**A** **TECHNICAL REPORT**

ON

**STUDENTS INDUSTRIAL WORK EXPERIENCE SCHEME (SIWES II)**

**EEE400**

UNDERTAKEN AT

**APPLIED ARTIFICIAL INTELLIGENCE & ROBOTICS RESEARCH LAB**

OBAFEMI AWOLOWO UNIVERSITY, ILE-IFE, OSUN STATE.

****

BY

**OLATEJU, EMMANUEL OLUWASEGUN**

**MATRIC NO: EEG/2016/054**

SUBMITTED TO

**THE SIWES COORDINATOR**

**DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING**

**OBAFEMI AWOLOWO UNIVERSITY, ILE-IFE, OSUN STATE**

**IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF**

**BACHELOR OF SCIENCE (B.Sc.) DEGREE IN**

**ELECTRONIC AND ELECTRICAL ENGINEERING**

**DECEMBER 2021**

Department of Electronic and Electrical

Engineering,

Obafemi Awolowo University,

Ile-Ife, Osun State.

1st August, 2021.

The SIWES Coordinator,

Department of Electronic and Electrical Engineering,

Obafemi Awolowo University,

Ile-Ife, Osun Sate.

Dear Sir,

**LETTER OF TRANSMITTAL**

In partial fulfillment for the requirement of the award of B.Sc. (Hons.) in Electronic and Electrical Engineering, I, OLATEJU Emmanuel Oluwasegun hereby submit for grading, the report of the Students’ Industrial Work Experience Scheme (SIWES) – EEE400 which was undertaken at Applied Artificial Intelligence and Robotics Laboratory, Obafemi Awolowo University, Ile-Ife, Osun State.

Yours faithfully

……………….………………………

**OLATEJU Emmanuel Oluwasegun**

**EEG/2016/054**

**CERTIFICATION**

This is to certify that, OLATEJU Emmanuel Oluwasegun, with matric number EEG/2016/054, carried out the Students’ industrial Work Experience Scheme (SIWES) programme under my supervision and submit this report for grading and that the contents of the report were written and compiled by him with respect to the skills acquired during the course of the programme.

……………………… …………………….

Dr. F.K. Ariyo Date

SIWES Coordinator

**DEDICATION**

I dedicate the period of the work experience scheme to God almighty my parents, my industry based supervisor and colleagues.

Most importantly, I dedicate the works done during the work experience scheme to my industry based supervisor, the person of Dr. K.P Ayodele and other colleagues who provided career and moral mentor-ship to me during and after the period of the work experience scheme.

**ACKNOLEDEMENT**

I give thanks to God Almighty, who thought of me as worthy of holding on to his breath of life.

I sincerely appreciate my mother, MRS. F.O Olateju who despite all odds gave me the rare chance of being educated and who despite needing my presence during the period of the work experience scheme understood my desires and needs, thus allowing me to be physically distant from her for the period of the work experience scheme.

My sincere appreciation also goes to my industry based supervisor, Dr. K.P. Ayodele for his consistent mentoring and guidance, not just in technical areas and career-wise, but also in areas of moral-development and handling of personal matters.

With every drop of gratitude in my heart, I appreciate Mr. Segun Akinniyi and all other staffs of Applied Artificial Intelligence and Robotics Laboratory for their continued support and continually trusting in my abilities enough to leave certain issues and projects into my little imperfect and not yet capable hands.

**TABLE OF CONTENTS**

CERTIFICATION ii

DEDICATION iii

ACKNOWLEDGEMENT iv

TABLE OF CONTENTS v

LIST OF ABBREVIATIONS viii

LIST OF FIGURES x

LIST OF PLATES xi

ABSTRACT xii

EXECUTIVE SUMMARY xiii

**CHAPTER ONE: INTRODUCTION**

* 1. Historical Background of Students Industrial Work Experience Scheme 1
  2. Aims and Objectives Of SIWES 1
  3. About A2IR2 1
  4. A2IR2 Mission 1
  5. A2IR2 Vision 1
  6. Organizational Structure of A2IR2 1

**CHAPTER TWO: LITERATURE REVIEW**

* 1. Brain Computer Interfacing (BCI) 1
     1. Areas of Application of Brain Computer Interfacing 4
     2. Challenges of Brain Computer Interfacing 4
     3. Ethical and Legal Issues in Brain Computer Interfacing 4
  2. Platform for Upper-Limb Stroke Rehabilitation 5
     1. Stroke Neuro-Rehabiltation 5
     2. Stroke Robotic-Rehabilitation 5
     3. Robotic Devices-Machine for Upper Limb Rehabilitation 5
  3. Electronic Speed Controller 6
     1. Brushed Motor 6
     2. Advantages of Brushed Direct-Current Motor 6
     3. Disadvantages of Brushed Direct-Current Motor 6
     4. Brush-less Direct-Current Motor 6
     5. Brush-less Direct-Current Motor Control 6
        1. Positional Feedback 6
     6. Advantages of Brush-less Direct-Current Motor 6
     7. Disadvantages of Brush-less Direct-Current Motor 6

**CHAPTER THREE: PLATFORM FOR UPPER-LIMB STROKE REHABILITATION (PULSR)**

* 1. Introduction 1
  2. Block Diagram of System 4
  3. Statement of Previously Done Work 4
  4. Control-Board Re-Design 4
  5. Software Design 4
     1. Block Diagram 4
     2. User Interface Design 4
     3. Hardware Communication 4
  6. System Problem Identification, Proposed and Adopted Solutions 4
     1. Problem Specification 4
     2. Troubleshooting Approach 4
     3. Proposed Solutions 4
     4. Adopted Solution 4
        1. Hall Sensor Emulation 4
        2. Developing an electronic speed controller 4

**CHAPTER FOUR: ELECTRONIC SPEED CONTROLLER**

* 1. Introduction 1
  2. Arduino Code and Testing 4
     1. State Transition Diagram of Arduino Code 4
     2. Testing and Results 4
  3. MOSFET Gate Drivers 4
     1. Working Principle 4
     2. Testing and Results 4
  4. Half-Bridges 4
     1. Working principle 4
     2. Testing and Results 4
  5. Final Circuit Testing and Results 4

**CHAPTER FIVE: BCI TASK DISCRIMINABILITY ANALYSIS**

* 1. Introduction 1
  2. Pre-processing Steps 4
     1. Stream to RAW Format Conversion 4
     2. Marker Stream to Events Conversion 4
     3. Band-pass Filtering 4
     4. Dropping of Empty Channels 4
  3. Feature Extraction 4
     1. Band-pass Filtering 4
     2. Laplace Averaging 4
  4. Analysis for Class Discriminability 4
     1. Epoching 4
     2. Power Spectral Density Analysis 4
     3. Percental Change Analysis 4

**CHAPTER SIX: BCI MOVEMENT AGAINST REST CLASSIFIER**

* 1. Introduction 1
  2. Data Acquisition and Protocols 4
  3. Signal Pre-processing 4
     1. Data-Set Merging 4
     2. Merged Data-Set Filtering 4
     3. Epoching Merged Data 4
     4. Standardization of Epochs 4
  4. Feature Extraction 4
     1. Common Spatial Patterns (CSP) 4
     2. Results of Common Spatial Patterns 4
  5. Classification 4
     1. Logistic Regression 4
     2. Results of Logistic Regression 4

**CHAPTER SEVEN: SUMMARY, CONCLUSION AND RECOMMENDATION**

* 1. Summary of Work Done 1
  2. Relevance of Experience to Course of Study 4
  3. Conclusion 4
  4. Recommendations 4

**APPENDIX** 4

A PULSR GUI/software code 4

B ESC firmware code 4

C Hall sensor emulator firmware code 4

**REFERENCES** 50

**LIST OF ABBREVIATIONS**

**BCI:** Brain Computer Interfacing

**BLDC:** Brush-less Direct Current

**BOM:** Bill of Materials

**CSP:** Common Spatial Patterns

**EEG:** Electroencephalography

**EPs:** Evoked potentials

**ERD:** Event related de-synchronization

**ERP:** Event related potentials

**ERS:** Event related synchronization

**ESC:** Electronic Speed Controller

**FBCSP:** Filter banks common spatial patterns

**FFT:** Fast Fourier Transforms

**fMRI:** Functional Magnetic Resonance Imaging

**fNIR:** Functional Near Infrared Imaging

**FES:** Functional Electrical Stimulation

**LSL:** Lab Streaming Layer

**MOSFET:** Metallic Oxide Semiconductor Field Effect Transistor

**PET:** Positron Emission Tomography

**PULSR:** Platform for Upper Limb Stroke Rehabilitation

**PWM:** Pulse Width Modulation

**MEG:** Magnetoencephalography

**SMR:** Sensory-motor Rhythm

**USART:** Universal Synchronous Asynchronous Receiver and Transmitter

**LIST OF FIGURES**

* Table 2.1 Brain-Lobes and associated body function
* Table 3.1: PULSR damaged parts and resultant behavior of system
* Table 3.2: Results Of Scoping And Resistance Tests On PULSR BLDC Hall Sensors
* Table 4.1 IR2110 Inputs and Output Source

**LIST OF PLATES**

* Figure 2.1: Components of a Brain Computer Interfacing system
* Figure 2.2: The 10-20 electrode placement system
* Figure 2.3: Inner Structure Of A Brushed DC Motor
* Figure 2.4: A Brush-less DC Motor Model
* Figure 2.5: Inner Structure of a 3-Phase BLDC Motor
* Figure 3.1: Block Diagram of PULSR Components
* Figure 3.2:
* Figure 3.3:
* Figure 3.4: BLDC ESC Controller Schematic
* Figure 3.5: PULSRs Previous Control Board
* Figure 3.6: PULSRs Current Control Board
* Figure 3.7: Block Diagram Showing Components of PULSR software
* Figure 3.8: Agreed Information Transfer Discipline between PULSR PC and firmware
* Figure 3.9: Block Diagram of Hall Sensor Emulator
* Figure 4.1: ESC Block Diagram
* Figure 4.2: ESC state transition diagram
* Figure 4.3: MOSFET gate driver and half-bridge circuit
* Figure 4.4: Proteus Circuit Model for ESC
* Figure 5.1: EEG Data Before Filtering
* Figure 5.2: EEG Data After Filtering and Dropping of Bad Channels
* Figure 5.3: Plot of Computed Laplace Average
* Figure 5.4: Plot of Extracted Close Epochs
* Figure 5.5: Power Spectral Density Plot for relax and close Epochs
* Figure 5.6: Percental Analysis Plot Over Averaged Class Epochs
* Figure 6.1: Lab Recorder Merging Streams for EEG Data Recording
* Figure 6.2: Digital Filters Frequency Response

**ABSTRACT**

Brain Computer Interfacing(BCI) is a developing and inter-disciplinary field that deals with the extraction of intelligence of human-activity from the human brain as signals, using either invasive or non-invasive methods and the interpretation of these signals to their equivalent associated human-activity using intelligent processing methods from computer-science, electronic-engineering and mathematics.

PULSR which stands for Platform for Upper-Limbs Stroke Rehabilitation is a two degrees of freedom robot designed to employ controlled physical exercise with feedback from brain-computer-interfacing(BCI) and machine-learning systems in the rehabilitation process of early stage stroke patients having their upper-limbs inactive due to the stroke condition.

A brush-less direct current (BLDC) motor is a three-phase coil (star configuration) motor making use of the three-phase coiled electromagnets as its stator, a set of permanent magnets as its rotor part and hall sensors for generation of rotor position and next control-signal information, unlike the traditional servo motor which requires no information on rotor position and does not have transient states (changing control-signal). The special configuration of a BLDC motor thus makes it require the use of a smart controller generally called an Electronic Speed Controller (ESC).

**CHAPTER ONE**

**INTRODUCTION**

* 1. **HISTORICAL BACKGROUND OF STUDENT INDUSTRIAL WORK EXPERIENCE SCHEME**

The Student Industrial Work Experience Scheme (SIWES) was established by Industrial Training Fund (ITF) in 1973 by decree No. 47 of 1973 due to the identified need in graduates of tertiary institutions in Nigeria. SIWES was introduced as a planned and structured program based on stated and specific career objectives, geared towards developing the occupational competences of participant. SIWES commenced in 1974 with Seven hundred and forty-eight (748) students from eleven institutions. In 1978, Federal Ministry of Education made it compulsory for all students of Polytechnic, College of Technology to undergo one (1) year Industrial Training program. The ITF solely funded the scheme during its formative years, but as the financial involvement became unbearable to the fund, it withdrew from the scheme in1978. The Federal Government handed over the scheme in 1979 to both the National Universities Council (NUC) and National Board for Technical Education. They were managing the activities of SIWES until 1984. During this period of their management, NBTE renamed SIWES to Compulsory Supervised Industrial Training Attachment (COSITA). In November, 1984 Federal Government reverted the management and implementation of SIWES to ITF and it was effectively taken by the Industrial Training Fund in 1985 in collaboration with the supervising agencies (NUC, NBTE). In 1985, Federal Government legally backed the scheme by decree No. 16 of 1985 which says ―all students enrolled in specialized engineering, technical, business, applied arts should have supervised industrial attachment as part of their studies. As at 2017, the number of institutions has increased to 311 to 360,341students.

* 1. **AIMS AND OBJECTIVES OF SIWES**

The Industrial Training Funds Policy Document No. 1 of 1973 which established SIWES

1. Provide an avenue for students in higher institutions of learning to acquire industrial skills and experience during their course of study.
2. Prepare students for industrial work situations that they are likely to meet after graduation.
3. Expose students to work methods and techniques in handling equipment and machinery that may not be available in their institutions.
4. Make the transition from school to the world of work easier and enhance students’ contacts for later job placement.
5. Provide students with the opportunities to apply their educational knowledge in real work situations, thereby bridging the gap between theory and practice.
6. Enlist and strengthen employers’ involvement in the entire educational process and prepare students for employment in industry and commerce.
   1. **ABOUT A2IR2**

Applied Artificial Intelligence and Robotics Research Lab (A2IR2) is a research laboratory located within Obafemi Awolowo University, Ile-Ife. The research lab is an affiliate of the department of Electronic and Electrical Engineering and the College of Health Science of the previously mentioned university.

The research lab was established in the year \_\_\_\_\_\_ with \_\_\_\_\_\_ founding members. Currently the research lab has high ranking industry and academic engineers as its members some of which are within the shores and some off-shore.

The research lab focuses on development of human-task easing solutions, medical-rehabilitation robots, technology driven agriculture solutions, and development of other cutting edge solutions that apply to human activities of daily living. The research lab employs knowledge from all sectors of sciences and engineering particularly the electronic engineering, computer-science and mathematics. Not limited to knowledge from the engineering sector, the research lab employs tools from the medical sector, agricultural sector and from other areas as long as a problem is being solved by providing a solution.

The research lab since its establishment has participated in \_\_\_\_\_ academic and industrial events and \_\_\_\_\_ exhibitions. The research lab has published \_\_\_\_\_ papers and has received \_\_\_\_\_ grants and awards.

