CHAPTER 1 : INTRODUCTION

1.1 Introduction

The most convenient hardware interface for a computer would be one which directly detects brain signals or thoughts and converts them into appropriate input signals for the computer. This is possible, however this requires some training for the user and is complex to implement and requires a more sophisticated EEG with ~32 electrodes. These "sophisticated" EEG sensors tend to be 3 or more times more expensive than simpler ones. Since we wanted to make a conduit to make computers more accessible it would make more sense for the interface to be affordable and available to the masses. A simple EEG on the other hand has re limitations making it useful for mainly binary(yes/no) actions. This meant that there was a requirement for another sensor to get the complete idea of what a user wanted to do.

So it was decided that the hardware interface will implement only the mouse in hardware since the keyboard is too complicated to handle with 1 or 2 sensors without making too complicated to use just like a real keyboard. A simple EEG which could handle yes/no questions would be ideal to record mouse clicks and a second sensor, an accelerometer would through the tilt angle decide in which direction and how fast the mouse cursor would move. This 2 sensor configuration reduced the cost to a 3rd. As far as ease of use is concerned this system has been designed for patients, and these patients shall be able to adapt to the system with some training sessions.

The system has been made to be almost as accurate as a mouse and keyboard this could vastly improve the lives the people who have lost dexterity or complete use their hands and need constant supervision. Since through the interface the affected people could firstly communicate a lot more easily with the outside world and with further development in AI the very computer they use could work as there personal assistant.

1.2 Aim And Objectives

We aim to make the operation of a computer completely hands free for the paralyzed patient. This system replaces the need of a physical mouse setup with an EEG headset mounted on the head of the patient and replacing the physical keyboard with a keyboard application. Also an application layout for the keyboard that is optimized for quick and error free typing using a word prediction engine. The paralyzed patient will be able to manage the system easily with minimal effort and type using the keyboard application.

1.3 Organization of Report

First chapter of the report contains Introduction to this project which explains why this project is undertaken and the concepts associated with it. Second chapter of the report consists of Literature survey where key for improvements in existing technologies is determined. Chapter 3 explains problem statement of this project, Chapter 4 deals with designing of the system. Chapter 5 is divided into two phases, Module-I deals with the hardware components of the system emulating a mouse and module-II deals with the software component which is the custom virtual keyboard. Chapter 7 discusses results in details. Rest of the report deals with conclusion and future scope.

CHAPTER 2 : LITERATURE REVIEW

The initial idea for the system was to use solely EEG as the sensor for the input signal for the computer. However as pointed by [1] the EEG being a non-intrusive sensor is riddled by low signal-noise ratio and the signal quality depends on the quantity of electrodes and their proximity to the source of the signal, namely the brain. A typical EEG headset gives P300 signals i.e. alpha(attention levels),beta(meditation levels) waves taken from the brain and eye blinks. Since when using a computer the user will hardly be in a meditative state. So the signals which could be used were attention and eyeblinks. The study mentioned in [2] and [3] used attention levels as a way to decide the next action in a game. Both of the studies had problems with the usability of attention signal, since it tended to frequently fluctuate. [2] used a dynamic algorithm which adjusted the acceptable attention level threshold as per the average attention level in the recent past. It[2] also proposes a t9 keyboard (a 9 button layout in old mobile phones) as a simpler version of a conventional keyboard with over 200 keys . Another study established an estimate of the attention levels with noise over time for an average humans along with the same for eye blinks and found that the fluctuations were too high for attention signals while being a lot more reliable for eye blinks.So eye blinks were a selected as the primary signal from EEG.

A similar interface system as the one proposed was discussed in [4] which used a combination of EEG and a gyroscope. The team initially considered other alternatives such as tongue-based mouse movement, which involved a magnet being attached to the tongue. Based on the movement of tongue there was a change in the magnetic field through which the desired mouse movement could be deciphered. This system was convenient and easy to use, however it was a little too intrusive. Also considering the possibility of facial paralysis this system may not prove to be good enough. So it was rejected in favour EEG-gyroscope system which was less intrusive but slightly more difficult to use because the user had to continuously move their head to keep the mouse moving which could be tiresome to do over a long period. This lead us to use accelerometer which could detect tilt and hence enable mouse movement through tilting the head and keeping it so till the cursor reaches destination which is less tiresome.

Another product manages to work with a computer using a single input. So potentially only an EEG could be used to run a computer. The name of this product is Tilvus [4]. It works by constantly shifting over selectable options, and once the desired option is highlighted the user has to trigger the input, say by eye blink which could be detected by an EEG.

But for the system to be usable the highlighting of options had to be fairly slow which in turn made the overall operations slow and rather tedious and hence this idea was rejected as being too slow.

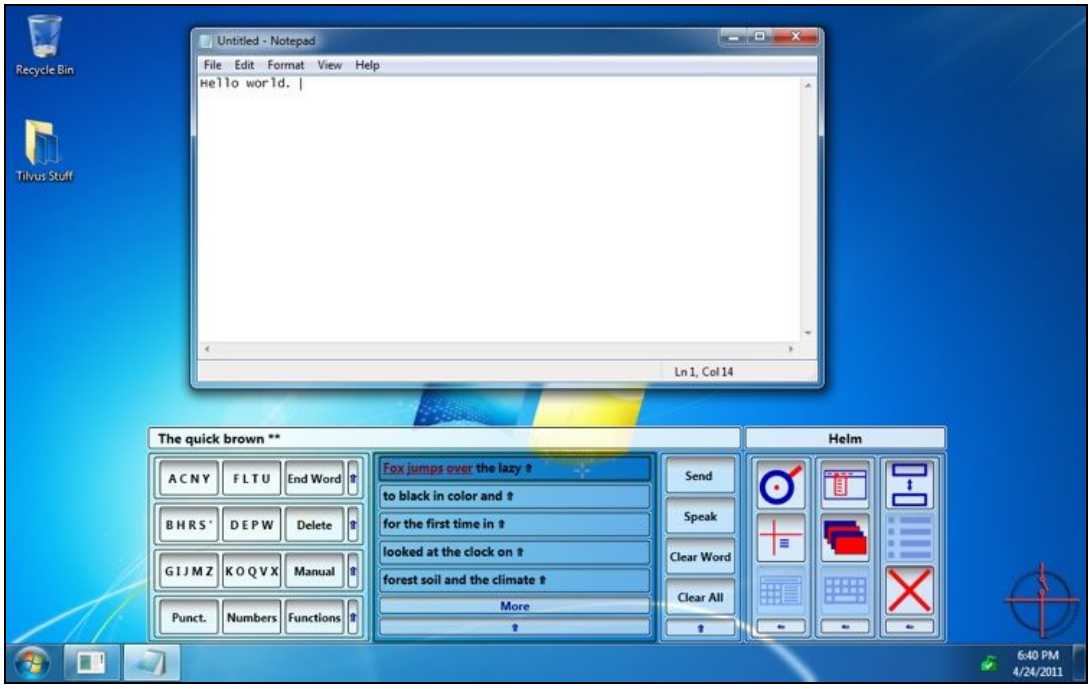

Fig. 2.1 *Tilvus Program*

A new system design in [6] used HC-05 as bluetooth device for interfacing arduino and EEG. The system worked to move the wheelchair for the paraplegic using EEG.The authors of this papers cited HC-05 as a reliable and well supported device for bluetooth interfacing with arduino.

There are also discussions about the concepts and conventions of morse code and how it could be adapted to be used with arduino [9]. The morse code is based on a binary search tree where the path taken to reach the required letter decides its respective morse code as shown below [16].

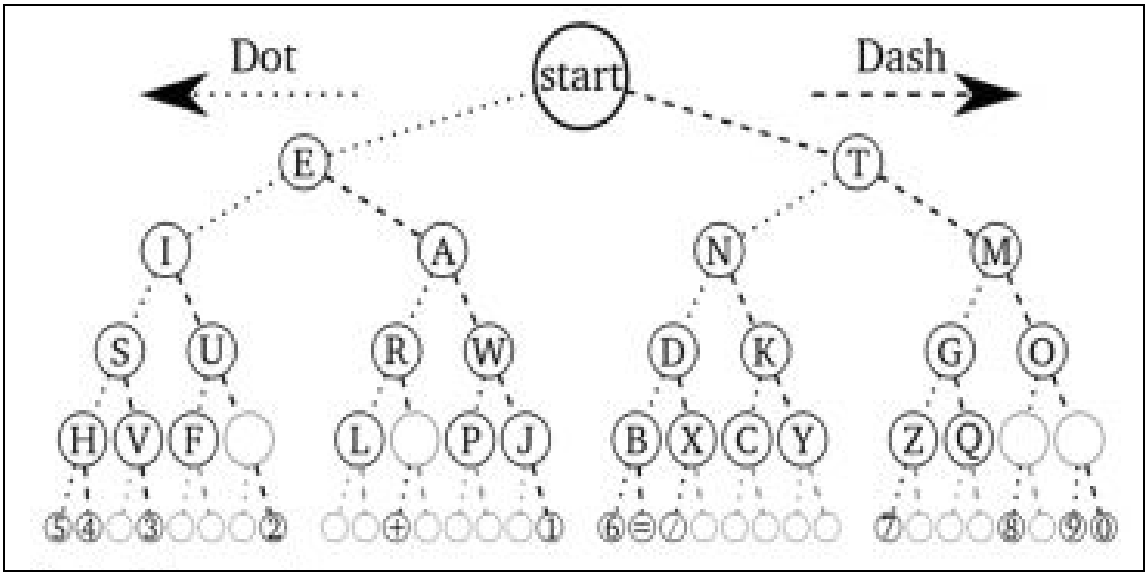

Fig. 2.2 *International Morse Code Letters and Numerals*

Dots and dashes could be decided on the basis of the duration for which the user blinks or keeps their eyes closed. According to study in [10] morse code typing should produce about 6-7 words per minute and that voluntary eye blinks tend to be about 2 times longer than involuntary ones.

