial.Serial(self.serial_port, 9600)
            print("[Serial_2] serial connected on:"+str(self.serial_port))
            return True
        except Exception as e:
            print(e)
            print("[Serial_2]       serial       failed       to       connect
on:"+str(self.serial_port))
            return False
```

```
        # Serial starting
        def serial_starting(self):

            #Bağlantı kurulana kadar 5 sn boyunca dener
            while self.serial_setup() is False:
                time.sleep(5)

            if self.loop_read:
                self.serial_loop()



        def serial_loop(self):
            print('[SerialRead_2] read loop started...')
            while True:
                data = self.arduino.readline()
                print('[SerialRead_2]',data)
                #data="41.2587:22.789;174#"
                self.listener(data, self.serial_port)
                #time.sleep(self.serial_sleep)


        def writeSerial(self, data):
            #print(data[0])
            if self.debug:
                return
            with self.serial_lock:
                self.arduino.write(data[0]) #TODO


    class M_Sensors_2():


        def __init__(self,main_control=None,
_listenPort='/dev/ttyACM0',debug=False):
            self.controller = main_control
            self.debug=debug
            self.sensor_lock = Lock()
            #M_Serial.__init__(self, _listenPort,sleep=0.04,debug=debug)
            self.serial =
M_Serial_2(_port=_listenPort,sleep=0.04,debug=debug,listener=self.onSerialMSGReceive
d)
            print("[Sensors 2] initializing...")
            self.ultrasonic_front = 0
            self.ultrasonic_left = 0
            self.ultrasonic_right = 0

        #28:213;12/\r\n
```

```
def onSerialMSGReceived(self, msg, port):
    temp = ""

    #print(msg)
    try:
        msg=msg.decode()
    except:
        print("[Serial_2][ERR] UTF-8")
    for c in msg:
        if c == ':':
            with self.sensor_lock:
                self.ultrasonic_front = int(temp)
                temp=""
                #print("LO:"+str(self.gps_lon))
        elif c == ';':
            with self.sensor_lock:
                self.ultrasonic_left = int(temp)
                temp = ""
                #print("LA:"+str(self.gps_lat))
        elif c == '/':
            if len(temp) == 0 :
                return
            with self.sensor_lock:
                self.ultrasonic_right = int(temp)
                temp = ""
                #print("R:"+str(self.rotation))
        else:
            temp += str(c)

    if self.controller.get_last_msg() == 'forward':
        if self.getFront() <= 20:
            self.controller.motor.sendMsgMotor("1500:1500;")
    elif self.controller.get_last_msg() == 'right':
        if self.getRight() <= 20:
            self.controller.motor.sendMsgMotor("1500:1500;")
    elif self.controller.get_last_msg() == 'left':
        if self.getLeft() <= 20:
            self.controller.motor.sendMsgMotor("1500:1500;")



def getFront(self):
    with self.sensor_lock:
        return self.ultrasonic_front
def getLeft(self):
    with self.sensor_lock:
        return self.ultrasonic_left
def getRight(self):
    with self.sensor_lock:
```

```
                return self.ultrasonic_right
        def getDataStr(self):
            with self.sensor_lock:
                return str(self.ultrasonic_front) + ":" + str(self.ultrasonic_left)
+ ";" + str(self.ultrasonic_right)+"/"


    class M_Motors():


        def __init__(self,_listenPort='/dev/ttyUSB0',debug=False):
            print("[Motors] initializing...")

            self.serial_lock=Lock()
            self.debug=debug

            if not debug:
                self.arduino = serial.Serial(_listenPort, 9600)
                print("[Motors] Arduino created...")
                self.writeSerial("1500:1500;".encode())
                self.left = 1500
                self.right = 1500


        def writeSerial(self, data):
            if self.debug:
                print(f'[M_Motor] received data {str(data)}')
                return
            #print(data[0])
            with self.serial_lock:
                self.arduino.write(data) #TODO
                #print("[Seial-Debug] writing data"+str(data))


        def checkRange(self, value, max=1900, min=1100):
            return max if value > max else (min if min > value else value)


        def setMotorsM(self, left, right):  # range 1900-1100
            self.left = self.checkRange(left)
            self.right = self.checkRange(right)
            self.updateMotors()


        # -400 backward / 0 stop / 400 Forward
        def setMotors(self, left=0, right=0):
            self.left = self.checkRange(1500 + left)
            self.right = self.checkRange(1500 + right)
            self.updateMotors()
```