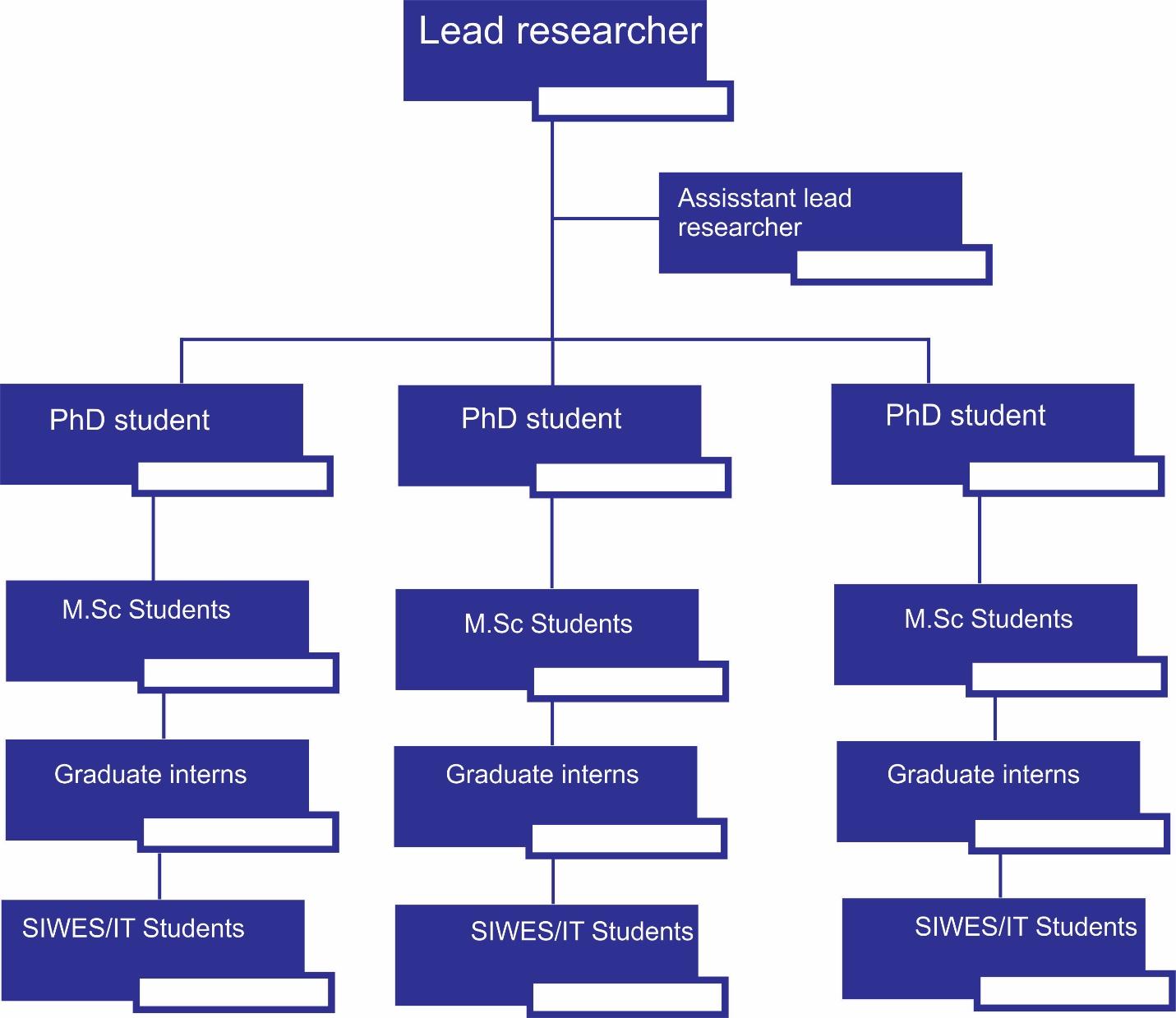
* 1. **A2IR2 MISSION**

The mission of the Applied Artificial Intelligence and Robotics Lab is to serve as a vehicle to fuse that collective Diaspora expertise and burgeoning local capacity to raise the quality and audacity of AI and robotics research and development nationally, towards long-term goals of

1. Making original and valuable contributions to the body of knowledge
2. Providing innovative, efficient and effective solutions to national and international problems.
   1. **A2IR2 VISION**

The vision for this research lab was borne out of the desire to formally bring together, graduate students at OAU Ile-Ife and Nigerian academics diaspora in the era of artificial intelligence and robotics.

* 1. **ORGANIZATIONAL STRUCTURE OF A2IR2**



**CHAPTER TWO**

**LITERATURE REVIEW**

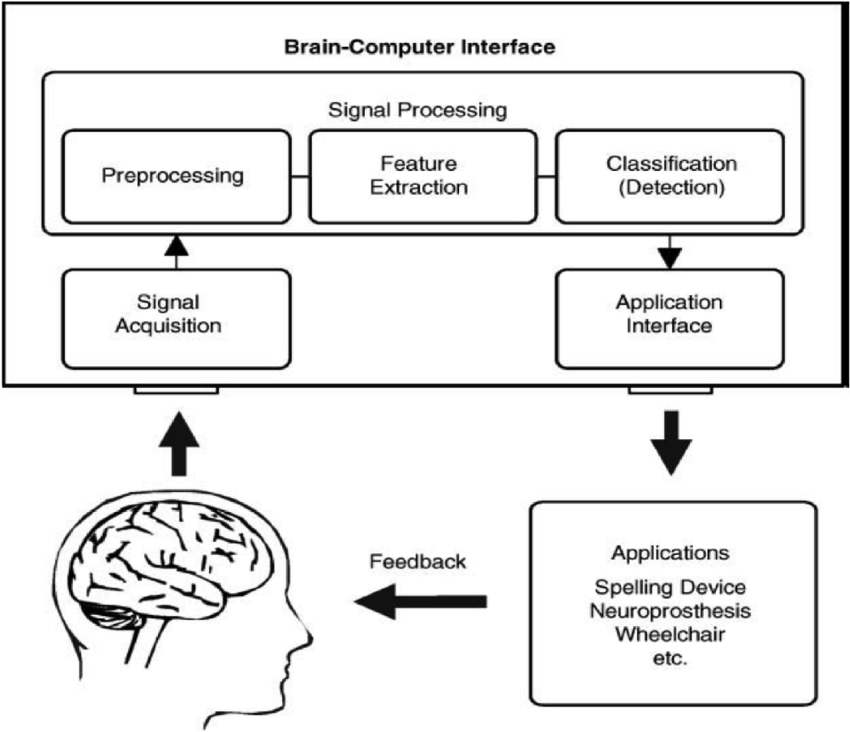
* 1. **BRAIN COMPUTER INTERFACING (BCI)**

Brain Computer Interfacing is a developing field of study that inquires ways to map various forms of signals of the brain, and optionally signals from other parts of the body (eye, muscle, etc.) to intelligent activity of the human body, as all activities and experiences of the human body are either pre-processed or post-processed by the brain for generation of interpretation or instruction for another set of activity.

The ability of BCI to map signals of the brain along with signals from other parts of the body to intelligent activity makes it a very important area of study in so many industries ranging from the medical-rehabilitation industry to the gaming industry, ggovernment intelligence and investigation agencies make use of BCI systems among many other industries. BCI systems can be used in lie detection, anomaly behaviour monitoring, control of prosthetic hands or legs, interpretation and then conversion of intended human-movement or activity to game control signals and so much more.

BCI applications can be divided into three classes based on the preferred state of user in development and usage stages. There is the passive BCI, active BCI and reactive BCI. In the passive BCI, the idle state or the state where the user is engaged in some cognitive task without intentions of controlling the BCI system is of interest. In active BCI, the activity of interest has to be properly specified by some predefined cue signals. The users’ goal is to control the BCI system by modulating his or her brain signals. The activity of interest which is some predefined cognitive task can be movement of specific parts of the body or being engaged in a mentally tasking exercise such as mathematical computations or artistic development which is to be dictated by the current cue signal. In a reactive BCI, the activity of interest is the brains response to external stimulus, the stimulus being visual, auditory, and sensual or a combination of any or all.

A typical BCI system consists of four stages (components) with an optional user or supervisor feedback system as shown in Figure1.



***Figure2.1 Components of a Brain Computer Interfacing system***

1. The signal-acquisition block:

The signals from the brain can be acquired by invasive or non-invasive methods. The non-invasive methods are generally preferred over the invasive methods as they require less protocols and supervision to be used, though the invasive methods give signals with better time and spatial resolution. There are majorly five non-invasive technologies used in acquiring the brain signals used in BCI systems.

1. Electroencephalography(EEG)
2. Functional Magnetic Resonance Imaging(fMRI)
3. Functional Near Infrared Imaging(fNIR)
4. Magnetoencephalography(MEG)
5. Positron Emission Tomography(PET)

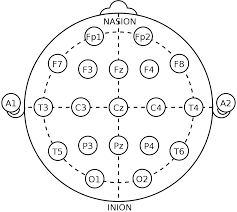
The EEG method is usually preferred over others due to its low-cost compared to other technologies and its being available and accessible as needed. EEG records electrical activity on the cortex of the brain (brain surface) by placing a group of electrodes, either wet (requiring electrode-gel and extra preparatory methods) or dry electrodes on the scalp. The arrangement of neurons in the cerebral cortex and its proximity to the skull favor recording by EEG by reading from the lateral electric fields of the cortex detected on the scalp by the EEG electrodes. It is common knowledge in BCI, (Rajesh, 2013) that the spatial resolution of EEG is typically poor (in the square centimeter range) but the temporal resolution is good (in the milliseconds range). The recorded signals are called EEG signals and are multi-space time-series of voltages in the micro-volts range. During the recording of EEG data in a BCI development exercise, labels of user activity are attached to time windows of user (subject) activity and are commonly called markers or event-markers. Thus, the EEG data is stored as 3D matrix of space, time and event-marker.

In the placement of the EEG sensors, a defined scalp mapping system called the 10-20 electrode placement system has to be followed precisely. Also, impedance between the scalp and the sensor has to be measured to be within the range defined by EEG device manufacturer for acquisition of meaningful data. The 10-20 electrode placement system makes use of the division of parts of the brain according to function into lobes and numbers electrode sites on the left-hemisphere of the scalp with odd numbers and electrode sites on the right hemisphere of the scalp with even numbers. The lobe division of the brain, their associated label and commonly indicative function on the 10-20 electrode placement system as shown in Figure2.2 is given in Table2.1

**Table2.1 Brain-Lobes and associated body function**

|  |  |  |
| --- | --- | --- |
| **Brain Lobe** | **10-20 placement representation** | **Commonly Associated Body Function** |
| Frontal | Fp |  |
| Parietal | P |  |
| Central | C |  |
| Temporal | T |  |
| Occipital | O |  |

The EEG recording is a summation of spontaneous-EEG which encodes human-activity information and background-EEG which is majorly noise also called artifacts and contains little or no information on neural-state or human-activity. The artifacts can be generated by movement of certain body parts or can be from external electrical devices, for this reason BCI users are usually advised not to move during a BCI development exercise and well detailed and instructive protocols are employed during BCI experiments. These artifacts need to be removed using algorithms or filters depending on the application so as to effectively decode EEG data.



***Figure2.2 The 10-20 EEG electrode placement system***

1. The Signal Preprocessing block:

Signal preprocessing requirement varies depending on the technology used in acquiring the brain signals. In cases where the brain signals are acquired using EEG technology, (Rajesh, 2013) states that the weak amplitude of the EEG signal makes it highly susceptible to noise contamination called artifacts from other parts of the body and from nearby electrical devices.

The presence of EEG artifact from external electrical devices can be reduced by making use of a 50-60Hz notch-filter for removal of power-line interference due to instrumentation. Also a 1Hz to 30Hz band-pass filter is traditionally used with EEG signals as EEG information carrying signals are encoded within this range of frequency. The band-pass filtering helps separate information bearing EEG signals from background EEG signals, bearing artifacts and other forms of unwanted noise.

Artifacts due to changes in body parts are quite difficult to remove as they are not easily distinguishable from meaningful EEG data. Artifacts due to eye-movements are called ocular-artifacts and those due to movements of other body parts are called muscular-artifacts. There exists powerful algorithms for the removal of body artifacts from EEG data, but depending on application, these bodily generated artifacts can be considered useful information. In cases where we wish to also detect movement of the eye, ocular-artifacts in EEG data due to eye-blinking are no longer considered as artifacts. Thus signal-preprocessing approach varies according to BCI application and signal-acquisition technology employed.

After successful separation of meaningful EEG data from artifacts, the EEG signals may need to be scaled depending on the requirements of successive blocks in the BCI system and also mapped to human/neural-activity (classification). The method of scaling adopted also depends on the requirement of successive blocks in the BCI system. The min-max scaling method or the centralization method can be adopted as required for successive blocks.

1. The Features Extraction block:

In order to map EEG signal to neural-activity, variables indicating such neural-activity need be computed from the pre-processed EEG signals around the time window of interest. This is done by extracting the time-window of interest around the time-frame indicated by the event-marker in the EEG signal, a process commonly called epoching or segmentation. The resulting EEG signals are called epochs and are the multi-space time-series from which variables indicative of neural activity are computed.

The variables computed can be in the time-domain, frequency-domain or a combination of both. Since the epochs are multi-space time-series data, there exists time-domain analysis methods, frequency-domain analysis methods and also spatial filtering methods for computing variables that give information on neural activity also called features.

Depending on the activity of interest, associated cortex area of activity and its commonly associated brain signal elicitation, the feature extraction method is chosen. These common brain signal elicitations associated with certain neural-state activity are generally referred to as event-related-potentials (ERPs) or evoked-potentials (EPs). A common class of event-related-potentials are event-related-de-synchronizations (ERDs), event-related-synchronizations (ERSs) and sensory-motor-rhythms (SMRs) which are commonly associated with cognitive functions involving motion and imagination of motion called motor-imagery and are largely observed in the frequency domain. There is also the P300 and P100 signals which are positive deflections in the EEG signals that occur around 300ms and 100ms respectively after the occurrence of some target stimuli which can be single or a combination of visual, auditory or tactile stimuli. A common class of evoked-potentials is the steady state evoked potential(SSVEP) which occurs in response to steadily changing visual patterns as electrical activity in EEG signals in the occipital lobe of frequency similar to the frequency of the visual pattern..

Fourier transforms and wavelet transforms are frequency analysis methods commonly used for extracting frequency-domain features such as those indicative of ERDs and ERSs in motor-imagery experiments. The wavelet transform is preferred due to its trade-off balance between time-domain resolution and frequency-domain resolution. Hjorth parameters and auto-regressive modelling are time-analysis methods among others used for time-domain features extraction. There exists spatial filters for feature extraction among which common spatial patterns (CSP) and the filter banks common spatial patterns (FBCSP) are more prominent as they produce better classification results on EEG data.

1. Classification:

After extraction of features from the EEG data, a function that maps the feature vector to neural-activity is required. This function can be called a prediction function or as commonly called a (machine-learning) classifier. This function is developed by a common machine-learning technique called supervised-learning in which, the features are passed to the machine-learning system along with the event-markers as labels.

* + 1. **Areas of Application of Brain Computer Interfacing**

A BCI system is also a communication system between the brain and external devices, making it possible for brain signals to be translated into control commands for external devices. This feature of BCI systems makes it useful in replacement of physically disabled body-parts, by controlling prosthetic parts using the brain alone. The communication and control capabilities of BCI makes it applicable in various areas and in recent times BCI systems have been validated for use in noisy structured environments such as homes and hospitals resulting in the BCI systems gaining popularity among regular people. In the last years, some research efforts have been done on its use about smart environments, smart control systems, fast and smooth movement of robotic arm prototypes, motion planning of autonomous or semi-autonomous wheelchairs, as well as controlling orthosis and prostheses. A number of research endeavors confirmed that different devices such as a wheelchair or robot arm can already be controlled by a BCI device.

1. Applications to unmanned aerial vehicle:

(LaFleur *et al,* 2013) gained extensive media attention for demonstrating the potential of noninvasive EEG-based BCI systems in 3D control of a quad-copter. Five subjects were trained to modulate their sensory-motor rhythms to control an AR drone navigating a physical space using an obstacle course. Visual feedback was provided through a first person view camera attached to the drone. The subjects were able to quickly pursue a series of foam ring targets by passing through them in a real-world environment. They obtained up to 90.5% of all valid targets through the course, and the movement was performed in an accurate and continuous way.

1. Applications to robotic arms and electrical prostheses:

Control of the movement of a cursor in a three dimensional scene is the most significant pattern in BCI based control studies. EEG changes, normally associated with left-hand, right-hand, or foot movement imagery can be used to control cursor movement.

(Xu *et al,* 2011) used a robot-assisted upper-limb rehabilitation system, in translating patient’s intention to direct control of the rehabilitation robot. Features were extracted from the acquired signal through wavelet transform and a linear discriminant analysis (LDA) classifier was used to classify the pattern of left and right-upper-limb motor imagery. A personal computer triggers the upper-limb rehabilitation robot movements.

1. Applications to wheelchair control and autonomous vehicles

Traditional wheelchair control is done using joystick, but in certain cases of paralysis, brain control is a better option.

(Tanaka *et al,* 2005) used a discrete approach for the navigation problem, in which the environment is discretized and composed by two regions (rectangles of 1m2 , one on the left and the other on the right of the start position), and the user decides where to move next by imagining left or right limb movements.

(Rebsamen *et al,* 2010) used a P300-based (slow-type) BCI to select the destination in a list of predefined locations.

(Iturrate *et al,* 2009) used a brain-actuated wheelchair based on P300 signals integrated in a real-time graphical scenario builder.

* + 1. **Challenged of Brain Computer Interfacing**

According to (Bonci *et al,* 2021), some of the general limitations of BCI technology are:

1. Inaccuracy in terms of classification of neural activity
2. Limited ability to read brain signals using non-invasive technology
3. In some cases, the requirement of surgery
4. The bulk nature of system leading to uncomfortable user experience
5. Ethical issues on reading inner thoughts and security of personal data from attackers or intruders.

