Head-movement Gesture Recognition using Artificial Neural Networks with Multi-Layer Perceptron Algorithm

by

Bitlla Sabitha

A thesis submitted in partial fulfillment of the requirements for the
degree of Master of Engineering in
Microelectronics and Embedded Systems

Examination Committee:	Dr. Mongkol Ekpanyapong (Chairperson) Assoc. Prof. Erik L. J. Bohez Dr. A.M. Harsha S. Abeykoon
Nationality:	Indian
Previous Degree:	Bachelor of Technology in Electronics and Communication Engineering Jawaharlal Nehru Technological University Hyderabad Telangana, India
Scholarship Donor:	AIT Fellowship

Asian Institute of Technology
School of Engineering and Technology
Thailand
May 2018

ACKNOWLEDGEMENTS

I would like to extend my sincere gratitude to my advisor and Chairperson Dr. Mongkol Ekpanyapong for guiding me and sharing his expertise during the study. I would like to thank the committee members, Dr. A. M. Harsha S. Abeykoon and Assoc. Prof. Erik L. J. Bohez for their kind guidance and support. I would like to thank my parents for their support and their encouragement throughout my work.

ABSTRACT

Assistive technologies have recently emerged to improve the quality of life of severely disabled people by enhancing their independence in daily activities. Since many of those individuals have limited or non-existing control from the neck downward, alternative hands-free input modalities have become very important for these people to access assistive devices. In hands-free control, head movement has been proved to be a very effective user interface as it can provide a comfortable, reliable and natural way to access the device. Therefore, the main aim of this thesis is to help the bed-ridden patients to communicate and also control the electrical appliances with the help of head movements. The patient can totally depend on themselves and with the help of their head movements, they can control basic electronic devices and also ask for their basic needs. Recently, neural networks have been shown to be useful not only for real-time pattern recognition but also for creating user-adaptive models. In this study multi-layer perceptron (MLP) neural networks trained using standard back-propagation technique has been proposed to improve the generalisation of the networks. The samples were collected and are trained and classified by the multi-layer perceptron model. The overall system accuracy is improved by optimizing the various parameters of the multilayer perceptron network.

Keywords: Neural networks, multi-layer perceptron (MLP), back-propagation technique

TABLE OF CONTENTS

LIST OF FIGURES

LISTOF TABLES

LIST OF ABBREVIATIONS

IMU	Inertial Measurement Unit
MEMS	Micro Electro Mechanical System
NEMS	Nano Electro Mechanical System
IC	Integrated Circuit
PWC	Electric Powered Wheelchair
ARM	Advanced RISC Machines
WSN	World Network Services
ADC	Analog-to-Digital Converter
UART	Universal Asynchronous Receiver-Transmitter
HMI	Human Machine Interface
EEG	Electro Encephalo Gram
CPU	Central Processing unit
LED	Light emitting diode
LCD	Liquid Crystal Diode
GSM	Global System for Mobile communications
DOF	Degree of Freedom
RAM	Random Access Memory
DSP	Digital signal processing
PIC	Peripheral Interface Controller
SSH	Secure Shell
SOC	System on Chip
EMF	Electromotive Force
USB	Universal Serial Bus
HDMI	High-Definition Multimedia Interface
GPIO	General-purpose input/output
NOOBS	New Out Of Box
SD	Secure Digital Card
TTL	Transistor–transistor logic
SIM	Subscriber Identification Module
AT	Attention
GPRS	General Packet Radio Service
RF	Radio-frequency
ASCII	American Standard Code for Information Interchange
CMOS	Complementary Metal-Oxide-Semiconductor
MCU	Multipoint Control Unit
EVM	Electronic Voltmeter
AHRS	Attitude and Heading Reference System
AAL	Ambient Assisted Living

CHAPTER 1
INTRODUCTION

1.1 Background

Approximately 6 million people in the world face the problem of disability due to paralysis of various degrees. Paralysis is caused by impairment of nervous system disabling the people from performing various common functions [1]. Paralysis is a loss of muscle function for one or more muscles. Paralysis can be accompanied by a loss of feeling (sensory loss) in the affected area if there is sensory damage, this means of their locomotion has to be increasingly sophisticated & cost effectively work in order to enhance their quality of life and cement their integration into their working world. In reality, there are many types of paralysis because there are innumerable ways that the body can be injured. There are four main categories of paralysis, however, which have to do with the portion of the body that is affected.

- Monoplegia is paralysis of a single area of the body, most typically one limb.
- Hemiplegia affects an arm and a leg on the same side of the body, and as with monoplegia
- Paraplegia refers to paralysis below the waist and usually affects both legs, the hips.
- Quadriplegia, which is often referred to as tetraplegia, is paralysis below the neck. All four limbs are typically affected.

The person affected with paraplegia normally has dis-functioning of both their legs whereas quadriplegics are individuals who are not able to use any of the limits. The reasons for such declined motion potentials can be different: stroke, arthritis, and high blood pressure, degenerative illnesses of bones and joins and cases of paralysis defects which are caused from the time of birth itself. Also, quadriplegia appears as a consequence of accidents or age. The patients with such plain debilities are not able to attain their daily activities, such as feeding, toilette usage, and movement through space. Depending on the type of the disability, a patient can maintain freedom of movement to a certain level by means of various medical devices.

Today's technology is fast shifting towards automation which minimizes the need for human intervention. Those mechanization-cum-human operators with the aid of machinery require muscular work whereas automation greatly reduces the need for not only human sensory but also mental requirements as well. Thus, the present day automated systems have less manual operations, more flexibility, reliability and high accuracy. Due to this demand, every field prefers automated control systems. Especially in the field of electronics automated systems as they are giving reliable performance. Gesture recognition is one of the ways where machines get to understand the human body statement and communicate with it, accordingly develops a comfortable support between human and machines [2]. The system mounted on helpfully and controls its movement based on control signs. Micro electromechanical system (MEMS) is the innovation of small mechanical gadgets driven by power and it meets at the Nano scale into Nano electromechanical frameworks (NEMS) and nanotechnology.

Microphones and cameras are rich in sensing capabilities but come with severe privacy concerns. Many physiological sensors such as respiration and electrocardiogram (ECG) sensors must be positioned at certain locations on the body (e.g., chest, neck, head), with some even requiring tight skin contact. The intrusiveness of these contact-based sensors makes people resistant to use them in practice. Various methods have been proposed for allowing disabled persons, including a quadriplegic to control a motorized wheelchair [3]. There are various proposed methodologies in recent times which involve various gestures like hand gesture[4], head gesture, accelerometer & voice controlled[5], EEG based system, tongue motion based operations[6], eye movement based operations [7]etc. The increased popularity of the wide range of applications of which head movement detection is a part, such as assistive technology, teleconferencing, and virtual reality,

have increased the size of research aiming to provide robust and effective techniques of real-time head movement detection and tracking. During the past decade, the field of real-time head movement detection has received much attention from researchers. There are many different approaches to head movement detection.
Head movement is likewise observed to be a characteristic, basic and natural method for identifying the items, interaction, and communication with them. In the same manner, gives the capacity to governor various gadgets with the help of control signals by finding the required position of the head[8].The head movement devices are increasing and there are numerous future methodologies. Though there many systems useful to some degree, and many researchers have proposed head movement detection using the wheelchair, and many failed to apply it to totally bed-ridden people. Therefore in this thesis, I would propose the head movement based head band which can detect the head movements and operates the functions for each distinguished directions especially for the people suffering from quadriplegia.

Figure 1.1: Example of a quadriphegia patient
Source: www.spinalcord.com

The picture above is an example for the quadriplegia people. Quadriplegia is paralysis caused by illness or injury to a human that results in the partial or total loss of use of all their limbs and torso; Paraplegia is similar but does not affect the arms. The loss is usually sensory and motor, which means both sensation and control are lost.

1.2 Purpose of the project

The main purpose of my thesis is to design and develop a head movement head band device system for physically challenged people. The customer can wear this gadget to head and with the straightforward head development's he can ask for the essential needs like water, nourishment or solution by using IMU sensor with MEMS technology. The client can likewise control the electrical gadgets like light; fan and so forth with the assistance of head developments. The major aspect here is to design a cost-effective, mountable and handy MEM based control system for the physically handicapped with high accuracy. The idea focuses on head movement detection in different directions. The main purpose of this project is to track the position of the head using IMU sensor. The IMU will be placed on the head. When the head rotates, the movement of the head is

recognized by IMU sensor. It converts this head movement into equivalent electrical signals and sends it to the raspberry pi via RFX240 Xbee module and as per the gesture according to the head movements, the audio is provided through speakers like food, water, medicine, doctor, fan, light and the text messages for the respective gesture is sent via GSM module to the mobile

1.3 Problem statement

The main problem is primarily with the sensor themselves. MEMS devices just do not have exceptionally good performances characteristics. The readout electronics are noisy and do not have very good dynamic range. There is really a less way to improve the precision or the accuracy of an IMU. If we want a real gyro, then we need a high-precision mechanical gyro, fiber optics gyro or a ring laser gyro which are high precision devices that requires expensive precision components and time-consuming algorithms and calibration. They are quite large and perfect for commercial military airplanes, missiles, and rockets but outside of that, they are far too expensive.

1.4 Objectives

The main objective of this thesis is to outline and build a head movement controlled head band for physically disabled people by using an IMU sensor. The patient can wear this headband on the head and with the head movement, the patient can communicate for their essential needs like food, water, medicine etc. This sensor finds the tilt and works the electrical gadgets and reports the necessary needs relying upon tilt. The system here acts as the assistance to the patient who cannot do their basic daily activities and with the help of the device they can minimize the need for the special assistant or the caretaker. GSM module was included to send the text messages to the caretaker or the doctor to inform the current status or the activity of the patient.

1.5 Scope and limitations

The project is regarding some of the tasks defined as follow:

- The head belt must work with maximum accuracy.
- The accurate outputs.
- The head belt must not work when the patient is in sleep or when not required.
- The patient must be convenient for the device and must feel free to use without straining the head.

1.6 Research outline

The outline of this thesis is shown as follows.

In Chapter 2, the literature review of the related schemes will be provided.

In Chapter 3, the methodology of the proposed system will be provided.

In Chapter 4, the implementation and results will be provided.

In Chapter 5, the conclusion and recommendation will be provided.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

Around 6 million individuals on the earth are facing the issue of the disability because of Paralysis of different degrees. Paralysis is caused by debilitation of sensory system incapacitating the general population from performing different basic capacities. Paralysis is the loss of muscle capacity for at least one muscles. Loss of movement can be merged by a misfortune feeling in the people. Around 1 in 50 individuals have been determined to have some type of loss of motion, transient or lasting. Versatile gear can be utilized to help stroke patients have more prominent autonomy with regular daily existence abilities or exercises of everyday living. There is an extraordinary grouping of assistive gadgets though regions of self-couldn't care less including dressing, washing, prepping, cooking, sustaining, toileting, and portability helps. There are likewise assistive innovation gadgets to help with correspondence, utilizing a PC [9], working family unit gadgets, and driving. Head development designed to operate the electric powered wheelchair in view of an EEG device called Emotive EPOC that distinguish the head movements [10]. It has two ways: where first uses just a single make a beeline for control the wheelchair and alternate use four head developments. It is conceivable by eye following framework to control controlled wheelchair [8]. Eye developments of the client are meant to screen position utilizing the optical sort eye following framework.

2.2 Previous study

2.2.1 Head movement based control wheelchair in an indoor area

The research paper was demonstrated by Ericka Janet Richey *et al*. [10] where they equipped a user-friendly human machine interface (HMI) to control the wheelchair with the help of free hands. And it is called as the electric powered wheelchair (EPW). The system is of two different modes: The first mode utilizes just a single make a beeline to follow the orders, and another mode utilizes head movements. An EEG gadget, to be specific Emotive EPOC, sent in this HMI to acquire the head development data of clients. The HMI which was fixed is contrasted and the joystick controls the EPW in an indoor situation. The trial comes about demonstrated in a very quick point of time which is done by the second mode dependably, accomplishing an interim of 67.90 seconds for the two subjects. Be that as it may, Control Mode 1 has mediocre execution, accomplishing an interim of 153.20 seconds for the two subjects in spite of the fact that it needs just a single head development. Unmistakably the proposed HMI can be adequately utilized substitute the conventional joystick control for handicapped and elderly individuals.

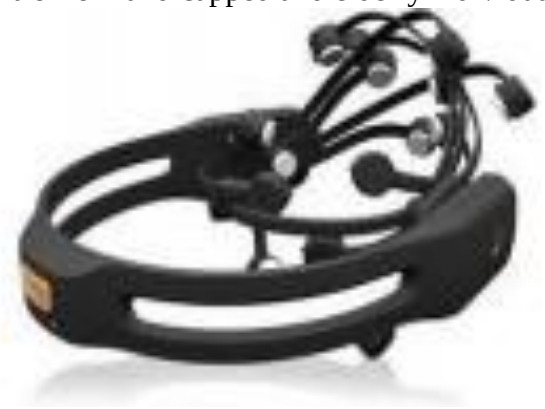


Figure 2.1: Emotive EPOC headset [10]

The control of an electric-powered wheelchair by using head movement is done with the support of an EEG sensor called emotive EPOC headset. For the most part, there are two control modes they are i) One head development mode ii) Four head development mode. With a specific end goal to recognize the head developments a whirligig is set in the emotive EPOC head set. The EPOC headset comprises 14 saline anodes. The game plan of anodes are finished by 10/20 outline and their parts are AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8 andAF4.Emotive EPOC headset provides four commands for controlling purpose: forward, right, left and stop. The gyroscope in Emotive EPOC headset has two axis X and Y through which the EPOC headset receives the information about the movements of the head. The x and y axis detects the horizontal and vertical movements of the head respectively. The negative and the positive represents the left and right movements of the head respectively. HMI proposed the four head developments based regulator which utilizes the movement information by a two-hub whirligig inside the Emotive sensor to recognize the movements. In this way, brightening impact is removed. The device offers four mechanisms; each command is given by another head movement, in the same way, four head advancements are necessary to regulate the wheelchair. The patients may simply have the ability to achieve each head gestures, thus the approach was planned for operating the wheelchair using just a single head advancement.

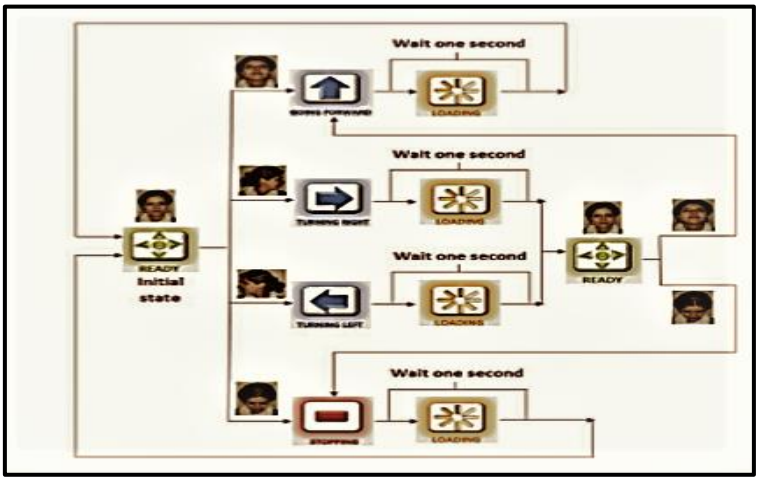


Figure 2.2: Algorithm of the four head movements based control mode
Source: (Paper from sysiass.eu)

The head movement based HMI was implemented by using the free hands to control the EPW. The whole setup is to identify the motion by using the gyroscope sensor. There are two controller modes to control the head motion. One mode is employed to control up and down movements whereas the second mode will is used to control the forward, right, left and stop directions. Overall the modes will control the head movements with the commands and the result will be displayed at HMI while controlling the wheelchair.

2.2.2 Tongue movement based operation of support system for paralysis patients

The tongue movement based wheelchair control is a decent technique in light of the fact that amid spinal rope wounds tongue, for the most part, escapes from this as it is associated with mind by hypoglossal nerves. Tongue development is quick when contrasted with different strategies and it is exact no need of much considering this was proposed by Junji Takahashi *et al*.[6]. Despite the fact that in resting position tongue can be effectively moved by the client. This technique gives another control gadget in light of tongue movements to control and speak with an emotionally

supportive network for an incapacitated patient. The tongue is one of the capable parts of the movements and is not affected by spinal cord
damage. The tongue movement is effectively seen from his/her mouse inside, it is, be that as it may, hard to watch them from outside. For the most part, there are two segments Transmitter segment and collector area. This framework utilizes a variety of Hall Effect sensors and a changeless magnet. The clients summon is changed over to control order by identifying the tongue movements. At the point when the client moves the tongue, the attractive field made by the perpetual magnet likewise shifts and these varieties are distinguished by Hall Effect sensors and relying upon the quality of attractive field the yield will fluctuate. The yield of the sensor is simple so a simple to the advanced converter is utilized to change it into the computerized flag. The microcontroller shown in the transmitter area is furnished with predefined esteems, and the microcontroller will contrast the output of the sensor and the predefined esteem and as indicated by the programming it will distinguish which summon the client have created. The microcontroller will send the certain order to the transmitter and the encoded information is transmitted remotely. Collector will get the information and decodes it and gives to the microcontroller of the receiving section. The microcontroller controls the development of wheelchair. With the assistance of DC engines wheels of the wheelchair shown rotates. A dual full bridge is as of now stacked in microcontroller utilizing Embedded C programming.

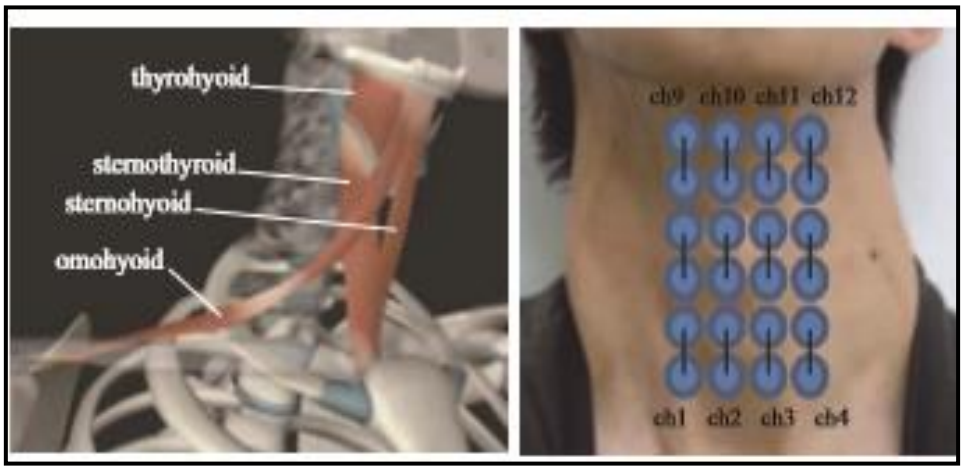


Figure 2.3: Anatomical figure of muscles related to tongue and the electrodes position [6]

Depending on the input microcontroller will deliver predefined logic to the dual full bridge driver. Driver IC in go will control the rotation of DC motor (clockwise and anticlockwise rotation) due to which wheelchair can move in a left, right and forward direction. In this method, we are using five commands. Out of this three of them are directional commands that are LEFT, RIGHT, FORWARD and two of them are selection commands that is Stand-by and Active Mode. When driving PWCs, FORWARD is used to change the wheelchair forward, while LEFT and RIGHT are used to turn left and right individually. To disable the system during eating and talking we can shift the TDS from active to stand-by mode, during which wheelchair will remain static.