```
        def updateMotors(self):
            self.writeSerial((str(self.left)    +    ":"    +    str(self.right)    +
";").encode())


        def sendMsgMotor(self, ss):
            self.writeSerial(ss) # TODO

            """
        def writeSerial(self, data):
            #print(data[0])
            with self.serial_lock:
                self.arduino.write(data) #TODO
                #print("[Seial-Debug] writing data"+str(data))
                """


        def checkRange(self, value, max=1900, min=1100):
            return max if value > max else (min if min > value else value)


        def setMotorsM(self, left, right):  # range 1900-1100
            self.left = self.checkRange(left)
            self.right = self.checkRange(right)
            self.updateMotors()


        # -400 backward / 0 stop / 400 Forward
        def setMotors(self, left=0, right=0):
            self.left = self.checkRange(1500 + left)
            self.right = self.checkRange(1500 + right)
            self.updateMotors()


        def updateMotors(self):
            self.writeSerial((str(self.left)    +    ":"    +    str(self.right)    +
";").encode())


        def sendMsgMotor(self, ss):
            self.writeSerial(ss.encode('UTF-8')) # TODO

    class MainController:

        """
        Expected commands:
            Headset Commands:
```

```
                forward-left(turning around)-right(turning around)-stop
            Server Commands:
                1500:1500;      ->For   motors   value,   1100:Backward,   1500:Stop
1900:Forward
                s           ->To sends the sensor data back
                c           ->Block the headset
                h           ->Enable the headset
                p           ->Close pinging

        """
        def __init__(self):
            DEBUG=True
            self.last_msg_lock=Lock()
            self.last_msg=None
            self.motor=M_Motors(_listenPort='/dev/ttyUSB1',debug=False)

self.sensors=M_Sensors_2(main_control=self,_listenPort='/dev/ttyUSB0',debug=True)
            #self.broadcast=M_Broadcast()
            self.tag='[H-Controller]'

            #Server Configuration
            self.connection=M_UDP()
            self.connection.add_listener(self.on_server)
            self.connection_lock=Lock()

            self.headset_control=True

        def get_last_msg(self):
            with self.last_msg_lock:
                return self.last_msg

        def set_last_msg(self):
            with self.last_msg_lock:
                self.last_msg="1500:1500;"

        def on_server(self,data,adress):
            print(f'{self.tag}            server            "{data}"            -
{datetime.now().strftime("%H:%M:%S")} - {self.headset_control}')
            self.last_msg=data
            if data.endswith(';'):
                if "1500:1500;" != data:
                    self.headset_control=False
                self.motor.sendMsgMotor(data)
            elif len(data)==1:
                if data=='h':
                    self.headset_control=True

self.connection.sender("headset",adress=adress[0],port=adress[1])
                    print(f'{self.tag} headset controller Enabled')
```

```
                elif data=='c':
                    self.headset_control=False

self.connection.sender("blocked",adress=adress[0],port=adress[1])
                    print(f'{self.tag} headset controller Blocked')
                elif data=='s':

self.connection.sender(self.sensors.getDataStr(),adress=adress[0],port=adress[1])
                elif data=='p':
                    print(f'{self.tag} aware pinging turned off')
                    self.connection.stop_aware()

            elif self.headset_control:
                print(f'{self.tag} Headset_Control')
                if 'forward' == data:
                    if self.sensors.getFront() > stop_distance:
                        #self.change_motors(200,200)

self.motor.sendMsgMotor(f"{motor_speed_base+motor_speed}:{motor_speed_base+motor_spe
ed};")
                    else:

self.motor.sendMsgMotor(f"{motor_speed_base}:{motor_speed_base};")
                elif 'left' == data:
                    if self.sensors.getLeft() > stop_distance:
                        #self.change_motors(-200,200)
                        self.motor.sendMsgMotor(f"{motor_speed_base-
motor_speed}:{motor_speed_base+motor_speed};")
                    else:

self.motor.sendMsgMotor(f"{motor_speed_base}:{motor_speed_base};")
                elif 'right' == data:
                    if self.sensors.getRight() > stop_distance:
                        #self.change_motors(200,-200)

self.motor.sendMsgMotor(f"{motor_speed_base+motor_speed}:{motor_speed_base-
motor_speed};")
                    else:

self.motor.sendMsgMotor(f"{motor_speed_base}:{motor_speed_base};")
                elif 'stop' == data:
                    #self.change_motors(0,0)

self.motor.sendMsgMotor(f"{motor_speed_base}:{motor_speed_base};")

        def change_motors(self,left,right):
            print(f'{self.tag} {left},{right}')
            return
```

```
if __name__ == '__main__':

    M=MainController()
```

2_ Dedicated Linux Server Codes:

```
    """
    Created on Thu Jun  1 12:16:27 2023


    @author: Utku ARSLAN

    This code is responsible from transmitting the data between RaspberyPI and
User/Controller
    """


    import socket
    from threading import Lock,Thread
    import time
    import traceback
    from datetime import datetime




    class M_UDP:

        """
        def on_receive(data,adress):

        """
        udp_target="utkuarslan.xyz"

        def
__init__(self,aware_ping=False,aware_time=60,reader=True,port=6565,tag='[UDP]'):
            self.tag=tag
            self.reader=reader
            self.receive_listeners=[]
            self.aware=aware_ping
            self.aware_time=aware_time
            self.port=port

            print(f'{self.tag} initializing.')
```

```
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        ip = socket.gethostname()
        self.sock.bind(( "utkuarslan.xyz",self.port))
        self.lock = Lock()
        self.aware_lock=Lock()
        self.thread_start_loop()
        if reader:
            self.reader_lock=Lock()
            self.thread_start_reader()

        #self.sender('Raspi Available!')

    #////////////////////[Sender]
    #msg sender, takes argument as a str
    def sender(self,msg,adress= 'utkuarslan.xyz', port=6565):
        msg=msg.encode('utf-8')
        with self.lock:
            try:
                self.sock.sendto(msg, (adress, port))
            except Exception:
                print(f'{self.tag}[ERR] couldnt send the data to target')
                print(traceback.format_exc())


    #////////////////////[Aware-Functions]
    def stop_aware(self):
        with self.aware_lock:
            self.aware=False

    def start_aware(self):
        if not self.aware:
            self.aware=True
            self.thread_start_loop()

    def thread_start_loop(self):
        return
        print(f'{self.tag} thread starting.')
        Thread(target=self.aware_loop).start()

    def aware_loop(self):
        while True:
            with self.aware_lock:
                if not self.aware:
                    break
            self.sender('Raspi Available!')
            time.sleep(self.aware_time)

    #////////////////////[Reader]
    def thread_start_reader(self):
```

```
            Thread(target=self.reader_loop).start()


        def reader_loop(self):
            while True:
                with self.reader_lock:
                    if not self.reader:
                        break
                data, address = self.sock.recvfrom(1024)
                print(data)
                data=data.decode("utf-8")
                self.l_send(data, address)


        def l_send(self,data,address):
            #print(f'{self.tag}[Receive] {data}')
            for f in self.receive_listeners:
                try:
                    f(data,address)
                except Exception:
                    print(f'{self.tag}[OnListener]   error   accured   on   running
listener function')
                    print(traceback.format_exc())

        #////////////////////[Reader]
        def add_listener(self,listener_function):
            self.receive_listeners.append(listener_function)

        def remove_listener(self,listener_f):
            self.receive_listeners.remove(listener_f)

        #////////////////////[Shutdown]
        def shutdown(self):
            self.reader=False
            self.stop_aware()
            try:
                self.sock.shutdown(socket.SHUT_RDWR)
            except:
                pass
            print(f'{self.tag}[Shutdown] Socket closed')


    class MainServer:

        def __init__(self):
            self.con_pi=M_UDP(port=6565,tag='[UDP-Pi]')
            self.con_user=M_UDP(port=6464,tag='[UDP-User]')
            self.con_pi.add_listener(self.on_pi_message)
            self.con_user.add_listener(self.on_user_message)
```

```
            self.user_adress=None
            self.pi_adress=None
            self.pi_last="-1"

        def on_pi_message(self,data,address):
            print('[OnPi] ',data)
            self.pi_last=datetime.now()
            self.pi_adress = address#Raspi Available!
            if data=='Raspi Available!':
              self.con_pi.sender('Hey!',address[0],address[1])
              print('OnPi--',address[0],address[1])
              return
            if self.user_adress!=None:
                self.con_user.sender(data,self.user_adress[0],self.user_adress[1])


        def on_user_message(self,data,address):
            print('[OnUser] ',data)
            self.user_adress = address
            if data=='last':

self.con_user.sender(str(self.pi_last)+str(self.pi_adress),address[0],address[1])
              return
            if self.pi_adress!=None:
                self.con_pi.sender(data,self.pi_adress[0],self.pi_adress[1])
                print("Sent                                                    to:
",data,self.pi_adress[0],self.pi_adress[1],"data:",data)

    MainServer()
```

3_ Simple Remote Control App for Testing/Debugging without Headset connection

```
    """
    Created on Sun May 28 16:48:17 2023


    @author: Utku ARSLAN


    Apache-2.0 license
    """


    # Import the Required libraries
    from tkinter import *
    import socket


    # Create an instance of tkinter frame or window
```

```
win= Tk()
print('1')
# Set the size of the window
win.geometry("700x350")