Methods of recording brain activity are mostly causes of other forms of limitations. A major limitation of fMRI is the lack of image contrast. Setup and maintenance cost of PET is its major limitation. Motion artifacts, environmental noise, or eye movements can reduce the reliability of data acquired and can limit the ability of extracting relevant patterns. The features extracted from EEG are non-stationary due to changes among sessions that might not be monitored or controllable. For instance, the change in mental state or different levels of attention can affect the EEG signal characteristic and can increase its variability in different experimental sessions. In EEG signal acquisition, challenges concerning the identification of the optimal location for reference electrodes and the control of impedance when testing with high-density sponge electrode nets should be properly resolved. A relevant aspect related to the use of BCI concerns the trade-off between the difficulty interpreting brain signals and the quantity of training needed for efficient operation of the interface (Bonci *et al,* 2021).

* + 1. **Ethical and Legal Issue in Brain Computer Interfacing**

The possibilities of mind-reading and mind-conditioning in BCI poses serious ethical and legal issues for BCI technology. Brain-hacking and brain-phishing are possibilities that cannot be ignored, thus the need for policy-makers in the development of BCI technology.

As regards to mind-reading (Vlek *et al,*2012), discusses topics such as the representation of persons with communication impairments dealing with technological complexity and moral responsibility in multidisciplinary teams, and managing expectations, ranging from an individual user to the general public.

With reference to the mind-conditioning issue, the case of deep brain stimulation (DBS) is discussed in Schermer, M. (2011). DBS is currently used to treat neurological disorders such as Parkinson’s disease, essential tremor, and dystonia and is explored as an experimental treatment for psychiatric disorders such as major depression and obsessive compulsive disorder. Fundamental ethical issues arise in DBS treatment and research, the most important of which are balancing risks and benefits and ensuring respect for the autonomous wish of the patient.

* 1. **POST STROKE UPPER-LIMB REHABILITATION**

A stroke or brain attack occurs due to blocked or burst brain vessel in the brain. As the brain cannot store oxygen, it relies on a network of blood vessels to provide it with blood that is rich in oxygen. A stroke results in a lack of blood supply, causing surrounding nerve cells to be cut off from their supply of oxygen thus damaging these nerve cells. When tissue is cut off from its supply of oxygen for more than three to four minutes, it begins to die. Depending on the function of nerve cell damaged, body parts function are impaired or lost, due to lost communication between the damaged nerve cells and other cells of the body.

Stroke rehabilitation is a process by which those with disabling strokes undergo treatment by regaining and relearning the skills of everyday living. Stroke rehabilitation should begin almost immediately. It is believed that most of the recovery from strokes occur within the first 3-6 months but thereafter improvement is less likely. This can differ on a case-by-case basis.

Stroke can result in upper-limb motor impairment which results in functional limitations in using the affected upper-limb after stroke. Impairments may occur as impairment in body function or body structure. These impairments are not static, the nature of the impairment may change as motor recovery proceeds and there may be multiple impairment (Preeti Raghavan, 2015).

Upper-limb impairments are best approached from the perspective of damaged or lost function. There are three main consequences of stroke on the upper limb which are: learned non-use, learned bad use and forgetting as determined by behavioral analysis of a task (Preeti Raghavan, 2015).

Learned non-use occurs due to disuse of the affected upper-limb due to weakness/paralysis or sensory loss which may become habitual overtime and the limb may not be involved in functional activities, even though the individual can move it eventually becoming a learned behavior (Preeti Raghavan, 2015).

Learned bad use occurs in the absence of appropriate feedback and correction of abnormal motor behavior when using compensatory strategies to complete movement tasks in cases of abnormal movement due to forced limb movement. Thus the patient is learning to use the limb in the wrong way (Preeti Raghavan, 2015).

Continually retaining a motor skill learned through training is expected in the absence of consistent training. However, rats with motor cortex injury show a decline in performance during intervals of no training and additional training is required to recover performance to previous levels before training. Breaks in rehabilitation similarly lead to forgetting of upper extremity motor skills in humans after stroke. Thus new skills, although reasonably stable in healthy individuals, are more transient after stroke (Krakauer JW, 2006).

* + 1. **Stroke Neuro-Rehabilitation**

Until the 1950s, post-stroke patients were discouraged from being active after stroke. After the 1950s, therapeutic exercises were prescribed to patients from which good outcome was considered to be achieving a level of independence in which patients are able to transfer from the bed to the wheelchair without assistance.

In the early 1950s, Twitchell studied recovery pattern in stroke-patients. He found that if there is some recovery of hand function within four weeks, there is a 70% chance of making a full or good recovery. Twitchell reported that most recovery happens in the first three months and minor recovery occurs after six months.

There are majorly three classes of therapeutics for stroke neuro-rehabilitation which are:

1. Motor Re-learning: This is achieved through constrain-induced movement therapy or mental-imagery of movement tasks. In the former, affected patients are forced to use affected parts while constraining movement of normal working parts. In the latter patients are taught and made to imagine movement of affected body parts.
2. Brain Repair: This can be achieved through a technology called Functional Electrical Simulation (FES) where electrical stimulation mimics the action of healthy muscles to improve weak muscles. There is also robotic-rehabilitation where robot-assisted training enables stroke patients with moderate or severe upper limb impairment perform repetitive tasks in a highly consistent manner, tailored to their motor abilities and disabilities. These therapies have achieved the highest level of evidential support by the American Heart Association (Class I, Level of Evidence A) for the outpatient and chronic care settings and Class IIa Level of Evidence for the inpatient setting (Winstein CJ etal, 2016).
   * 1. **Stroke Robotic-Rehabilitation**

Stroke robotic-rehabilitation emphasizes intense, highly repetitive and task-oriented movements so as to increase accuracy and functional use in subjects with stroke-induced paresis.

Stroke robotic-rehabilitation promotes cost-effective use of human resources and rehabilitation programs standardization. It also allows physiotherapist focus on functional rehabilitation during individual training and supervising patients simultaneously during robot-assisted therapy sessions, thus paying less attention to labor-intensive parts (Masiero, Poli,Rosati,et al).

To decrease the dependence on therapists, automated electro-mechanical gait machines have been developed. Gait machines comprise an electro-mechanical solution with two driven foot plates that simulate the phases of gait (i.e., the ‘Gait Trainer’,Schimdt H,et al) or a robot-driven exoskeleton orthotic system such as ‘Lokomat’ (Riener H,et al). ‘Lokomat’ is a computer-controlled robotic gait orthotic system that guides the patient in which gait training is automated, following a pre-programmed gait pattern. Gait training with the ‘Gait Trainer’ is also automated.

(Wagner et al, 2011) performed an economic analysis of robot assisted therapy for long-term upper limb impairments after stroke compared with intensive traditional therapy and standard care. At 36 weeks post-randomization, the total costs were comparable between the three groups, and changes in quality of life were modest and did not differ significantly.

Stroke robotic-rehabilitation offers optional extra-features such as interaction and stimulation, extrinsic feedback.

* + 1. **Robotic Devices-Machine for Upper Limb Rehabilitation**

(Massiero, Poli, Rosati et al, 2014).Existing upper limb robotic systems can be classified by:

1. The upper limb function on which they focus
   1. Unilateral
   2. Bilateral shoulder, elbow, wrist or hand movements
2. Mechanical characteristics:
   1. Exoskeleton
   2. Operational machines(End effector based machines)
3. Control strategy:
   1. Passive movement
   2. Active non-assisted
   3. Active assisted or Resistive mode
   4. Bi-manual exercise
   5. **ELECTRONIC SPEED CONTROLLER (ESC)**

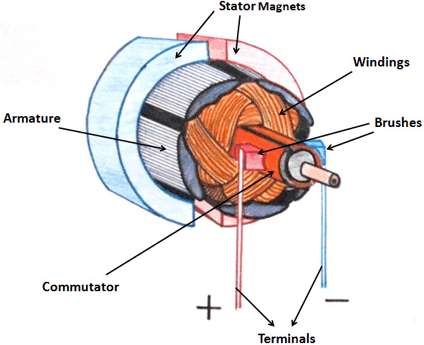
An electronic speed controller (ESC) is an electronic device designed to generate control signals which control the direction of angular-motion and angular-speed of brush-less direct-current (BLDC) motors.

* + 1. **Brushed Motor Working**

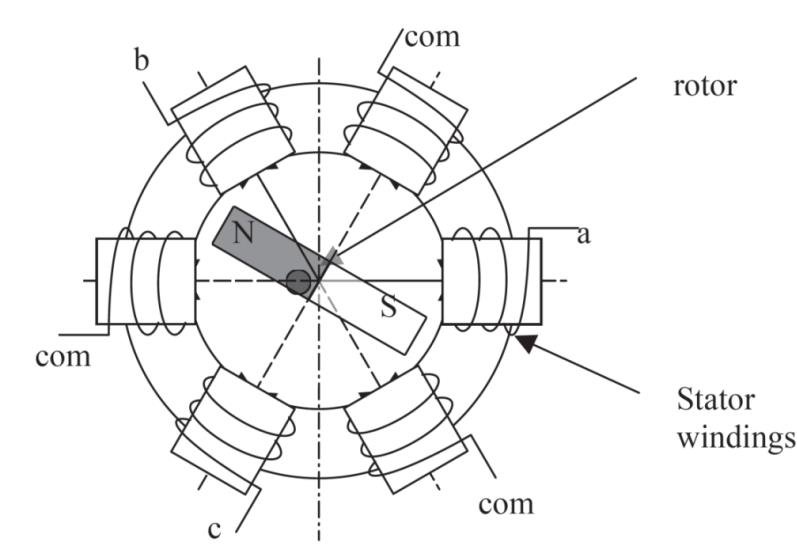
A BLDC motor is a direct-current(DC) motor which makes use of three-phase electromagnetic coils(star-connected) as its stator and makes use of pairs of permanent magnets as its rotor.

Brushed motors on the other hand make use of permanent magnets on the outer body as the stator, electromagnets inside as the rotor and mechanical commutation through brushes. The structure of brushed DC motors (the placement of the brushes and commutator) and interaction between the excited electromagnet and the permanent magnets is such that there is no need for an electrical control-scheme to generate control signals for the electromagnet so as to start and maintain rotational-motion. The electromagnet (rotor) is placed within the magnetic field of the permanent-magnet (stator) such that current passed through the electromagnet is perpendicular to the permanent-magnet magnetic field while generating flux for the electromagnet (rotor) which interacts with the permanent-magnet field to generate force perpendicular to the current and the magnetic field in a direction given by flemmings left hand rule. The motion of the coil and the placement of brushes and commutator changes the direction of current through the electromagnet as required to sustain motion. Motion is stopped by disconnecting the electromagnet (rotor) from its electrical source.

Unlike the brushed motors, which employ mechanical commutation through the use of brushes, BLDC motors do not make use of brushes or any other mechanical means for commutation, thus the need for electrical means of commutation.

**

***Figure2.3 Inner Structure of a Brushed DC Motor***

**

***Figure2.4 a Brush-less DC Motor Model***

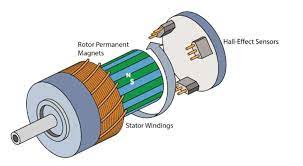
* + 1. **Advantages of Brushed Direct-Current Motor**

1. High starting torque: Brushed motors are preferred for applications that need to get up to speed very quickly as they have high starting torque compared to their brush-less counterpart.
2. Low cost: Brushed motors are of low cost compared to their brush-less counterpart, partly because they do not require special controllers to run.
3. Compatibility with industrial environments: Brushed motors are preferred in industries as they generally have high-starting torque.
   * 1. **Disadvantages of Brushed Direct-Current Motor**
4. Increased Maintenance: Due to the effects of friction on a motor’s carbon brushes, they will naturally wear over time. As a result, brushed electric motors are more likely to require some sort of maintenance in the form of brush cleaning or replacement.
5. Lower speed:  Despite a high starting torque, brushed motors are not as capable of maintaining high-level speeds. This is because running a brushed motor at a consistent high speed can cause it to get warm.
   * 1. **Brush-less Direct-Current Motor**

Electronic commutation replacing mechanical commutation make BLDCs popular. This replaces the conventional mechanics, comprised of brushes rubbing on the commutator to energize the windings in the armature of a DC motor. Electronic commutation provides greater efficiency over conventional DC motors with improvements of 20 to 30% for motors running at the same speed and load. Furthermore, the BLDC motor is more durable. It retains its high performance while the efficiency and power of an equivalent conventional motor declines due to wear, causing poor brush contact, arcing between the brushes and the commutator dissipating energy, and dirt compromising electrical conductivity. Greater efficiency allows BLDC motors to be made smaller, lighter and quieter for a given power output, further increasing their popularity in sectors such as automotive; white goods; and heating, ventilation and air conditioning (HVAC). Other advantages of BLDC motors include superior speed versus torque characteristics (with the exception of torque at start-up), a more dynamic response, noiseless operation, and higher speed ranges. The downside of BLDC motors is their complexity and the associated increase in cost. Electronic commutation demands supervisory circuits to ensure precise timing of coil energization for accurate speed and torque control, as well as ensuring the motor runs at peak efficiency.

All electric motors, whether mechanically or electronically commutated, adhere to the same basic method of converting electrical energy into mechanical energy. Current through a winding generates a magnetic field, which in the presence of a second magnetic field (typically initiated by permanent magnets) generates a force on that winding that reaches a maximum when its conductors are at 90° to the second field. Increasing the number of coils raises the motor output and smooth’s power delivery.

A BLDC motor overcomes the requirement for a mechanical commutator by reversing the motor set-up; the windings become the stator and the permanent magnets become part of the rotor. Because the windings are stationary, permanent connections can be established to energize them. In order for the stationary winding's to move the permanent magnet, the winding's need to be energized (or commutated) in a controlled sequence by the supervisory circuit generally called en electronic speed controller(ESC) to produce a rotating magnetic field. Because the rotating magnetic field generated by the stator causes the rotor to revolve at the same frequency, a BLDC motor is known as a “synchronous” type. BLDC motors can come in one, two or three phase configuration.

**

***Figure2.5 Inner Structure of a 3-Phase BLDC Motor***

* + 1. **Brush-less Direct-current Motor Control**

The most common method for BLDC motor control is the use of three half-bridge MOSFET configuration, each one for switching each phase of the BLDC motor. The high-side of the half-bridges are controlled using pulse-width-modulation(PWM) which limits start-up current and offers precise control over speed and torque and also trade-off between high-frequency switching losses and low-frequency ripple current. Typically, designers use a PWM frequency of at least an order of magnitude higher than the maximum motor rotation speed. There might be need for positional feedback to the controller so as to generate the next control signal for the half-bridges depending on the adopted control scheme.

There are three control schemes for electronic commutation which are trapezoidal, sinusoidal and field-oriented control.

1. Trapezoidal control scheme: At each step, two windings are energized (one “high” and one “low”) while the other winding floats. The downside of the trapezoidal method is that this stepped commutation causes the torque to ripple, especially at low speeds.
2. Sinusoidal control scheme: This is more complex, but it reduces torque ripple. All three coils remain energized with the driving current in each of them varying sinusoidally at 120° from each other. The result is a much smoother power delivery compared with the trapezoidal technique.
3. Field-oriented control scheme: This relies on measuring and adjusting te stator current so that the angle between the rotor and stator flux is always 90°. This technique is more efficient at high speeds than the sinusoidal method and gives better performance during dynamic load changes compared to all other techniques. There is virtually no torque ripple, and smoother, accurate motor control can be achieved at both low and high speeds.
   * + 1. **Positional feedback**

Two technologies offer a solution for positional feedback. The first and most common uses three Hall-effect sensors embedded in the stator and arranged at equal intervals, typically 60° or 120°. A second, sensor-less control technology comes into its own for BLDC motors that require minimal electrical connections.

1. Sensored: In a sensor-equipped BLDC motor, each Hall-effect sensor is combined with a switch which generates a logic “high” (for one magnetic pole) or “low” (for the opposite pole) signal. The commutation sequence is determined by combining the logic signals from the Hall-effect sensors and associated switches. At any time, at least one of the sensors is triggered by one of the rotor’s magnetic poles and generates a voltage pulse.
2. Sensor-less: A sensor-less BLDC motor makes use of the electromotive force (EMF) that gives rise to a current in the windings of any DC motor with a magnetic field that opposes the original change in magnetic flux as described by Lenz’s Law. The EMF tends to resist the rotation of the motor and is therefore referred to as “back” EMF.

By monitoring the back EMF, a suitably programmed micro-controller can determine the relative positions of the stator and rotor without the need for Hall-effect sensors. This simplifies motor construction, reducing its cost as well as eliminating the additional wiring and connections to the motor that would otherwise be needed to support the sensors, thus improving reliability.

However, because a stationary motor generates no back EMF, the controller is unable to determine the motor position at start-up. The solution is to start the motor in an open loop configuration until sufficient EMF is generated for the controller to determine rotor and stator position and then take over supervision. A more sophisticated control regime is used if the motor is used in an application where reverse rotation is forbidden.