2.2.3 Head mounted input device using MEMS sensor

Head mounted input device using MEMS sensor was demonstrated by V. Anbarasu *et. al*. The prototype [11] system designed to create a cost-effective, less power hungry, portable and small device substitute for a mouse that can help the disabled. Handheld devices such as Personal Digital Assistant (PDA) and smart phones are now widely used for many

of our everyday tasks. However, there are at least two reasons that make the interaction on those devices difficult compared to desktop interfaces: small screen size and limited computing power. Pointing and scrolling are one of the most extensively used tasks in almost all computing applications. The environment in which mobile devices are used is different from that in which desktop is used. The user needs to manage the environment while using the device: holding something, writing notes, opening doors, etc. Also, some users cannot use both hands due to other reasons such as disability or accidents. So, we think that freeing one hand from interaction with the device is very useful. In this work, they study the use of tilt modality for pointing and scrolling on mobile devices. The effectiveness of using tilting in the two tasks then a model for predicting their execution time is developed.

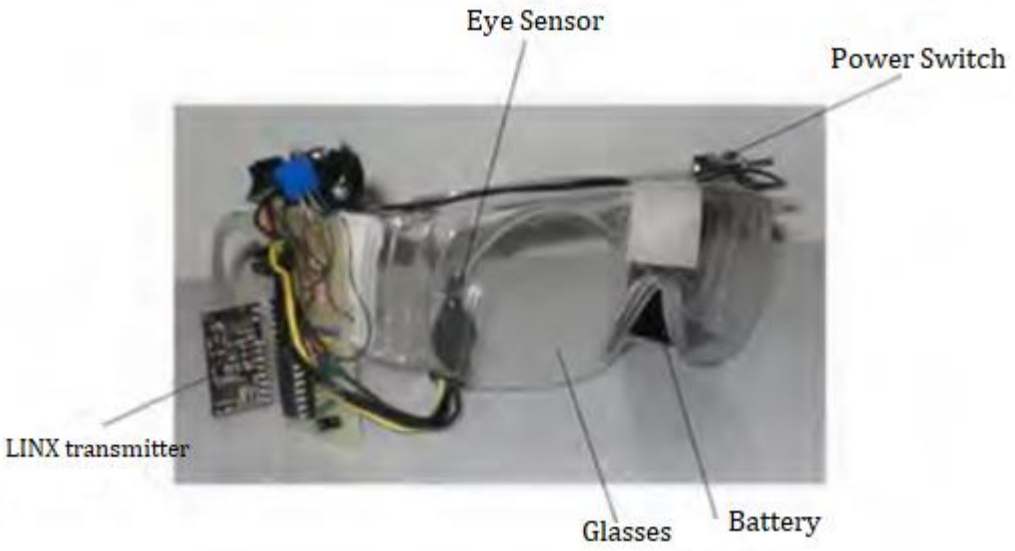


Figure 2.4: Wearable wireless spectacle prototype [11]

This paper describes the implementation of head operated mouse that uses tilt sensors placed on the headset to determine head position. Also, it uses 3 axis MEMS accelerometers to detect head tilt in order to direct mouse movement on the monitor. Accelerometer sends the information to the microcontroller. The microcontroller then passes the actual information to the encoder. Information encoded is then sent using Transmission to ZigBee receiver. ZigBee receiver will decode the received information. The microcontroller sends to PC through a RS232 cable. It will perform the operation. Same operation for selecting any documents with the help of eye blink. We constructed an interface system that would allow a similarly paralyzed user to interact with a computer with almost full functional capability. That is, the system operates as a mouse initially, but the user has the ability to toggle in and out of a keyboard mode allowing the entry of text. This is achieved by using the control from a single eye, tracking the position of the pupil for direction, and using blinking as an input. The proposed paper describes the design of a system that is compatible with all operating systems.

2.2.4 Hand gesture recognition input device using MEMS sensor

A nonspecific person gesture recognition system by using MEMS accelerometers was presented by R. Xu. *et al.* proposed a paper [4] which describes the recognition system which consists of sensor data collection, segmentation, and recognition. After receiving acceleration data from the sensing device, a segmentation algorithm is applied to determine the starting and end points of every input gesture automatically. The sign sequence of a gesture is extracted as the classifying feature, i.e., a gesture code. Finally, the gesture code is compared with the stored standard patterns to determine the most likely gesture.

Since the standard gesture patterns are generated by motion analysis and are simple features represented by 8 numbers for each gesture, the recognition system does not require a big data base and needs not to collect as many gestures made by different people as possible to improve their cognition accuracy.

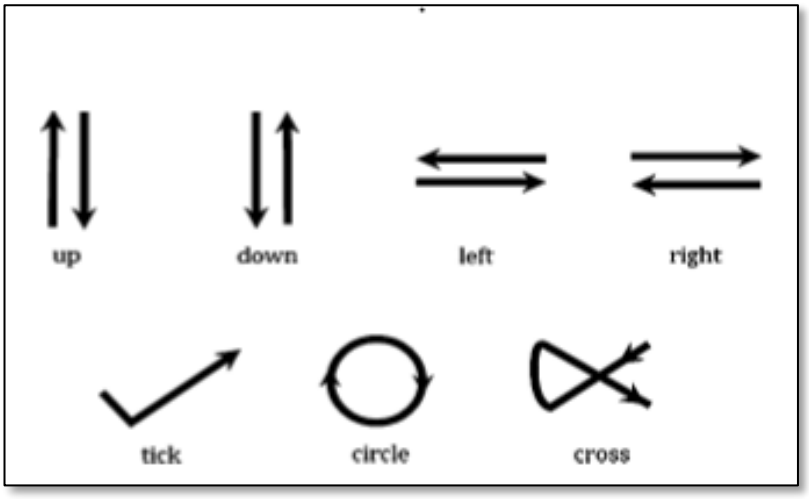


Figure 2.5: Motions of seven gestures [4]

When the sensing system is switched on, the accelerations in three perpendicular directions are detected by the MEMS sensors and transmitted to a PC via Bluetooth protocol. The gesture
Motion data then go through a segmentation program which automatically identifies the start and end of each gesture so that only the data between these terminal points will be processed to extract feature. Subsequently, the processed data are recognized by a comparison program to determine the presented gestures. The work flow of this system is shown in the figure below.

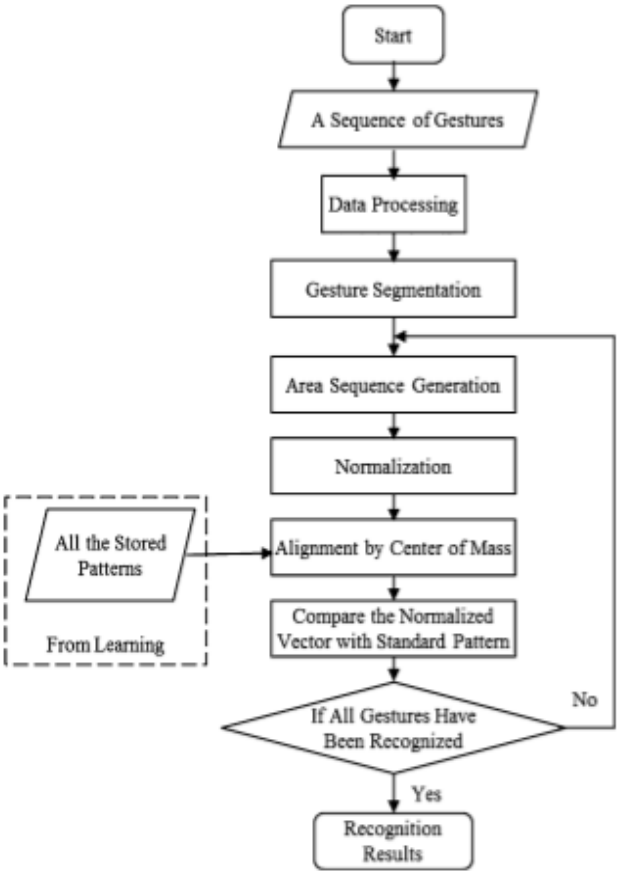


Figure 2.6: Flow chart for hand gesture recognition [4]

2.2.5 A wearable system for radio-based sensing of head and mouth-related activities

According to B. Fang *et al.* the HeadScan [12], a first-of-its-kind wearable for radio-based sensing of a number of human activities that involve head and mouth movements. HeadScan only requires a pair of small antennas placed on the shoulder and collar and one wearable unit worn on the arm or the belt of the user. HeadScan uses the fine-grained CSI measurements extracted from radio signals and incorporates a novel signal processing pipeline that converts the raw CSI measurements into the targeted human activities. To examine the feasibility and performance of HeadScan, they have collected approximate 50.5 hours data from seven users. Over wide-ranging experiments include comparisons to a conventional skin-contact audio-based sensing approach to tracking the same set of head and mouth-related activities. The experimental results highlight the enormous potential of our radio-based mobile sensing approach and provide guidance to future explorations.

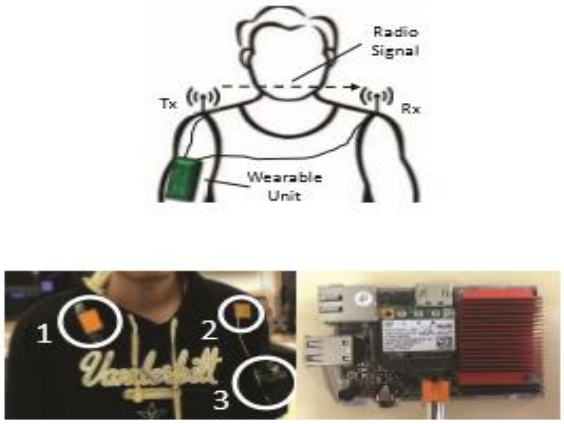


Figure 2.7: Head Scan wearable prototype [12]

HeadScan wearable prototype consists of two small (21 x 21 mm; 5 g) unobtrusive commercial off-the-shelf (COTS) 5GHz antennas, that can be worn on the shoulder and the collar respectively, as well as one wearable unit (85.0 x 60.0 x 36.0 mm; 390 g) that can be worn on the arm or the belt of the user. The two antennas are wired to the wearable unit. The figure provides a conceptual illustration of how HeadScan is worn while it tracks the movements of head and mouth of the user. The wearable unit consists of two Humming Board Pro (HMB) devices, each powered by one 3000 mAH battery. The HMB is a low-cost ARM-based mini-computer that contains an on-board 1.2 GHz ARM Cortex-A9 processor and 1GB RAM. Each HMB is connected to an antenna as well as an Intel WiFi Link 5300 card via the micro PCI socket for measuring CSI (Figure 2). In our prototype, two HMB is needed because a single HMB cannot support two Intel WiFi Link 5300 cards at the same time. One HMB acts as the radio transmitter that sends packets periodically and the other HMB acts as the radio receiver that continuously receives the packets. We installed the modified firmware released by the CSI measurement tool through the Debian Cubox-i Linux OS running on the receiver HMB, enabling the Intel WiFi Link 5300 card to extract CSI measurements from the received packets on the receiver HMB. During operation, the user uses a custom case to hold the wearable unit and attach it to the arm of the user with an elastic armband for data collection. The collected CSI measurements are exported to a desktop computer for offline processing.

2.2.6 Home automation systems control by head tracking in AAL applications

S. Spinsante *et al*. proposed a research paper[13] which is organized as follows: Section I provides details about the software application designed for head tracking, based on the OpenCV library for computer vision; Section II gives a brief overview of the reference assistive home automation system, the control of which is implemented by the application developed; Section IV discusses the main constraints to satisfy, related to the specific AAL context of application, and presents experimental tests and results; finally, Section V concludes the paper. This paper presents a software application based on computer vision techniques, designed to allow the control of an assistive home automation system, by tracking the head movements of the subject. The application includes both an engine, i.e. a set of algorithms, to perform real time video analysis, segmentation, and head tracking, and a Graphic User Interface (GUI) to interact with the user and issue commands towards the automation system. The GUI design took into account the requirements and the performance provided by the engine, to maximise the number of correct hits and minimise the number of false commands issued. Software applications to perform head tracking is not new, however, their adoption in home automation systems control is quite innovative, and, most of all,

the proposed implementation has been developed by taking into account the computational constraints posed by the target devices on which it is expected to run. As a matter of fact, in order to facilitate a truly spread of the proposed application, and to help the interested users in getting the technology required, commercial devices such as tablets have been chosen as the target execution platform: this way, it is possible to exploit the availability of an onboard and embedded digital camera, and the reduced burden of these devices, with respect to traditional laptop-based solutions. The paper presented a computer vision software application designed to allow the interaction between a user and a home automation system, through head movements tracking. The application runs a tracking algorithm based on skin color recognition, thus limiting the computational complexity of the process. Even if strong dis-uniformity in the luminance distribution may limit the application performance, experimental results show that the correct commands are issued in more than 75% of the total attempts.

2.3 Artificial neural network

An Artificial Neural Network (ANN) is a computational model that is inspired by the way biological neural networks in the human brain process information. Artificial Neural Networks have generated a lot of excitement in Deep Learning research and industry, thanks to many breakthrough results in speech recognition, computer vision, and text processing. In this study, we will try to develop an understanding of a particular type of Artificial Neural Network called the Multi-Layer Perceptron[22].

Multilayer Perceptron (MLP**):** It is a supervised learning algorithm that learns a function $f(\cdot): R^m \rightarrow R^O$ by training on a dataset, where *m* is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $X = x_1, x_2 \ldots\ldots, x_m$and a target, *y* it can learn a non-linear function approximate for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure2.8 shows a two hidden layer MLP with the output.

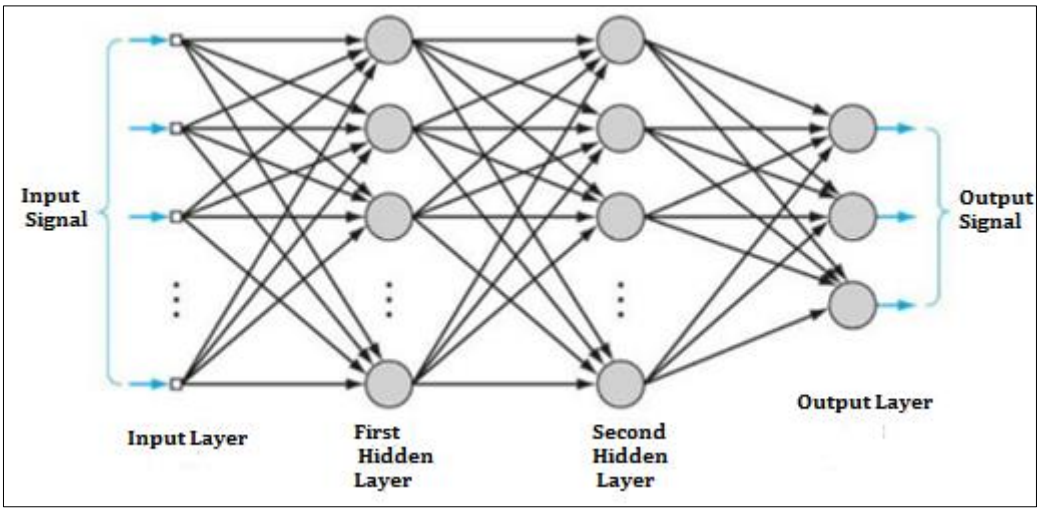
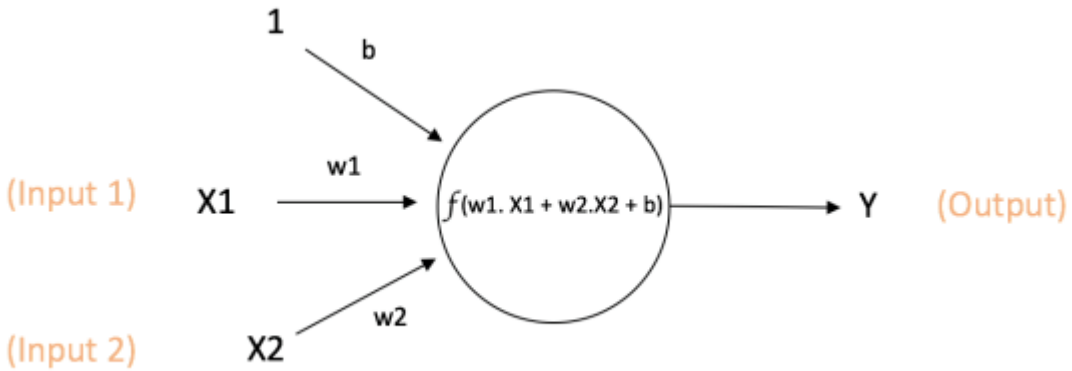


Figure 2.8: Two hidden layer MLP

The leftmost layer, known as the input layer, consists of a set of neurons $\{x_i | x_1, x_2 \ldots\ldots, x_m\}$ representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation, $w_1x_1 + w_2x_2 + \cdots + w_mx_m$ followed by a non-linear activation function $g(.): R \rightarrow R$ - like the hyperbolic tan function. The output layer receives

the values from the last hidden layer and transforms them into output values. MLP trains using Backpropagation. More precisely, it trains using some form of gradient descent and the gradients are calculated using backpropagation.

Backpropagation Algorithm [23]: Initially all the edge weights are randomly assigned. For every input in the training dataset, the artificial neural networks are activated and its output is observed. This output is compared with the desired output that we already know, and the error is "propagated" back to the previous layer. This error is noted and the weights are "adjusted" accordingly. This process is repeated until the output error is below a predetermined threshold. Once the above algorithm terminates, we have a "learned" artificial neural networks which, we consider is ready to work with "new" inputs. This artificial neural network is said to have learned from several examples (labeled data) and from its mistakes (error propagation). The basic unit of computation in a neural network is the neuron, often called a node or unit. It receives input from some other nodes, or from an external source and computes an output. Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs. The node applies a function *f* (defined below) to the weighted sum of its inputs as shown in Figure 2.9.