Now there are alternatives to EEG for detecting eye blinks, mainly through image processing. We can perform image processing using an arduino and openCV library. Using image processing could allow us to detect winks which are almost always voluntary and could greatly improve the speed of typing if used in morse code, where left and right blinks are used to traverse the BST instead of dots and dashes based on blink duration.This would also enable easy implementation of left and right click.

Though this might seem like a better solution than an EEG but from the system mentioned in [7] we numerous limitations crop up. Such as the fact that room should be well-lit. The angle of the face is very important and so at oblique angles the system could stop working altogether. having to maintain a constant posture for a system work is difficult for the user and also IP tends to take

a lot more processing power than using an EEG. The combination of lacking reliability and requiring more processing power meant that we had to stick with the EEG for eye blinks.

To ensure the registering of deliberate blinks we needed the reliable attention values from the user, the kind of values which would help differentiate voluntary blinks from involuntary ones.[14] and [15] explore the possibility of using artificial neural networks to enable learning and dynamically adjust the threshold value for attention. This nearly eliminates all errors after a certain period of time. However the network has to adapt for each user separately so the reliable usage might take some time.

CHAPTER 3: PROBLEM STATEMENT AND SCOPE

3.1 Problem Statement

The difficulty of operating a computer for a paralyzed patient or a amputee is tremendous. They have to be over dependent on the helpers or assistant to complete basic tasks. The solution here is to make a system that will help the paralyzed patients to do computer operation without any help from any other person. There are systems that might solve such issues but the technologies are either too costly or they are not available easily.

3.2 Scope

The system that we have implemented has the following scope limitations

- Basic control of a PC / Laptop- The patient has been enabled to basic tasks like opening a file, browsing through file systems, open applications etc.
- Hands-free mouse navigation function- The patient can easily achieve the above mentioned tasks by using the handsfree mouse and clicking on any target area of choice.
- Easy typing application interface with rudimentary text prediction - To make the typing more easier for the patient a novice keyboard has been developed on basis of T9 keyboard that shall help the patients to type with relative ease in order to communicate with the outside world.

CHAPTER 4 : ANALYSIS AND DESIGN

4.1 Existing System

The existing systems in market are very in number and very advanced in technology to be made easily available to common man. Most of these systems use high end technologies which makes it difficult for any common man to use it. One such system is Tilvus [4] as shown *fig. 2.1* The tivus program is very close to our system in terms of design and application, but on fronts of budget our system proves to be a more economical. There are other products that help in hands free computing but most of these are not specifically designed for paralysed patients and hence their usability by these patients is questionable.

4.2 Proposed System

The proposed system has been implemented with mouse functions in hardware because mouse as a pointing device unlike a physical keyboard is versatile and simple. So not only it is easier to use but also simpler to implement. The keyboard function have been accomplished using on-screen application based keyboard.

This system uses hardware components which are:-

- Neurosky Mindwave EEG headset.
- Arduino Leonardo.
- Accelerometer.
- Connecting wires.
- Casing hardware.

The onscreen keyboard has been developed using QT API which compiles applications for Linux and Windows and Mac PCs. The cross platform approach is the one that helps us convert any laptop or desktop machine to be usable with our system. The API can be availed for free as long as the application which is built on it remains open-source. The mouse implementation has been

successful with utmost degree of accuracy of the mouse movement. The keyboard has been successfully developed with a novice design to help paralyzed patients use the system better. A profound research was done before implementing the keyboard as it is and with help of [tilvus ref] , we designed the multiple the keyboard which looks similar to T9 keyboard and has 2 click usage to avoid any false entries.

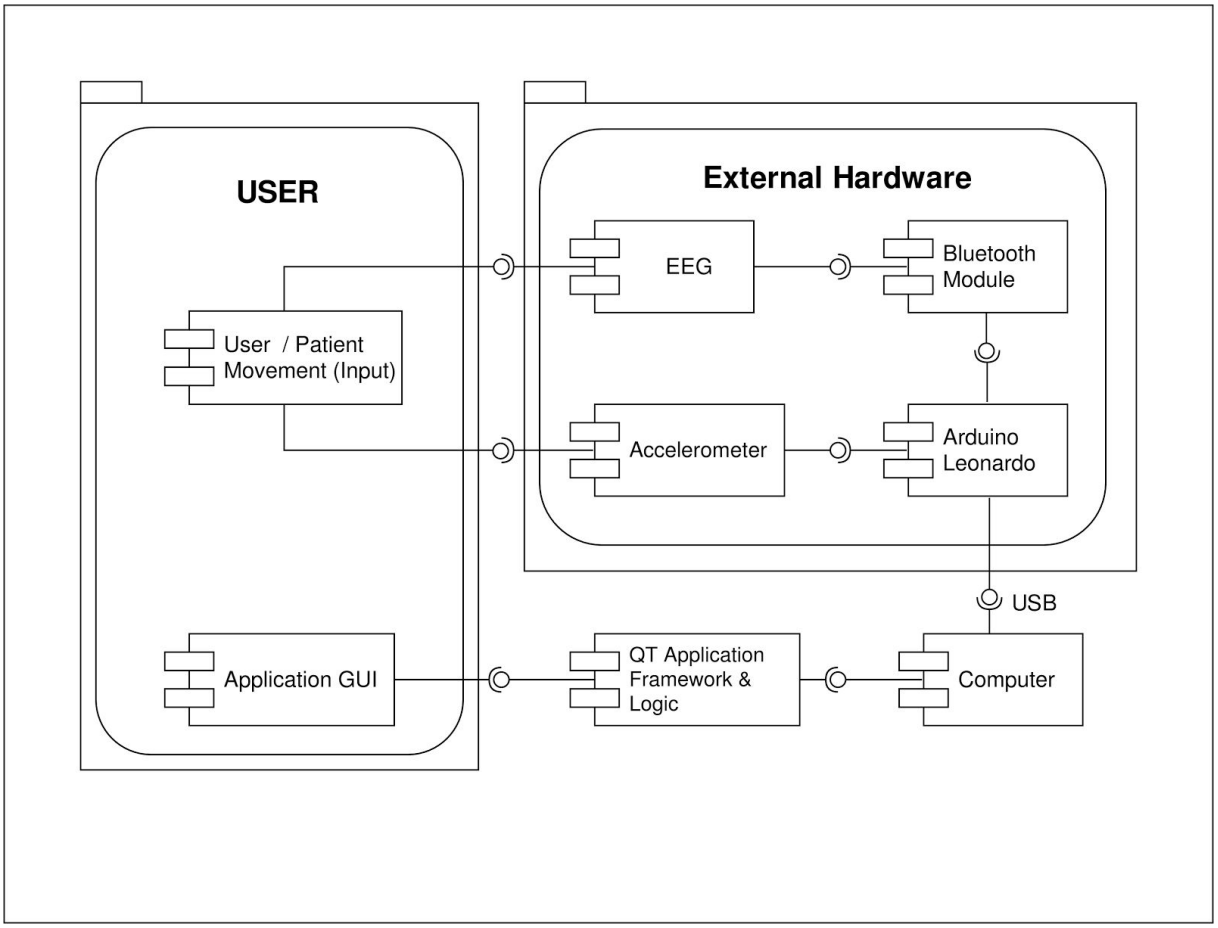

Fig. 3.1 *System Architecture*

work on all systems regardless of their PC hardware/software recommendation since USB HID drivers are widely accepted standards.

The user will utilise the hardware component to click on the required on the on-screen keyboard developed using Qt framework which supports OSX, windows RT and Linux. This means that the framework will build a platform dependent application based off a single source code,

greatlThe user will interact with the system in 2 main interfaces. One being with the arduino leonardo and EEG headset duo and the GUI of the on-screen keyboard and quick access bar on the PC to which the hardware components are connected.

The user's head movement is detected by the accelerometer and the user's blink is detected by the EEG. The data of these sensors is collected by the arduino leonardo for it to be converted to mouse clicks or move based on the inputs from the sensors(refer appendix 3 for the code). The leonardo produces a USB HID compliant mouse signals so the hardware component can y reducing our workload and at the same time increasing the accessibility of our system.

4.3 System Plan

As it has been made abundantly clear that the mouse emulation will be the the primary feature of the system. It is only natural that mouse functions be implemented first.So we needed a central device which could read sensor values from both an EEG and an accelerometer and give a USB HID compliant mouse signals.Accelerometer being a simpler sensor in terms of working and signal output was interfaced first with arduino leonardo and later mapped for mouse movement. After this was interfacing the neurosky EEG with arduino leonardo which required bluetooth capabilities which was made possible with HC-05 chip. With EEG interfacing done eye blinks could be easily detected which are being used to send mouse click signals to the PC to which the system is connected. However arduino leonardo having fundamentally different architecture to an arduino UNO is presenting a few challenges which is why the the EEG is connected to arduino leonardo through arduino UNO. This difference in architecture will be adjusted next, removing the need for and arduino UNO. During the transit time of the bluetooth module for arduino no mouse related work could be done, which is why basics of the Qt framework were learned by the team. It is primarily based on C++ and comes with its own IDE called as QtCreator. The IDE allows for easy UI design allowing more time for the functional backend.

4.4 Details of Hardware and Software Module

4.4.1 Hardware Module

The Hardware module has 4 major components

1. Arduino Leonardo
2. NeuroSky mindwave EEG
3. HC-05 Bluetooth Module.
4. Accelerometer ADXL335

The components and their functions are as follows:

4.4.1.1 Arduino Leonardo

The Arduino is a special micro controller that has keyboard and mouse primitives and is perfectly suited for the application of our project as we need to emulate the keyboard and mouse function onto the computer but the input shall actually be given by the accelerometer and EEG value inputs.