* + 1. **Advantages of Brush-less Direct-Current Motor**

1. Long lifespan: As brush-less DC motors do not have brushes, they require less maintenance as mechanical motion and friction does not wear out any commutation part
2. Efficiency: The absence of brushes in brush-less DC motors means that no speed is lost due to friction. This makes brush-less DC motors typically 5-90% efficient compared with their brushed counterparts at a typical efficiency 75-80%
3. Quiet operation: Due to the lack of brushes, brush-less motors run extremely quietly and have particularly smooth operation. This is useful for applications that require such properties such as the development of medical robots, patient hoists, etc.
   * 1. **Disadvantages of Brush-less Direct-Current Motor**
4. Requires a controller: Brush-less DC motors need to be wired to an electronic sped controller (ESC), to enable current to flow through the electromagnets.
5. Cost: Due to the requirement of an electronic speed controller, brush-less DC motors can be more expensive.

**CHAPTER THREE**

**PLATFORM FOR UPPER-LIMB STROKE REHABILITATION (PULSR)**

* 1. **INTRODUCTION**

Platform for upper-limb stroke rehabilitation (PULSR) is a two degree of freedom robot designed for monitored and controlled exercise for the purpose of upper-limb stroke rehabilitation through repetitive exercises monitored and controlled by a physiotherapist using a control and interactive interface. PULSR reduces the need of physiotherapist using crude and non-metricized means for generating the force for moving stroke affected parts of stroke-patients during exercise sessions, thus allowing the physiotherapist focus on monitoring stroke-patient response to exercise and controlling exercise sessions as required by the physiotherapist.

PULSR is an end-effector based stroke-rehabilitation robot with user feedback generated using a GUI. This robot can run in three modes which are the patient-controlled mode, the patient-assisted mode and the circular-path mode.

The patient-controlled mode is to be used by patients capable of some level of motion in the upper-limb, such patients are expected to move the effector in a circular path trying to catch up with a dot that is moving in a circular path on a screen-display which also shows the end-effector position.

The patient-assisted mode is to be used by patients totally incapable of any form of motion in the upper-limb. Such patients have their affected arm strapped to PULSRs end effector and the motion of the end effector is totally controlled by the physiotherapist via a keyboard attached to the computer that generates visual feedback. Instructions on whether the patient is to try to oppose the motion of the end-effector or not depends on the physiotherapist in charge of the exercise session.

The circular-path mode is to be used by patients that have gone through the patient-assisted mode and show symptoms of upper-limb motion recovery. With such patients arm strapped to the PULSR robot, the PULSR robot moves the end-effector in a circular path. Instructions on whether the patient is to try to oppose the motion of the end-effector or not depends on the physiotherapist in charge of the exercise session.

An explanation of the system structure is given along with a fitting block diagram and a skeletal structure of the PULSR robot in the next section.

* 1. **BLOCK DIAGRAM OF SYSTEM**

This sub-section shows a block-diagram explaining how the PULSR robot works, which is also explained in words in subsequent pages after the block-diagram.

Also included in this sub-section is a skeletal structure of PULSRs moving parts.

**Upper Motor ESC Driver**

**Upper Motor**

**Lower Motor**

**Lower Motor ESC Driver**

**Upper Rotary Encoder**

**Lower Rotary Encoder**

**Micro-controller**

**Keyboard**

**Python Script**

**GUI**

**PULSR-PC**

***Figure3.1 Block Diagram of PULSR Components***

*“insert PULSR GUI PICTURE”*

As shown in Figure, the PULSR robot is made up of two BLDC motors, two ESC’s, two rotary encoders, one arduino and a personal computer consisting of a python script, a keyboard and a GUI for feedback of the PULSR root end-effector position.

1. Rotary Encoders: The rotary encoders provide angular position of the motors to the arduino which in turn feeds these angles to the PULSR python script in the personal computer which performs a forward kinematics computation based on the geometry of PULSR to compute the end effector position. The forward kinematics algorithm was provided by Isaac Omotayo in *‘Sir Isaacs’ thesis reference’*
2. PULSR PC: The PULSR PC consist of three important components which are
   1. Keyboard: This is used to first of all select the mode of operation (patient-controlled, patient-assisted or circular-path mode) of the PULSR robot. When in circular-path or in patient-controlled mode, direction control is given to python script or patient respectively. If in patient-assisted mode, direction control signals are generated by the keyboard according to the keys:
      1. W-up
      2. S-down
      3. A-left
      4. D-right
   2. PULSR python script: This handles communication with the hardware, receiving angle readings of the PULSR motors from which it computes the end effector position. It also sends communication control signals to the hardware to control the PULSR motor direction of motion
   3. User Interface: This gives visual feedback of the position of the end-effector on a visual representation of the work-path of the end-effector
3. Microcontroller: The micro-controller is an arduino board which controls the ESCs of the upper and lower-motors to control the motion of each motor respectively. It communicates with the PULSR PC by sending angular position of the motors and receiving control signals which are to be translated to control signals for the appropriate ESC(s).
4. Rotary-Encoders: These are attached to the motors through a shaft to generate signals that indicate changes in angle and direction of change in angle of the motors from which the micro-controller computes the angular-position of the motors.
5. ESCs: These (Electronic Seed Controllers) handle the control of the motion of the motors as they are BLDCs by receiving appropriate control signals from the micro-controller as dictated by the data-sheet of the used ESCs.

*“insert picture of PULSR here with patient”*

* 1. **STATEMENT OF PREVIOUSLY DONE WORK**

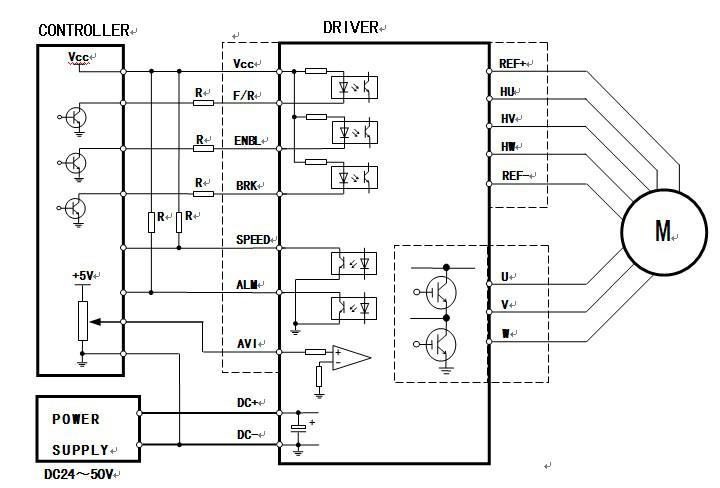
Before my assumption of duty in the research laboratory, the research laboratory had performed the mechanical fabrication and coupling of PULSR moving parts. The research laboratory already had a preexisting platform for controlling the direction of motion of PULSR, using buttons as inputs. Also there were no modes of operation of PULSR.

I re-implemented the electronics of the already existing platform, and made the direction control automated for a circular-path. I implemented a visual feedback user interface, implemented, replaced the button input with a personal computer keyboard input and also developed the modes of operation of PULSR. I also developed the communication protocol between the micro-controller and the PULSR PC.

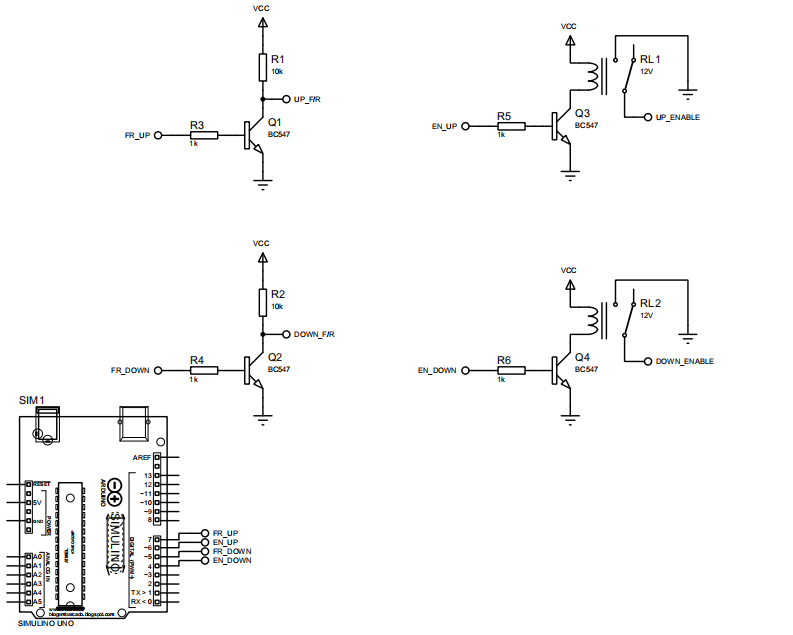
* 1. **CONTROL-BOARD RE-DESIGN**

The ESCs have two inputs EN and F/R for enabling/disabling motion and switching between clockwise/counter-clockwise motion of motor respectively which both require an open-collector to be controlled as they are pulled high and indicating one of two states and need to be pulled low to switch to the other state.

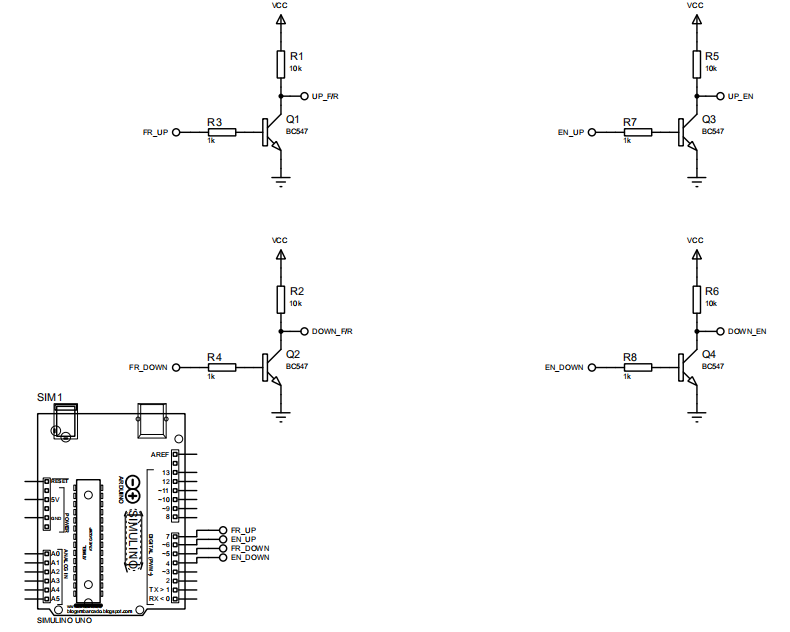
There was a pre-existing control-board for the ESCs before my assumption of duty at the research laboratory. I re-designed this control-board due to its intermittent failure.



***Figure3.4 BLDC ESC Controller Schematic***



***Figure3.5 PULSRs Previous Control Board Circuit***



***Figure3.6 PULSRs Current Control Board Circuit***

* 1. **SOFTWARE DESIGN**

The PULSR software is embedded within the PULSR PC and is basically a python script that does three things which are: to communicate with the micro-controller in the control-board for receiving angular-position data and sending control-signals, implements a visual feedback user interface that plots the work-path of the end-effector and the end-effector current position using received angular-position data and last but not the least, detects keyboard input for direction-control, mode of operation switching, pausing and resuming exercise session signals.

* + 1. **Block Diagram**

The block-diagram in Figure, shows the components of the PULSR software and also flow of data between these components.

**pyGame Keyboard Input**

**Keyboard to control-signal mapper**

**Serial**

**Output**

**pyGame**

**GUI**

**Angle to position function**

**Serial**

**Input**

**To Micro-controller**

**From Micro-controller**

***Figure3.7 Block Diagram Showing Components of PULSR software***

* + 1. **User Interface Design**

The user interface has been shown in Figure3.2 and its python script code for the user-interface can be found in the appendix section of this document,

* + 1. **Hardware Communication**

An information exchange discipline was developed for communication between the micro-controller and the python script of the PULSR software. Figure, shows and explains the agreed information discipline.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| R | ENU | FRU | END | FRD | Circ | M |

***Figure3.8 Agreed Information Transfer Discipline between PULSR PC and firmware***

M 0: PULSR in patient/physiotherapist mechanical control mode

1: PULSR in physiotherapist through computer control mode

Circ (Only works in computer control mode\_

0: direction control given to keyboard

1: direction control given to circle motion generating function

FRD 0: generates pushing force (down-motor in clockwise direction)

1: generate pulling force (down-motor in counter-clockwise direction)

END 0: disables the down motor (i.e. disables push and pull motion)

1: enables the down motor (i.e. enables push and pull motion)

FRU 0: generates right direction force (up-motor in clockwise direction)

1: generates left direction force (up-motor in counter-clockwise direction)

ENU 0: disables the up motor (i.e. disables left and right motion)

1: enables the up motor (i.e. enables left and right motion)

R (Unused)

* 1. **SYSTEM PROBLEM IDENTIFICATON, PROPOSED AND ADOPTED SOLUTIONS**
     1. **Problem Specification**

Table 3.1 shows PULSR damaged parts and their associated resulting behavior noticed after troubleshooting during the process of development:

**Table3.1 PULSR damaged parts and resultant behavior of system**

|  |  |
| --- | --- |
| **Part** | **System behavior** |
| BLDC motor Hall Sensor | 1. Jerky motion of motor 2. Stalling of motor with no-load |
| Electronic Speed Controller | 1. BLDC motor not moving on being enabled via ESC. |

* + 1. **Troubleshooting Approach**

1. BLDC motor Hall Sensor:
   1. Scoping of outputs of hall sensors:
      1. The hall sensors outputs were pulled up to +5V through 1k resistors, as they are open-collector output
      2. The halls sensor outputs were connected to three separate input channels of a four channel oscilloscope
      3. The BLDC motor was rotated manually
      4. The damaged hall sensor maintained its voltage level throughout rotation. The hall sensors in okay condition had their voltage level changing between +5V and 0V.
   2. Resistance between hall sensor output and ground:
      1. The resistance between each hall sensor output and the common ground of the hall sensors was measured.
      2. The bad hall sensor had low resistance compared to the hall sensors in okay condition which read infinite resistance indicating an open circuit.
2. Electronic Speed Controller:
   1. Scoping of phases output:
      1. The output phases U,V,W of the ESC were scoped
      2. The BLDC motor was rotated manually
      3. The bad phase was not energized at all throughout rotation.

**Table3.2 Results Of Scoping and Resistance Tests On PULSR BLDC Hall Sensors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MOTOR** | **HU** | **HV** | **HW** |
| **SCOPING TEST** | UP | Patterned square wave | static | Patterned square wave |
| DOWN | Patterned square wave | Patterned square wave | Patterned square wave |
| **RESISTANCE TEST** | UP | Open circuit | 53.9 Ohms | Open circuit |
| DOWN | Open circuit | Open circuit | Open circuit |

* + 1. **Proposed Solutions**

1. BLDC motor Hall Sensor:
   1. Replacing BLDC motor:
      1. Problem: Cost, BLDC motor part not readily available.
   2. Opening up motor to replace hall sensor module
      1. Problem: Custom hall sensor module used.
   3. Emulating hall sensor:
      1. Explanation: From the BLDC motor having all it hall sensor in okay and working condition, there is a pattern to the hall sensor plots during rotation. The correct behavior appropriate damaged hall sensor can be mimicked from the appropriate good hall sensor plot.
      2. Problem: Timing of signal mimicking not being exact.
2. Electronic Speed Controller:
   1. Replacing the damaged ESC:
      1. Problem: Cost, ESC not readily available thus time constraint.
   2. Building an ESC
      1. Problem: Complexity
      2. **Adopted Solution**
         1. **Hall sensor Emulator**

The code snippet and state-transition diagram for the hall sensor emulator is given in the appendix section of this document.

The hall sensor emulator was able to get the BLDC motor move in one direction smoothly, but jerkily in the reverse direction.

***Figure3.9 Block Diagram of Hall Sensor Emulator***

Micro-controller

Hall Sensor 1

Hall Sensor 1

Emulated Hall Sensor 2 signal

**CHAPTER FOUR**

**ELECTRONIC SPEED CONTROLLER (ESC)**

* 1. **INTRODUCTION**

The electronic speed controller is a device built to control and regulate the motion of an electric motor in terms of speed, optionally providing reversing of the motors rotation and dynamic breaking. The ESC follow some speed reference signal and varies the switching parameters (duty-cycle or frequency of switching) of a MOSFET network (commonly half-bridge), thus controlling the speed of the motor.

Brushed motors and brush-less motors, require different speed control scheme. While the former’s speed can be controlled by varying the voltage of the armature, the latter has its speed varied by adjusting the timing of pulses of current delivered to the several windings of the motor.