#####


sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
print('2')
ip = socket.gethostname()
sock.bind(( ip,6464))
sock.sendto('last'.encode('utf-8'), ('utkuarslan.xyz', 6464))
print('3')
data, address = sock.recvfrom(1024)
last_msg=data.decode('utf-8')
print('4')
print('[RECV] ',data.decode('UTF-8'))
#####


# Define a function to display the message
left_motor = 1500
right_motor = 100
max_speed = 100
base_motor = 1500


last_key_press='a'
def key_press(e):
   global last_msg,left_motor,right_motor
   label.config(text=f"{last_msg} {e.char}")
   c=e.char
   last_key_press=c
   if c == 'w':
      left_motor=base_motor+max_speed
      right_motor = base_motor+max_speed
      sock.sendto((f'{left_motor}:{right_motor};').encode('utf-8'),
('utkuarslan.xyz', 6464))
   elif  c == 'a':
      left_motor=base_motor+max_speed*(-1)
      right_motor = base_motor+max_speed
      sock.sendto((f'{left_motor}:{right_motor};').encode('utf-8'),
('utkuarslan.xyz', 6464))
   elif  c == 'd':
      left_motor=base_motor+max_speed
      right_motor = base_motor+max_speed*(-1)
      sock.sendto((f'{left_motor}:{right_motor};').encode('utf-8'),
```

```
('utkuarslan.xyz', 6464))
        elif  c == 's':
            left_motor=base_motor+max_speed*(-1)
            right_motor = base_motor+max_speed*(-1)
            sock.sendto((f'{left_motor}:{right_motor};').encode('utf-8'),
('utkuarslan.xyz', 6464))
        elif  c == 'q':
            left_motor=base_motor
            right_motor = base_motor+max_speed
            sock.sendto((f'{left_motor}:{right_motor};').encode('utf-8'),
('utkuarslan.xyz', 6464))
        elif  c == 'e':
            left_motor = base_motor+max_speed
            right_motor = base_motor
            sock.sendto((f'{left_motor}:{right_motor};').encode('utf-8'),
('utkuarslan.xyz', 6464))
        elif  c == 'l':
            sock.sendto('last'.encode('utf-8'), ('utkuarslan.xyz', 6464))
            data, address = sock.recvfrom(1024)
            last_msg=data.decode('utf-8')
            label.config(text=f"{last_msg} {e.char}")
        elif  c == 'h':
            sock.sendto('h'.encode('utf-8'), ('utkuarslan.xyz', 6464))
            #data, address = sock.recvfrom(1024)
            #last_msg=data.decode('utf-8')
            #label.config(text=f"{last_msg} {e.char}")
        elif  c == 'c':
            sock.sendto('c'.encode('utf-8'), ('utkuarslan.xyz', 6464))
            #data, address = sock.recvfrom(1024)
            #last_msg=data.decode('utf-8')
            #label.config(text=f"{last_msg} {e.char}")
        elif  c == 'm':
            sock.sendto('s'.encode('utf-8'), ('utkuarslan.xyz', 6464))
            data, address = sock.recvfrom(1024)
            last_msg=data.decode('utf-8')
            label.config(text=f"{last_msg} {e.char}")
        elif  c == 'f':
            sock.sendto('forward'.encode('utf-8'), ('utkuarslan.xyz', 6464))
        elif  c == 'r':
            sock.sendto('right'.encode('utf-8'), ('utkuarslan.xyz', 6464))
        elif  c == 'v':
            sock.sendto('left'.encode('utf-8'), ('utkuarslan.xyz', 6464))
        elif  c == 'p':
            sock.sendto('1500:1500;'.encode('utf-8'), ('utkuarslan.xyz', 6464))
```

```
        def key_released(e):
            label.config(text=f"{last_msg}")
            left_motor = base_motor
            right_motor = base_motor
            if(last_key_press=='v' or last_key_press=='f' or last_key_press=='r'):
                  return
            sock.sendto((f'{left_motor}:{right_motor};').encode('utf-8'),
('utkuarslan.xyz', 6464))




        label= Label(win, text= last_msg, font= ('Helvetica 17 bold'))
        label.pack(pady= 50)


        w = Canvas(win, width=300, height=300)
        #rectangle = w.create_rectangle(0, 0, 20, 140, fill='orange')
        #w.create_rectangle(0, 0, 100, 100, fill="blue", outline = 'blue')
        #w.create_rectangle(50, 50, 100, 100, fill="red", outline = 'blue')
        w.pack()


        #canvas.itemconfig(rectangle, fill='green')