Output of neuron = $Y = f(w1.X1 + w2.X2 + b)$

Figure 2.9: A single neuron

The above Figure2.9 network takes numerical inputs X1 and X2 and has weights w1 and w2 associated with those inputs. Additionally, there is another input 1 with weight b (called the Bias) associated with it. The output Y from the neuron is computed as shown in Figure 2.8. The function *f* is non-linear and is called the Activation Function.

- **Activation Function:**

It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function). The purpose of the activation function is to introduce non-linearity into the output of a neuron. This is important because most real-world data is nonlinear and we want neurons to learn these nonlinear representations. Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it. There are several activation functions[23] and two of the most common ones are the sigmoid and the tanh function nonlinearities that are symmetric around the origin are preferred because they tend to produce zero-mean inputs to the next layer (which is a desirable property), we have observed that the tanh has better convergence properties. But in this study, we have used tanh activation function. We used tanh in this study because it typically yields to faster training.

Sigmoid: takes a real-valued input and squashes it to range between 0 and 1

$$\sigma(x) = 1 / (1 + \exp(-x)) \quad (1)$$

tanh: takes a real-valued input and squashes it to the range [-1, 1]

$$\tanh(x) = 2\sigma(2x) - 1 \quad (2)$$

ReLU: ReLU stands for Rectified Linear Unit. It takes a real-valued input and thresholds it at zero (replaces negative values with zero)

$$f(x) = \max(0, x) \quad (3)$$

The below Figures 2.9 show each of the above activation functions.

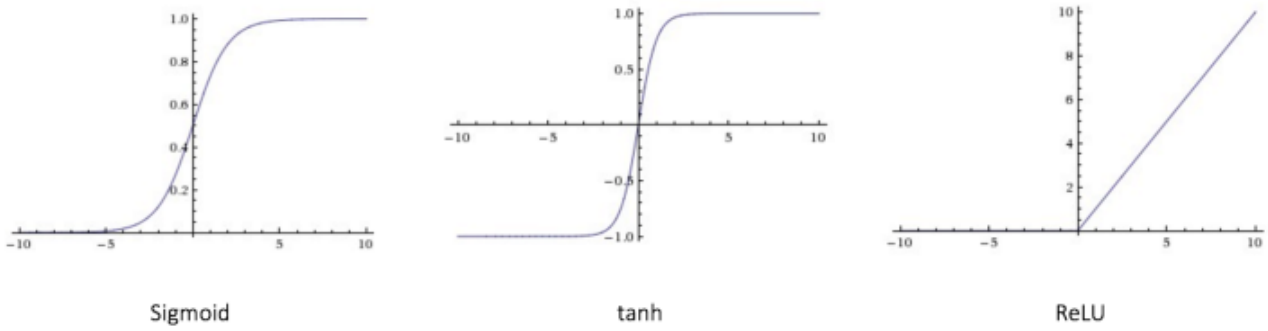


Figure 2.10: Different activation functions

- **Hidden Layer Selection:**

For non-linear nonseparable model classification problem and non-linear function approximation problem, we can use multi-layer perceptron. First, the concept of decision-making boundary is introduced, and the decision boundary is the dividing line, interface, or dimensional boundary of the classification problem. Assuming that our subject belongs to the multi-classification problem, the decision boundary is the dimensional boundary so at least choose two layers. The increase of the hidden layer will increase the computing time, considering we choose the two hidden-layer perceptions.

- **The solver for weight optimization:**

Adam,[24] an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to a diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. Some connections to related algorithms, on which Adam was inspired, the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best known results under the online convex optimization framework. The results demonstrate that Adam works well in practice and compares favourably to other stochastic optimization methods. MLP trains using Adam, Stochastic Gradient Descent, or L-BFGS. Adam updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation. Adam is an optimization algorithm that can use instead of the classical stochastic gradient descent procedure to update network weights iterative based on training data. Adam is similar to SGD in a sense that it is a stochastic optimizer, but it can automatically adjust the amount to update parameters based on adaptive estimates of lower-order moments. With Adam,

training supports online and mini-batch learning. Therefore in this study, we have used the solver 'Adam' which works well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score.

2.4 Performance of head movement detection methods

The head movement detection accuracy, angle accuracy of the different head movement detection methods are summarized in Table 1.1. The three different methods were considered in in the three methods there are four journals with the accelerometer and gyroscope methods and a journal based on the radio-based sensing and five papers with the computer vision-based head tracking method.

Table 2.1 Performance of different head movement detection methods

Methods	Detection Accuracy (%)	Angle Accuracy (degree)
Accelerometer and Gyro-Sensor based methods		
Nguyen *et al.* [14]	93.75%	N/A
King *et al.* [15]	97.50% for Disabled users & 99.85% for able-bodied users.	N/A
Nina Rudigkeit *et al.* [16]	N/A	N/A
Mohammed Faeik *et al.*[17]	97.4%	N/A
Radio-Based Sensing of Head		
Biyi Fang et al. [12]	86.3%	N/A
Computer-vision-based Head Tracking		
Zhao and H.Yan [18]	90%	N/A
Siriteerakul *et al.* [19]	N/A	3.32
Zhao *et al.* [20]	92.86%	N/A
Vamsikrishna *et al.* [21]	N/A	N/A
Susanna Spinsante *et al.* [13]	75%	N/A

2.5 Summary

The proposed system was compared with the various systems and for the relevant comparison five journals were studied and compared with the proposed system in the Table 1.2 and here by concluded that the proposed system is comparatively more accurate and easy to use and it is the most suitable device for the quadriplegia patients as is the user friendly device and can be the most conveniently used device with the maximum accuracy and not only in terms of accuracy but also in the application purpose, with the single head movement they can operate any device and can ask for any requirement and can save them from an emergency cases therefore in the application wise it is the most useful device for the paralysis patients.

Journal 1- R. Xu, S. Zhou, and W. J. Li, [4] implemented a head movement tracking system MEMS accelerometer is used which cannot measure either static accelerations or ones occurring at the low frequencies hence the output is not as accurate as IMU. The gesture recognition models which are capable of recognizing seven hand gestures.

Journal 2- S. Spinsante and E. Gambi, [13] Computer vision techniques, designed to allow the control of an assistive home automation system, by tracking the head movements. An approach for head movement detection is computer vision-based introduced a video-based technique for estimating the head pose and used it in a good image processing application for a real-world problem.

Journal 3- B. Fang, N. D. Lane, M. Zhang, and F. Kawsar, [12].Head Scan, is a kind of wearable for radio-based sensing of a number of human activities that involve only head and mouth movements. It only recognizes eating, drinking, coughing, and speaking activities.
Journal 4- L. King, H. Nguyen, and P. Taylor, [15] presents the head-movement gesture classification using neural networks. The data was collected using a dual axis accelerometer.
Journal 5- M. F. Ruzaij, S. Neubert, N. Stoll, and K. Thurow, [17] this paper is an auto-calibrated head orientation controller using the MEMS sensors.

Table 2.2 The comparison between the proposed system with the other systems

Proposed System	Journal 1	Journal 2	Journal 3	Journal 4	Journal 5
1. In the proposed system the Inertial Measurement Units (IMU) 9DOF is used which normally, contains accelerometers, gyroscopes, magnetometer sensors. The IMU sensors also used in aerospace applications, gyroscopes provide a very accurate estimation with a low drift.	[4]MEMS the accelerometer is used which cannot measure either static accelerations or ones occurring at the low frequencies hence the output is not as accurate as IMU.	[13]Computer vision techniques, designed to allow the control of an assistive home automation system, by tracking the head movements.	[12] Head Scan, is a kind of wearable for radio-based sensing of a number of human activities that involve only head and mouth movements.	[15] Presents the head-movement gesture classification using neural networks. The data was collected using a dual axis accelerometer.	[17] This paper is an auto-calibrated head orientation controller using the MEMS sensors.
2. There are totally eight possible outputs in the proposed device.	The gesture recognition models which are capable of recognizing seven hand gestures.	The commands generated by head tracking are working as a joystick, to move the focus left/right and up/down on the GUI.	It only recognizes eating, drinking, coughing, and speaking activities.	The head-movement classifications are done among the four basic head movements.	The system design to implement control command for wheelchair motors depending on the user head movements.
3. It is a head-mounted device mainly proposed for patients suffering from Quadriplegia. This device is a wearable headband worn on the head.	The device is tied to the hand. This device is not applicable for Quadriplegia patients.	It works totally on the software base. Even if strong dis-uniformity in the luminance distribution may limit the application performance.	The wearable unit is attached to the arm of the user with an elastic armband for data collection.	The user is prompted for one of four gestures using a dual axis accelerometer mounted inside a hat the user wore on their head.	Orientation module is fixed on the user's head as a part of a small wearable device.
4. Artificial neural networks with Multilayer perceptron	A recognition algorithm based on sign sequence and	Canny algorithm, to identify objects and their shapes	Head Scan uses the fine-grained CSI measurements	The head-movement gesture classification	Auto-calibrated orientation control algorithm, the

algorithm is used for detecting the eight head movements and classifying them.	template matching is used.	within images. Lucas-Kanade, Horn, and Schunck algorithms, for motion detection, object tracking, and block matching in video sequences.	extracted from radio signals and incorporates a novel signal processing pipeline that converts the raw CSI measurements into the targeted human activities.	system is shown using a Neural Network employing the Magnified Gradient Function (MGF) algorithm.	system uses two Orientation Detection (OD) units.
5. Each action of the patient is sent as the text message via GSM to the caretaker and the audio and the display of their action is given to avoid confusion and to make the device more user-friendly for the patient.	The gesture code is compared with the stored standard patterns to determine the most likely gesture.	The real-time video analysis, segmentation, and head tracking, and a Graphic User Interface (GUI) to interact with the user and issue commands towards the automation system.	Head Scan requires a pair of small antennas placed on the shoulder and collar and one wearable unit worn on the arm or the belt of the user which may not be comfortable for the user.	The results in the paper only show that the MGF is suitable for use in the detection of head movement gestures.	The user cannot operate any other devices with the help of the head except to control the wheelchair motors.
6. The overall accuracy of the head movements of the headband is 86.04%.	The overall mean accuracy is 95.6%, with the recognition accuracy of each gesture above 90%.	The experimental results show that the correct commands are issued in more than 75% of the total attempts.	The targeted head and mouth-related activities (i.e., eating, drinking, coughing, and speaking) with an average accuracy of 86.3%.	The overall accuracy results for the normal people or able bodies is 99.85% whereas for the disabled bodies is 97.50%.	The result shows that the control performance with accuracy is 97.4%. But the system fails to provide higher safety conditions such as obstacle avoidance as it is designed for handicapped and elderly users.

CHAPTER 3
METHODOLOGY

3.1 Working

From the working flowchart, as shown in Figure 3.1, each stage from the flow chart is discussed and explained in this manner.

i. Study and selection of modules

In this stage, the first idea is that selecting the suitable modules. There are advantages and disadvantages of selection of a particular module hence before selecting study and research about the proper suitable modules.

ii. **Concept design and concept selection**

There are many details discussed for the design of the head belt such as the size of the headband and the location where to mount the sensor on the head and the quality of the headband and the required details regarding the design.

iii. Working with each individual modules

To understand about each individual components and get familiar by working with them because directly cannot fix them without knowing about the module behavior under each condition and also communicating each module with one other in fact develops the new ideas and also makes the work easier.

iv. Detailed design and creating prototype

Some algorithms were referred and the suitable algorithm was selected and implemented for the head belt. The designing of the head belt and fixing the communication among each module.

v. System experiment and modifications

After the total head belt is finished, the experiment is designed and tested in this section. Other elements will be discussed for making the system more accurate.

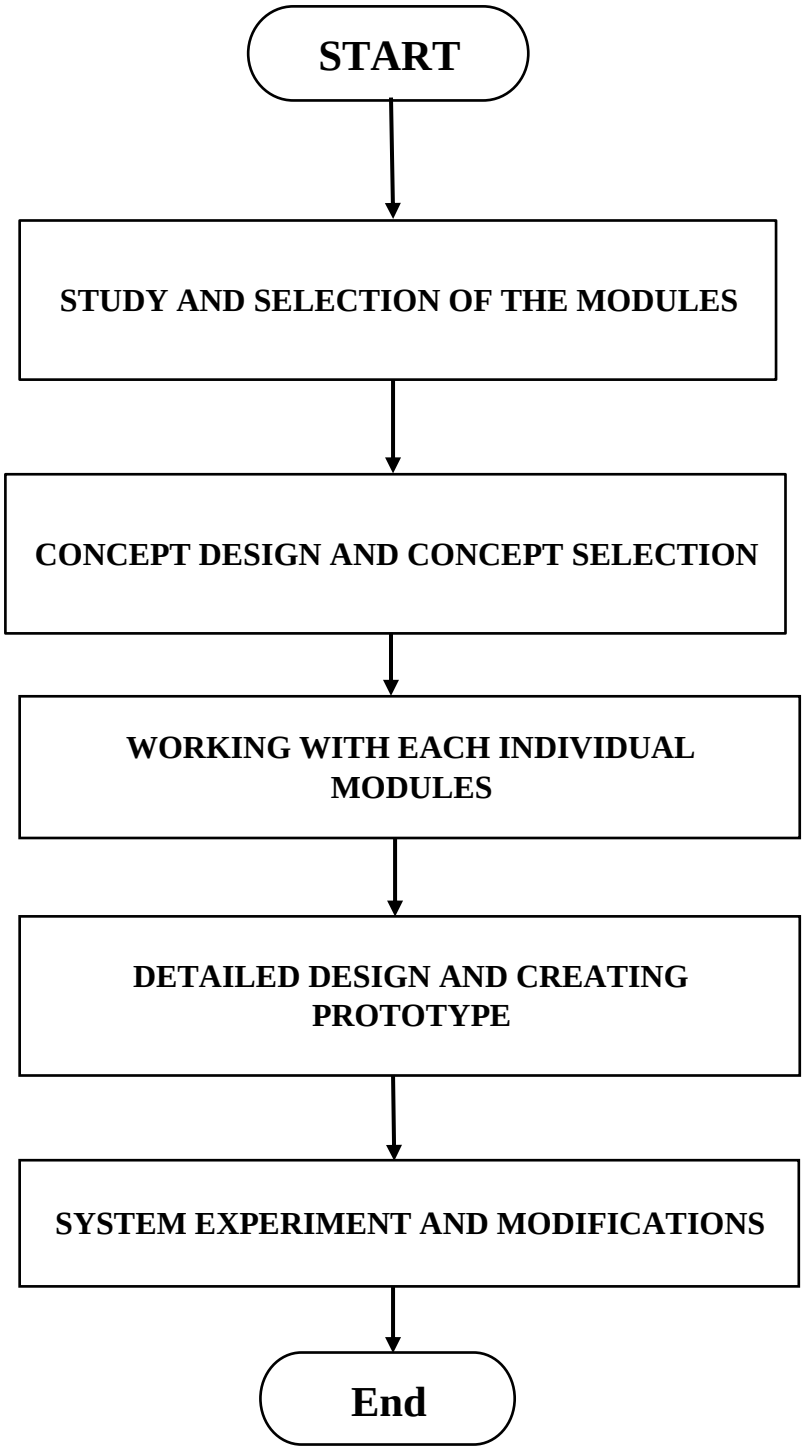


Figure 3.1: Methodology flowchart

3.2 System overview

The project contains totally two parts one is the transmitter and second one is a receiver. The transmitter part is incorporated with IMU razor, XBee module. This section is placed on the patient head. The receiver part contains the raspberry pi module, GSM, LCD, relay and a buzzer. Once we place this sensor on the patient's head, the sensor output changes according to the patient head moments. This sensor output is given to the XBee receiver via XBee transmitter which will pass

the data to the raspberry pi and the pi will give the outputs according to the instructions given by the sensor. The GSM module to send the message to the doctor or to the caretaker and depending on the patient moments patient requesting requirements using voice and the GSM module are playing them on the receiver side. For example, if patient head moves in forwarding direction the sensor output Rx, Ry, Rz when this value reaches threshold value in the program (transmitter side) will send the text message to the receive part, by receiving this message the receiver will give the signal to the voice module the voice module will play the desired voice which was saved to the memory location. Hence the remaining actions can be developed by the different types of head moments and different types of sensors outputs in this manner.

3.3 System design

The section explains about the whole system in the block diagram representation. The whole system was divided into two sections the transmitter section and the receiver section. It is very simple to construct the block diagram for the big and complicated systems, therefore, it is the most preferred way of explaining the system in the block diagram representation. The functional operation of the system was clearly observed from the block diagram and also it provides the information about the performance of the system.

Transmitter section

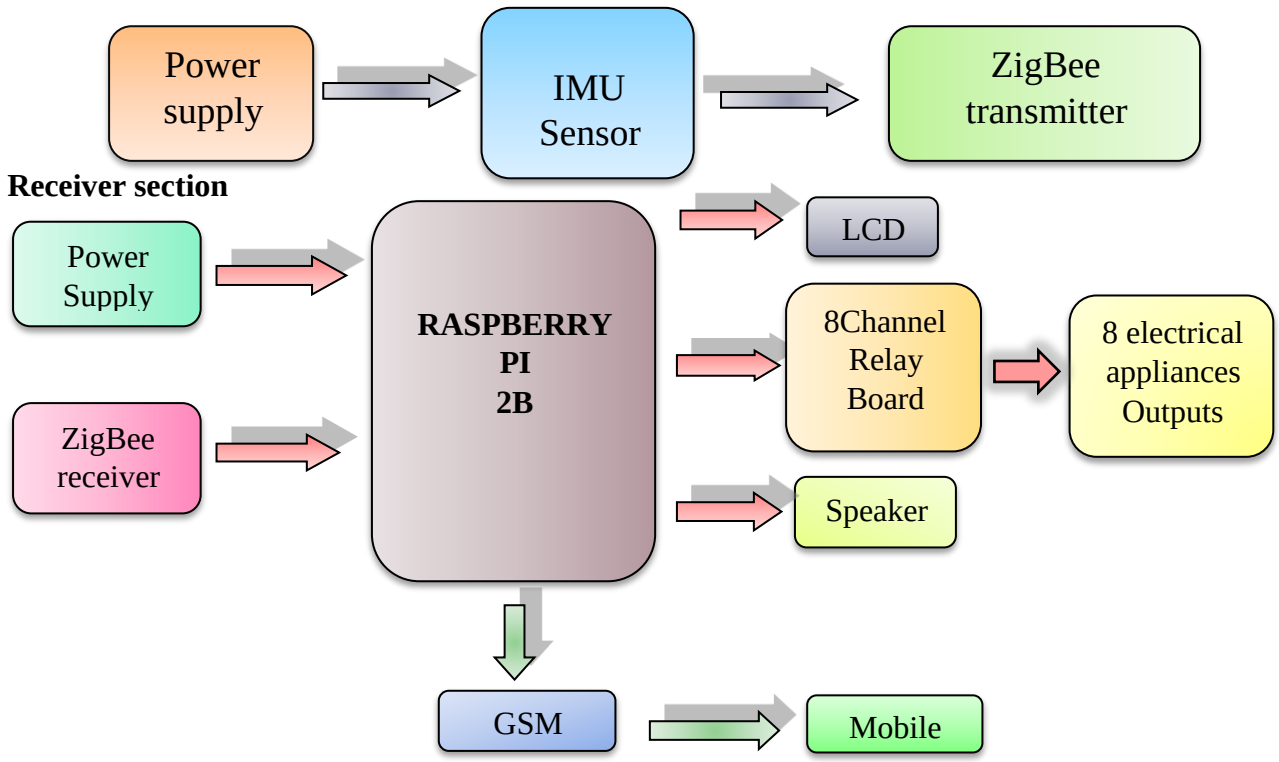


Figure 3.2: Block diagram

The project contains totally two parts one is the transmitter and second one is a receiver. The transmitter part is incorporated with IMU razor, Xbee module. This section is placed on the patient head. The receiver part contains the raspberry pi module, GSM, LCD, relay and a buzzer. Once we place this sensor on the patient's head, the sensor output changes according to the patient head moments. This sensor output is given to the XBee receiver via XBee transmitter which will pass the data to the raspberry pi and the pi will give the outputs according to the instructions given by the sensor. The GSM module to send the message to the doctor or

to the caretaker and depending on the patient moments we are playing the patient requesting requirements using voice and the GSM module on the receiver side.