Brush-less ESC systems basically create three-phase AC power, like a variable frequency drive, to run brush-less motors. The correct phase of the current fed to the windings of the motor varies with the motor rotation. The correct phase of the current to be fed to the windings can be predicted by generating information on the rotation via means of back EMF from the motor windings or use of separate magnetic hall sensors or optical detectors.

Method of creating three-phase AC power for brush-less ESC systems vary depending on the manufacturer. As stated in section 2.3.5 of this text, there are the trapezoidal method, sinusoidal method and field-oriented method. The trapezoidal method was adapted due to ease of implementation compared to the sinusoidal and field-oriented techniques. And our method of generating position (rotation) feedback information for the brush-less ESC was the use of the magnetic-hall sensors that are attached to the brush-less motors.

Figure and , gives a block-diagram and a state-transition diagram explaining the component of the trapezoidal control based brush-less ESC and each magnetic-hall sensor input and its associated three phase output respectively.

**Hall Sensor Inputs**

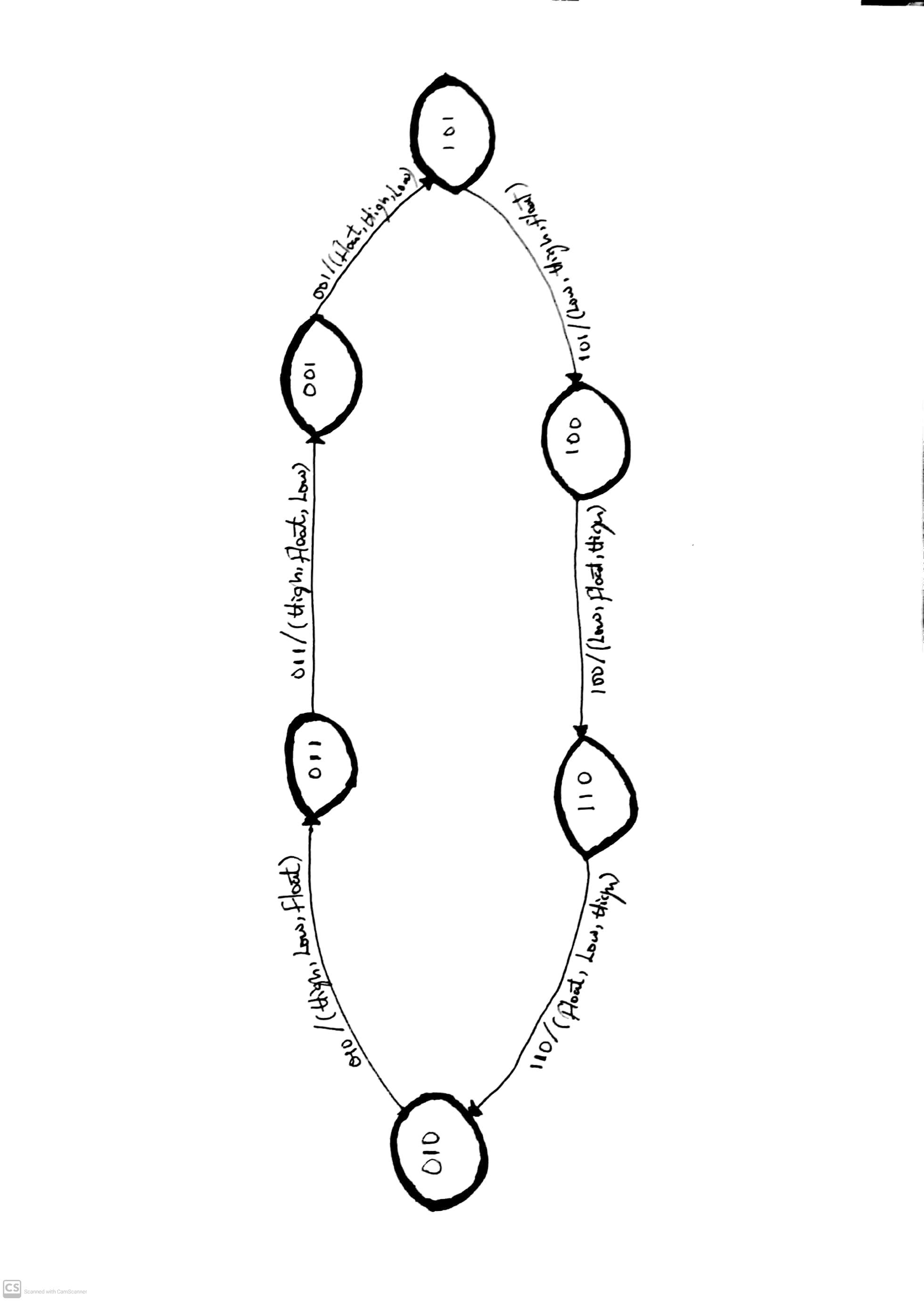
**Micro-controller**

**MOSFET Gate Drivers**

**MOSFET Half-Bridges**

**BLDC Motor**

***Figure4.1 ESC Block Diagram***

****

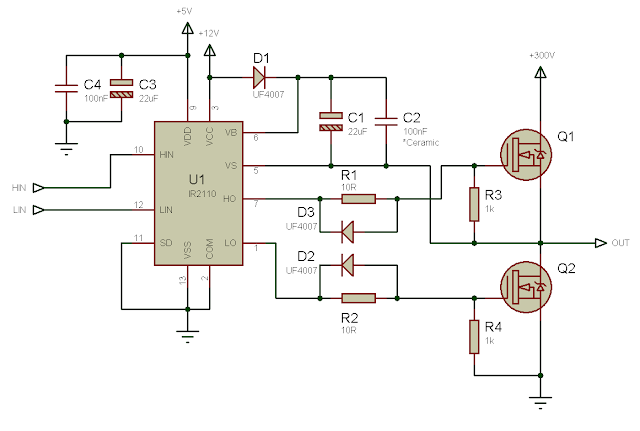
***Figure4.2 ESC state transition diagram***

* 1. **ARDUINO CODE AND TESTING**
     1. **State Transition Diagram of Arduino Code**

The C based codes of the developed ESC can be found in the appendix section of this document. The state transition diagram describing the working of the ESC is given in Figure4.2.

* 1. **MOSFET Gate Drivers**
     1. **Working Principle**

As indicated by Figure4.3 on the next page, the MOSFET gate driver IR2110 has three inputs which are the shutdown input (SD), high input (HI), low input (LO) and has majorly two outputs which are the high output (HO), low output (LO). The inputs VDD and VSS are the logic level voltage supply terminals requiring at least +5V across these terminals, the input pair Vcc and COM also form a voltage supply terminal. Likewise the Vb and Vs input pair also form another supply terminal.

**

***Figure4.3 MOSFET gate driver and half-bridge circuit***

The outputs LO and HO switch their source between Vcc, COM and Vb and Vs respectively according to logic input on inputs LI and HI respectively. Table 4.1 shows the logic level inputs to LI and HI and the expected source for LO and HO.

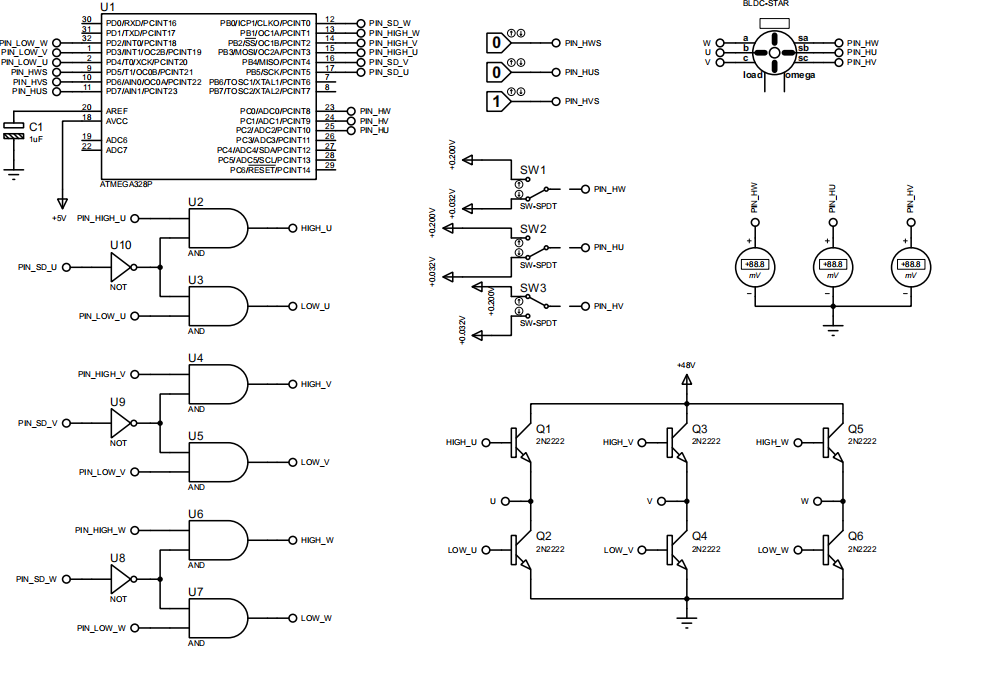
**Table 4.1 IR2110 Inputs and Output Source**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **INPUTS** | **LO** | **HO** |
| LI | 0 | COM | - |
| 1 | VCC | - |
| HI | 0 | - | Vs |
| 1 | - | Vb |

* 1. **HALF-BRIDGES**
     1. **Working Principle**

As shown in Figure4.4 on the next page, the half-bridge is made up of two IRF150 n-channel enhancement MOSFETS. These MOSFETS are to operate in one of three states which are the voltage-divider state, pull-up state and pull-down state. The state is determined by the input to the high and low side MOSFETs.

In the pull-up state, the high side MOSFET is driven to saturation while the low side is not activated thus connecting the high side MOSFET source (output of the half-bridge) terminal to the voltage supply. In the pull-down state, the low side MOSFET is driven to saturation while the high side is not activated, thus connecting the low side MOSFET drain (output of the half-bridge) terminal to ground. In the voltage-divider state, both high side and low side MSOFETs are not activated making the drain to source channel of both, highly resistive paths. This forms a voltage divider giving a voltage at the output of the half-bridge somewhere between the drain voltage of the high side MOSFET and ground depending on the value of the resistance formed.



***Figure4.4 Proteus Circuit Model for ESC***

**CHAPTER FIVE**

**BCI TASK DISCRIMINABILITY ANALYSIS**

* 1. **INTRODUCTION**

The aim of this project was to study the extent to which two BCI task events (movement and rest) differ from each other using spectral measures of EEG signals for both classes of events.

A data-set was provided by the Clinical Neuro-technology (CNT) team from Charitea University, Berlin. The data-set consisted of 100 instances of each class, each being EEG multi-channel time-series and the classes being movement and rest. In the provided data-set, movement cues are identified as close events while rest cues were identified as relax events. This project computed two parameters which are the average power spectral density for each class and the percental change over time averaged epochs of both classes as measures of discriminability between each class.

The provided data-set was in the .XDF format which stores streams and marker-stream information. Our tool for analysis was the MNE library provided within python’s jupyter notebook. As MNE deals with array information formatted into one of its basic type-classes such as RAW, epochs, event, definition of functions converting streams to RAW and marker-streams to events was required so analysis could be easily done using MNE.

* 1. **PRE-PROCESSING STEPS**
     1. **Conversion Of Stream to RAW**

A function named ‘stream2raw’ was defined to convert XDF data-stream to MNE RAW format. The function takes in as arguments, the XDF data-stream, marker-stream and two formatting arguments which are ‘marker\_out’ and ‘ch\_type\_stream’. The ‘marker\_out’ argument determines if the markers are to be returned as events and thus calls the ‘marker\_stream2events’ function if so. The ‘ch\_type\_transform’ argument dictates the channel type transformation and has a default value of ‘ch\_type\_transform\_default’ which is defined by MNE. The function then returns the RAW format of the data-stream and the event format of the marker-stream composed of the events and event\_id arrays.

* + 1. **Conversion Of Marker-Stream to Events**

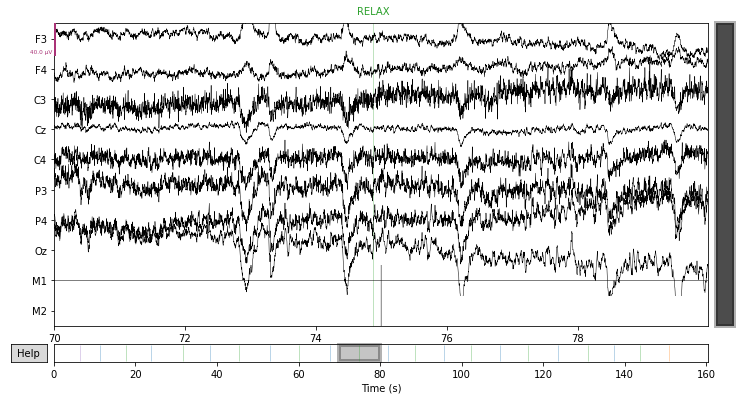
A function named ‘marker\_stream2events’ was defined to convert XDF marker-stream to event format. This function takes in the marker-stream, the time-reference, the nominal frequency rate, the RAW format of the data-stream and is therefore called within the ‘stream2raw’ function then returns the events and event\_id required by the event type-class of MNE.

* + 1. **Band-pass Filtering**

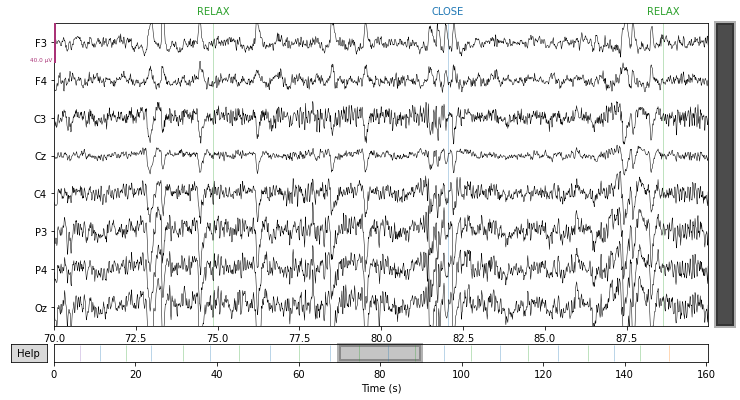
The RAW data format was filtered within the band 1Hz to 30Hz so as to retain frequency and time domain information of EEG which is commonly found within this range. Before using filtered data, empty channels in the filtered data were dropped as discussed in 5.2.4

* + 1. **Dropping Empty Channels**

The MNE method for dropping empty channels was employed after visual analysis on filtered data indicating the channels ‘M1’ and ‘M2’ were empty.

**

***Figure5.1 EEG Data before Filtering***

**

***Figure5.2 EEG Data after Filtering and Dropping Bad Channels***

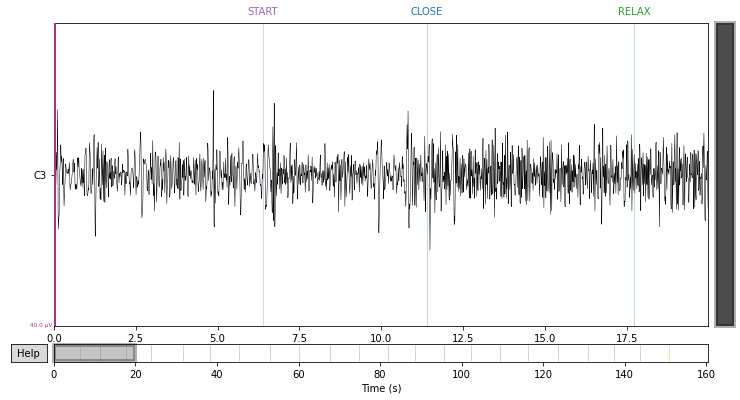
* 1. **FEATURE EXTRACTION**
     1. **Band-pass Filtering**

The filtered data was further filtered passing bands between 6Hz and 30Hz, as this range of frequency band in EEG data bears information on human motion.

* + 1. **Laplace-Averaging**

Laplacian averaging also called laplacian filtering, extracts local activity at certain electrode positions by subtracting the average activity present in the four orthogonal nearest neighbouring electrodes. This cause’s common activity such as body generated artefacts to be subtracted away from the electrode of interest. A closely related type of spatial filtering, common average referencing (CAR). Enhances the local activity if certain electrode positions by subtracting the average over all electrodes.

Motion bearing information is predominant in the C3 and C4 channels of EEG data according to the 10-20 electrode placement system. Activity in C3 indicates motion activity in the right hand part of the body. Activity in C4 indicates motion activity in the left hand part of the body. Laplace-Averaging was carried out on C3 using F3, P3 and Cz, which are channels surrounding C3 as our target was discriminating right hand activity from resting position.