Fig 4.1 *Arduino Leonardo*

4.4.1.2 Neurosky Mindwave EEG Headset

system we have taken into account the link strength values as a parameter to decide if the mouse click is to be generated or not. The raw data values are obtained via packets sent through bluetooth. These packets are sent at regular intervals hence maintaining a stream of data that keeps flowing hence making the operation very smooth. The NeuroSky Mindwave EEG headset

is a device which works on principles of Electroencephalography(EEG). The Model we have used has 3 Electrodes. The EEG gives 3 output values i.e for attention levels, meditation levels, blink strengths. Each of these are values are given in form of packets of data which hold raw data values. These raw data values can further be processed according to the needs of the application for suitable use. In our

Fig 4.2 *Mindwave EEG Headset*

The above headpiece has 3 electrodes that need to be placed on the patient's head as depicted in the following figure

4.4.1.3 Bluetooth Module HC-05

The bluetooth we have used for implementation of wireless communication between the Headset and arduino is called as the HC-05. It belongs to class II bluetooth. It has a average range of 50 meters and is efficient enough for our system. The documentation available for Hc-05 also prompted us to use it.

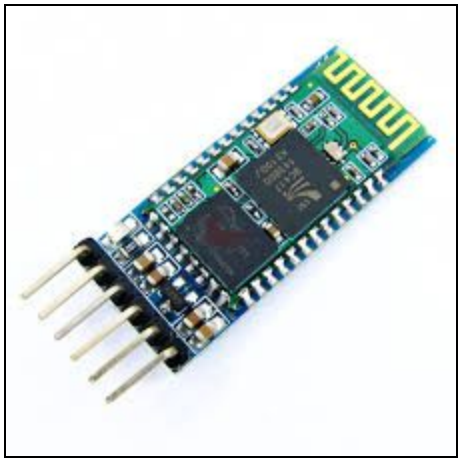

Fig 4.3 *HC-05 Bluetooth Module*

4.4.1.4 Accelerometer Module

The accelerometer is a device that detect the tilt of the crystal embedded in the chip to detect the movement and the direction of the movement, this movement can be recorded in a 3D plane along X-Y-Z axes respectively. We have used the ADXL335 accelerometer which shall help detect the tilt angle of the neck(as it is placed on the head) and give values in a 2D plane i.e X-Y plane. We have not used the Z axes as it would be excess to requirement and also difficult for the patient to handle the setup with neck movement alone.

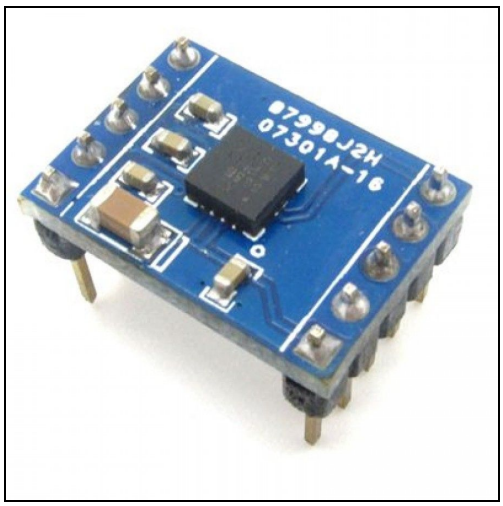

Fig 4.4 *Accelerometer Module*

4.4.2 Software Details

Arduino IDE: The open-source Arduino Software (IDE) makes it easy to write code and upload it to the Arduino Leonardo board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

Qt 5.8 - 64bit:
Application type : QtQuick Controls 2
Language: C++ and QML
Libraries: QProcess and LocalStorage
Database: Sqlite
Theme: Google Material Light

The application is based on the Qt framework. It is a cross-platform QtQuick Controls 2 application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while still being a native application with native capabilities and speed. We are using Qt version 5.7.0 for Linux 64-bit and Windows 32-bit & 64-bit architecture and can be also ported to MacOS as well making it a cross-platform application.

The user will utilize the hardware component to click on the required on the on-screen keyboard developed using Qt framework which supports OSX, windows RT and Linux. This means that the framework will build a platform dependent application based off a single source code, greatly reducing our workload and at the same time increasing the accessibility of our system.

4.5 System Design Details

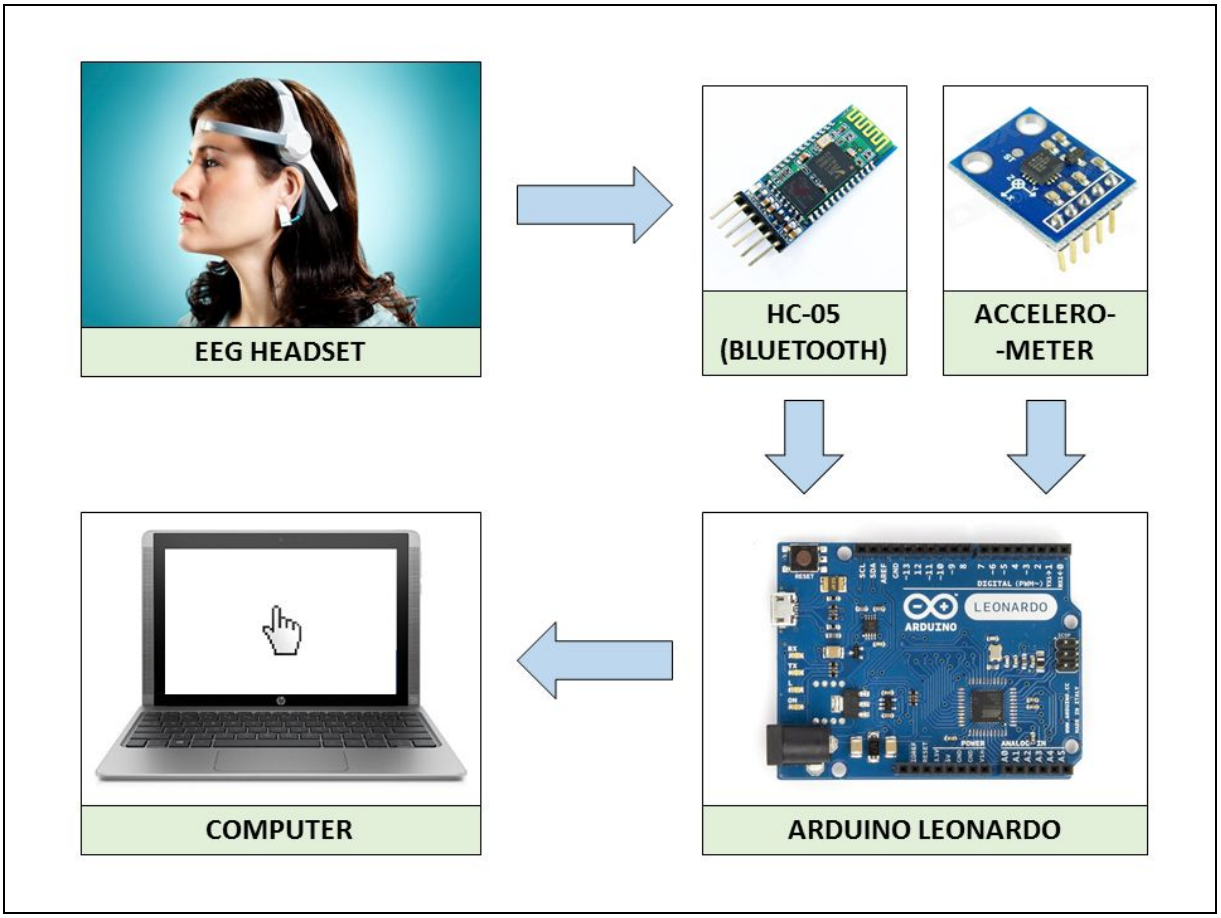

Fig 4.5 *System Design*

The overall system design is as follows. The EEG headset is connected to the Bluetooth module and sends data to it which is then transferred to the Arduino Leonardo to which it is connected to. The Accelerometer is connected directly to the Arduino and takes the sensor values on the analog pins of the Arduino. The accelerometer is responsible for the mouse pointer movements while the EEG headset is responsible for the mouse clicks. The Arduino Leonardo is directly connected to the computer or laptop using micro-b USB cable. The Arduino Leonardo is used for the mouse emulation and it works on the inputs by accelerometer and the EEG headset. Thus it can directly send commands to the computer for the mouse cursor and click functions.

The user will utilize the hardware component to click on the required on the on-screen keyboard developed using Qt framework which supports OSX, windows RT and Linux. This means that the framework will build a platform dependent application based off a single source code, greatly reducing our workload and at the same time increasing the accessibility
of our system.

The onscreen keyboard will be developed using QT API which compiles applications for Linux and Windows and Mac PCs. The SDK can be availed for free as long as the application which is built on it remains open-source. If the keyboard is implemented within the time limit. Text prediction capabilities help the users type faster using minimum number of clicks and reducing error.

CHAPTER 5: Implementation

5.1 Module I - Hardware Module - Mouse Emulation

Since the mouse is intended to be placed on the head. One part of this module, i.e. the EEG headset is already set for that purpose. The problem was to then place the remaining hardware on the head along with the EEG. So the first step was to put the circuitry (i.e. the Arduino Leonardo, bluetooth interface chip and accelerometer) inside an enclosure. For this purpose an ABS plastic case made for Arduino Mega was chosen, Arduino Mega ABS case is 10.9 cm long, 5.5 cm wide and 1.5 cm tall; compared that to Arduino Leonardo which is 6.8 cm long and 5.3 cm, means that Arduino Leonardo can comfortably fit inside the aforementioned enclosure and still have some space left. This space was utilised for fixing an accelerometer to the enclosure and also the Bluetooth interface chip (HC-05). Now that the second part of hardware which wasn't designed to be attached to a head was put in an enclosure. The hardware looks cleaner and closer to a finished product and more importantly it's now easier to attach this part to the head as now it was one unit. Next the enclosure was to be attached to the EEG headset. This would be the most practical since the entire system could be removed from the user in one go, unlike if it was attached separately to the head, which would require even more hardware as a head cap or another headset.