3.4 System construction

The proposed system is a part of a multi-input control system that uses different available body signals to control any intelligent application. The system design takes into consideration that it can be used easily and comfortably by quadriplegia and elderly patients. It includes also a voice output speaker, which gives the user the option to abort the head tilts when necessary. Figure 3.2 shows the block diagram of the system. It includes the core microcontroller and the main units and sensors. The system blocks and structures of each module will be explained in detail in the next section.

A. Microcontroller Unit

Since the proposed system is a part of a multi-input control system a careful selection of the microcontroller unit is required. It represents the "brain" of the system. It must have special characteristic and specification to cover the required input/output ports and peripherals for the system design. This will make the interface between the microcontroller and the input and output units more flexible. Raspberry Pi has been selected to be the main controller of the system. This powerful microcontroller is one of the most energy friendly products. It has all the required peripheral communication ports for the system design, including General Purpose Input Output (GPIO) pins. The wide range of ports diversity makes the system updates more easily.

B. Orientation Detection Unit

The orientation detection unit consists of the 9-axis Freedom Degree IMU Razor Module which is inbuilt with the three sensors those are Atmega328ITG3205, ADXL345, HMC5883L hence each sensor are triple axis respectively therefore altogether the module is said to be the nine-axis i.e. Nine degrees of freedom (9dof).

C. Output Unit

16x2 LCD display is used to view the control commands and the currently activated control mode. The buzzer is used for the indication of each activity selection. Voice output is included for each head movement selection for easily recognizing the particular action in this way patient would have clear idea of the device operation. GSM module is used to send a text message to the caretaker.

3.5 Hardware components specifications

3.5.1 9DOF Razor IMU

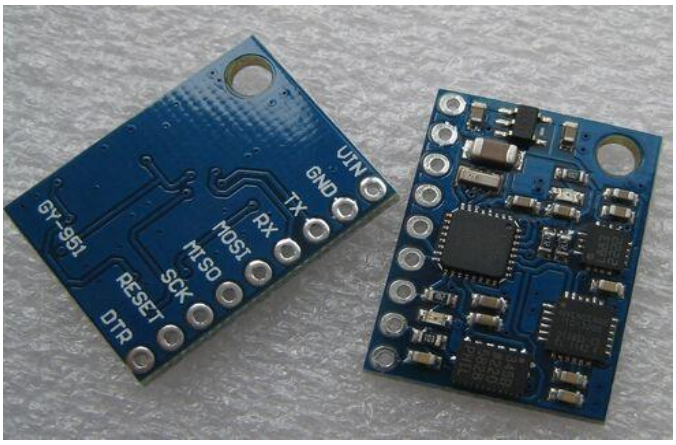


Figure 3.3: IMU Razor
Source :(eBay.com)

The 9-axis Freedom Degree IMU Razor Module which is inbuilt with the three sensors those are Atmega328ITG3205, ADXL345, HMC5883L hence each sensor are triple axis respectively therefore altogether the module is said to be the nine-axis i.e. Nine degrees of freedom (9dof). The device's output is taken by on-board ATmega328 via a serial interface. The device is programmed with the 8MHz Arduino with some firmware which gives an output of all the sensors by simply connecting the RX TX with 3.3FTDI then set the 57600bps for testing purpose. It can also be connected to XBee for wireless operations. The 9dof razor is the powerful controller for UAV's vehicles and also stabilization systems.
Specifications:
Power supply 3-5v
Serial port communication protocol
Reset switch and control switch for ON OFF

3.5.1.1 IMU with Three Sensor Types

This type of IMU consists of three sensors: accelerometer, gyroscope, and magnetometer. The three sensors have tri-axial to get measurements in three different axes to make the total of 9 DOF. The magnetometer is used to measure yaw angle rotation, thus it can be calibrated to the gyroscope data to improve the big drift issue. The importance of this type of sensor appears when it used for dynamic orientation calculation in the short and long run when fewer drift errors occur. One disadvantage of using this sensor. If the IMU is being used in the environment that is surrounded by ferromagnetic, the measurements might be affected due to the disturbance to the magnetic field.

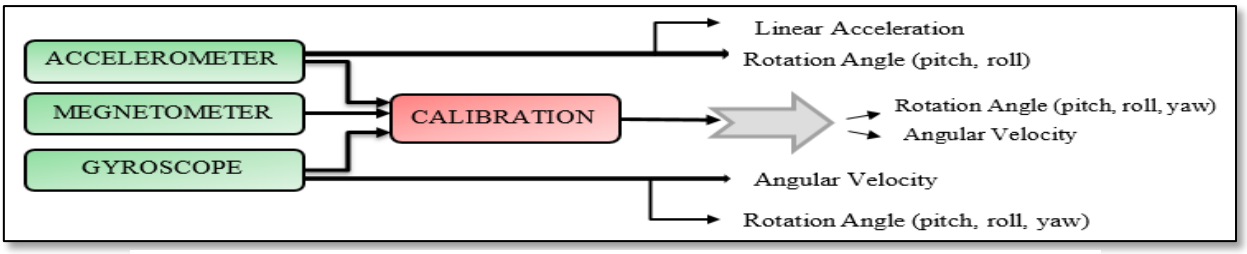


Figure 3.4: IMU based on three sensors

3.5.2 Introduction to raspberry pi

Figure 3.5: Raspberrypi 2B module
Source: (raspberrypi.org)

The one-word explanation of Raspberry Pi is a mini plug and play computer. The Raspberry Pi is an energizing gadget that will ideally bring PC programming (and Linux) to the masses. The low-valued, charge card estimated single-board PC. It is a low controlled, ease web server, media focus, or SSH customer. The working frameworks are given as pictures that are flashed to an SD card. A parcel of Linux flavors is accessible for the gadget: Debian, Fedora and Arch Linux, Android 2.3, XBMC and so on.

An overview about Raspberry Pi:

i. Processor: The main core of Raspberry Pi is the processor which is BCM2835. It is of the 32bit system on chip with ARM11 architecture. RAM for B model is 512MB whereas for model A it is of 256MB.

ii. SD slot: Hard drive is not available on this raspberry pi, so the SD card is an option for booting the device. Connecting an external USB hard disk is possible with raspberry pi. You should have a minimum of 4GB SD card with class 4.

iii. USB Port: Model B there are two USB ports and pi A has only one USB ports. You can use your external powered hub so that you can use the additional USB peripherals.

iv. Ethernet port: To connect to the internet.

v. HDMI connector: HDMI port is provided for pi to connect it to other pc for the digital output and also for the audio purpose.

vi. Input power supply: Here the micro USB connector is used to supply power. This port is not for like any other purpose it is only for powering the pi. Micro USB power supply is selected to reduce the cost.

vii. Status LED's:

Table 3.1: Raspberry Pi has five status LED

	Colour	Function	Status
ACT	GREEN	SD CARD	When SD card is inserted it starts blinking
PWR	RED	Power	Steady power(3.3V)
FDX	GREEN	Full Duplex	On when Ethernet connection is full duplex
LNK	GREEN	Link	On when Ethernet is connected
100	YELLOW	100 Mbps	On if the internet connection is 100 Mbps

viii. GPIO (General Purpose Input and Output):

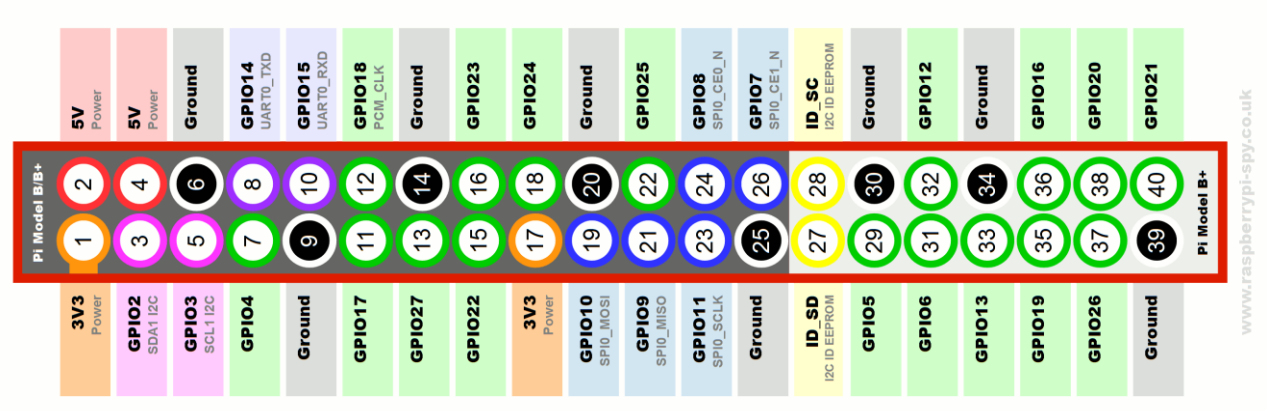


Figure 3.6: GPIO pins
Source: (raspberrypi.org)

One effective section of raspberry pi is the row of GPIO pins at the end of the board. GPIO remains for General-Purpose Input/Output. These GPIO pins are interfaced via pins between the Raspberry

and the external module. The GPIO pins can be easily operated because they act like switches which can be turned on off by using these pins. GPIO pins enable the Raspberry Pi to control and screen the outside world by being associated with electronic circuits. The Pi can control LEDs, turning them on or off, run engines, and numerous different things. It's additionally ready to recognize whether a switch has been squeezed, the temperature, and light. GPIO pins can be gotten to for controlling the equipment, for example, LED, Motors, LCD, and transfer. There are 40 sticks on the Raspberry Pi (26 sticks on early models), and they give different distinctive capacities.

ix. CSI (Camera Serial Interface):

This port allows you to connect the camera module directly to the board. Now the official camera module (5MP) is available. We can record and capture images using that camera.

- Various Distributions available for Raspberry Pi:

The raspberry pi uses the operating system called the Linux operating system. Variety of flavors of Linux OS is available. Common distributions are Ubuntu, Arch Fedora, and Debian. All these distributions have their individual community for users who are using for any of these specific applications.

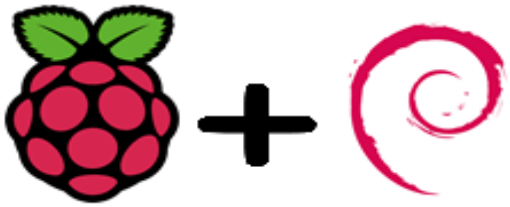


Figure 3.7: Debian+Raspberry pi=Raspbian
Source: (raspberrypi.org)

Raspbian is an "official recommended" distribution from the foundation based on Debian. We can directly download the pre-built image from the www.raspberrypi.org download section.

3.5.2.1 NOOBS (New out of Box Software) Installation

On the first boot, it will list out the choices of four operating systems including Raspbian, Pidora, Raspbmc, Openelec, Archlinux, RiscOS.
Raspbmc is the distribution for the people interested to use raspberry as an open source. The distribution is available from the following website.www.raspbmc.org
The Arch Linux is based on which is not suitable for the beginners, the latest version of the boots to command prompt in ten seconds. The pre-built image is available in raspberrypi.org download section.

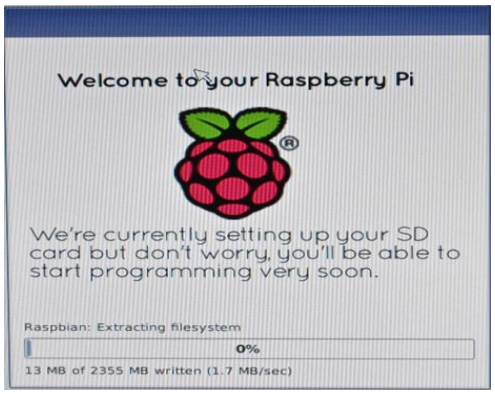


Figure 3.8: Files extracting

3.5.2.2 Flashing an SD card

Downloading an image:
The pre-built images are readily available in the http://www.raspberrypi.org/downloads .You can download it by using torrent or direct download. It will download as zip file format.

3.5.2.3 Porting an image to SD card

Extract the zip file and open it. Note the file should be ended with .img. Porting an image to the SD card is not like a drag and drop to the SD card. We should copy the image to the SD card using a special tool called win32diskimagewriter.I have downloaded the image from the following link. http://sourceforge.net/projects/win32diskimager/
Step 1:
SD card should be inserted into the card reader or the adaptor and open win32diskiamgewriter software. Ensure that the device drive is assigned or not as the image shown below.

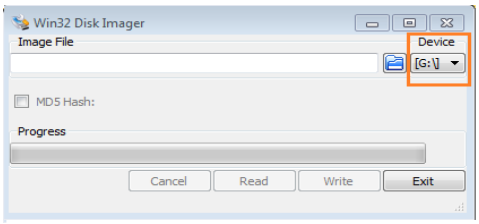


Figure 3.9: Selection of the device drive

Step 2:
Format SD card or else win32diskimager will hang and it may corrupt your card. Then click the browse button and select your corresponding Rasbian image file and click write. Selecting the extracted image file. Click write

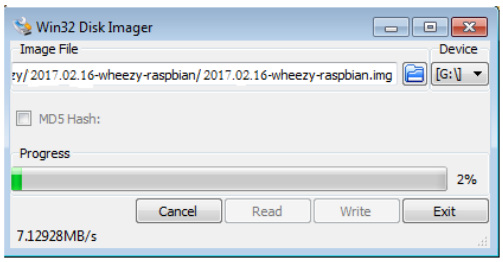


Figure 3.10: Extracting the image file

After 10-15 Minutes (depending on the class of SD card) it will display the write successful message.

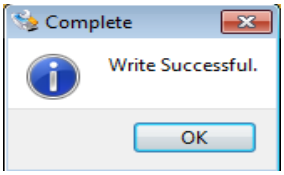


Figure 3.11: Write successful dialog box

Step 3. Booting up your Pi

- Insert the SD card in to SD slot of raspberry pi.
- Plug in the USB mouse and keyboard. For model B

- Plug in the power supply.
- If all goes well then it could able to see the log entries on the screen with raspberry pi log
- Now we can also see these log messages later by typing **dmesg** in the command prompt.

3.5.2.4 Configuring your raspberry pi

In the very first boot, it should open the raspi-config tool as per the image is shown below. You have to use up and down arrow key to move across the configuration tool.

Network: After connecting your raspberry pi to the network we have n number of utility to play with the raspberry pi. If you wish to check your IP address type the command **"ifconfig"** it will display the network interfaces and the IP address of the pi.

"Ping" command is a basic tool for troubleshooting your network convention. This command is mainly intended for network testing, measurement, and management.

Example**:** ping google.com, ping yahoo.com

3.5.3 GSM Module – SIM900A

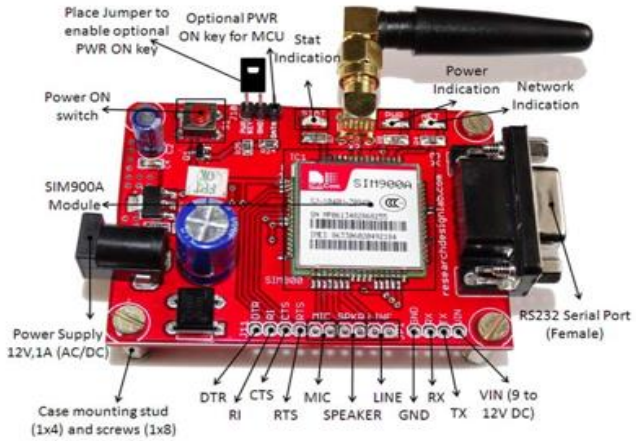


Figure 3.12: GSM module

Source: (researchdesignlab.com)

GSM module is a dual-band breakout board with the system of SIM900A. This GSM device will communicate with controllers by using the AT commands. It is a reliable ultra-compact wireless device. The processor in the module is responsible for the sim card which should be placed in the sim slot. Sim900A is used to communicate via text messages, voice calls, Fax and data with low power consumption.

General specification:

Low power: 1.5mA

Dual quad-band 900/1900MHz

Constant voltage: 3.4V-4.5V

Operating temperature -40C to +85C

3.5.4 Xbee pair RFX240

The module is optimized to give all functionality of transmitting power amplification for 802.11b/g/n used in the 2.4GHz recurrence go. Xbee RFX240 is implemented in CMOS process

with huge linearity and high power amplifier. The module has CMOS logic system-on-chip input impedance coordinating, and in addition, incorporated RF decoupling for the power supply. It requires insignificant outside parts to incredibly streamline RF front-end usage.

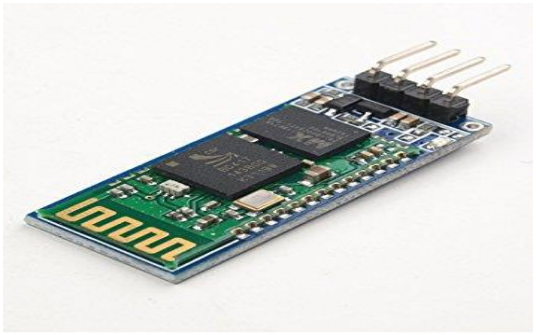


Figure 3.13: Xbee module
Source: (adafruite.com)

Features
Xbee is a simple and easy to use the module.
External Antenna is not necessary. It can be directly Plugged and played gadget.
Works at 5 DC power supply,
Range 100 Meters
TTL Outputs
Frequency 2.4GHz

3.5.5 Relay board

In my project, I used a 4 channel relay board. This relay is used to control the lights, electrical appliances, and equipment. The module is designed in such a way to allow its communication with any microcontroller boards. Relay interface board that allows you to control various appliances, and other equipment's with high voltages.