**

***Figure5.3 Plot of Computed Laplace Average***

* 1. **ANALYSIS OF DISCRIMINABILITY**
     1. **Epoching**

Using MNE Epochs method, across all instance of classes in the Laplace averaged data, epochs of 5seconds window were extracted. All epochs for the movement class was stored in one variable and all epoch of the rest class was store in another for analysis of discriminability between the two classes.

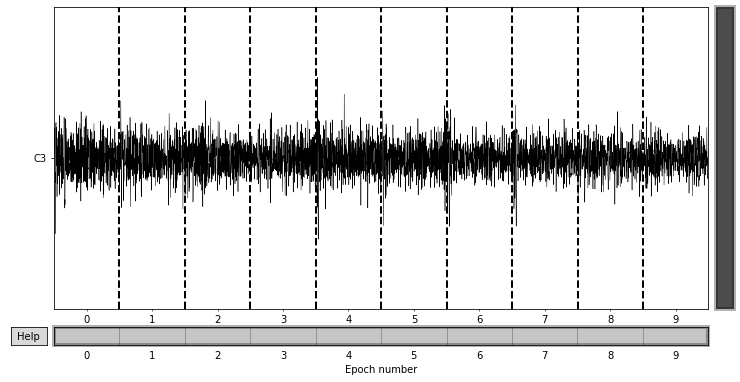
* + 1. **Power Spectral Density Analysis**

Across all the movement epochs and the rest epochs, the power spectral density was computed using fast Fourier transform (FFT).

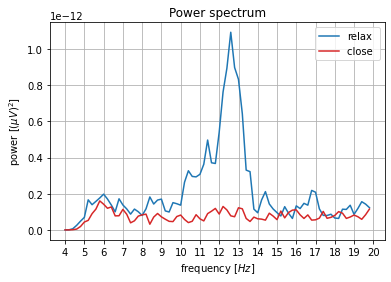
Within the 10Hz to 14Hz band , the power spectral density plot showed high level of difference between both classes, thus indicating discriminability in the 10Hz to 14Hz frequency band.

* + 1. **Percental Change Analysis**

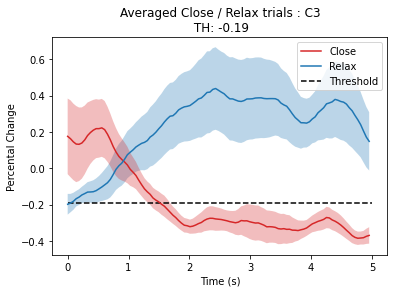
The epochs of each class(movement and rest) were averaged and plotted against time, within the 1s to 5s window, the plot showed large correlating change in the average between both classes.

**

***Figure5.4 Plot of Extracted Close Epochs***

**

***Figure5.5 Power Spectral Density Plot for relax and close Epochs***

**

***Figure5.6 Percental Analysis Plot over Averaged Class Epochs***

**CHAPTER SIX**

**BCI TASK MOVEMENT AGAINST REST-CLASSIFIER**

* 1. **INTRODUCTION**

The aim of this project was to develop a system that decodes information on active human movement or intent of movement from EEG signals.

The EEG signals were acquired using the openBCI electrode cap, cyton board and blue-tooth dongle. The signals were parsed from the blue-tooth using the pyOpenBCI and pylsl library and the training signals were recorded using the LabRecorder software which made use of a laboratory communication protocol called the Lab Streaming Layer (LSL).

EEG data for movement and rest positions was acquired from six healthy volunteers. The marker cues were generated randomly so as to avoid adaptation.

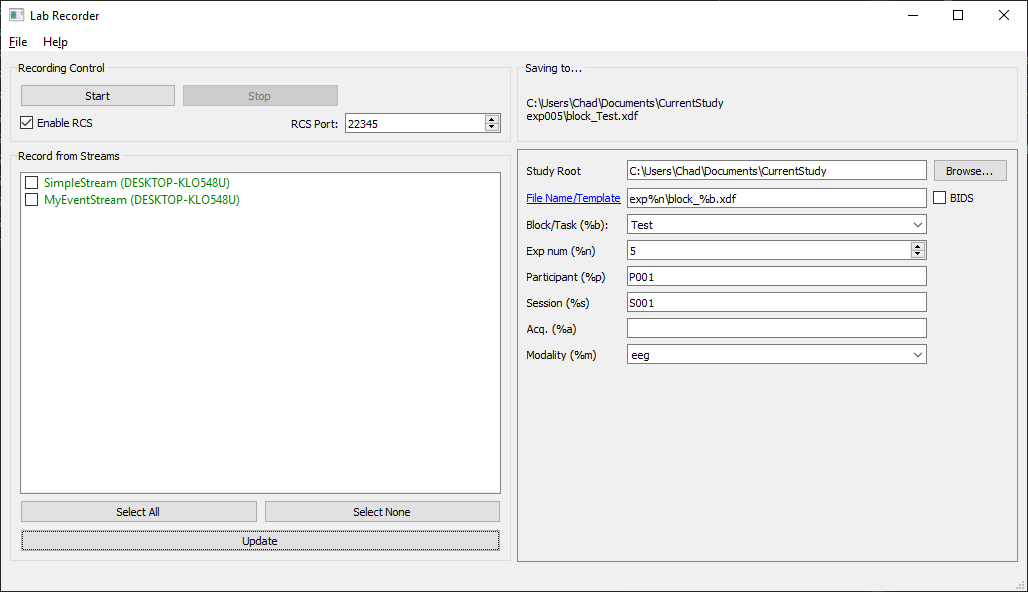
The acquired EEG signals were processed using the python based jupyter notebook, pyriemann library and sci-kit learn imputer. Online data-processing and classification was implemented using the neuropype suite pipeline designer and a python script for EEG data and marker streaming.

* 1. **DATA ACQUISITION AND PROTOCOLS**

Cues were displayed by a monitor using the python matplotlib library and passed as markers to the LabRecorder using the same script displaying the cues. The cue instructions included:

1. Calib-Begin: This cue notifies the volunteer that a data-acquisition session is starting.
2. Movement: This cue notifies the volunteer to move his or her right hands.
3. Rest: This cue notifies the volunteer to remain at ease, not moving or thinking of anything.
4. Calib-End: This cue notifies the volunteer that the started data-acquisition session is ended.

The LabRecorder receives markers via a LSL stream and EEG data from another LSL stream generated and fed by the openBCI. The codes for generating openBCI EEG data LSL stream can be found in Appendix



***Figure6.1 Lab Recorder Merging Streams for EEG Data Recording***

* 1. **SIGNAL PRE-PROCESSING**
     1. **Data-Set Merging**

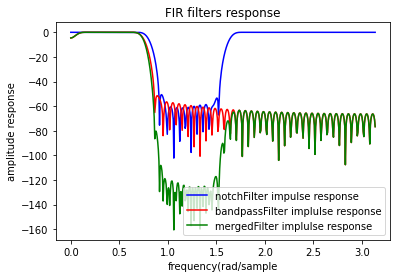
The EEG data acquired from the six subjects were merged into one multi-dimensional array having 204 instances as rows, with each instances having 8 columns as channels and each channel(column) containing 1275 samples corresponding to 5.1s of activity from the onset of a marker. The merged data-set array was converted to a pandas data-frame and a marker column was added to the data-frame and saved as a pickle file.

* + 1. **Merged Data-Set Merging**

The merged data-frame was imported in another jupyter note-book which was used to filter the data-set.

A digital notch filter kernel of length 101 with nyquist frequency 500Hz was used to attenuate signals within the frequency band 32Hz to 65Hz so as to attenuate power line interference noise. A digital band-pass filter kernel of length 100 with nyquist frequency of 500Hz was used to pass signals in the frequency band 1Hz to 30Hz so as to retain EEG data intelligent information found in the frequency domain.

The length of the resulting signal was 1474 samples. The filtered EEG data was in between the 205th sample and the 1275th sample.

**

***Figure6.2 Digital Filters Frequency Response***

* + 1. **Epoching Merged Data**

The filtered data had a length of 2,374 samples. The EEG data was found between the 205th sample and the 1,275th sample corresponding to 1,070 samples. The EEG data was extracted from the filtered data and further pre-processing was carried out on the merged and filtered EEG data-set.

* + 1. **Standardization of Epochs**

The epochs were standardized using the standard-deviation and mean of the samples in the each epoch. The standardized form for each sample in each epoch was stored in a data-frame and saved to a pickle file to be used later in feature-extraction. The formula for the standardization used is given below:

* 1. **FEATURE EXTRACTION**
     1. **Common Spatial Patterns (CSP)**

CSP is a supervised technique. The class to which each data-vector belongs in a training data-set is given. CSP finds spatial filters such that the variance of the filtered data from one class is maximized while the variance of the filtered data from the other class is minimized. The resulting feature vectors thus enhance the discriminability between the two classes. CSP has emerged as a popular filtering method for EEG BCIs because these BCIs rely on the power in a frequency band for control.

Since the variance of EEG signals filtered in a given frequency band corresponds to the power in this band, CSP essentially maximizes the discriminability of the features used in the BCI. We are given input data from trial i for class . Each is an N×T matrix, where N is the number of channels and T the number of samples in time per channel. We assume that the is centered and scaled.

The goal of CSP is to find M spatial filters, given by an N×M matrix W (each column is a spatial filter), that linearly transforms the input signals according to:

* 1. **CLASSIFICATION**
     1. **Logistic Regression**

Logistic Regression (also called Logit Regression) is commonly used to estimate the probability that an instance belongs to a particular class (e.g., what is the probability that this email is spam?). If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled “1”), or else it predicts that it does not (i.e., it belongs to the negative class, labeled “0”). This makes it a binary classifier.

A Logistic Regression model computes a weighted sum of the input features (plus a bias term), but instead of outputting the result directly like the Linear Regression model does, it outputs the logistic of this result. The logistic function is given by:

**CHAPTER SEVEN**

**SUMMARY, CONCLUSION AND RECOMMENDATION**

* 1. **SUMMARY OF WORK DONE**

During the industrial training, I worked on developing the control and instrumentation system for a robot to be used at Obafemi Awolowo University Teaching Hospital (OAUTHC) for upper-limb stroke rehabilitation, this robot is called PULSR (Platform for Upper Limb Stroke Rehabilitation). I worked on developing brain computer interfaces for decoding human-movement and intent of human movement from electroencephalogram (EEG) signals. Also, I worked on developing an electronic speed controller (ESC) for brush-less DC (BLDC) motors. I also developed an electronic solution for damaged three phased BLDC motors, using three hall-sensors for feedback with one hall sensor damaged. This was called the hall sensor emulator project. I also was in charge of supervising and coordinating other projects and also making preparations for exhibitions and events.

The PULSR robot had its mechanical frame-work and couplings ready alongside a preexisting non-automated control-box which failed intermittently due to using relays in place of open collector NPN transistors. I built another version of the control-box which does not fail intermittently due to replacement of the relays by open collector NPN transistor. I made the control box automated by creating an arduino firmware that communicates with a PC (Personal Computer) through a python script based in the PC to receive control instructions from the PC keyboard and in turn the arduino switches the open collector transistors base accordingly. The PC based python script also receives angular position data of the PULSR links from the arduino and implements a graphical representation of PULSR end-effector position and the workspace. I developed a information exchange discipline between the arduino and the PC python script to achieve all stated above. I also troubleshooted some of PULSR electrical parts due to some abnormal behavior and tracked the causes of such behavior to some damage ESC and a damaged hall sensor of the upper BLDC motor. This led to the development of an ESC and a hall sensor emulator. Images of PULSR and its graphical user interface are given in

I developed an ESC for driving BLDC motors using the hall sensor feedback signals of the BLDC motors, assuming that all hall sensors of the BLDC motor to be used are in good working condition. I firstly reviewed past works on similar topics using google, youtube and electronic-project example websites as sources. I then created a firmware and simulated the behavior of the firmware and designed circuit on proteus design suite. After simulation, a bill of material (BOM) was generated so as to order the needed parts from outside the state and some other parts from outside the continent. I audited all parts and implemented the ESC.

I carried out a BCI experiment to determine to what extent EEG signals of human movement or human intent of movement can be differentiated from EEG signals while the human body is in a state of rest (relaxed). The EEG data used was provided by the Cllinical Neurotechnology Team from Charitea University, Berlin for the jointly organized workshop on BCI experiment design. I computed and compared the power spectral density for the movement class (close) and the rest class (relax). I also computed and compared the percental change in averaged epochs for each class. The results indicated high level of discriminability of both classes.

Also, I developed a BCI system that classifies in between two classes of human activity, the first being movement or intent of movement and the second being rest. I employed filtering and standardization algorithms for pre-processing of EEG data-set which was acquired by myself from six subjects for both classes and then applied a CSP algorithm in computing features which were fed to a Logistic-Regression classifier. The accuracy of the system built was 81.2%. Images on data acquisition process is given in appendix

I also developed a hall sensor emulator for mimicking the behavior of a three phase BLDC motor that has all hall sensors working well in a three phase BLDC motor having two of its hall sensors working well and just one damaged.

* 1. **RELEVANCE OF EXPERIENCE TO COURSE OF STUDY**

**EEE203 & 204**

So as to develop an ESC for BLDC motors, there was the need to determine the power consumption of the motor coils whilst running, having one coil excited by a variable duty-cycle PWM signal of 1 KHz, another pulled to ground and the last floating, through the three-phase half-bridges.

The BLDC motor datasheet gave the line to line resistance of the BLDC motor coils and the line to line inductance of the BLDC motor coils, while also specifying its peak current if excited by a 1 KHz signal source. The line to line resistance and line to line inductance of the BLDC motor provided allowed the BLDC motor to be modelled as a first order RL circuit, excited by a square wave function having an off and on time of 0.5ms each, while using the line to line impedances provided by the datasheet for the resistor and inductor value. The current in one cycle of a 1 KHz square wave was computed from the model circuit and was confirmed not to exceed the peak continuous or impulse current of the BLDC motor.

Also the datasheet of the MOSFETs used in the half-bridges were used to determine the gate to source voltage that would drive the MOSFETs to saturation s the MOSFETS are to be used as switches and to determine the maximum drain-source current and voltage at saturation. The MOSFET used for the half-bridges were picked based on the ability of the drain-source current at saturation to handle the current requirement of the BLDC motor.

During the development of another control circuit for PULSR, previous knowledge of BJT transistors was used in computing the base resistance values that would turn on the BJT collector to emitter channel using a +5V input

**EEE305 &306**

The use of arduino UNO in the development of the ESC, the hall-sensor emulator for BLDC motors and PULSR firmware required that I had a good understanding of the architecture and working of the arduino UNO mother chip, atmega328p.

I used the datasheet of the atmega328p as my primary source of information on the architecture and limitations of the atmega328p chip.

First off, using the pre-requisite knowledge of 306 which shows that every form of computer has its own basic instruction sets, my first task was to understand the atmega328p chip instruction sets. Secondly, I took time to understand the digital circuitry, instruction set for manipulating and the limitations of, the digital modules that came with the atmega328p chip such as the USART module, the TIMER module, and its INTERRUPT modules, majorly amongst others.

In developing PULSR, I had to understand the digital circuitry for configuring the atmega328p USART module and also the instruction sets for accessing inputs to the digital circuitry, for communication between PULSR hardware and software.

In developing the ESC and the hall-sensor emulator, deep understanding if the atmega328p TIMER module and INTERRUPT module was needed. I interpreted the digital circuits for these modules and searched for instruction sets that help configure the TIMER and INTERRUPT modules.

**EEE309**

In the development of BCI systems, FIR (Finite Impulse Response) digital filters were used for band-passing and band-removal. The impulse response of this digital filters were calculated and returned by the python sci-kit library by just specifying the cut-off frequencies, type of filter and the filter length.

The knowledge of convolution from signals and systems courses was useful in convolving the impulse response of the digital filters and the raw EEG data to be further processed after the convolution which indicate filtering.

From signals and systems courses, convolution expands the length of input signals by the formula given below:

In the formula above, is the length of the impulse response (the digital-filter impulse response), is the length of the signal (EEG signal to be filtered) and is the length of the resulting convolved signal. Obviously, the resulting convolved signal has a larger length and fits the actual filtered EEG data in between segments of the digital-filter impulse response. This knowledge was used in visualizing the data and extracting only the filtered EEG data.

In the development of BCI systems, the knowledge of the representation of filters and domain transforms as matrixes helped in investigating the mathematics behind feature-extraction techniques and trouble-shooting code that implements the mathematics of the feature-extraction techniques when errors occurred.

In frequency analysis of EEG signals, computing power spectral densities for each class of the EEG signals, the knowledge and understanding of Fourier’s analysis which describes any time varying signal as a sum of sinusoids of different frequencies and amplitude was needed in understanding the Fast Fourier Transform(FFT) which was used in computing the power spectral density plot.