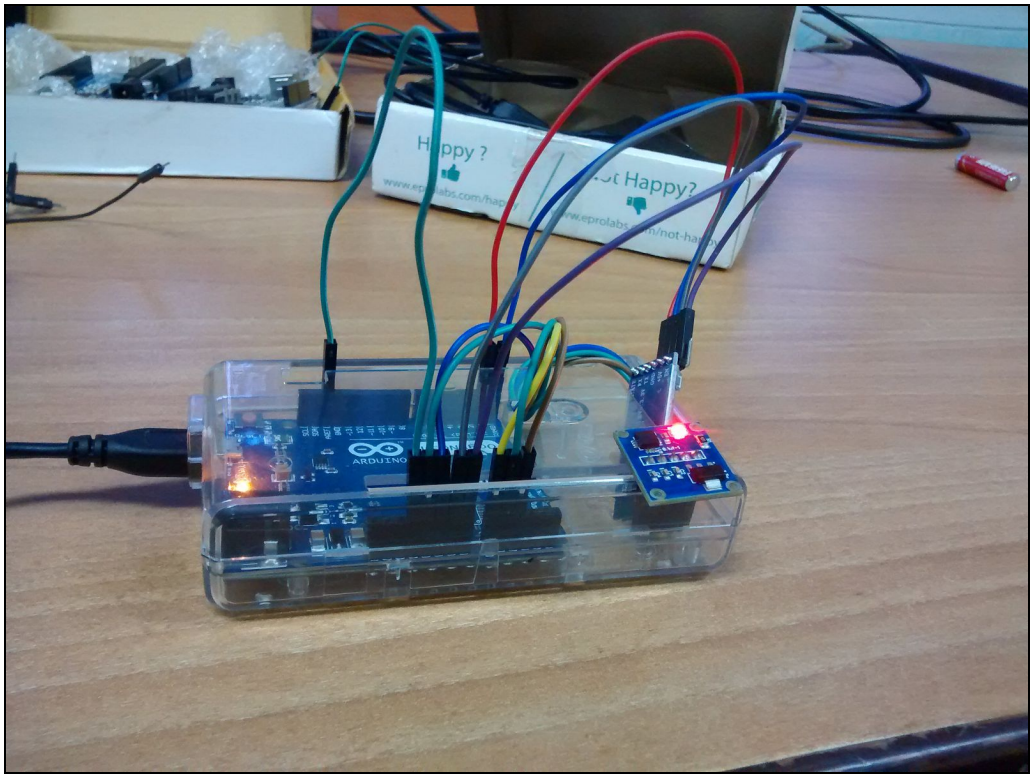

Fig 5.1-Arduino circuit with casing

At first the enclosure was directly attached to the EEG headset using a double sided adhesive tape. However this setup caused the enclosure and by extension accelerometer to be slightly tilted. This caused great difficulty for user operation for mouse movements, in other words when the user's head was in a neutral, upright position the accelerometer must not register any mouse movement. In order to solve the problem we resorted to cutting an arch with a flat top made of thermocol. The radius of the arc was measured to fit the curvature of the EEG and the enclosure was placed on the flat top. Thermocol was chosen since it was easy to obtain and cut to make whatever was required. However the ease of use thermocol was also it's downfall, thermocol isn't sturdy and after moderate amount of time in usage and also in storage the thermocol arch buckled and bent so that when it was put back in the head. It didn't register 0 mouse movement in upright head position and required constant adjustment. This would be simply be infeasible for our intended user, for whom this adjustment just wouldn't be possible.Secondly the thermocol fixture was custom to each user which meant the arc position perfect for one patient may not be feasible for another (i.e may droop in some way). So it was decided to use a sturdy plastic to

make the same arch but instead of fixing the plastic to the EEG directly, the arch was instead fixed to sleeve. The sleeve is put on the EEG headset and can freely move on the headset. This solved the problem of both sturdiness and varying head size of end users. This meant the enclosure could be easily adjusted for every user and once adjusted it would be sturdy so it wouldn't need anymore adjustments beyond that.

Moreover the distance between the bluetooth interface chip and EEG would be greater,meaning more noise.If however the bluetooth interface chip and the accelerometer are placed on the head, it would solve the problem of noise but would exacerbate the messiness of longer and more jumper cables. It is also noteworthy that the patients on whom the system was tested didn't find the decided system bulky or heavy, meaning the decided configuration is optimal.

To conclude the hardware module successfully emulated the mouse and wasn't bulky for the patients on whom it was tested. The ease of use the hardware module is discussed in se Alternatively it could have been possible to not aggregate the hardware into the enclosure and instead only fixing the accelerometer on EEG headset was also possible. This would certainly would have reduced the weight on the user's head, however that would mean longer jumper cables would be needed to connect the arduino to accelerometer; which would be action

5.2 Module II - Software Module - Virtual Keyboard

The objective of making the mouse handsfree was achieved using the above mentioned hardware module of the project, but we thought of a software solution that could eliminate the use of a physical keyboard and also help them type with more ease and less movement of the head. There are virtual keyboards present on most of the operating systems nowadays but those are still based on the QWERTY keyboard layout style since it is more widely used layout style also on physical keyboards. This approach is acceptable for normal users activities but for our use case would have been a much harder task since the keyboard spans across the whole screen on the bottom half. This involves moving of the head from one end to other end for typing letters which can cause fatigue and in some cases neck pain because of the movement of neck.

Thus we thought of using a custom application that acts like a virtual keyboard and has a keyboard layout that is compact and easy to use even for beginners. We chose on the 9 key layout keyboard which was used majorly in mobile phones eg. Nokia 3310 since mobiles needed to be a lot more compact fitting easily in a person's hands. This layout uses 3 keys per button and letter typing is done based on the number of successive button presses. So for example to type 'b' button 'abc' is pressed twice in quick succession.

Fig 5.2 *Nokia 3310 using the 9 key layout*

The virtual keyboard application is split into various sections depending on their functionality and usage. The sections are as follows:

- Text Area
- Function Keys
- Prediction Area &
- Typing Area

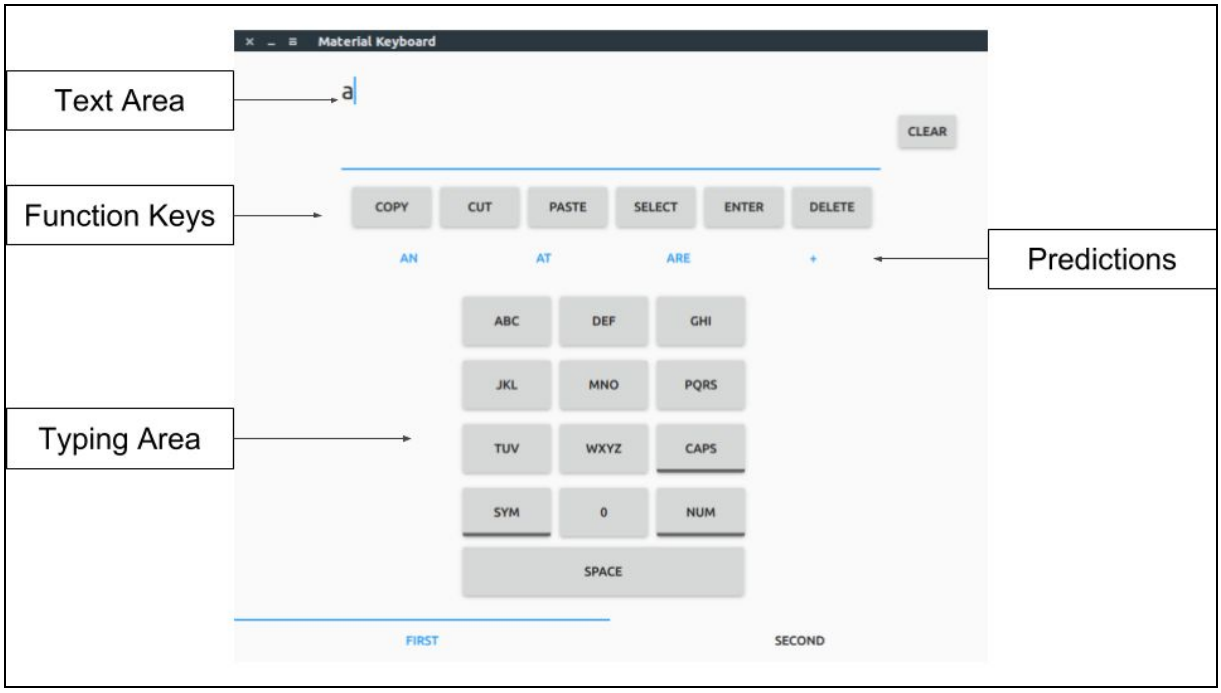

Fig 5.3 *Complete Keyboard Layout*

We have used the layout using expansion and contraction of keys so that it is much more easier than the physical counterpart which needs multiple button presses for a single character to be pressed. So in our program a click on ‘abc’ expands ‘a’, ‘b’ & ‘c’ on that row and whenever any button out of these three are pressed the letter is typed on the text area and the buttons contract and go back to it’s normal state.