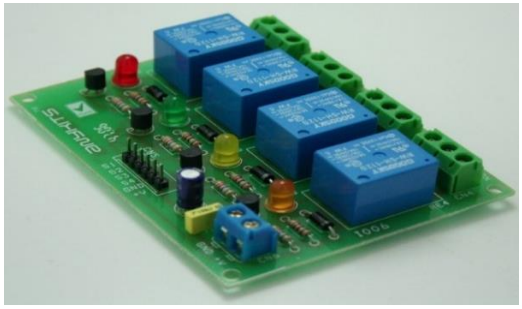


Figure 3.14: Channel 12V relay module
Source: (Electronics-lab.com)

Specifications:
Working voltage: 5v or 12v
No of Channels: 4
Module: 4-channel interfaced board and each channel uses 15-20mA driver current.
LEDs are used to indicate the relay output status.

3.6 System hardware working

3.6.1 Setting up the hardware 9DOF Razor IMU

I have used some soldering equipment and some pin headers which was there in the box along with the sensor. Starting with the software: Firstly the latest Razor AHRS Firmware must be downloaded from GitHub website.

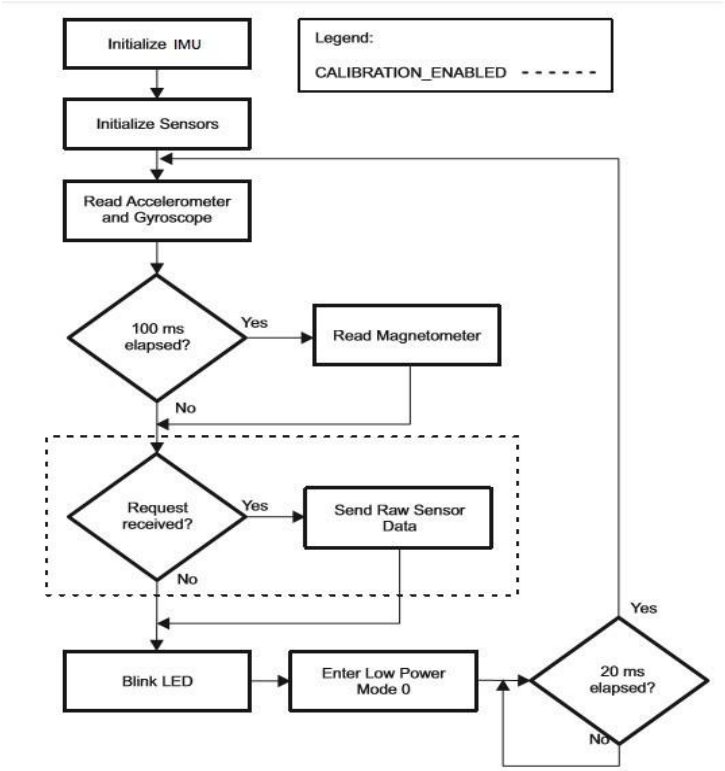


Figure 3.15: Calibration firmware overview

Firstly firmware must be downloaded and must be uploaded on to the IMU. After downloading the Firmware open downloaded RazorAHRS. Firmware package ie. Razor_AHRS/Razor_AHRS now in the Razor_AHRS.ino file, view the suitable data related to the firmware. Also, there is a unit labelled “USER SETUP AREA” which we can set few firmware by Default and by selecting the hardware under “HARDWAREOPTIONS”. In “Tools” select “Board” and select the board we remain using and select Pro Mini (3.3v, 8mhz) ATmega328”.Then in “Tools” and “Serial Port” and choose the port for the Razor. Now in “File” and hit “Upload” I/OBoard”. Finally, at the end of the window, there must say "Done uploading”.

In the serial monitor:

See the serial monitor of Arduino under “Tools” → “Serial Monitor”. Set the baud rate to 57600 then it gets roughly output values in this manner:

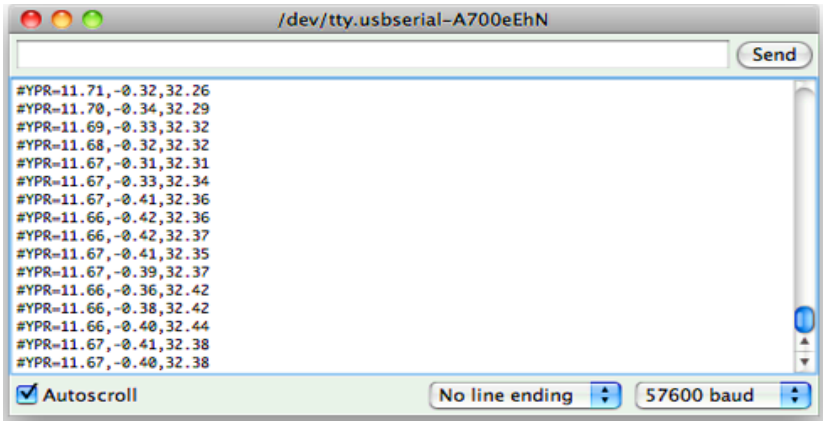


Figure 3.16: Serial monitor

The Processing test sketch to test the tracker:
Processing software must be downloaded and installed it is used compile the code and run code. Open file processing Razor_AHRS_test.pde using the processing.
Click on "Sketch" and then on "Run". The movements of the sensors are shown clearly in the screen.

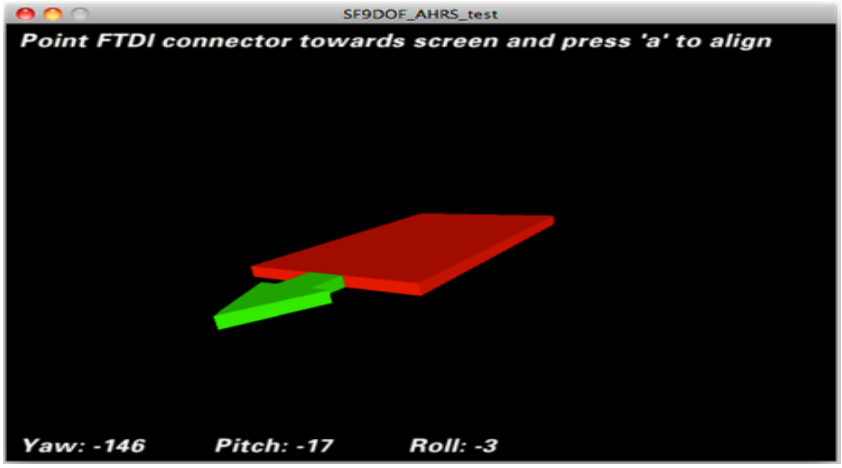


Figure 3.17: In Processing

3.6.2 Sensor calibration

Dependent how good or bad the sensors are, the accuracy and sensitivity of Razor AHRS be able to be upgraded by calibrating the sensors. Uncertainty if the sensor calibration is bad then there is a chance of getting drifts in yaw when the sensor is rolled on. Directing the sensor up does not actually give change in the attitude. One should know the definition of the axes differs from what is printed on the board. The firmware use.

- X axis directing forward (towards the edge of the connector holes)
- Y axis directing to the right
- Z axis directing down

Which represents the right-handed organise structure.

3.6.3 Design of DC power supply

As I bought the components and started to work with them the first thing I noticed is the power supply for the components is a need, so I decided to design the required. DC power supply, since it's impossible to work with the modules without giving power to them. Therefore to make the power supply circuit I used PCB board because I need 5v input for my sensor. Firstly I have designed the circuit in Proteus software. The 5v power supply was designed in Proteus where I used voltage regulator IC, which is usually known as 7805. Two100uF capacitors and a 1N4007 diode are also used. This voltage regulator is used to regulate or change the voltage level of supply voltage. As most of the batteries available in the market are of 9volts. So, 9V has become the normal for electrical batteries. Here voltage source is 9volts battery and the operating equipment sensor and ZigBee needs 5volts to operate. So a transitional source or such type of DC Power Supply is needed, which converts the 9V battery voltage to 5V operative voltage for sensor and ZigBee module. This problematic issue is reduced by means of 7805 IC, and therefore it is called voltage regulating IC. I finally designed a small circuit to maintain the voltage level and supply 5V required current for my sensor.

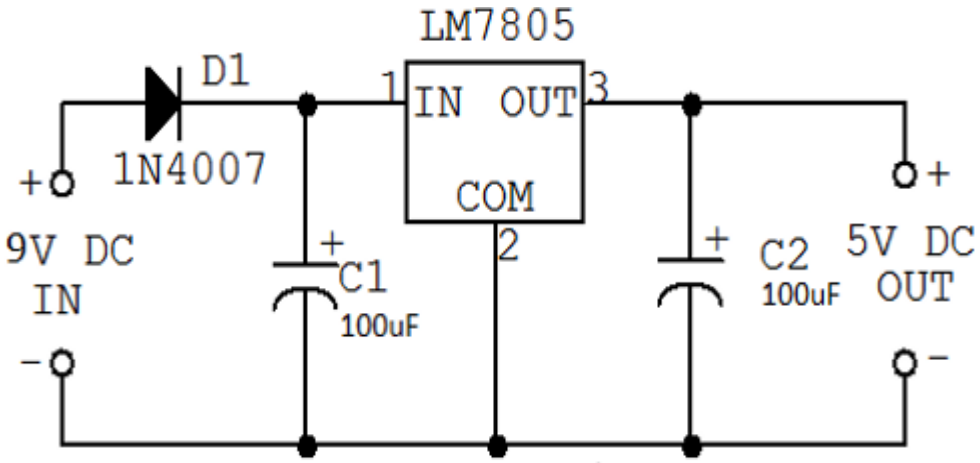


Figure 3.18: 5V power supply circuit diagram
Source: (Image developed from Proteus Software)

After designing the 5V supply then I have attached the sensor to the battery which is attached to the 5V supply pcd board. Which finally looks like this:

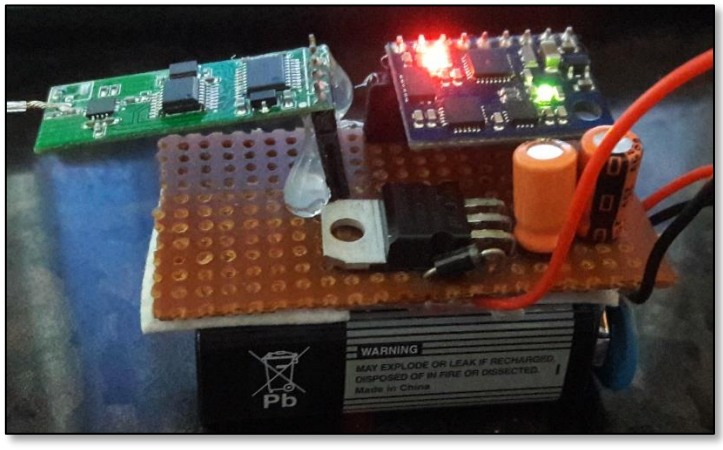


Figure 3.19: 5V power supply to IMU

I have started working with the individual components.
Each component interfacing with the raspberry pi. Firstly I worked with the basic components like led and then LCD.

3.7 Working on each component interfacing with raspberry pi

3.7.1 Here is the LCD interface with the raspberry pi. As appeared in the circuit chart underneath, I have Interfaced Raspberry Pi with LCD show by associating 10 GPIO pins of PI to the 16*2 LCD's Control and Data Transfer Pins. I have utilized GPIO Pin 21, 20, 16, 12, 25, 24, 23, and 18 as a BYTE and made "PORT" ability to send data to LCD. GPIO 21 is LSB (a least significant bit) and GPIO18 is MSB (a most significant bit). 16x2 LCD Module has 16 pins, which can be separated into five classifications, Power Pins, differentiate stick, Control Pins, Data pins and Backlight pins.

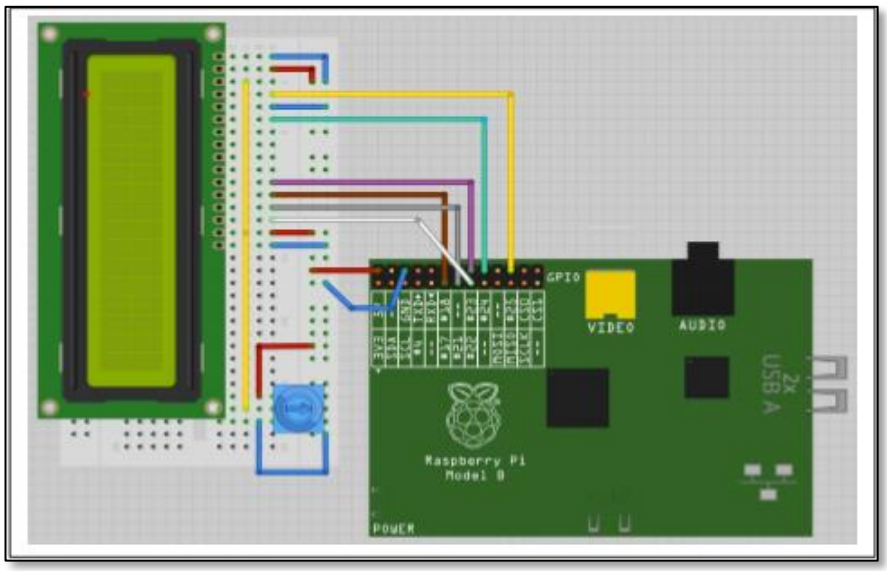


Figure 3.20: Image developed using Fritzing

As said before that we will send the characters in a steady progression. The characters which are sent are in the form of ASCII codes (American standard Code for Information Interchange). When everything is associated according to the circuit graph, we can turn ON the PI to compose the program in python. I have taken around few summons which will use in a python program. We will import GPIO document from the library, beneath capacity empowers us to program GPIO pins of PI. Subsequent to composing the program and executing it, the Raspberry Pi sends characters to LCD one by one and the LCD shows the characters on the screen.

```
import RPi.GPIO as GPIO
import time

# Define GPIO to LCD mapping
LCD_RS = 33
LCD_E  = 35
LCD_D4 = 37
LCD_D5 = 40
LCD_D6 = 38
LCD_D7 = 36

# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants
E_PULSE = 0.001
E_DELAY = 0.001

def lcd_init():
  # Initialise display
  GPIO.setwarnings(False)
  GPIO.setmode(GPIO.BOARD)        # Use BCM GPIO numbers
  GPIO.setup(LCD_E,GPIO.OUT)  # E
  GPIO.setup(LCD_RS,GPIO.OUT) # RS
  GPIO.setup(LCD_D4,GPIO.OUT) # DB4
  GPIO.setup(LCD_D5,GPIO.OUT) # DB5
  GPIO.setup(LCD_D6,GPIO.OUT) # DB6
  GPIO.setup(LCD_D7,GPIO.OUT) # DB7

  lcd_byte(0x02,LCD_CMD) # 110011 Initialise
```

Figure 3.21: Code for LCD GPIO pin connection

3.7.2 Interfacing GSM module with raspberry pi

Next the GSM module interfacing with raspberry pi. When the patient operates the sensor with his head movement for each movement each operation is done and at the same time, the same respective operation is sent as the text message to the patient care taker. Here I give the details for how to interface a GSM TTL UART Modem-SIM900A with Raspberry Pi2 and to send and receive a message through it.

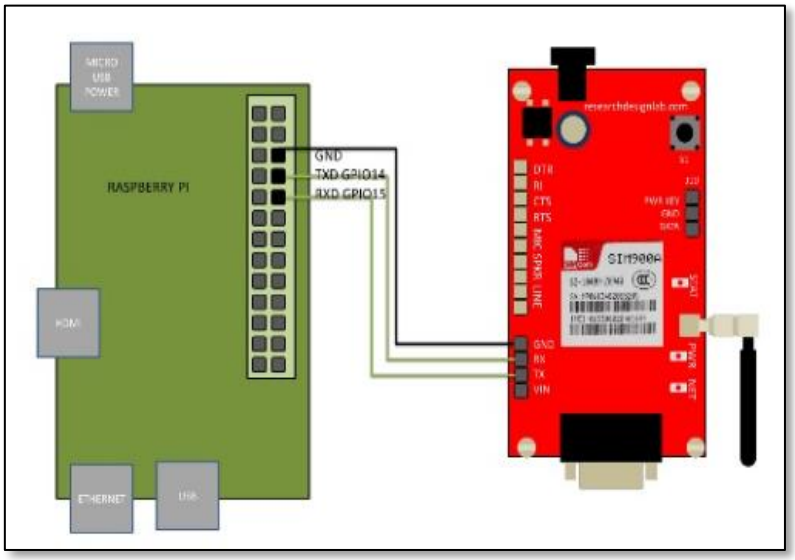


Figure 3.22: Image developed using Digi-Key electronics

Insert a sim card in to the GSM modem and make the appropriate connections which are according to the figure shown above. A suitable factor that should keep in mind is that the sim used in the module should have enough balance to send the sms and it should be kept in a place having the suitable range for the proper network. The transmission and reception pins of GSM and raspberry pi should be given in the reverse order and the ground pins must be shorted. Give the power supply to the GSM module and should wait for few seconds for sim initialization.

```
import time
import serial
gsmser = serial.Serial('/dev/ttyAMA0',9600)

def Gsminit():
    gsmser.write('AT\r\n')
    print'AT'
    time.sleep(1)
    gsmser.write('AT+CMGF=1\r\n')
    print'AT+CMGF=1'
    time.sleep(1)
    gsmser.write('AT+CNMI=1,2,0,0\r\n')
    print'AT+CNMI=1,2,0,0'
    time.sleep(1)
    print'Initilized'
    return

def Sendmsg(num,msg):
    gsmser.write('AT+CMGS=\"' + num + '\"\r\n')
    print num
    time.sleep(2)
    gsmser.write(msg)
    print msg
    gsmser.write(serial.to_bytes([0x1a]))
    return
```

Figure 3.23: Code to send SMS via GSM modem

Now send messages is done through this modem with the help of a python script given above. In the code, import the proper libraries and enable the serial communication. GSM modem, control is done with the AT commands, where suitable commands can be transmitted to the modem for each purpose and the modem will get the respond to these commands by transmitting appropriate messages that should receive and display.