* 1. **CONCLUSION**

My being involved in research and development during the period of the industrial training helped in learning the right process to approaching research works, and the right approach towards literature review, information gathering, selection and filtering.

The introduction of the SIWES program into the undergraduate curriculum is a necessity as it helps students gain perspective on their engineering career opportunities. It also helps students relate acquired information from class and school practical sessions to uses in the engineering industry giving students insights into what is expected of them.

A2IR2 provides a good platform for students on industrial training attachment to undergo trainings that shapes them well for the engineering industry and not neglecting the business aspect of the industry. A2IR2 provides exposure to the nature, structure and organization of the engineering industry within Nigeria and in all other parts of the world while also ensuring its interns are trained on the latest required engineering skillset.

* 1. **RECOMMENDATIONS**

It is becoming increasingly apparent that it is only through an idea like SIWES that tertiary institutions can produce employable graduates to meet the needs of a technologically dynamic society and also compete on an international scale.

I would like to suggest that the Industrial Training Fund should liaise with companies to make arrangements for absorption of students into their field of specialization after undergoing a compulsory period of education on available areas of specialization and if there are no openings available companies should make it known to student long before commencement of the training. This would help students learn more about their field of interest, their options and help in their decision making.

I therefore suggest that this program should be continued for all undergraduates with the stipulated six months or more, so that student can get the adequate training required.

**APPENDIX**

**A** PULSR GUI/software code

1. import pyautogui
2. import pygame
3. import math
4. # import csv
5. # import os
6. from pulsr2API import \*
7. import random
9. pygame.init()
11. pulsr2=pulsr()
12. pulsr2.initialize\_communication(2000000,'COM17')
14. #get screen\_width and screen\_height
15. #and initialize windows
16. screen\_width,screen\_height=pyautogui.size()
17. starting\_dialog\_box\_size=(int(screen\_width/2.6),int(screen\_height/4))
18. starting\_dialog\_box=pygame.display.set\_mode(starting\_dialog\_box\_size)
19. background=pygame.display.set\_mode((screen\_width,screen\_height))
20. screen=pygame.display.set\_mode(starting\_dialog\_box\_size)
21. pygame.display.set\_caption('PULSR2.0')
22. oaufont = pygame.font.SysFont('Arial', 25)
23. oaufont2 = pygame.font.SysFont('Arial', 15)
24. oaufont3 = pygame.font.SysFont('Arial', 10)
25. meaningtext = oaufont.render('Platform for Upper Limb Stroke Rehabilitation', False, (0, 0, 0))
26. oautext = oaufont2.render('Obafemi Awolowo University, Ile-Ife', False, (0, 0, 0))
27. dialog\_message = oaufont3.render('PULSR2 starting up.........', False, (0, 0, 0))
28. clock=pygame.time.Clock()
30. background.fill((255,255,255))  #fill dialog box background with white
31. screen.blit(meaningtext,(10,60))
32. screen.blit(oautext,(135,90))
33. screen.blit(dialog\_message,(7,170))
34. pygame.display.update()
35. clock.tick(20)
36. time.sleep(1)
37. '''
38. code to display pulsr2 starting small dialog box
39. i.check communication status and report on dialog box
40. '''
42. #i.check pulsr2 communication status and report on dialog box
43. if pulsr2.check\_communication()==True:
44. dialog\_message = oaufont3.render('communication port active', False, (0, 0, 0))
45. background.fill((255,255,255))  #fill dialog box background with white
46. screen.blit(meaningtext,(10,60))
47. screen.blit(oautext,(135,90))
48. screen.blit(dialog\_message,(7,170))
49. pygame.display.update()
50. clock.tick(20)
51. time.sleep(1)
52. else:
53. dialog\_message = oaufont3.render('communication port not active', False, (0, 0, 0))
54. background.fill((255,255,255))  #fill dialog box background with white
55. # background.blit(image,(0,0))
56. screen.blit(meaningtext,(10,60))
57. screen.blit(oautext,(135,90))
58. screen.blit(dialog\_message,(7,170))
59. pygame.display.update()
60. clock.tick(20)
61. time.sleep(1)
62. pygame.quit()
63. quit()

66. '''
67. code to run motor and encoder readings tests and report on dialog box
68. i.send control-data to arduino and receive motor data from arduino
69. ...if succesful report communication test passed
70. ...if failed report communication test failed
71. ii.test motions and reported encoder readings for correct behaviour
72. '''
73. #i.send control-data to arduino and receive motor data from arduino
74. pulsr2.send\_command()
75. if pulsr2.motor\_data==str():
76. dialog\_message = oaufont3.render('communication test failed', False, (0, 0, 0))
77. background.fill((255,255,255))  #fill dialog box background with white
78. # background.blit(image,(0,0))
79. screen.blit(meaningtext,(10,60))
80. screen.blit(oautext,(135,90))
81. screen.blit(dialog\_message,(7,170))
82. pygame.display.update()
83. clock.tick(20)
84. time.sleep(1)
85. pygame.quit()
86. quit()
87. else:
88. dialog\_message = oaufont3.render('communication test passed', False, (0, 0, 0))
89. background.fill((255,255,255))  #fill dialog box background with white
90. # background.blit(image,(0,0))
91. screen.blit(meaningtext,(10,60))
92. screen.blit(oautext,(135,90))
93. screen.blit(dialog\_message,(7,170))
94. pygame.display.update()
95. clock.tick(20)
96. time.sleep(1)
97. #ii.test motions and reported encoder readings for correct behaviour
98. dialog\_message = oaufont3.render('testing motors............', False, (0, 0, 0))
99. background.fill((255,255,255))  #fill dialog box background with white
100. # background.blit(image,(0,0))
101. screen.blit(meaningtext,(10,60))
102. screen.blit(oautext,(135,90))
103. screen.blit(dialog\_message,(7,170))
104. pygame.display.update()
105. clock.tick(20)
106. time.sleep(1)
107. dummy=pulsr2.upper.angle
108. pulsr2.motion1(1)
109. pulsr2.update\_motor\_angles()
110. #check for current upper motor angle if it is greater or lesser than previous
111. #if correct set check variable to True else False
112. check=True
113. if check==True:
114. dummy=pulsr2.upper.angle
115. pulsr2.motion2(1)
116. pulsr2.update\_motor\_angles()
117. #check for current upper motor angle if it is greater or lesser than previous
118. #if correct set check variable to True else False
119. if check==True:
120. dummy=pulsr2.lower.angle
121. pulsr2.motion3(1)
122. pulsr2.update\_motor\_angles()
123. #check for current lower motor angle if it is greater or lesser than previous
124. #if correct set check variable to True else False
125. if check==True:
126. dummy=pulsr2.lower.angle
127. pulsr2.motion4(1)
128. pulsr2.update\_motor\_angles()
129. #check for current lower motor angle if it is greater or lesser than previous
130. #if correct close dialog\_box and continue
131. #else report motors/encoders test failed, quit pygame and program
132. time.sleep(5)
134. pulsr2.define\_geometry(22,43,43,65,42)
135. pulsr2.set\_origin()
137. '''
138. all tests succesful, so run main GUI
139. '''
140. black = (0,0,0)
141. white = (255,255,255)
142. cream = (255,250,170)
143. green=(255,0,255)
144. blue=(0,0,255)
145. red=(255,0,0)
146. grey=(222,222,222)
147. gui\_size=((int(screen\_width),int(screen\_height-250)))
148. gui=pygame.display.set\_mode(gui\_size)
149. gameDisplay=pygame.display.set\_mode((screen\_width,screen\_height))
150. pygame.display.set\_caption('PULSR2.0')
151. pygame.font.init()
152. pulsrfont = pygame.font.SysFont('Arial Bold', 100)
153. pulsrtext = pulsrfont.render('PULSR', False, white)
154. oaufont = pygame.font.SysFont('Arial', 15)
155. meaningtext = oaufont.render('Platform for Upper Limb Stroke Rehabilitation', False, white)
156. #meaningtext2 = oaufont.render('Stroke Rehabilitation', False, (0, 0, 0))
157. oautext = oaufont.render('Obafemi Awolowo University, Ile-Ife', False, white)
158. pygame.display.set\_caption('Image')
159. image = pygame.image.load(r'OpenBCI figure.jpg')
161. new\_x=0
162. new\_y=0
164. yOffset=24
165. xOffset=0
167. screen\_workspace\_height=screen\_height\*0.7
168. screen\_workspace\_width=screen\_height\*0.7
170. # old\_x=pulsr2.x\*(screen\_workspace\_width/50)
171. # old\_y=pulsr2.y\*(screen\_workspace\_height/50)
172. # pulsr2.get\_effector\_coordinate()
174. global running
175. running=True
177. def runSession():
178. global running
179. global new\_x
180. global new\_y
181. global old\_x
182. global old\_y
183. locationIterator = 0
184. gameDisplay.fill(black)
185. # gameDisplay.blit(image, (0, 0))
186. gui.blit(pulsrtext, (40, 40))
187. gui.blit(meaningtext, (30, 150))
188. gui.blit(oautext, (65, 180))
189. radius = screen\_workspace\_height\*0.35
190. screenCenter = [screen\_width \* 0.5, screen\_height \* 0.5]
191. movingCircleCenter = [0, 0]

194. while running:
195. # pulsr\_key\_controls()
196. for event in pygame.event.get():
197. if event.type == pygame.QUIT:
198. running = False
199. if event.type == pygame.KEYDOWN:
200. if event.key == pygame.K\_SPACE:
201. running = False
202. if event.key == pygame.K\_q:
203. pygame.quit()
204. quit()
205. if event.key == pygame.K\_r:
206. pulsr2.upper.angle=0
207. pulsr2.lower.angle=0
208. pulsr2.set\_origin()
209. if event.key == pygame.K\_w:
210. pulsr2.motion2(0)
211. if event.key == pygame.K\_s:
212. pulsr2.motion1(0)
213. if event.key == pygame.K\_a:
214. pulsr2.motion4(0)
215. if event.key == pygame.K\_d:
216. pulsr2.motion3(0)
217. if event.key == pygame.K\_m:
218. pulsr2.control\_data &= 191  #computercontrolled mode
219. pulsr2.send\_command()
220. if event.key == pygame.K\_u:
221. pulsr2.control\_data |= 64   #user controlled mode
222. pulsr2.send\_command()
223. if event.key == pygame.K\_c:
224. pulsr2.control\_data |= 32 #computer controlled mode move in circle
225. pulsr2.send\_command()
226. pulsr2.circleMode=True
227. if event.key == pygame.K\_x:
228. if pulsr2.circleMode==True:
229. pulsr2.disable\_motions()
230. pulsr2.circleMode=False
231. pulsr2.control\_data &= 223 #computer controlled mode move according to pre-defined keys direction
232. pulsr2.send\_command()
233. if event.key == pygame.K\_t:
234. pulsr2.disable\_upper\_motor()    #disable upper motor
235. if event.key == pygame.K\_g:
236. pulsr2.enable\_upper\_motor() #enable upper motor
237. if event.key == pygame.K\_y:
238. pulsr2.disable\_lower\_motor()    #disable lower motor
239. if event.key == pygame.K\_h:
240. pulsr2.enable\_lower\_motor() #enable lower motor
241. # print((pulsr2.control\_data&128)>>7,(pulsr2.control\_data&64)>>6,(pulsr2.control\_data&32)>>5,(pulsr2.control\_data&16)>>4,(pulsr2.control\_data&8)>>3,(pulsr2.control\_data&4)>>2,(pulsr2.control\_data&2)>>1,(pulsr2.control\_data&1))
243. workspaceBorder=pygame.draw.rect(gameDisplay,white,pygame.Rect(screenCenter[0]-(screen\_workspace\_height/2),screenCenter[1]-(screen\_workspace\_height/2),screen\_workspace\_height+10,screen\_workspace\_height+10))
245. angleInDegrees = (locationIterator/250.0)\*360
246. oldAngleInDegrees = ((locationIterator-1)/250.0)\*360
248. #compute current and old position angle in radians
249. currentTheta=math.radians(angleInDegrees)
250. oldTheta = math.radians(oldAngleInDegrees)
252. #compute old position cartesian coordinates
253. movingCircleCenter[0] = screenCenter[0]+(radius\*math.cos(oldTheta)) #x-coordinates on screen for center
254. movingCircleCenter[1] = screenCenter[1]-(radius\*math.sin(oldTheta)) #y-coordinates on screen for center
256. redrawGray = pygame.draw.circle(gameDisplay, black, (screenCenter[0],screenCenter[1]), radius, width=10)    #draw circular motion path
257. clearDot = pygame.draw.circle(gameDisplay, black, (movingCircleCenter[0], movingCircleCenter[1]), 3, width=3) #clear current position
259. #compute current position coordinates
260. movingCircleCenter[0] = screenCenter[0] + (radius \* math.cos(currentTheta))
261. movingCircleCenter[1] = screenCenter[1] - (radius \* math.sin(currentTheta))
263. newDot = pygame.draw.circle(gameDisplay, white, (movingCircleCenter[0],movingCircleCenter[1]), 14, width=14) #draw current position
264. # pygame.draw.arc(gameDisplay, red, [(screen\_width\*0.5)-radius, (screen\_height\*0.5)-radius, radius\*2, radius\*2], 0, currentTheta, width=10) #draw red arc from origin to current position
266. # workspace=pygame.draw.rect(gameDisplay,black,pygame.Rect(screen\_workspace\_width,screen\_workspace\_width,screen\_workspace\_height,screen\_workspace\_height))
268. old\_x=new\_x
269. old\_y=new\_y
270. clearOldCoordinate=pygame.draw.circle(gameDisplay, black, (screenCenter[0]-old\_y, screenCenter[1]-old\_x), 3, width=3)
272. pulsr2.send\_command()
273. # pulsr2.upper.angle=random.randint(1,5)\*10
274. # pulsr2.lower.angle=random.randint(1,5)\*10
275. pulsr2.update\_motor\_angles()
276. pulsr2.get\_effector\_coordinate()
277. global pointColor
278. new\_x=(pulsr2.x+xOffset)\*(screen\_workspace\_width/61)
279. new\_y=(pulsr2.y-yOffset)\*(screen\_workspace\_height/61)
280. if abs(math.sqrt((new\_x\*\*2)+(new\_y\*\*2))-radius)<=10:
281. if abs((screenCenter[0]-new\_y)-movingCircleCenter[0])<=50 and abs((screenCenter[1]-new\_x)-movingCircleCenter[1])<=50:
282. pointColor=blue
283. else:
284. pointColor=red
285. else:
286. pointColor=black
287. drawNewCoordinate=pygame.draw.circle(gameDisplay, pointColor, (screenCenter[0]-new\_y, screenCenter[1]-new\_x), 4, width=4)
288. drawPathLine=pygame.draw.line(gameDisplay,black,(screenCenter[0]-old\_y, screenCenter[1]-old\_x), (screenCenter[0]-new\_y, screenCenter[1]-new\_x), width=2)
290. # if new\_x==250:
291. #   new\_x=0
292. # else:
293. #   new\_x+=1
295. # if new\_y==250:
296. #   new\_y=0
297. # else:
298. #   new\_y+=1
300. '''
301. #     ACTIVITIES PER LOOP
302. '''

305. # i. retrieve the angles of the encoders
306. #compute theta from angle's 0 and 1 as rotaryTheta
308. # ii. check through the array of 250 angles to determine the closest
309. #compute closestTheta between currentTheta and rotaryTheta
311. # iii. determine the difference between the angles
312. #determine difference between closestTheta and rotaryTheta
314. # iv. move the motors closer to the angle
315. #move rotaryTheta to closestTheta
317. # v. advance the iterator to take on the value closest to the index of the current location in the location array

320. #vi. Advance the arc of the actual movement

323. if locationIterator==250:
324. locationIterator=0
325. else:
326. locationIterator+=1
328. pygame.display.update()
329. clock.tick(20)
331. return
333. def waitForKey():
334. global running
335. while running == False:
336. events = pygame.event.get()
337. for event in events:
338. if event.type == pygame.KEYDOWN:
339. if event.key == pygame.K\_SPACE:
340. running = True
341. if event.key == pygame.K\_q:
342. pygame.quit()
343. quit()
344. return


348. while True:
349. if running == False:
350. print("running Wait function")
351. waitForKey()
352. if running == True:
353. print("running Rehab function")
354. runSession()