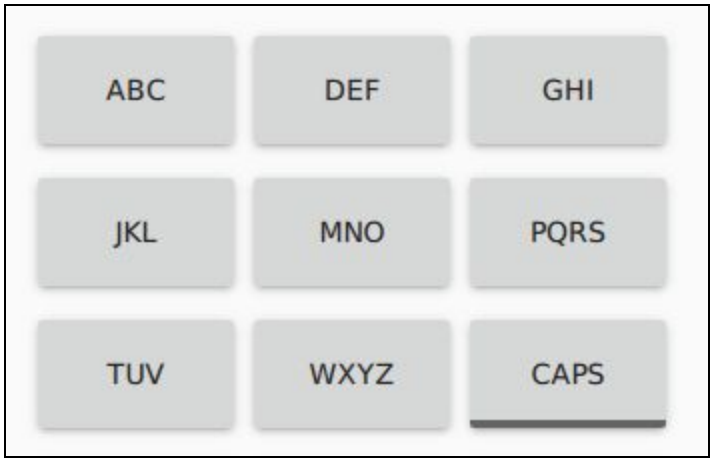

Fig 5.4 *Default State of Keyboard*

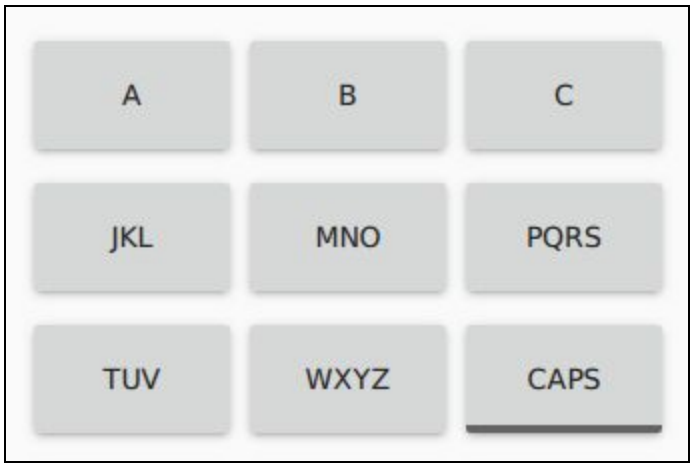

Fig 5.5 *Expanded State of Keyboard*

The virtual keyboard also includes commonly used functions for typing of which some are key combination shortcuts like

- Copy
- Cut
- Paste
- Select All Text
- Enter
- Delete / Backspace last Character
- Clear

These function keys are placed just below the text area and help the user do quick modifications to the text present on the text area. The function keys are sorted based on their usage from left to right and the backspace or delete button kept on the extreme right for quicker access.

The typing area also consists of two state buttons like CAPS which is used for switching the case of the letters from lowercase to uppercase and vice versa, by default the button is at lowercase mode. There are also buttons like SYM which is for various common symbols and NUM which is used to enter numbers from 0-9. These two state buttons light up blue when they are in ON state and go back to OFF state when they are pressed again.

Fig 5.6 *Symbol Entry Mode*

Fig 5.7 *Number Entry Mode*

The on screen keyboard in many computer operating systems lack predictions which on the other hand is majorly used in mobile keyboards to help users type faster using relevant text predictions and also correct typing errors if there are any. The prediction region is the important part of our application since it will be hard for a user to type very long words using just eye blinks. So we

designed a prediction engine that is present above the typing area and suggests words as the user is typing them. The user can type some part of the word and use the suggestion from the prediction area to quickly type out the whole word using one click on the suggestion. This minimizes the number of clicks which is turn is directly related to the number of blinks that a user has to do to type out a word.

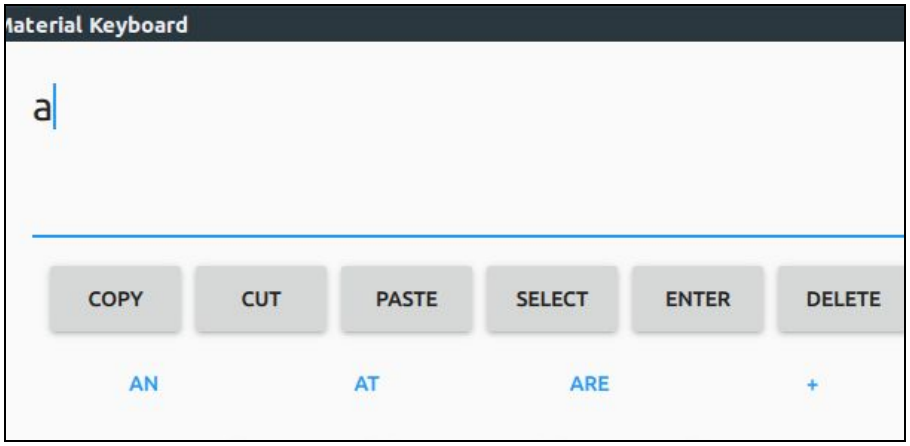

Fig 5.8 *Prediction Area*

We have also added a '+' icon that is used to add a custom word to the dictionary. The user has to just type out the word on the textarea and click on the '+' icon to add it to the dictionary. Also the words are ranked based on their usage which means the more a particular word is used it will appear on the left in the prediction region. Words are placed on the region based on their ranks and in a descending order of usage/ranks.

Chapter 6 : System Operation

6.1 Initialization

6.1.1 Initialization of the system.

The system has a keyboard application made as mentioned above. It needs to be installed or an executable file needs to be present in the system in order to make the system available to the user. The QT application needs all the libraries that have been used to develop the application to be present in machine along with the executable files. The libraries used are listed along with their function in section **6.1.1.1**

6.1.1.1

The libraries used for application development are:

1. QT Process:The QProcess class is used to start external programs and to communicate with them.

The header is given as: #include<QProcess>

QProcess allows you to treat a process as a sequential I/O device. You can write to and read from the process just as you would access a network connection using QTcpSocket. You can then write to the process's standard input by calling write(), and read the standard output by calling read(), readLine(), and getChar(). Because it inherits QIODevice, QProcess can also be used as an input source for QXmlReader, or for generating data to be uploaded using QNetworkAccessManager.

When the process exits, QProcess reenters the NotRunning state (the initial state), and emits finished().

The finished() signal provides the exit code and exit status of the process as arguments, and you can also call exitCode() to obtain the exit code of the last process that finished, and exitStatus() to obtain its exit status. If an error occurs at any point in time, QProcess will emit the errorOccurred() signal. You can also call error() to find the type of error that occurred last, and state() to find the current process state.

6.2 Bluetooth Connection Setup

The system uses bluetooth communication for connecting the EEG Headset with Arduino Leonardo. The Bluetooth we have used is HC-05 which belongs to the class II Bluetooth. The communication between these two components is used to send raw data from the EEG regarding the attention, meditation, blink strength levels to the arduino leonardo for further processing.
There are settings that need to be done that enable us to connect the HC-05 and EEG which is named as AT settings for the Bluetooth.
The following are the commands used for setting up the connections:

First Enter into command mode by long pressing the button provided on HC-05 Module, after then enter the following commands for HC-05 module to communicate our Mindwave Kit.

1. AT+NAME=”EEGPROJ”

2. AT+UART=”57600,0,0”

3. AT+ROLE=”1”

4. AT+PSWD=”1234”

5. AT+CMODE=”0”

6. AT+BIND=”U9AD,Z3,57KJ” (Mindwave Unique Number)

7. AT+IAC=”9E8B33”

8. AT+CLASS=”0”

VERIFICATION COMMANDS FOR HC-05 MODULE

1. AT

OK

2. AT+UART?

UART:57600,0,0

OK

3. AT+ROLE?

+ROLE:1

OK

4. AT+PSWD?

+PSWD:1234

OK

5. AT+CMODE?

+CMODE:0

OK

Once the bluetooth is set, the LED on HC-05 starts to blink with one second.

Procedure to start the communication:

1. Connect HC-05 to Arduino
2. Connect arduino to computer.
3. Start NeuroSky Mindwave EEG and press the start button on connection mode until a solid RED light appears.
4. The system then ready for use.

6.3 System setup

The system takes around 2-3 minutes for initialization once the headset is correctly placed on head and the bluetooth connections are made. The time is need to test the data packets received from the EEG via bluetooth. The EEG initially sends incorrect data packets until full connections are established. The values then gradually start to come to a constant range. From this point the click function gets activated as the user blinks as the difference in blink strength is noted. The mouse sensitivity can be adjusted using the software module where in the values are feeded to the arduino via a python script in order to fit to users comforts.

Chapter 7: Results and Testing

The System was built with the aim to make a hands-free computer assist for paralyzed patients. The system has been implemented as proposed, with utmost efforts to make it usable by a paralyzed patients. The mouse movement was implemented with an accelerometer with accurate movement of cursor with respect to the tilt of neck by the user. The eye blink function ,which was used to click on any given target area on a screen was implemented and tested with success. The keyboard application was developed on basis of T9 layout with beneficial two click input to avoid any chances of false clicks or entries in text fields. The success of the system cannot be judged unless and until used by a paralyzed patient and hence we decided to test the system with paralyzed patients.

A formal approach was made to the doctors from Neurogen Hospital, Navi-Mumbai. The doctors were impressed with the idea of the project and decided to go ahead with the testing of the project.

The following patients were a part of the testing phase:

1. Dr. Faizal Khan- A patient suffering from Quadriplegia that has resulted in negligible movement of both hand and both legs. This patient was specially selected looking at the medical condition and free movement of neck.
2. Dr. Hemangi Sane.- A patient suffering from ALS and is also a doctor who works as head of research and development in Neurogen hospital.