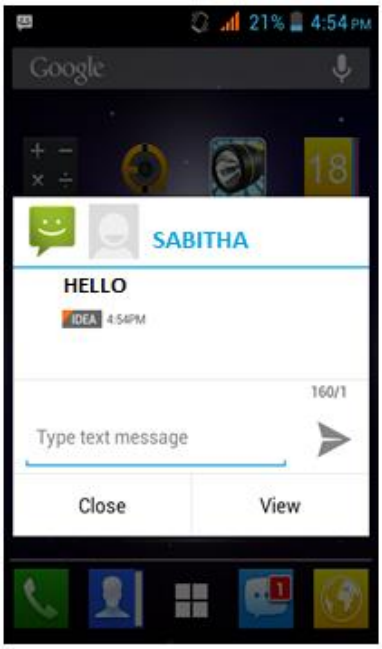


Figure 3.24: Screenshot of text message from GSM module

3.7.3 Buzzer interface with raspberry pi

Buzzer or the Beepers are the audio signalling device which may be mechanical, electromechanical or piezoelectric Piezo-electric buzzer should be connected to GPIO pins. Where we can connect the buzzer pins straight to the Raspberry Pi using female-to-female headers. Raspberry Pi is connected to the voltage supply to make it high (taking the value 1) or to make it to low (taking the value 0) by joining the ground and to give an output to it, can sense the pin as an input. These buzzers use very little current. The program works by simply toggling the GPIO pin 18 on and off with a short delay in between.

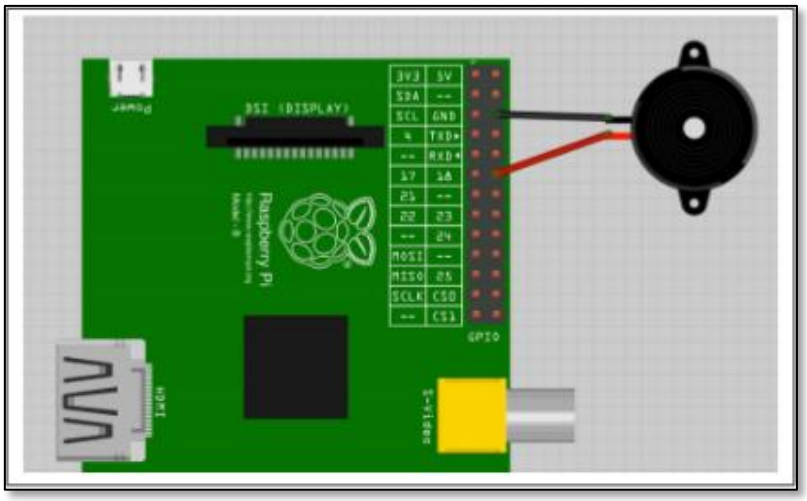


Figure 3.25: Image developed using Fritzing

Coding in Raspberry Pi

Step 1: On the home screen of the raspberry pi. Open LXTERMINAL which is available on the left corner of the screen.
Step 2: New file should be created with the sudo nano filename.py command.
Step 3: Click ENTER key takes the new window and can type the code.
Step 4: After the code is typed press the CTRLX to save the code. And it will prompt that save mode at the bottom of the window. Press Y and hit enter key.
Step 5: By using sudo python filename.py command runs the code. Hit enter key. On hitting ENTER key, the program starts to run.

3.7.4 Relay interfacing with raspberry pi

A relay basically acts like a switch whose contacts are closed when an electromagnet pulls them together. As the electromagnet and switch are not connected electrically, this protects the circuit driving the relay coil from any of the high voltages on the switch side. I used relay here to operate my electrical appliances. It acts as the switch to ON and OFF the devices for the respective head movement positions.

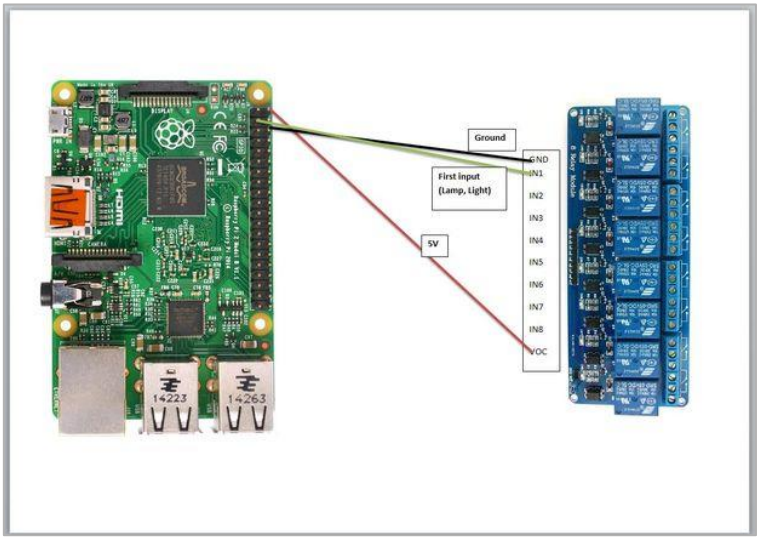


Figure 3.26: Relay with pi connections
Source: (instructables.com)

3.7.5 Xbee pair RFX240 interface with raspberry pi

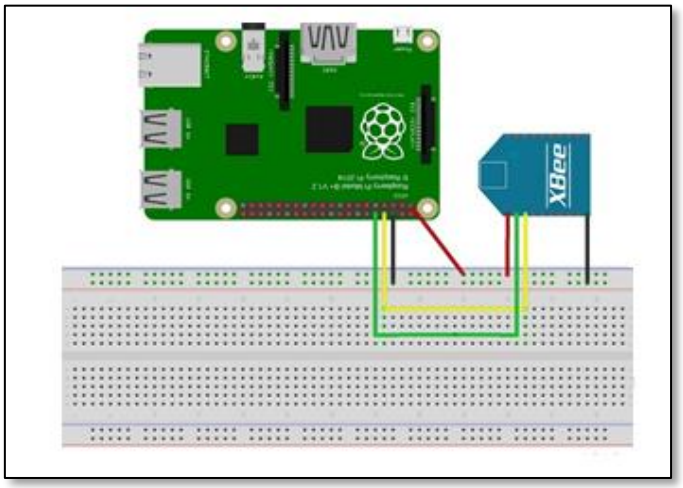


Figure 3.27: Image developed using Fritzing

XBee is a popular ZigBee item. For the experiment purpose, I bought two xBee Pro. I started by connecting the xBee to my PC and xBee Router to my raspberry pi and connected the xBee directly by utilizing the female to female wires, which is shown in the above image developed using fritzing. Interface raspberry pi vin 3.3v to xBee 3.3v stick, and ground to xBee GND. Rx pin to xbee TX pin (data out) and connect TX to xBee RX(data in) now few information from Router xbee to Coordinator will be sent. Presently the information from Router xBee to my raspberrypi to the organizer associated with my Computer is sent by interfacing a wire from xbee DOUT to pi's RXD as it will just utilize it to get the information and another xbee DIN is given to pi TX as per this raspberry pi GPIO pinout. Raspberry pi is ready to get its primary bytes by means of the Xbee now.

3.7.6 The Pin mapping for the complete system

Table 3.2: Pin mapping of the system from Raspberry Pi to LCD

Raspberry Pi 2B	LCD
+5V	+5V
GND	Contrast(3)
GPIO7	RS(4)
GND	RW(5)
GPIO8	E(6)
GPIO25	Data 4(11)
GPIO24	Data 5(12)
GPIO23	Data 6(13)
GPIO18	Data 7(14)
P1-06	GND

Table 3.3: Pin mapping of the system from Raspberry Pi to Xbee RFX240 module

Raspberry Pi 2B	Xbee pair RFX240
GPIO14(8)	TX
Pi2(3.3V)	VDD
Pi6(GND)	GND

Table 3.4: Pin mapping of the system from Raspberry Pi to Buzzer

Raspberry Pi 2B	Buzzer
Pi11 (GPIO14)	Active1(+ve)
Pi9 (GND)	GND 0

Table 3.5: Pin mapping of the system from Raspberry Pi to GSM module

Raspberry Pi 2B	GMS module
TX	RX
RX	TX
Pi(02)	+5v
GND	GND

Table 3.6: Pin mapping of the system from Raspberry Pi to Relay

Raspberry Pi 2B	Relay
Pin7	IN1
Pin5	IN2
Pin3	IN3
Pin11	IN4
Pin13	IN5
Pin15	IN6
Pin19	IN7
Pin21	IN8
5V	VCC
GND	GND

Table 3.7 Pin mapping of the system from IMU module to Xbee RFX240

IMU	Xbee RFX240	Battery(9V)
Vin	VDD	Positive Terminal(+ve)
GND	GND	Negative Terminal(-ve)
TX	RX	-
RX	TX	-

3.8 Schematic

The schematic design explains the wiring between the hardware involved in the proposed system. The pin mapping is as explained in the above section 3.6.6.

Figure 3.32 shows the wiring of the head band section. The IMU module RX is connected to TX of the Xbee module and TX of IMU is connected to the RX of the Xbee module and VDD, GND of both the IMU and Xbee module is connected to the 5V i.e. VDD to positive wire and GND to the negative wire of the PBC board which was connected to the 9V battery.

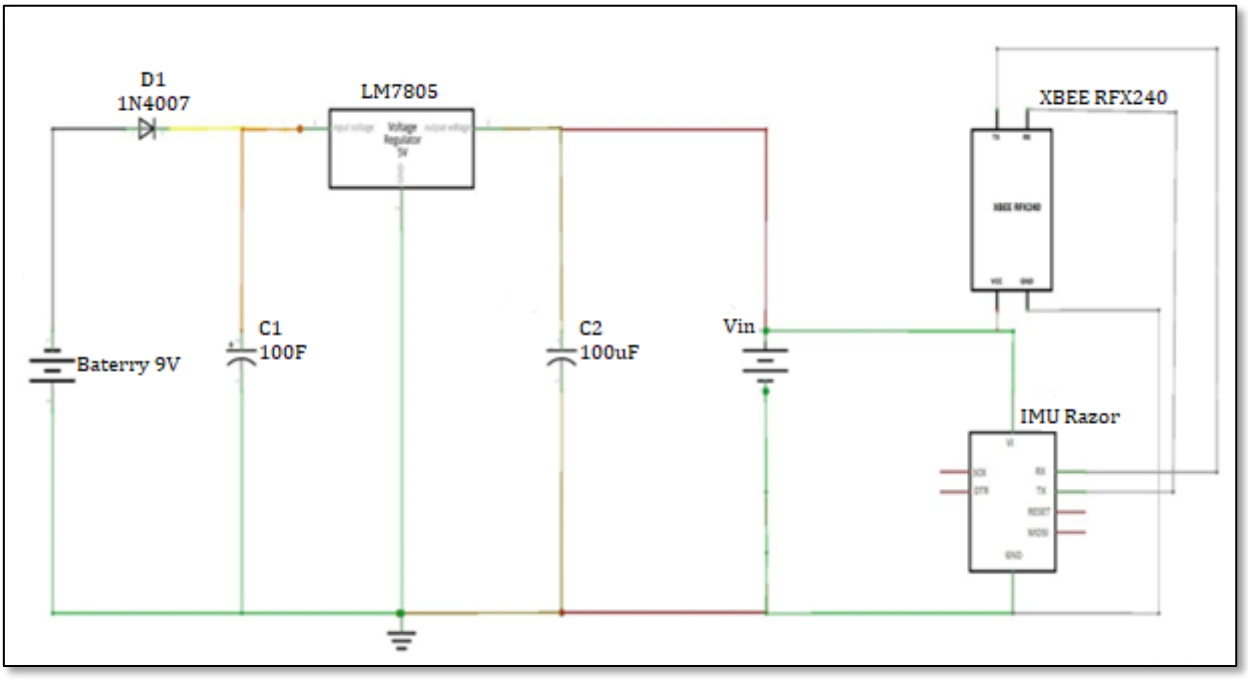


Figure 3.28: Head band unit Schematic

Figure 3.33 shows the wiring between the raspberry pi and the other modules.

Raspberry pi wiring with the LCD display is connected to these pins GND, +5V, GND, GPIO7, GND, GPIO8, GPIO25, GPIO24, GPIO23, GPIO18, P1-06 are wired to +5V, Contrast(3), RS(4), RW(5), E(6)Data 4(11), Data 5(12), Data 6(13), Data 7(14), GND pins to the LCD respectively.

Raspberry pi wiring with the GSM module TX of pi to RX pin of GSM, RX of pi wired with TX of GSM, and Pi(02) pin connected to +5v of the input of GSM and raspberry pi GND connected to GSM GND.

Raspberry pi wiring with the buzzer here Pi11 (GPIO14) pin of raspberry is connected to Active1 (+ve) pin of the buzzer and Pi9 (GND) is wired to GND 0 of the buzzer.

Raspberry pi wiring with the RFX240 the raspberry pi GPIO14 (8) is connected to the TX of the RFX240 and Pi2 pin (3.3V) is connected VDD of RFX240 and raspberry pi pin Pi6(GND) is given to the GND of RFX240.

Raspberry pi wiring pins Pin7, Pin5, Pin3, Pin11, Pin13, Pin15, Pin19, Pin21, 5V, GND are connected to the relay board input pins IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8, VCC, GND respectively.

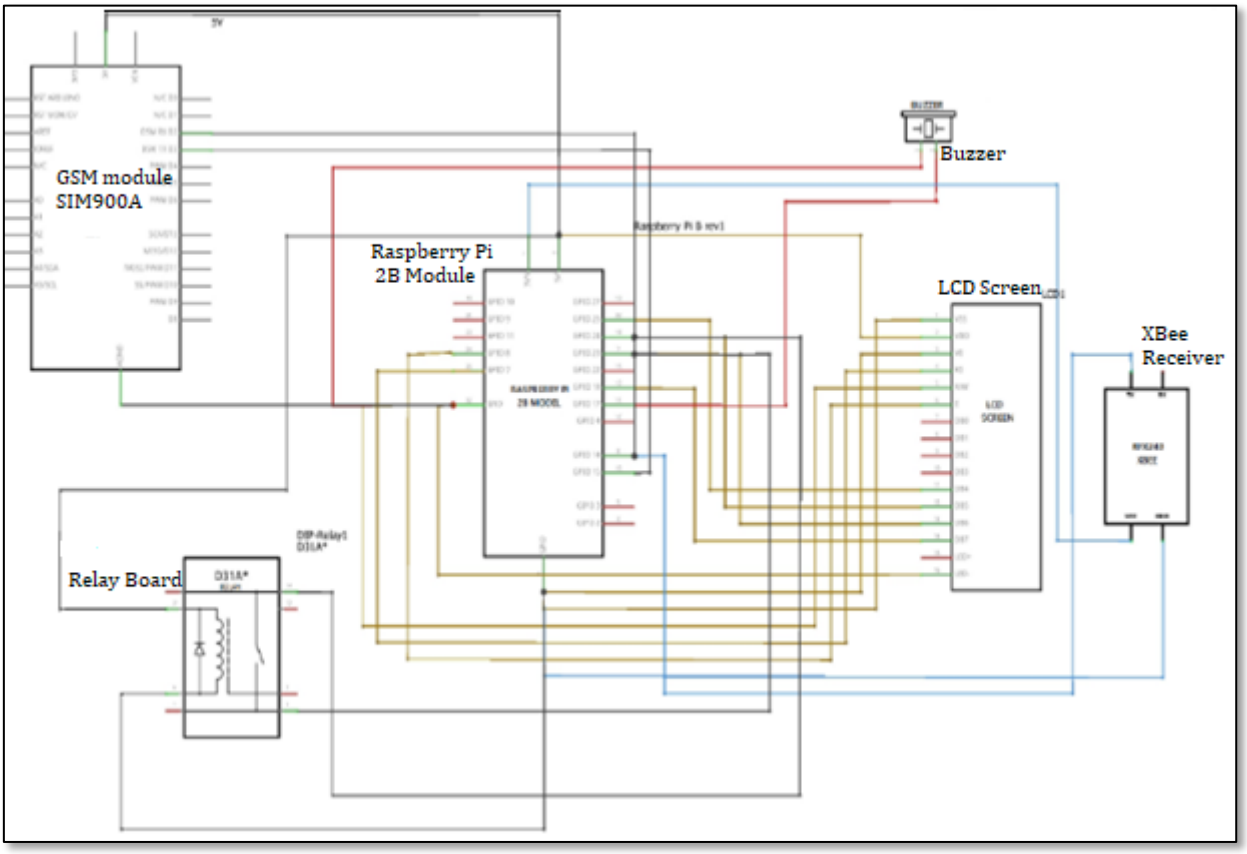


Figure 3.29: Receiver unit Schematic

CHAPTER 4
RESULT AND ANALYSIS

4.1 Test and result

The experiment end result and the final module are discussed in this result and analysis section. The experiment is of prototype consists of two different parts or sections. In the first portion ie. transmitter section which consists of the headband. This headband includes the IMU, Xbee which are powered up using a battery. The headband is a wearable belt and patient can wear the belt on the forehead. The second section is the receiver section which contains the raspberry pi, GSM module, LCD, relay, buzzer, and voltage regulator. The receiver part i.e Figure 4.1 is can be placed under the patient's bed as the headband it is a wireless wearable device. All the module interfacing was discussed in the methodology accordingly the overall sections i.e. transmitter section and the receiver section was obtained.