        # Bind the Mouse button event
        win.bind('<KeyPress>',key_press)
        win.bind('<KeyRelease>',key_released )
        win.mainloop()
```

4_ Motor Controller Arduino:

```
        /*
        //Coded by Utku ARSLAN
        //
        //10.06.2023
        //Apache-2.0 license


        //This code is responsible from rotating motors with L298 with received data
from Raspberry PI via Usb/Serial Protocol
        */
```

```
String temp="";
int left_motor=1500;
int right_motor=1500;


int dir1PinA = 2;
int dir2PinA = 3;
int speedPinA = 9; // PWM pini, hız kontrolü için

// Motor 2

int dir1PinB = 4;
int dir2PinB = 5;
int speedPinB = 10; // PWM pini, hız kontrolü için
//d3(1) - EN1/ D5(9)-EN2 / D9(13)-EN3 / D10(14)-EN4



void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  pinMode(dir1PinA,OUTPUT);
  pinMode(dir2PinA,OUTPUT);
  pinMode(speedPinA,OUTPUT);
  pinMode(dir1PinB,OUTPUT);
  pinMode(dir2PinB,OUTPUT);
  pinMode(speedPinB,OUTPUT);



}


void update_motors(){
  if(left_motor==1500){

    analogWrite(speedPinA, 0);


    //digitalWrite(l1,LOW);
    //digitalWrite(l2,LOW);
  }else{
    if(left_motor>1500){
      int v = (int)( (1-( (1900-left_motor) /400.0)) *255);
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, HIGH);
      analogWrite(speedPinA, v);
    }else{
```

```
      int v=(int)((1-((left_motor-1100)/400.0))*255);
      digitalWrite(dir1PinA, HIGH);
      digitalWrite(dir2PinA, LOW);
      analogWrite(speedPinA, v);
    }
  }


  if(right_motor==1500){

    analogWrite(speedPinB, 0);
    //digitalWrite(r1,LOW);
    //digitalWrite(r2,LOW);
  }else{
    if(right_motor>1500){
      int v = (int)( (1-( (1900-left_motor) /400.0)) *255);
      digitalWrite(dir1PinB, LOW);
      digitalWrite(dir2PinB, HIGH);
      analogWrite(speedPinB, v);
      //digitalWrite(r1,LOW);
      //analogWrite(r2,(int)((1900-right_motor)/400.0*255));
    }else{
      int v=(int)((1-((left_motor-1100)/400.0))*255);
      digitalWrite(dir1PinB, HIGH);
      digitalWrite(dir2PinB, LOW);
      analogWrite(speedPinB, v);
      //digitalWrite(r2,LOW);
      //analogWrite(r1,(int)((1900-right_motor)/400.0*255));
    }
  }

}
void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available()>0){
      char c =Serial.read();
      if(c==':'){
        left_motor=temp.toInt();
        temp="";
        update_motors();
      }else if(c==';'){
        right_motor=temp.toInt();
        temp="";
        update_motors();
      }else{
        temp+=c;
      }
  }
}
```

5_ Sensor Transmittir Arduino:

```
/*
//Coded by Utku ARSLAN
//
//10.06.2023
//Apache-2.0 license


//This code is responsible from transmitting the Sensor data to Raspberry PI vai Usb/Serial Protocol
*/



// Define the ultrasonic sensor pins
const int TRIGGER_PIN_1 = 2;
const int ECHO_PIN_1 = 3;
const int TRIGGER_PIN_2 = 2;
const int ECHO_PIN_2 = 3;
const int TRIGGER_PIN_3 = 2;
const int ECHO_PIN_3 = 3;


// Define variables for calculating distance
long duration;
//int distance_1=0,distance_2=0,distance_3=0;


void setup() {
  Serial.begin(9600); // Initialize serial communication
  setup_sensor();
}



void setup_sensor(){
  pinMode(TRIGGER_PIN_1, OUTPUT); // Set trigger pin as output
  pinMode(ECHO_PIN_1, INPUT); // Set echo pin as input
  pinMode(TRIGGER_PIN_2, OUTPUT); // Set trigger pin as output
  pinMode(ECHO_PIN_2, INPUT); // Set echo pin as input
  pinMode(TRIGGER_PIN_3, OUTPUT); // Set trigger pin as output
  pinMode(ECHO_PIN_3, INPUT); // Set echo pin as input
}
int measureDistance(int s_id){


  int tp=0;
  int ep=0;
```

```
  switch(s_id){
    case 0://Front
      tp=TRIGGER_PIN_1;
      ep=ECHO_PIN_1;
      break;
    case 1://Left
      tp=TRIGGER_PIN_2;
      ep=ECHO_PIN_2;
      break;
    case 2://Right
      tp=TRIGGER_PIN_3;
      ep=ECHO_PIN_3;
      break;
  }