Both the patients were given a brief idea regarding the working of the system before the system testing was done. Following are some illustrations related to the same:

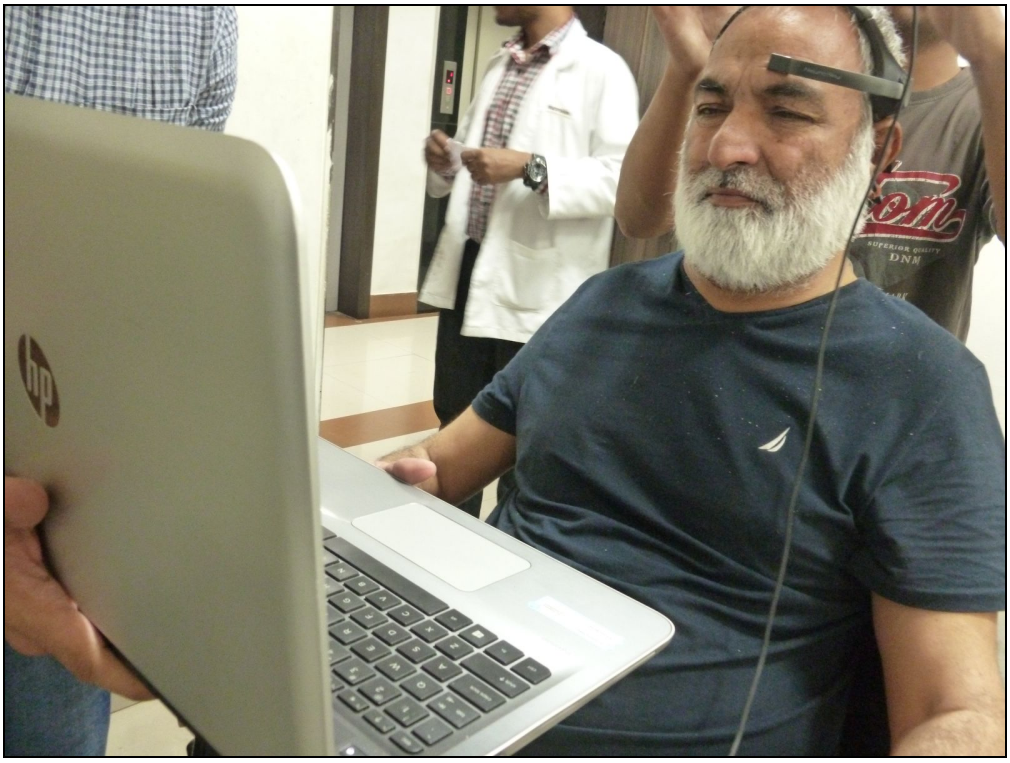

Fig 7.1 *Setting up the system*

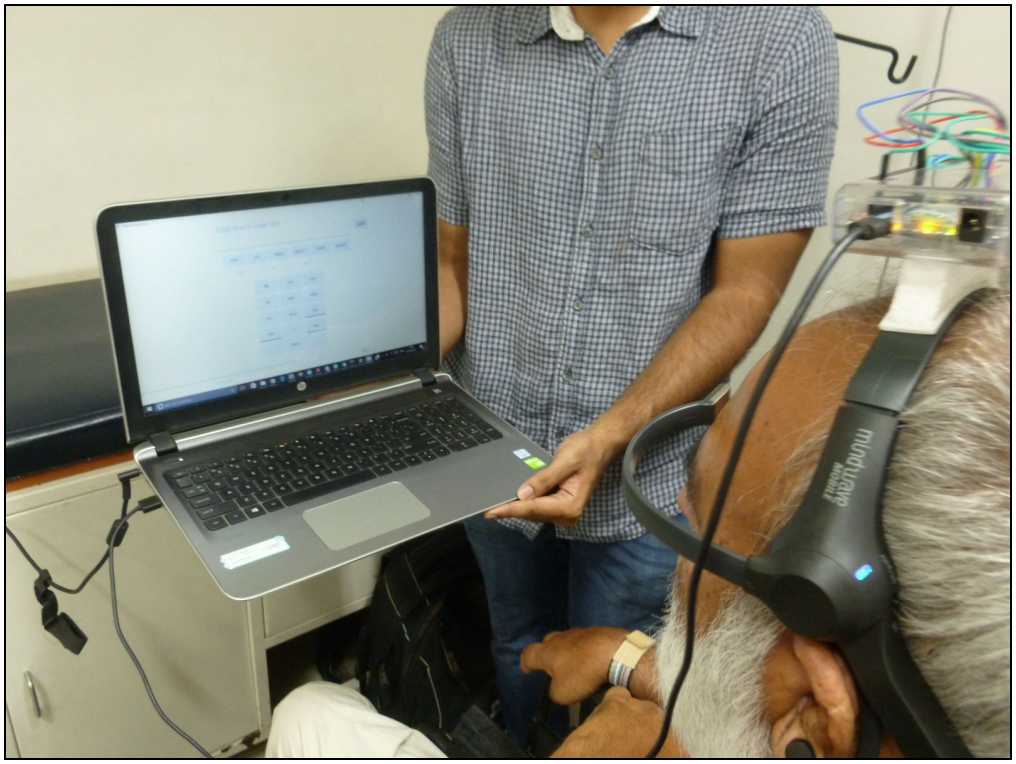

Fig 7.2 *First Test of the system on Dr. Faizal*

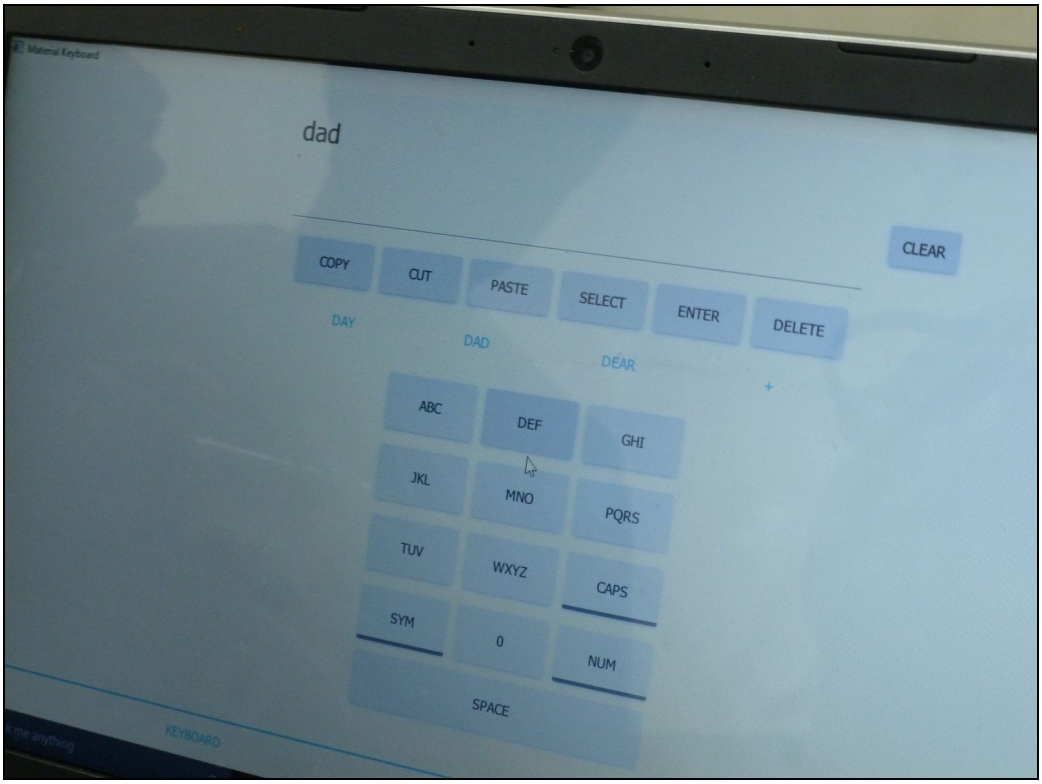

Fig 7.3 *Dr. Faizal typing words using our keyboard application*

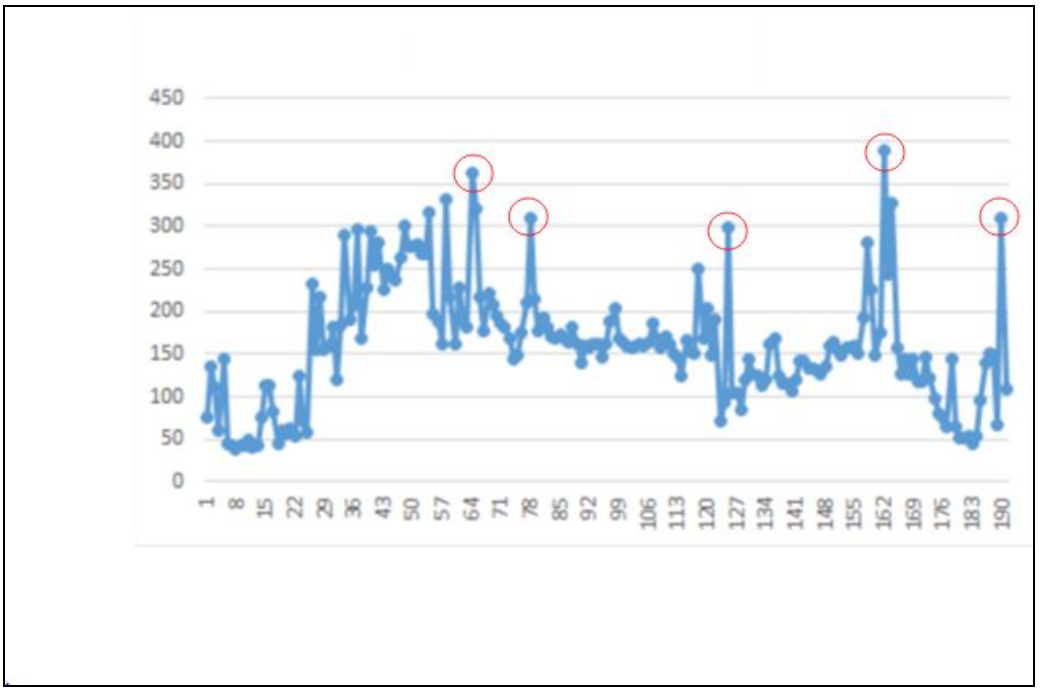

Fig 7.4 Dr. Faizal's Attention level mapping for Clicks Generated(The circle depict the clicks that were registered)

The above pictures depict the system being used by paralyzed patients as a part of testing as mentioned. In the testimonials they confirmed the ease of use and regarded the project a usable, practical product for the paralyzed patients.