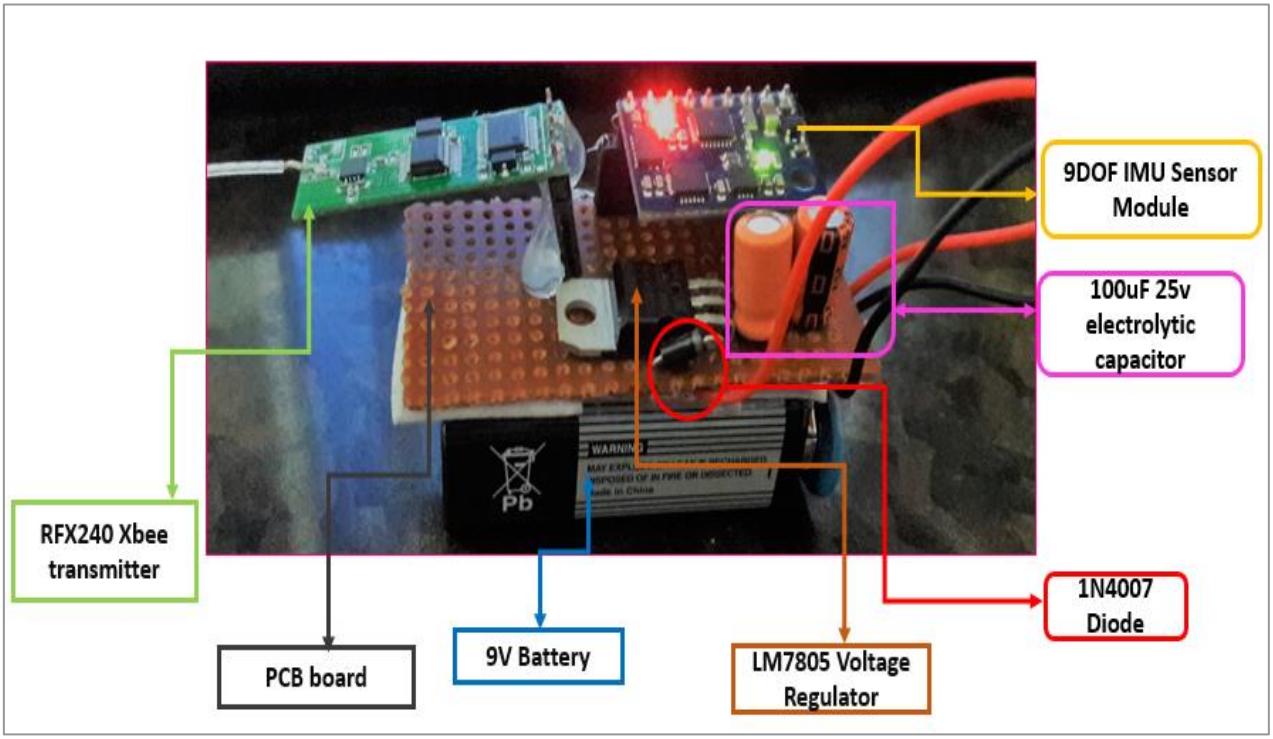


Figure 4.1: IMU and Xbee module powered up

Figure 4.1 is the IMU and ZigBee module after the 5V supply is given from the 9V battery using the voltage regulator.This voltage regulator is used to regulate or change the voltage level of supply voltage. As most of the batteries available in the market are of 9volts. So, 9V has become the normal for electrical batteries. Here voltage source is 9volts battery and the operating equipment sensor and ZigBee needs 5volts to operate. So a transitional source or such type of DC Power Supply is needed, which converts the 9V battery voltage to 5V operative voltage for sensor and ZigBee module. This problematic issue is reduced by means of 7805 IC, and therefore it is called voltage regulating IC. I finally designed a small circuit to maintain the voltage level and supply 5v required current for my sensor.

The Figure 4.2 receiver section includes the Xbee receiver which receives the sensor data from the Xbee transmitter and gives it to the raspberry pi this pi will run the interfaced modules according

to the commands. Each module has its specific task. The GSM module will send each output as the text message to the mobile whose number is mentioned in the code. In this way, one can have the clear update of the patient's activity and any immediate help can be given if there is any emergency for the patient. LCD will be displaying every action that is performed by the head band. The buzzer will give a beep sound for each action this is for the conformation message to the patient that the selected action is been implemented. The relay is placed to operate the electronic appliances such as fan and light. When the fan or light command is given then automatically the respective operation is done. So the paralysis patients suffering from quadriplegia will be able to communicate with the other people for their basic needs in this manner.

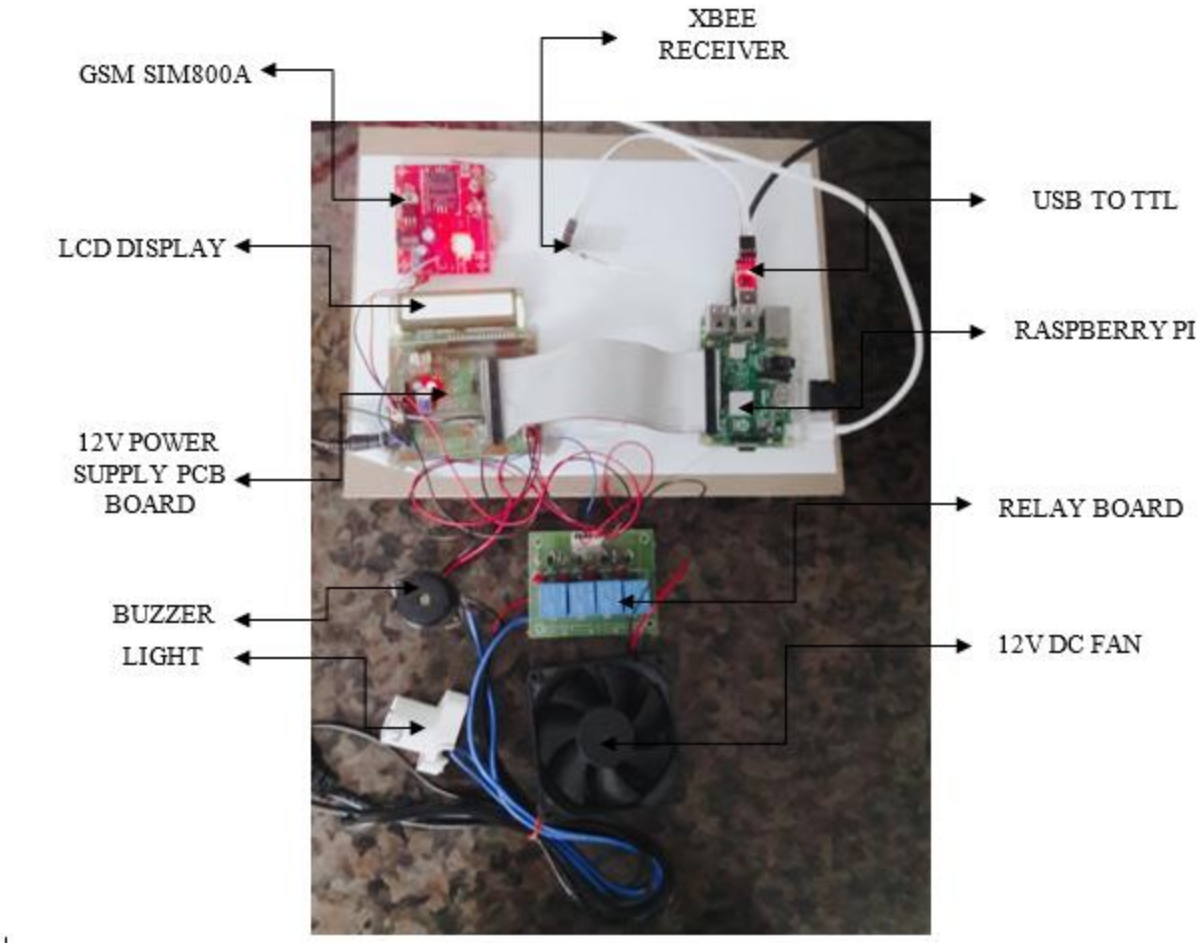


Figure 4.2: Receiver section

Figure 4.3 is the flow chart which describes the working flow of the headband. The start is to power on the device and posts it in the stable position and the head movement to start the loop is the command to start the device and it must be implemented with the up direction of the head. Once the start loop is activated than other head movements can be given to operate other devices. For example, if the right movement is given to on the fan this would first check the condition whether the device is ON or OFF and then operates accordingly and finally, after every operation is done then it will return to the start or stop the loop.

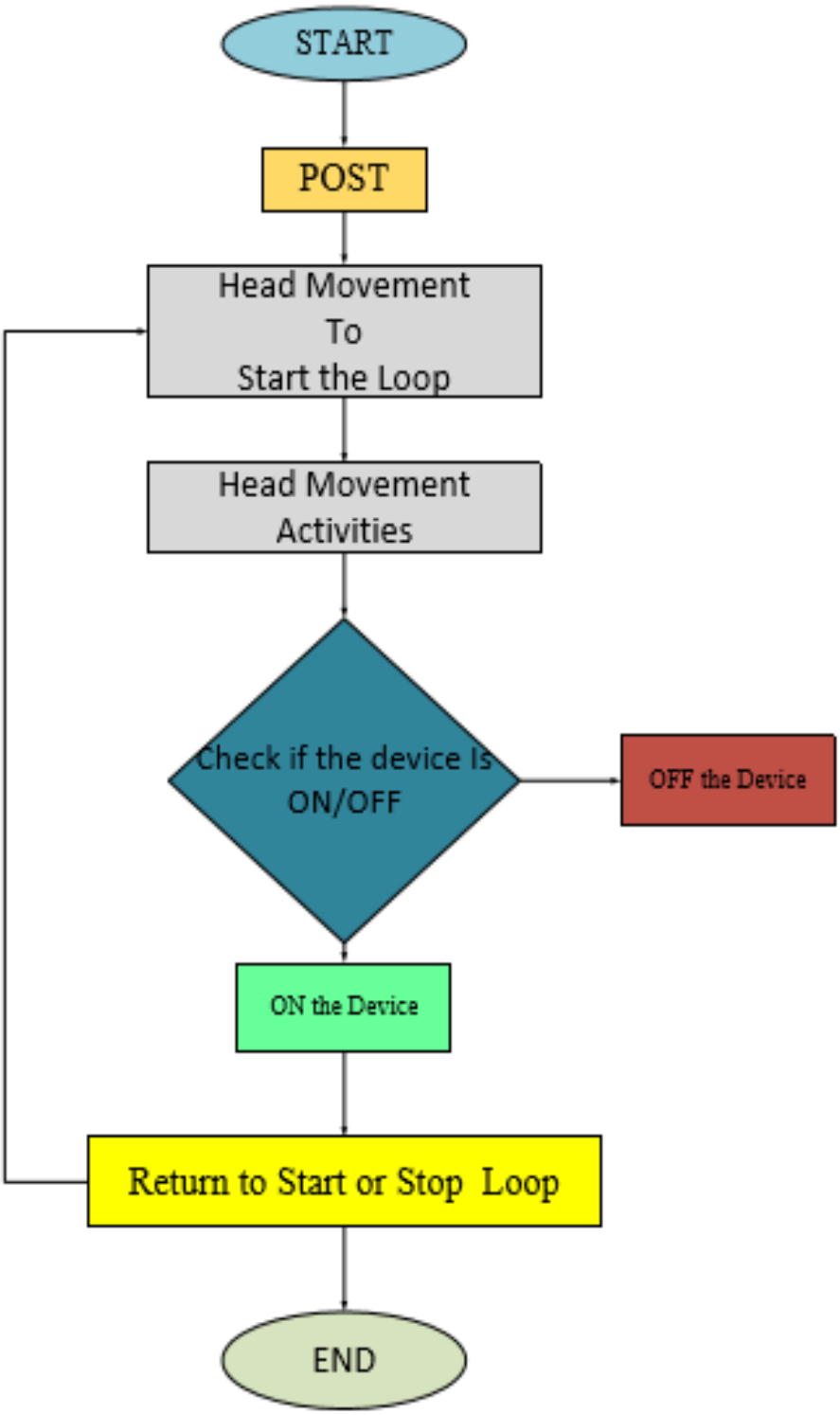


Figure 4.3: Flow chart

Figure 4.4 shows head movements. Here the 0 position where the sensor is placed exactly on the head in this position no action is done therefore the LCD display shows that the sensor is stable. The up movement i.e. 4, is the most important movement as it is used to start the device which

means whenever the patient wants to use the band for any requirement then the head must be tilted in the up direction in order to make the device active for use. Now after the device is ready to use the other movements comes into play. The above figure shows the eight different head movements. Here the zero is the stable position where the head band with the sensor is placed exactly on the forehead and the head should be facing the ceiling in this position no action is done therefore the LCD display shows that the sensor is stable. The first movement is the right side here the output is patient requests for food for this action the LCD shows display as food and the text message request for food is sent to the mobile. The second direction is the left side which gives the output as a request for water. The third movement is downwards this gesture is used for water. The fifth movement is the down left which is used to call the doctor and the sixth gesture upright used to operate the fan, the seventh direction is towards up left which is used to operate the light. The last gesture is the towards downright which is used to ask for medicine. These are the all eight different gesture movements with their respective outputs.

Figure 4.4: Head gesture recognition

4.2 Systems algorithm

In the present work, the Euler angles were used to detect the orientation. The space of orientations can be parameterized by Euler angles. When Euler angles are used, a general orientation is written as a series of rotations about three mutually orthogonal axes in space. Usually, the x, y, and z-axes in a Cartesian coordinate system are used. The rotations are often called x-roll, y-roll, and Z-roll. For each type of roll, there is a corresponding rotation matrix, i. e. an x rotation matrix, a y rotation matrix, and a z rotation matrix. The matrices rotate by multiplying them to the position vector for a point in space, and the result is the position vector for the rotated point. A general rotation is obtained by multiplying the three matrices corresponding to the three Euler angles. The resulting matrix embodies the general rotation and can be applied to the points that are to be rotated. Euler angles represent the orientation of a body rotation around a principal axis. Figure.4.3 shows the Euler Angles axes.

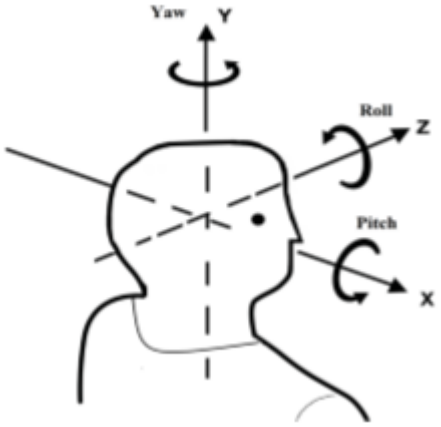


Figure 4.5 Euler angle axes

The rotation of Yaw angle around the z-axis is defined as:

$$R_{z\,(\alpha)} = \begin{vmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (4)$$

The rotation of Pitch angle around the x-axis is defined as

$$R_{Y\,(\beta)} = \begin{vmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{vmatrix} \quad (5)$$

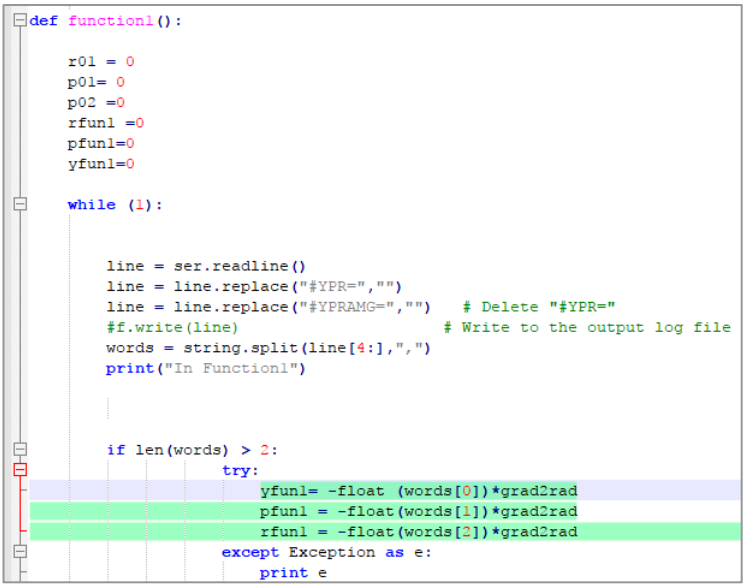
The rotation of Roll angle around the y- axis is defined as

$$R_{x(\gamma)} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{vmatrix} \quad (6)$$

A single rotation vector can be formed by multiplying the yaw, pitch, and roll rotation axes to obtain by the angles $(\alpha\beta\gamma)$ represent pitch, roll and yaw which are Euler angles

Rotation vector $(\mathrm{R}(\alpha\beta\gamma)) = R_{X\,(\alpha)}\ R_{Y\,(\beta)}\ R_{Z\,(\gamma)}$

When the sensor is connected to the serial port we get the yaw pitch and roll values according to changes made in the IMU Razor code but as these angles are in the wide range we convert them to radians in order to minimum the range for the obtained angles and after converting them to radians we get the outputs in yaw pitch roll in radians which is shown in Figure 4.6.



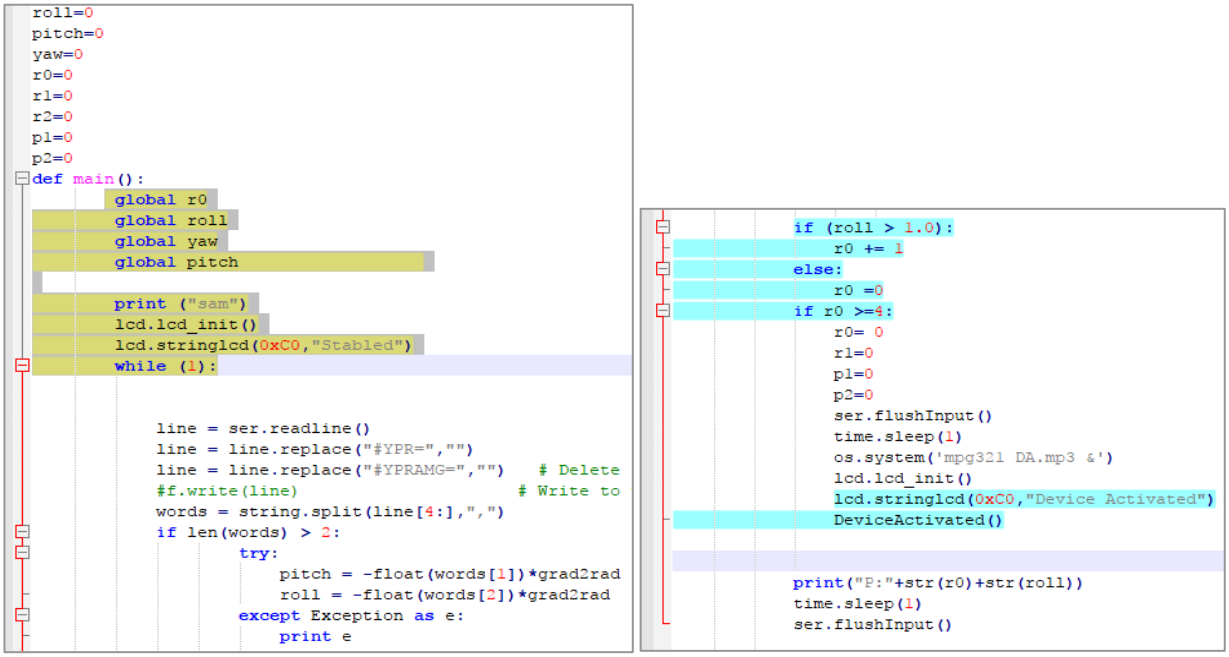
```
def function1():

    r01 = 0
    p01= 0
    p02 =0
    rfun1 =0
    pfun1=0
    yfun1=0

    while (1):


        line = ser.readline()
        line = line.replace("#YPR=","")
        line = line.replace("#YPRAMG=","")   # Delete "#YPR="
        #f.write(line)                       # Write to the output log file
        words = string.split(line[4:],",")
        print("In Function1")


        if len(words) > 2:
                    try:
                        yfun1= -float (words[0])*grad2rad
                        pfun1 = -float(words[1])*grad2rad
                        rfun1 = -float(words[2])*grad2rad
                    except Exception as e:
                        print e
```

Figure 4.6: Code to calculate YPR angles into radians

4.3 Logic explanation

The code written is the python code in the raspberry pi.