  // Send a short pulse to the trigger pin
  digitalWrite(tp, LOW);
  delayMicroseconds(2);
  digitalWrite(tp, HIGH);
  delayMicroseconds(10);
  digitalWrite(tp, LOW);


  // Measure the duration of the pulse on the echo pin
  duration = pulseIn(ep, HIGH);


  delayMicroseconds(10);
  // Calculate the distance in centimeters
  int distance = int(duration * 0.034 / 2);
  return distance;


}


void loop() {
  //String                                    s                                    =
String(measureDistance(0))+":"+String(measureDistance(1))+";"+String(measureDistance
(2))+"/";
  String s = String(10)+":"+String(10)+";"+String(10)+"/";
  Serial.println(s);
  delay(100);
}
```

APPENDIX D

```
// Motor 1
int dir1PinA = 2;
int dir2PinA = 3;
int speedPinA = 9; // PWM pini, hız kontrolü için
// Motor 2
int dir1PinB = 4;
int dir2PinB = 5;
int speedPinB = 10; // PWM pini, hız kontrolü için
void setup() { // Setup runs once per reset
Serial.begin(9600);// 9600 baud rate ile iletişimi başlat:
//L298N motor kontrolcü pinleri
pinMode(dir1PinA,OUTPUT);
pinMode(dir2PinA,OUTPUT);
pinMode(speedPinA,OUTPUT);
pinMode(dir1PinB,OUTPUT);
pinMode(dir2PinB,OUTPUT);
pinMode(speedPinB,OUTPUT);
}
void loop() {
Dur();
delay(1000);
Ileri();
delay(5000);
Dur();
```

```
delay(1000);
Geri();
delay(5000);
Dur();
delay(1000);
Sag();
delay(5000);
Dur();
delay(1000);
Sol();
delay(5000);
}
void Ileri(){
analogWrite(speedPinA, 80);//Sets speed variable via PWM
digitalWrite(dir1PinA, LOW);
digitalWrite(dir2PinA, HIGH);
analogWrite(speedPinB, 80);
digitalWrite(dir1PinB, LOW);
digitalWrite(dir2PinB, HIGH);
}
void Geri(){
analogWrite(speedPinA, 80);
digitalWrite(dir1PinA, HIGH);
digitalWrite(dir2PinA, LOW);
analogWrite(speedPinB, 80);
```