CHAPTER 8 : PROJECT TIMELINE AND TASK DISTRIBUTION

8.1 Project Timeline

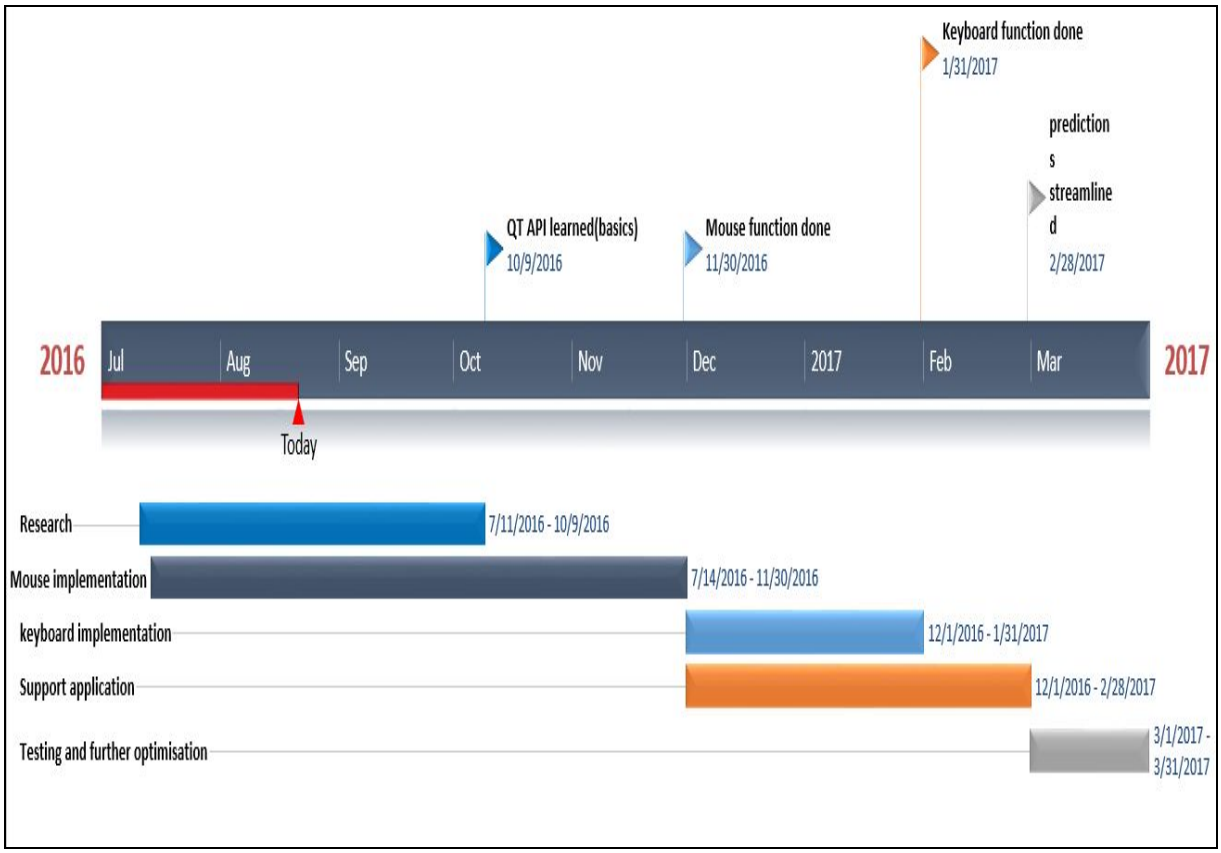

Fig 8.1- Timeline for Project implementation

The Project timeline was decided at the beginning of the development of the system. The timeline was fairly accurate and was followed for most of the stages during the development in the project.

CONCLUSION

The System has been developed with features that could help a paralyzed patient to operate a computer without any use of hands. The constraint being free movement of neck is necessary and with with a few training sessions the patient shall be able to do to most of the basic operations related to computer such as opening files , playing music etc. The breakthrough being the custom keyboard made to suit the patient. The T9 keyboard in particular was liked by the patients during the testing phase of the project and the found it very helpful. The testing phase has clearly proved that the system is very much helpful.

The System was developed and delivered at the end of the development cycle as promised in the proposal of the project. The software modules have been developed with a scope of future improvements and can be modified for individual personnels. The circuit can be integrated to a much smaller custom PCB Chip that can implement all the tasks the current circuit implements. This will lead to the circuit being very handy and smaller in size, weight which will make it even more easier for the user. A 3D printed support for the new chip would be suggested as to keep the EEG design constant.

The system can be made very generic to all patients with usage of a 32 electrode EEG instead of the 3 electrode EEG. This will surely increase the cost of the system but would eliminate any chances of false clicks etc and will enable development of system where in the user can do more complex tasks with ease. Having said that it would be suggested that the project could be carried out to be useful for one patient. The system can be targeted to solve the issues of one patient to maximum extent. This complete system can then be twitched to fit other patients requirements.

REFERENCES

1. *EEG-Based Online Two-Dimensional Cursor Control*

   Dandan Huang, Student Member, IEEE, Peter Lin, Ding-Yu Fei, Member, IEEE, Xuedong Chen, and Ou Bai, Member, IEEE .1.Published in: Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE Pages: 4547 - 4550 DOI: 10.1109/IEMBS.2009.5332722

2. *Binary EEG Control for Two-Dimensional Cursor Movement: An Online Approach*

   T. Kayagil 2, 0. Bail, P. Lin', S. Furlanil, S. Vorbach', M. Hallett' National Institute of Neurological Disorders and Stroke, National Institutes of Health, Duke University, Durham, NC 27708 USA Published in: Complex Medical Engineering, 2007. CME 2007. IEEE/ICME International Conference on Pages: 1542 - 1545, DOI: 10.1109/ICCME.2007.4382005

3. *NeuroPhone: Brain- Mobile Phone Interface using a wireless EEG Headset*

   Andrew T. Campbell, Tanzeem Choudary, Shaohan Hu, Hong Lu, Matthew K. Mukerjee, Mashfiqui Rabbi and Rajeev D. S. Raizada - Dartmouth College, Hanover, NH , USA.

4. *Hands Free Mouse*

   Aaron Castillo, Graciela Cortez, David Diaz, Rayton Espiritu, Krystle Ilistastigui, Kiran George. Published in: Wearable and Implantable Body Sensor Networks (BSN), 2016 IEEE 13th International Conference on 14-17 June 2016 Pages: 109 - 114, DOI: 10.1109/BSN.2016.7516242

5. *A Brain-Computer Interface (BCI) system to use arbitrary Windows applications by directly controlling mouse and keyboard*

   Martin Spuler. Published in: Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE Pages: 1087 - 1090, DOI: 10.1109/EMBC.2015.7318554

6. *Mind-Controlled Wheelchair using an EEG Headset and Arduino Microcontroller*

   Nikhil Sharma , Alson & Karthik Rajagopalan. Published in: Technologies for

Sustainable Development (ICTSD), 2015 International Conference on 4-6 Feb 2015 Pages: 1 - 5, DOI: 10.1109/ICTSD.2015.7095887

7. *Real-Time Nonintrusive Monitoring and Detection of Eye Blinking in view of Accident Prevention due to Drowsiness* Published in: Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on 18-19 March 2016 Pages: 1 - 6, DOI: 10.1109/ICCPCT.2016.7530224
8. *Electrodermal Activity Based Study on the Relationship Between Visual Attention and Eye Blink* Published in: Sensing Technology (ICST), 2015 9th International Conference on 8-10 Dec 2015 Pages: 596 - 599, DOI: 10.1109/ICSensT.2015.7438468
9. *Morse Code Translator Using the Arduino Platform: Crafting the Future of Microcontrollers* Published in: SAI Computing Conference (SAI), 2016 Pages: 675 - 680, DOI: 10.1109/SAI.2016.7556055
10. *Communication by Eye Closure-A Microcomputer Based System for the Disabled* Published in: IEEE Transactions on Biomedical Engineering (Volume: BME-33, Issue: 10, Oct. 1986) Pages: 977 - 982, DOI: 10.1109/TBME.1986.325671
11. *Real-Time Nonintrusive Monitoring and Detection of Eye Blinking in view of Accident Prevention due to Drowsiness* Published in: Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on 18-19 March Pages: 1 - 6, DOI: 10.1109/ICCPCT.2016.7530224
12. *Brain Computer Interface and Electro Stimulation of muscles to assist paralyzed patients in limited movement of hands.*
13. *An EEG-based BCI System for 2D Cursor Control*

    Yuanqing Li, Chuanchu Wang, Haihong Zhang and Cuntai Guan Published in: Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on 1-8 June 2008 Pages: 2214 - 2219, DOI: 10.1109/IJCNN.2008.4634104

14. *Enabling Computer Decisions Based on EEG Input*
Benjamin J. Culpepper ,Robert M. Keller, Harvey Mudd College Published in: IEEE Transactions on Neural Systems and Rehabilitation Engineering (Volume: 11, Issue: 4, Dec. 2003) Pages: 354 - 360, DOI: 10.1109/TNSRE.2003.819788

15. *Brain machine interface system with artificial intelligent for a person with disability*
-Ujwala Marghade, Vinay Keswani

16. *International Morse Code Recommendation* ITU-R M.1677-1
http://www.itu.int/rec/R-REC-M.1677-1-200910-I/

17. *Tilvus - Single Switch Assistive Computer Interface*
http://tilvus.net

18. *Face detection and tracking with Arduino and OpenCV*
http://www.instructables.com/id/Face-detection-and-tracking-with-Arduino-and-OpenC/step4/Resources/

APPENDIX

CODING CONVENTIONS:

1. General

- Avoid using names that are too general or too wordy. Strike a good balance between the two.
- Bad: data_structure, my_list, info_map, dictionary_for_the_purpose_of_storing_data_representing_word_definitions
- Good: user_profile, menu_options, word_definitions
- Don't be a jackass and name things "O", "l", or "I"
- When using CamelCase names, capitalize all letters of an abbreviation (e.g. HTTPServer)

2. Packages

- Package names should be all lower case
- When multiple words are needed, an underscore should separate them
- It is usually preferable to stick to 1 word names

3. Modules

- Module names should be all lower case
- When multiple words are needed, an underscore should separate them
- It is usually preferable to stick to 1 word names

4. Classes

- Class names should follow the UpperCaseCamelCase convention
- Python's built-in classes, however are typically lowercase words
- Exception classes should end in "Error"

5. Global (module-level) Variables

- Global variables should be all lowercase
- Words in a global variable name should be separated by an underscore

6. Instance Variables

- Instance variable names should be all lower case
- Words in an instance variable name should be separated by an underscore
- Non-public instance variables should begin with a single underscore
- If an instance name needs to be mangled, two underscores may begin its name

7. Methods

- Method names should be all lower case
- Words in an method name should be separated by an underscore
- Non-public method should begin with a single underscore
- If a method name needs to be mangled, two underscores may begin its name

8. Method Arguments

- Instance methods should have their first argument named 'self'.