**B** ESC firmware code

1. void phaseUint();
2. void phaseVint();
3. void phaseWint();
5. const int pwmFrequency = 50;  //in Hz
6. const int pwmPeriod = 20; //in miliseconds
7. const int pwmCycles = 24;
8. #define xtal 16000000
10. #define pinHU 7 //PCINT[23]
11. #define pinHV 6 //PCINT[22]
12. #define pinHW 5 //PCINT[21]
14. #define pinHighU 11 //PWM OC2A
15. #define pinHighV 10 //PWM OC1B
16. #define pinHighW 9  //PWM OC1A
17. #define pinLowU 4
18. #define pinLowV 3
19. #define pinLowW 2
20. #define pinSDu 13
21. #define pinSDv 12
22. #define pinSDw 8

25. #define hallThreshold 7
26. #define AU A2
27. #define AV A1
28. #define AW A0
30. uint8\_t nextStep=1;
31. uint8\_t hallSensors;
33. void sensorlessModeInterrupt();
34. void sensoredModeInterrupt();
36. ISR (TIMER2\_COMPA\_vect){
37. digitalWrite(pinLowU,!digitalRead(pinHighU));
38. }
39. ISR (TIMER1\_COMPB\_vect){
40. digitalWrite(pinLowV,!digitalRead(pinHighV));
41. }
42. ISR (TIMER1\_COMPA\_vect){
43. digitalWrite(pinLowW,!digitalRead(pinHighW));
44. }
45. ISR (PCINT2\_vect){
46. digitalWrite(A0,HIGH);
47. delay(500);
48. sensorlessModeInterrupt();
49. }
50. ISR (TIMER0\_COMPA\_vect){
51. sensoredModeInterrupt();
52. }
54. void setup() {
55. // put your setup code here, to run once:
56. pinMode(AU,INPUT);pinMode(AV,INPUT);pinMode(AW,INPUT);
57. //set input/output directions of PORTD and PORTB for hall sensor inputs and ESC control outputs
58. DDRB=B00111111;  //pins 13,12,11,10,9,8 --> OUTPUTS
59. DDRD=B00011110;  //pins 7,6,5,0 --> INPUTS, pins 4,3,2,1 --> OUTPUTS
61. PCICR |= \_BV(PCIE2);  //enable PCINT23....16
62. PCMSK2 = B11100000; //enable interrupts for PCINT23...21 alone for Hall sensors inputs on pin 7,6,5 named pinHU,pinHV,pinHW respectively
64. TCCR0A = \_BV(WGM01) | \_BV(CS00);
65. OCR0A=9;
66. TIMSK0 = \_BV(OCIE0A);
67. TCCR0A &= ~\_BV(COM0A1) | ~\_BV(COM0A0) | ~\_BV(COM0B1) | ~\_BV(COM0B0);
69. /\*-----------SET OC1A(PHASE W HIGH) AND OC1B(PHASE V HIGH) TO 50Hz/50%DutyCycle PWM----------\*/
70. TCCR1A=0;
71. TCCR1B=0;
72. TCNT1=0;
73. TCCR1B = \_BV(WGM12) | \_BV(CS11);
74. ICR1 = int(xtal/(2\*8\*pwmFrequency));
75. OCR1A = int(xtal/(2\*8\*pwmFrequency));
76. OCR1B = int(xtal/(2\*8\*pwmFrequency));
78. TCCR2A = 0;
79. TCCR2B = 0;
80. TCNT2 = 0;
81. TCCR2A =  \_BV(WGM21);
82. TCCR2B = \_BV(CS22) | \_BV(CS21) | \_BV(CS20);
83. //  TIMSK2 = \_BV(OCIE2A);
84. OCR2A=157;
86. sei();
88. //  digitalWrite(pinLowU,HIGH);digitalWrite(pinLowV,HIGH);digitalWrite(pinLowW,HIGH);
89. //  digitalWrite(pinSDu,LOW);digitalWrite(pinSDv,LOW);digitalWrite(pinSDw,LOW);
90. digitalWrite(pinSDu,LOW);
91. digitalWrite(pinSDv,LOW);
92. digitalWrite(pinSDw,LOW);
93. sensoredModeInterrupt();
94. }
96. void loop() {// put your main code here, to run repeatedly:
97. switch(nextStep){
98. case 0://UV
99. //make pwm interrupt and pwm active on highU
100. TIMSK2 |= \_BV(OCIE2A);
101. TCCR2A |= \_BV(COM2A0);
102. //disable pwm interrupt and make pwm output 0 on highV so as to be able make lowV active
103. TIMSK1 &= ~\_BV(OCIE1B);
104. TCCR1A &= ~\_BV(COM1B0);
105. //disable pwm interrupt and make pwm output 0 on highW so as to be able make W float
106. TIMSK1 &= ~\_BV(OCIE1A);
107. TCCR1A &= ~\_BV(COM1A0);
109. digitalWrite(pinHighV,LOW);
110. digitalWrite(pinLowV,HIGH);
111. digitalWrite(pinHighW,LOW);
112. digitalWrite(pinLowW,LOW);
113. break;
114. case 1://UW
115. //make pwm interrupt and pwm active on highU
116. TIMSK2 |= \_BV(OCIE2A);
117. TCCR2A |= \_BV(COM2A0);
118. //disable pwm interrupt and make pwm output 0 on highV so as to be able make V float
119. TIMSK1 &= ~\_BV(OCIE1B);
120. TCCR1A &= ~\_BV(COM1B0);
121. //disable pwm interrupt and make pwm output 0 on highW so as to be able make lowW active
122. TIMSK1 &= ~\_BV(OCIE1A);
123. TCCR1A &= ~\_BV(COM1A0);
125. digitalWrite(pinHighW,LOW);
126. digitalWrite(pinLowW,HIGH);
127. digitalWrite(pinHighV,LOW);
128. digitalWrite(pinLowV,LOW);
129. break;
130. case 2://VW
131. //disable pwm interrupt and make pwm output 0 on highU so as to be able make U float
132. TIMSK2 &= ~\_BV(OCIE2A);
133. TCCR2A &= ~\_BV(COM2A0);
134. //make pwm interrupt and pwm active on highV
135. TIMSK1 |= \_BV(OCIE1B);
136. TCCR1A |= \_BV(COM1B0);
137. //disable pwm interrupt and make pwm output 0 on highW so as to be able make lowW active
138. TIMSK1 &= ~\_BV(OCIE1A);
139. TCCR1A &= ~\_BV(COM1A0);
141. digitalWrite(pinHighW,LOW);
142. digitalWrite(pinLowW,HIGH);
143. digitalWrite(pinHighU,LOW);
144. digitalWrite(pinLowU,LOW);
145. break;
146. case 3://VU
147. //disable pwm interrupt and make pwm output 0 on highU so as to be able make lowU active
148. TIMSK2 &= ~\_BV(OCIE2A);
149. TCCR2A &= ~\_BV(COM2A0);
150. //make pwm interrupt and pwm active on highV
151. TIMSK1 |= \_BV(OCIE1B);
152. TCCR1A |= \_BV(COM1B0);
153. //disable pwm interrupt and make pwm output 0 on highW so as to be able make W float
154. TIMSK1 &= ~\_BV(OCIE1A);
155. TCCR1A &= ~\_BV(COM1A0);
157. digitalWrite(pinHighU,LOW);
158. digitalWrite(pinLowU,HIGH);
159. digitalWrite(pinHighW,LOW);
160. digitalWrite(pinLowW,LOW);
161. break;
162. case 4://WU
163. //disable pwm interrupt and make pwm output 0 on highU so as to be able make lowU active
164. TIMSK2 &= ~\_BV(OCIE2A);
165. TCCR2A &= ~\_BV(COM2A0);
166. //disable pwm interrupt and make pwm output 0 on highV so as to be able make V float
167. TIMSK1 &= ~\_BV(OCIE1B);
168. TCCR1A &= ~\_BV(COM1B0);
169. //make pwm interrupt and pwm active on highW
170. TIMSK1 |= \_BV(OCIE1A);
171. TCCR1A |= \_BV(COM1A0);
173. digitalWrite(pinHighU,LOW);
174. digitalWrite(pinLowU,HIGH);
175. digitalWrite(pinHighV,LOW);
176. digitalWrite(pinLowV,LOW);
177. break;
178. case 5://WV
179. //disable pwm interrupt and make pwm output 0 on highU so as to be able make U float
180. TIMSK2 &= ~\_BV(OCIE2A);
181. TCCR2A &= ~\_BV(COM2A0);
182. //disable pwm interrupt and make pwm output 0 on highV so as to be able make lowV active
183. TIMSK1 &= ~\_BV(OCIE1B);
184. TCCR1A &= ~\_BV(COM1B0);
185. //make pwm interrupt and pwm active on highW
186. TIMSK1 |= \_BV(OCIE1A);
187. TCCR1A |= \_BV(COM1A0);
189. digitalWrite(pinHighV,LOW);
190. digitalWrite(pinLowV,HIGH);
191. digitalWrite(pinHighU,LOW);
192. digitalWrite(pinLowU,LOW);
193. break;
194. }
195. }
197. void sensorlessModeInterrupt(){
198. cli();
199. //  //disable pwm interrupt and make pwm output 0 on highU so as to be able make lowU active
200. TIMSK2 &= ~\_BV(OCIE2A);
201. TCCR2A &= ~\_BV(COM2A0);
202. //disable pwm interrupt and make pwm output 0 on highV so as to be able make lowV active
203. TIMSK1 &= ~\_BV(OCIE1B);
204. TCCR1A &= ~\_BV(COM1B1);
205. //disable pwm interrupt and make pwm output 0 on highW so as to be able make W float
206. TIMSK1 &= ~\_BV(OCIE1A);
207. TCCR1A &= ~\_BV(COM1A1);
208. digitalWrite(pinHighU,LOW);digitalWrite(pinHighV,LOW);digitalWrite(pinHighW,LOW);
209. digitalWrite(pinLowU,LOW);digitalWrite(pinLowV,LOW);digitalWrite(pinLowW,LOW);
210. hallSensors=(B00000111)&((digitalRead(pinHW)<<2) | (digitalRead(pinHU)<<1) | (digitalRead(pinHV)));
211. switch(hallSensors){
212. case 1:
213. nextStep=0;
214. break;
215. case 5:
216. nextStep=1;
217. break;
218. case 4:
219. nextStep=2;
220. break;
221. case 6:
222. nextStep=3;
223. break;
224. case 2:
225. nextStep=4;
226. break;
227. case 3:
228. nextStep=5;
229. break;
230. }
231. sei();
232. }
233. void sensoredModeInterrupt(){
234. cli();
235. int u=analogRead(AU);int v=analogRead(AV);int w=analogRead(AW);
236. bool hu,hv,hw;
237. if(u>hallThreshold)hu=1;else hu=0;
238. if(v>hallThreshold)hv=1;else hv=0;
239. if(w>hallThreshold)hw=1;else hw=0;
240. hallSensors=(B00000111)&((hw<<2) | (hu<<1) | (hv));
241. switch(hallSensors){
242. case 1:
243. nextStep=0;
244. break;
245. case 5:
246. nextStep=1;
247. break;
248. case 4:
249. nextStep=2;
250. break;
251. case 6:
252. nextStep=3;
253. break;
254. case 2:
255. nextStep=4;
256. break;
257. case 3:
258. nextStep=5;
259. break;
260. }
261. sei();
262. }

**C** Hall sensor emulator firmware code

1. #define pinHU 7 //PCINT[23]
2. #define pinHV 6 //PCINT[22]
3. #define pinHW 5 //PCINT[21]
5. #define emulatedHV 4
7. long TimeUtoV=0;
8. long TimeUtoW=0;
10. long counter=0;
11. bool counterIncrement=false;
12. int diff;
14. bool event1=0;
15. bool event2=0;
16. bool event3=0;
18. bool lastHU=0;
19. bool lastHW=0;

22. ISR(PCINT2\_vect){
23. if(lastHU==!digitalRead(pinHU)){
24. //    counter=0;
25. counterIncrement=true;
26. }else if(lastHW==!digitalRead(pinHW)){
27. TimeUtoW=counter;
28. counterIncrement=false;
29. counter=0;
30. TimeUtoV=TimeUtoW/2;
31. }
32. }
34. void setup() {
35. // put your setup code here, to run once:
36. Serial.begin(9600);
37. while(!Serial);
38. Serial.write(1);
39. pinMode(pinHU,INPUT);
40. pinMode(pinHV,INPUT);
41. pinMode(pinHW,INPUT);
42. pinMode(emulatedHV,OUTPUT);
43. pinMode(9,OUTPUT);
44. digitalWrite(9,LOW);
46. PCICR |= \_BV(PCIE2);  //enable PCINT23....16
47. PCMSK2 = B10100000; //enable interrupts for PCINT23...21 alone for Hall sensors inputs on pin 7,6,5 named pinHU,pinHV,pinHW respectively
49. lastHU=digitalRead(pinHU);
50. lastHW=digitalRead(pinHW);
52. sei();
53. }
55. void loop() {
56. // put your main code here, to run repeatedly:
57. if(counterIncrement==true)counter++;
58. lastHU=digitalRead(pinHU);
59. lastHW=digitalRead(pinHW);
60. diff=abs(counter-TimeUtoV);
61. if(counter>=TimeUtoV)digitalWrite(emulatedHV,digitalRead(pinHU));
62. }

**REFERENCES**

Bonci, A.; Fiori, S.; Higashi, H.; Tanaka, T.; Verdini, F. (2021). ‘An Introductory Tutorial on Brain– Computer Interfaces and Their Applications’. *Electronics*, *10*, 560. https://doi.org/10.3390/electronics 10050560

Dave Hitchins. (2016). ‘What is the difference between brushed and brushless DC motors’.*<https://www.parvalux.com/what-is-the-difference-between-brushed-and-brushless-dc-motors/#:~:text=A%20brushed%20DC%20motor%20has,with%20a%20rotating%20armature%20inside.&text=The%20rotating%20armature%20contains%20an,is%20applied%20to%20the%20armature>* 3 January, 2022.

DigiKey.(2019). ‘How to power and control brush-less DC motors’. <<https://www.digikey.com/en/articles/how-to-power-and-control-brushless-dc-motors>> 7 December, 2016.

Iturrate, I.; Antelis, J.; Kubler, A.; Minguez, J.(2009). ‘A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation’. IEEE Trans. Robot. 25, 614–627.

Krakauer JW.(2006). ‘Motor learning: its relevance to stroke recovery and neurorehabilitation’. Curr Opin Neurol 2006;19(1):84–90.

LaFleur, K.; Cassady, K.; Doud, A.; Shades, K.; Rogin, E.; He, B.(2013). ‘Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface’. J. Neural Eng. 10, 046003.

Stefano Maserio, Marco Iosa, Giovanni Morone, Damiano Zanotto, et al. ‘The value of robotic systems in stroke rehabilitation’. Expert Review of Medical Devices, 2014.

Rajesh P.N. Rao.(2013). Brain-Computer Interfacing, An Introduction, 1st ed. Cambridge University Press, New York city.

Rebsamen, B.; Guan, C.; Zhang, H.; Wang, C.; Teo, C.; Ang, M.(2010). ‘A brain controlled wheelchair to navigate in familiar environments’. IEEE Trans. Neural Syst. Rehabil. Eng. 18, 590–598.

Riener R, Lu¨nenburger L, Jezernik S, et al. ‘Patient-cooperative strategies for robot-aided treadmill training: first experimental results’. IEEE Trans Neural Syst Rehabil Eng 2005; 13(3):380-94

Schermer, M. (2011). ‘Ethical issues in deep brain stimulation’. *Front. Integr. Neurosci.* *5*, 17.

Schmidt H, Werner C, Bernhardt R, et al. ‘Gait rehabilitation machines based on programmable footplates’. J Neuroeng Rehabil 2007;4:2

Tanaka, K.; Matsunaga, K.; Wang, H.(2005). ‘Electroencephalogram-based control of an electric wheelchair’. IEEE Trans. Robot. 21, 762–766.

Vlek, R.J.; Steines, D.; Szibbo, D.; Kübler, A.; Schneider, M.J.; Haselager, P.; Nijboer, F.(2012). ‘Ethical issues in brain-computer interface research, development, and dissemination’. *J.Neurol. Phys. Therapy,* *36*, 94–99.

Wagner TH, Lo AC, Peduzzi P, et al. An economic analysis of robot-assisted therapy for long-term upper-limb impairment after stroke. Stroke 2011;42(9):2630-2

Winstein CJ, Stein J, Arena R, Bates B, Cherney LR, Cramer SC, et al. (2016). ‘Guidelines for Adult Stroke Rehabilitation and Recovery: A Guideline for Healthcare Professionals From the American Heart Association/American Stroke Association’. *Stroke*. **47** (6): e98–e169

Xu, B.; Peng, S.; Song, A.; Yang, R.; Pan, L.(2011). ‘Robot-aided upper-limb rehabilitation based on motor imagery EEG’. Int. J. Adv. Robot.Syst. 8.