```
digitalWrite(dir1PinB, HIGH);
digitalWrite(dir2PinB, LOW);
}
void Sag(){
analogWrite(speedPinA, 80);//Sets speed variable via PWM
digitalWrite(dir1PinA, LOW);
digitalWrite(dir2PinA, HIGH);
analogWrite(speedPinB, 80);
digitalWrite(dir1PinB, HIGH);
digitalWrite(dir2PinB, LOW);
}
void Sol(){
analogWrite(speedPinA, 80);
digitalWrite(dir1PinA, HIGH);
digitalWrite(dir2PinA, LOW);
analogWrite(speedPinB, 80);
digitalWrite(dir1PinB, LOW);
digitalWrite(dir2PinB, HIGH);
}
void Dur(){
analogWrite(speedPinA, 0);
analogWrite(speedPinB, 0);
}
```

REFERENCES

1. Das, J., Anosike, K., & Asuncion, R. (2022). Locked-in Syndrome. Statpearls Publishing. Retrieved from https://www.ncbi.nlm.nih.gov/books/NBK559026/
2. Spinal Cord Injury Facts & Statistics. (2019). Retrieved 12 December 2022, from https://www.sci-info-pages.com/spinal-cord-injury-facts-and-statistics/
3. Madanhire, I., Gudukeya, L., & Mushonga, R. (2020). Development of a Rough Terrain Wheelchair Design. Public Health In Developing Countries - Challenges And Opportunities. doi: 10.5772/intechopen.91267
4. (2022)Accessdata.fda.gov.Availableat:https://www.accessdata.fda.gov/cdrh_docs/reviews/DEN200046.pdf (Accessed: 20 December 2022).
5. Are EMOTIV products medical devices? (2022). Available at: https://www.emotiv.com/knowledge-base/are-emotiv-products-medical-devices/ (Accessed: 20 December 2022).
6. Regulatory Compliance - EPOC Flex User Manual. (2022). Retrieved 20 December 2022, from https://emotiv.gitbook.io/epoc-flex-user-manual/epoc-flex/regulatory-compliance
7. Palumbo, A., Gramigna, V., Calabrese, B., & Ielpo, N. (2021). Motor-Imagery EEG-Based BCIs in Wheelchair Movement and Control: A Systematic Literature Review. Sensors, 21(18), 6285. doi: 10.3390/s21186285
8. F. Achic, J. Montero, C. Penaloza and F. Cuellar, "Hybrid BCI system to operate an electric wheelchair and a robotic arm for navigation and manipulation tasks," 2016 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), 2016, pp. 249-254, doi: 10.1109/ARSO.2016.7736290.
9. D. W. -K. Ng, Y. -W. Soh and S. -Y. Goh, "Development of an Autonomous BCI Wheelchair," 2014 IEEE Symposium on Computational Intelligence in Brain Computer Interfaces (CIBCI), 2014, pp. 1-4, doi: 10.1109/CIBCI.2014.7007784.
10. X. Deng, Z. L. Yu, C. Lin, Z. Gu and Y. Li, "A Bayesian Shared Control Approach for Wheelchair Robot With Brain Machine Interface," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 28, no. 1, pp. 328-338, Jan. 2020, doi: 10.1109/TNSRE.2019.2958076.
11. A. C. Lopes, G. Pires, L. Vaz and U. Nunes, "Wheelchair navigation assisted by Human-Machine shared-control and a P300-based Brain-Computer Interface," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 2438-2444, doi: 10.1109/IROS.2011.6094748.
12. J. Philips et al., "Adaptive Shared Control of a Brain-Actuated Simulated Wheelchair," 2007 IEEE 10th International Conference on Rehabilitation Robotics, 2007, pp. 408-414, doi: 10.1109/ICORR.2007.4428457.
13. Bobrov, P., Frolov, A., Cantor, C., Fedulova, I., Bakhnyan, M., & Zhavoronkov, A. (2011). Brain-Computer Interface Based on Generation of Visual Images. Plos ONE, 6(6), e20674. doi: 10.1371/journal.pone.0020674
14. Batres-Mendoza, Patricia & Ibarra-Manzano, Mario & Guerra-Hernandez, Erick & Almanza-Ojeda, Dora & Montoro-Sanjose, Carlos & Romero-Troncoso, René & Rostro-Gonzalez, Horacio. (2017). Improving EEG-Based Motor Imagery Classification for Real-Time Applications Using the QSA Method. Computational Intelligence and Neuroscience. 2017. 10.1155/2017/9817305.
15. Wolpaw JR, Birbaumer N, McFarland DJ, Pfurtscheller G, Vaughan TM. Brain-computer interfaces for communication and control. Clin Neurophysiol. 2002 Jun;113(6):767-91. doi: 10.1016/s1388-2457(02)00057-3. PMID: 12048038.
16. Tong Y, Pendy JT Jr, Li WA, Du H, Zhang T, Geng X, Ding Y. Motor Imagery-Based Rehabilitation: Potential Neural Correlates and Clinical Application for Functional Recovery of Motor Deficits after Stroke. Aging Dis. 2017 May 2;8(3):364-371. doi: 10.14336/AD.2016.1012. PMID: 28580191; PMCID: PMC5440115.
17. Technical Specifications - EPOC+. (2022). Retrieved 27 December 2022, from https://emotiv.gitbook.io/epoc-user-manual/introduction-1/technical_specifications
18. A. Vallabhaneni, T. Wang, and B. He, “Brain—Computer Interface,” in Neural Engineering, Springer US, 2007, pp. 85–121.
19. A. Delorme and S. Makeig, “EEGLAB: An open source toolbox for analysis of singletrial EEG dynamics

including independent component analysis," J. Neurosci. Methods, vol. 134, no. 1, pp. 9–21, 2004.