- Class methods should have their first argument named ‘cls’

9. Functions

- Function names should be all lower case
- Words in a function name should be separated by an underscore

10. Constants

- Constant names must be fully capitalized
- Words in a constant name should be separated by an underscore

SOURCE CODE:

FOR ARDUINO PROGRAMMING

```
#include<Mouse.h>

#define   BAUDRATE          57600

#define   LED               13

#define   Theshold_Eyeblink  170

#define   EEG_AVG           70

//accelerometer
const int xout = A0;
const int yout = A1;
const int zout = A2;
const int threshold=10; //mapped reading at which mouse gets triggered
float MouseSensitivity=0.5;

int px,py,pz;
int x,y,z; //volt reading on x,y,z axes
```

```
int xMap,yMap,zMap; //axes voltage mapping for mouse triggering
float xMove,yMove,wheelMove;
int st;
int mouseState;
int inPin = 8;   // EEG O/P connected to digital pin 8
int val = 0;     // variable to store the read value
int clickstate=0;


int invertY=1;
//end acc

long payloadDataS[5] = {0};

long payloadDataB[32] = {0};

byte checksum=0,generatedchecksum=0;

unsigned int Raw_data,Poorquality,Plength,Eye_Enable=0,On_Flag=0,Off_Flag=1 ;

unsigned int j,n=0;

long Temp,Avg_Raw,Temp_Avg;



 void setup()

 {
```

```
  Serial.begin(9600);
  Mouse.begin();
  mouseState=true;
  pinMode(inPin, INPUT);      // sets the digital pin 8 as input
  st=millis();


  Serial1.begin(BAUDRATE);
  pinMode(LED, OUTPUT);
  pinMode(8,OUTPUT);//change it

}



byte ReadOneByte()          // One Byte Read Function

{

  int ByteRead;
   //Serial.println("12test");
  while(Serial1.available()){
   //Serial.println("test");
    ByteRead = Serial1.read();
  }
 //Serial.println(ByteRead);
  return ByteRead;

}
```

```
 boolean approxeq(int a,int b)
{
  int approximation=threshold;
  if(a>=b-approximation && a<=b+approximation)
  {
    Serial.print("yeee");
    return true;
  }
  else
    return false;
}

void mouseToggle()
{
  if(mouseState==true)
    mouseState=false;
  else
    mouseState=true;
}

float setMoveSensitivityY(int m){ //takes mouse move b/w -128 to 128
  m=abs(m);
  if(m<3){
    return 0.3;
  }else if(m<5){
    return 0.5;
  }else if(m>=5 && m<10){
    return 0.6;
```

```
  }else if(m>=10){
    return 1;
  }
}

float setMoveSensitivityX(int m){
  m=abs(m);
  if(m<3){
    return 0.5;
  }else if(m<5){
    return 0.5;
  }else if(m>=5 && m<10){
    return 0.6;
  }else if(m>=10){
    return 1;
  }
}

 void loop()                    // Main Function

 {

   if(Serial.available()>0){
    String message="";
    while(Serial.available()>0){
      //Serial.println("yo");
      message+=char(Serial.read());
      delay(50);
    }
```

```
    MouseSensitivity=message.toFloat();
    //Serial.println(message);
  }
   //DO NOT REMOVE DELAYS IT MAY CAUSE THE COMPUTER TO HANG!
  // put your main code here, to run repeatedly:
  analogReference(EXTERNAL);
  //mouseState=true;

  x=analogRead(xout); //
  xMap=map(x,0,1023,-512,512); // map from range 0 - 1023 to -512 to 512
  //if(xMap>threshold || xMap<-(threshold))
  //{
    xMove=map(xMap,-512,512,-128,128); // map from -512 - 512 to range supported by
mouseMove()
    xMove*=(setMoveSensitivityX(xMove)*MouseSensitivity);
  //}
  //else
    //xMove=0;

  y=analogRead(yout);
  yMap=map(y,0,1023,-512,512);
//  if(yMap>threshold || yMap<-(threshold))
//  {
    yMove=map(yMap,-512,512,-128,128); //invert y
    yMove*=(invertY*setMoveSensitivityY(yMove)*MouseSensitivity);
//  }
//  else
//    yMove=0;
```

```
//  z=analogRead(zout);
//  zMap=map(x,0,1023,-512,512);
//  if(zMap>5 || zMap<-5)
//  {
//    wheelMove=map(xMap,-512,512,-100,100);
//    if(wheelMove>30 || wheelMove<-30) // used to turn mouse on or off (the jerking)
//    {
//      Serial.println("ending!!!!");
//      Serial.println(wheelMove>20);
//      Serial.println(wheelMove<-20);
//      mouseToggle();
//      wheelMove=0;
//      delay(500);
//    }
//  }

if(mouseState==true)
  {
    Mouse.begin();
    Mouse.move((int)xMove,(int)yMove,0);
  }
//  Serial.print("mouse state");
//  Serial.print(mouseState);
//  Serial.print("\tx=");
//  Serial.print((int)xMove);
//  Serial.print("\ty=");
//  Serial.print(yMove);
//  Serial.print("\tz=");
//  Serial.println(wheelMove);
```

```
//delay(5);


//Serial.println("test");

 if(ReadOneByte() == 170)        // AA 1 st Sync data

 {

  if(ReadOneByte() == 170)      // AA 2 st Sync data

  {

   Plength = ReadOneByte();

   if(Plength == 4)   // Small Packet

   {
    //Serial.println("sp");
     Small_Packet ();

   }

   else if(Plength == 32)   // Big Packet

   {
     //Serial.println("BP");
    Big_Packet ();
```

```
      }
    }
  }
}


void Small_Packet ()
{
  generatedchecksum = 0;
  for(int i = 0; i < Plength; i++)
  {
    payloadDataS[i]     = ReadOneByte();      //Read payload into memory
    generatedchecksum  += payloadDataS[i] ;
  }
  generatedchecksum = 255 - generatedchecksum;
```

```
checksum  = ReadOneByte();

if(checksum == generatedchecksum)        // Varify Checksum

{

 if (j<512)

 {

  Raw_data  = ((payloadDataS[2] <<8)| payloadDataS[3]);

  if(Raw_data&0xF000)

  {

   Raw_data = (((~Raw_data)&0xFFF)+1);

  }

  else

  {

   Raw_data = (Raw_data&0xFFF);

  }

  Temp += Raw_data;
```

```
      j++;

    }

    else

    {

      Onesec_Rawval_Fun ();

    }

  }

}



void Big_Packet()

{

  generatedchecksum = 0;
  //Serial.println("test");
  for(int i = 0; i < Plength; i++)

  {
```

```
 payloadDataB[i]     = ReadOneByte();      //Read payload into memory

 generatedchecksum  += payloadDataB[i] ;

}

generatedchecksum = 255 - generatedchecksum;

checksum  = ReadOneByte();

if(checksum == generatedchecksum)        // Varify Checksum

{

 Poorquality = payloadDataB[1];
 Serial.println(Poorquality);

 if (Poorquality==0 )

 {

  Eye_Enable = 1;

 }

 else

 {
```

```
    Eye_Enable = 0;
   }
  }
}

void Onesec_Rawval_Fun ()
{
  Avg_Raw = Temp/512;
  if (On_Flag==0 && Off_Flag==1)
  {
   if (n<3)
   {
    Temp_Avg += Avg_Raw;
    n++;
   }
```

```
    else
    {
      Temp_Avg = Temp_Avg/3;
      if (Temp_Avg<EEG_AVG)
      {
        On_Flag=1;Off_Flag=0;
      }
      n=0;Temp_Avg=0;
    }
  }
  Eye_Blink ();
  j=0;
  Temp=0;
}
```

```
void Eye_Blink ()

{

 if (Eye_Enable)

 {

  if (On_Flag==1 && Off_Flag==0)

  {
   Serial.println(Avg_Raw);
   if ((Avg_Raw>Theshold_Eyeblink) && (Avg_Raw<550))

   {

    digitalWrite(LED,HIGH);
    //Serial.println("Initiating click");
    Mouse.click(MOUSE_LEFT);
    //digitalWrite(8,HIGH);

   }

   else

   {
```

```
      if (Avg_Raw>350)
      {
        On_Flag==0;Off_Flag==1;
      }
      digitalWrite(LED,LOW);
    }
  }
  else
  {
    digitalWrite(LED,LOW);
  }
}
else
{
  digitalWrite(LED,LOW);
```

```
    }

}
```

ACKNOWLEDGEMENT

We would like to thank our guide, Mrs. Sejal Chopra and Mrs. Kalpita Wagaskar for their constant support and guidance throughout this two semesters, without which we would not have been able to progress so far with our project.

We would like to express our sincere gratitude to Dr. Amiya Tripathy for being a constant help throughout the project. The suggestions provided by Dr. Amiya Tripathy were excellent and helped us focus on all areas of the project.

A special thanks to Dr. Joji Joseph and his team of doctors who helped us immensely in the testing phase of the project with their valuable inputs. We would also like to express sincere gratitude towards Dr. Hemangi sane and Dr. Faizal Khan for their co-operation and active participation in the testing of the system.

Project Team Members :

1. Akash Gund B. E. – (24)
2. Bronson Mendonca B. E. – (45)
3. Siddhant Reddy B. E. – (58)