20. Decety, J., & Jeannerod, M. (1996). The cognitive neuroscience of motor intention and imagery. Behavioral and Brain Sciences, 19(2), 269-294.
21. Mokienko, Olesya & Lyudmila, Chernikova & Frolov, Alexander & Bobrov, Pavel. (2014). Motor Imagery and Its Practical Application. Neuroscience and Behavioral Physiology. 44. 483-489. 10.1007/s11055-014-9937-y.
22. Schalk, Gerwin & Mellinger, Jürgen. (2010). A Practical Guide to Brain-Computer Interfacing with BCI2000. 10.1007/978-1-84996-092-2.
23. Wang, Yijun & Gao, Xiaorong & Hong, Bo & Gao, Shangkai. (2010). Practical Designs of Brain–Computer Interfaces Based on the Modulation of EEG Rhythms. 10.1007/978-3-642-02091-9_8.
24. Hoffmann, Ulrich & Fimbel, Eric & Keller, Thierry. (2009). Brain-computer interface based on high frequency steady-state visual evoked potentials: A feasibility study. 466 - 469. 10.1109/NER.2009.5109334.
25. Javatpoint. (n.d.). Regression vs classification in Machine Learning . www.javatpoint.com. Retrieved January 15, 2023, from https://www.javatpoint.com/regression-vs-classification-in-machine-learning
26. Ph. D., M. & Das, Sujoy & Kabir, Md & Lima, Aklima & Watanobe, Yutaka. (2021). Brain-Computer Interface: Advancement and Challenges. Sensors (Basel, Switzerland). 21. 10.3390/s21175746.
27. AlZoubi, Omar & Koprinska, Irena & Calvo, Rafael. (2008). Classification of Brain-Computer Interface Data. 7th Australasian Data Mining Conference. 87. 123-131.
28. Arslan, U. (2022). USV Robot – competation. Retrieved January 15, 2023, from https://utkuarslan.net/2022/08/20/usv-robot-competation/
29. Witten, I. H and Frank, E. (2005): Data mining: practical machine learning tools and techniques. Second edition, San Francisco, Morgan Kaufmann
30. "Brushless DC Motors for Electric Wheelchair Applications." (https://www.maxonmotorusa.com/applications/electric-wheelchairs/)
31. "Power Wheelchair Controls: An Overview." (https://www.rehabmart.com/article/power-wheelchair-controls-overview)
32. https://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/1168/What-are-the-four-fundamental-methods-of-requirement-verification.aspx
33. Choi, K., & Cichocki, A. (2008). Control of a Wheelchair by Motor Imagery in Real Time. Lecture Notes In Computer Science, 330-337. doi: 10.1007/978-3-540-88906-9_42
34. Ramoser, H., Müller-Gerking, J., & Pfurtscheller, G. (2000). Optimal spatial filtering of single trial EEG during imagined hand movement. IEEE transactions on rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society, 8(4), 441–446. https://doi.org/10.1109/86.895946
35. C. Brunner, R. Leeb, G. Muller-Putz, A. Schl ¨ ogl, and G. Pfurtscheller,¨ "Bci competition 2008–graz data set a," Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology, p. 16, 2008.
36. Movement imagery classification in EMOTIV cap based system by Naïve Bayes. (2023). Retrieved 19 April 2023, from https://ieeexplore.ieee.org/document/7591711
37. labstreaminglayer. (2023). GitHub - labstreaminglayer/liblsl-Matlab: Matlab bindings for liblsl. Retrieved 20 April 2023, from https://github.com/labstreaminglayer/liblsl-Matlab
38. FranzHell. (2023). EEG-data-analysis-Course/topoplotIndie.m at master · FranzHell/EEG-data-analysis-Course. Retrieved 20 April 2023, from https://github.com/FranzHell/EEG-data-analysis-Course/blob/master/topoplotIndie.m
39. (2023).Retrieved27April2023,fromhttps://www.youtube.com/watch?v=metlFBa_NdQ&t=1070s&ab_channel=NeuralOscillations
40. (2023).Retrieved27April2023,fromhttps://ourspace.uregina.ca/bitstream/handle/10294/7875/Ali_Syed_Salman_200351184_MASC_ESE_Fall2017.pdf
41. (2023).Retrieved28April2023,fromhttps://www.youtube.com/watch?v=lCfYbwT0AA4&list=PLXc9qfVbMMN2uDadxZ_OEsHjzcRtlLNxc&index=3&ab_channel=EEGLAB
42. (2023). Retrieved 29 April 2023, from https://www.youtube.com/watch?v=YK1F0-3VvQI&t=503s&ab_channel=MikeXCohen
43. Batres-Mendoza, P., Montoro-Sanjose, C., Guerra-Hernandez, E., Almanza-Ojeda, D., Rostro-

Gonzalez, H., Romero-Troncoso, R., & Ibarra-Manzano, M. (2016). Quaternion-Based Signal Analysis for Motor Imagery Classification from Electroencephalographic Signals. Sensors, 16(3), 336. doi: 10.3390/s16030336

44. Tangermann, M., Müller, K., Aertsen, A., Birbaumer, N., Braun, C., Brunner, C., . . . Blankertz, B. (2012, March 30). Review of the BCI competition IV. Retrieved May 5, 2023, from https://www.frontiersin.org/articles/10.3389/fnins.2012.00055/full
45. R. Scherer, G. R. Muller, C. Neuper, B. Graimann and G. Pfurtscheller, "An asynchronously controlled EEG-based virtual keyboard: improvement of the spelling rate," in IEEE Transactions on Biomedical Engineering, vol. 51, no. 6, pp. 979-984, June 2004, doi: 10.1109/TBME.2004.827062.