Figure 4.7 shows the main part of the program where the sensor is in a stable position and the roll, pitch, yaw values are in the initial stage. This main function is crossed only in one condition which is when the roll value is >1.0 in this condition the device will be in the device activation stage now the device is ready to be used to select the functions. There are eight activities written for every eight different directions. Initially, all the yaw, pitch, roll values are zero and once the sensor is stabled and when the roll value is >1.0 then the device will cross the main function and is ready to operate the other devices with the other seven head movements.



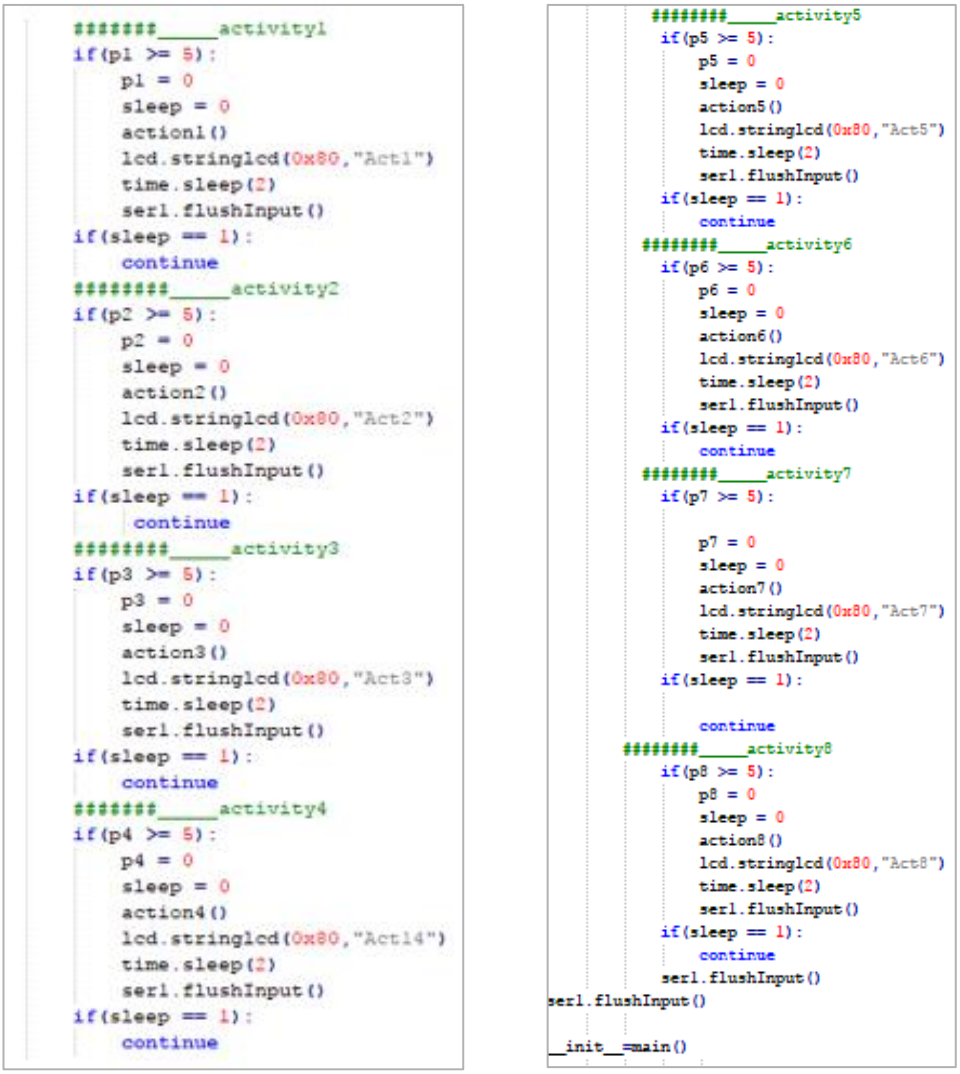
```
roll=0
pitch=0
yaw=0
r0=0
r1=0
r2=0
p1=0
p2=0
def main():
        global r0
        global roll
        global yaw
        global pitch

        print ("sam")
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"Stabled")
        while (1):


            line = ser.readline()
            line = line.replace("#YPR=","")
            line = line.replace("#YPRAMG=","")    # Delete
            #f.write(line)                        # Write to
            words = string.split(line[4:],",")
            if len(words) > 2:
                    try:
                        pitch = -float(words[1])*grad2rad
                        roll = -float(words[2])*grad2rad
                    except Exception as e:
                        print e
```

```
            if (roll > 1.0):
                r0 += 1
            else:
                r0 =0
            if r0 >=4:
                r0= 0
                r1=0
                p1=0
                p2=0
                ser.flushInput()
                time.sleep(1)
                os.system('mpg321 DA.mp3 &')
                lcd.lcd_init()
                lcd.stringlcd(0xC0,"Device Activated")
                DeviceActivated()


            print("P:"+str(r0)+str(roll))
            time.sleep(1)
            ser.flushInput()
```

Figure 4.7 Main program code

Figure 4.7 shows the code for device activation. When the head is tilted up direction and the roll value must reach the >1.0 at this stage the device will be activated for use. This part is implemented as it acts as the switch where we can ON the device with the up direction and after activation other seven movements can be given to operate each function and after activation of each activity automatically the device will reach back to the device activation stage in order to over come confusion the continuous operations at a time.

Figure 4.8 is the code for the all eight activities. There are eight positions for 8 different directions p1,p2,p3,p4,p5,p6,p7,p8 are the up,down, left, right, upright, upleft, downright, downleft. Here each direction is used to operate respective devices.



```
#######_____activity1
if(p1 >= 5):
    p1 = 0
    sleep = 0
    action1()
    lcd.stringlcd(0x80,"Act1")
    time.sleep(2)
    ser1.flushInput()
if(sleep == 1):
    continue
########_____activity2
if(p2 >= 5):
    p2 = 0
    sleep = 0
    action2()
    lcd.stringlcd(0x80,"Act2")
    time.sleep(2)
    ser1.flushInput()
if(sleep == 1):
    continue
########_____activity3
if(p3 >= 5):
    p3 = 0
    sleep = 0
    action3()
    lcd.stringlcd(0x80,"Act3")
    time.sleep(2)
    ser1.flushInput()
if(sleep == 1):
    continue
#######_____activity4
if(p4 >= 5):
    p4 = 0
    sleep = 0
    action4()
    lcd.stringlcd(0x80,"Act14")
    time.sleep(2)
    ser1.flushInput()
if(sleep == 1):
    continue
```

```
        ########_____activity5
        if(p5 >= 5):
            p5 = 0
            sleep = 0
            action5()
            lcd.stringlcd(0x80,"Act5")
            time.sleep(2)
            ser1.flushInput()
        if(sleep == 1):
            continue
    ########_____activity6
        if(p6 >= 5):
            p6 = 0
            sleep = 0
            action6()
            lcd.stringlcd(0x80,"Act6")
            time.sleep(2)
            ser1.flushInput()
        if(sleep == 1):
            continue
    ########_____activity7
        if(p7 >= 5):

            p7 = 0
            sleep = 0
            action7()
            lcd.stringlcd(0x80,"Act7")
            time.sleep(2)
            ser1.flushInput()
        if(sleep == 1):

            continue
    ########_____activity8
        if(p8 >= 5):
            p8 = 0
            sleep = 0
            action8()
            lcd.stringlcd(0x80,"Act8")
            time.sleep(2)
            ser1.flushInput()
        if(sleep == 1):
            continue
        ser1.flushInput()
ser1.flushInput()

__init__=main()
```

Figure 4.8 Code for eight activities

4.4 Classification using Multilayer Perceptron neural networks

4.4.1 Layer Architecture in MLP

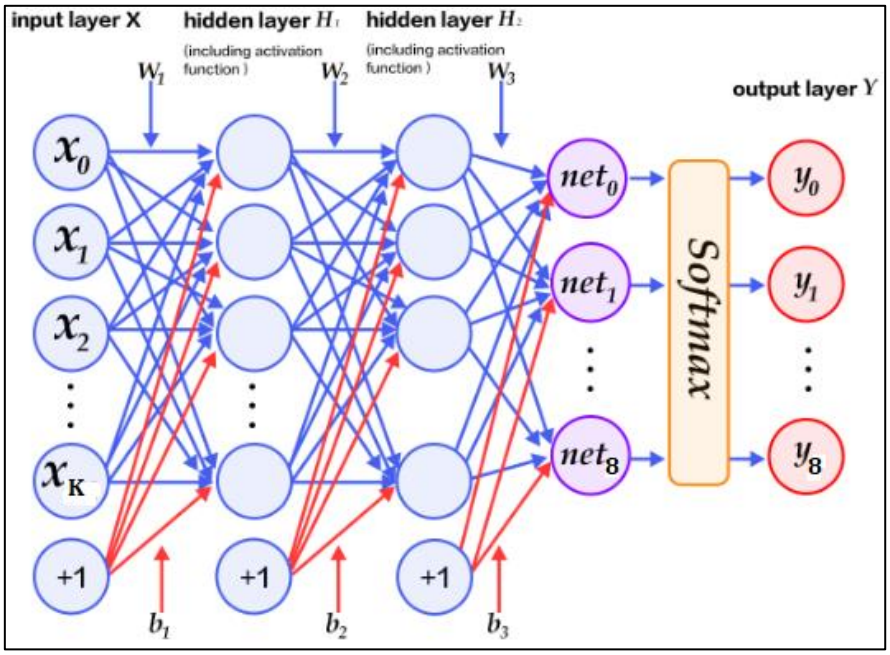


Figure 4.9: Multilayer Perceptron network architecture

As outlined in Figure 4.9, shows Multilayer Perceptron network, with the weights, and the bias, +1 indicates that the bias is 1. MLPC consists of multiple layers of nodes including the input layer, hidden layers (also called intermediate layers), and output layers. Each layer is fully connected to the next layer in the network. Where the input layer, intermediate layers, and output layer can be defined as follows:

- The **input layer** consists of neurons that accept the input values. The output from these neurons is same as the input predictors. Nodes in the input layer represent the input data. All other nodes map inputs to outputs by a linear combination of the inputs with the node's weights **w** and bias **b** and applying an activation function. This can be written in z form for MLPC with *K*+1 layers as follows:

$$y(x) = f_k(\dots f_2(w_2^T f_1(w_1^T x + b_1) + b_2) \dots + b_k) \quad (7)$$

- **Hidden layers** are in between input and output layers. Typically, the number of hidden layers range from one to many. It is the central computation layer that has the functions that map the input to the output of a node. Nodes in the i**ntermediate layers** use the hyperbolic tan function, as follows:

$$f(z_i) = \frac{e^{z_i} - e^{-z_i}}{e^{z_i} + e^{-z_i}} \quad (8)$$

- The **output layer** is the final layer of a neural network that returns the result back to the user environment. Based on the design of a neural network, it also signals the previous

layers on how they have performed in learning the information and accordingly improved their functions.

Nodes in the **output layer** use softmax function:

$$f(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}} \tag{9}$$

The number of nodes N, in the output layer, corresponds to the number of classes.

4.4.2 Mathematical formulation of Adam, Loss function, Softmax classification

MLP trains using Adam, updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation, i.e.

$$\omega \leftarrow \omega - \eta \left(\alpha \frac{\partial R(\omega)}{\partial \omega} + \frac{\partial LOSS}{\partial \omega} \right) \tag{10}$$

where η is the learning rate which controls the step-size in the parameter space search. $LOSS$ is the loss function used for the network. $\alpha > 0$ is a non-negative hyperparameter. The loss function for classification is Cross-Entropy log loss function. Given a set of training examples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ here $x_i \in \boldsymbol{R}^n$and,$y_i \in \{0,1\}$ a one hidden layer one hidden neuron MLP learns examples where function $f(x) = W_{2g}(W_1^T x + b_1) + b_2$ here $W_1 \in \boldsymbol{R}^m$ and $W_{2,} b_1, b_2 \in \boldsymbol{R}$ and are model parameters were $W_1 \in \boldsymbol{R}^m$ W_1, W_2 represents the weights of the input layer and hidden layer, respectively; b_1, b_2 represent the bias added to the hidden layer and the output layer, respectively. $g(.)R \rightarrow R$ is the activation function, set by default as the hyperbolic tan. It is given as,

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{11}$$

If there are more than two classes, $f(x)$ itself would be a vector of size (n_classes,). Instead of passing through logistic function, it passes through the softmax function, which is written as,

$$Softmax(z)_i = \frac{\exp(z_i)}{\sum_{l=1}^{k} exp(z)_i} \tag{12}$$

where z_i represents the *i* th element of the input to softmax, which corresponds to class *i*, and *k* is the number of classes. The result is a vector containing the probabilities that sample *x* belong to each class. The output is the class with the highest probability.

MLP uses different loss functions depending on the problem type. The loss function used in this study for classification is Cross-Entropy, is given as,

$$LOSS(\hat{y}, y, W) = -y ln\, \hat{y} - (1 - y) \ln(1 - \hat{y}) + \alpha ||W||_2^2 \tag{13}$$

where $\alpha ||W||_2^2$ is an L2-regularization term that penalizes complex models; and $\alpha > 0$ is a non-negative hyperparameter that controls the magnitude of the penalty.
In gradient descent, the gradient $\nabla Loss_W^i$ of the loss with respect to the weights is computed and deducted from W .More formally, this is expressed as,

$$W^{i+1} = W^i - \in \nabla Loss_W^i \tag{14}$$

where *i* is the iteration step, and ∈ is the learning rate with a value larger than 0.
The algorithm stops when it reaches a pre-set maximum number of iterations; or when the improvement in loss is below a certain, small number.

4.4.3 The networks parameter specification

i. The hidden_layer_sizes element represents the number of neurons in the hidden layer here two hidden layers are used.
ii. The weights and biases in nine different groups were initialized by random selections from zero-mean, activation function for the hidden layer is 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
iii. The solver for weight optimization. The solver 'Adam' works well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score.
iv. Learning rate schedule for weight updates, 'adaptive' keeps the learning rate constant to 'learning_rate_init' as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least to l or fail to increase validation score by at least to 1 if 'early stopping' is on, the current learning rate is divided by 5.
v. random_state: int, RandomState instance or None, optional, default None
If int, random_state is the seed used by the random number generator;

4.4.4 MLP code execution

The Figure 4.10 is the code which includes the multilayer perceptron algorithm and in this firstly the set of data from the imu in all the 9 directions direction was taken under the training file in line 236. MLP trains on two arrays: array X of size (n_samples, n_features), which holds the training samples represented as floating point feature vectors; and array y of size (n_samples,), which holds the target values (class labels) for the training samples. In the line 248 classifier selection the activation 'tanh', the hyperbolic tan function, returns f(x) = tanh(x) s used. The solver for weight optimization 'Adam' is used which works well on relatively large datasets with thousands of training samples or more, in terms of both training time and validation score. A maximum number of iterations the solver iterates until convergence or this number of iterations. For solvers ('Adam'), this determines the number of epochs (how many times each data point will be used), not the number of gradient steps. hidden_layer_sizes: there are two hidden layers in the 1st layer there are 900 neurons and in the 2nd layer 90 neurons. Learning rate schedule for weight updates, 'adaptive' keeps the learning rate constant to 'learning_rate_init' as long as training loss keeps decreasing.
Line 255 is used to save the file after training and line 261 is used to print the accuracy of the experiment.

```
'''
Training File and Features and Label selecion
'''

df = pd.read_csv('ppr_4500.txt')#Filename
df1 = df.replace(np.nan, '', regex=True)
print (df)#-------------------------------->print traning data
#define X & Y
X = np.array(df1.drop(['position'], 1))
y = np.asarray(df1['position'] )#dtype = '|S6'
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state = 5)


'''
MLP Realtime live training
'''
clf = MLPClassifier(activation='tanh',solver='adam', max_iter=400, hidden_layer_sizes=(900,90),learning_rate='adaptive', verbose=True, random_state=1)
clf.fit(X_train, y_train)
joblib.dump(clf, 'final.sav')

'''
Trained Model
'''
clf = joblib.load('final.sav')


'''
Print Accuracy
'''
accuracy = clf.score(X_test, y_test)*100
print ("Accuracy:"+str(accuracy)+"%")#--------------------->print acurracy
```

Figure 4.10: MLP algorithm

The multilayer perceptron (MLP) is an artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers and each layer is fully connected to the following one. The nodes of the layers are neurons using nonlinear activation functions, except for the nodes of the input layer. There can be one or more non-linear hidden layers between the input and the output layer. We have trained for our classifier clf. We have nine input nodes X0 to X8 called the input layer and nine output neuron. We have two hidden layers the first one with the neurons H00H00 ... H900H900 and the second hidden layer consisting of H00H90. Each neuron of the hidden layers and the output neuron possesses corresponding Bias, i.e. B00B00 is the corresponding Bias to the neuron H00H00, B01B01 is the corresponding Bias to the neuron H01H01 and so on.

Each neuron of the hidden layers receives the output from every neuron of the previous layers and transforms these values with a weighted linear summation.

$$\sum_{i=0}^{n-1} w_i x_i = W_0 X_O + W_1 X_1 + \cdots + W_{n-1} X_{n-1} \qquad (15)$$

into an output value, where n is the number of neurons of the layer and *wi* corresponds to the i^{th} component of the weight vector. The output layer receives the values from the last hidden layer. It also performs a linear summation, but a non-linear activation function

$$g(\cdot):R \rightarrow Rg(\cdot):R \rightarrow R \qquad (16)$$

like the hyperbolic tan function will be applied to the summation result. The attribute coefs_ contains a list of weight matrices for every layer. The weight matrix at index i holds the weights between the layer i and layer i + 1.

print (clf.coefs_)

The Python code above returned the following:

```
('current loss computed with the loss function: ', 0.2972775233289579)
('coefs: ', [array([[ 0.00467315,  0.03699793, -0.12668505, ...,  0.02137048,
         0.02815205, -0.00175058],
       [-0.11848539,  0.07991897, -0.08181881, ...,  0.05861824,
         0.03133305, -0.01462918]]), array([[ 0.0256824 ,  0.0478058 , -0.01064547, ..., -0.05694136,
        -0.0344993 ,  0.07296767],
       [-0.02709938,  0.03239971, -0.04138781, ...,  0.07088572,
        -0.03452317,  0.00177088],
       [ 0.04012   ,  0.02922316, -0.01231783, ...,  0.06038042,
         0.04122688,  0.04165533],
       ...,
```

Figure 4.11: Weights representation

Figure 4.11 shows the weights associated with each node.
The summation formula of the neuron H_{00} is defined by:

$$\sum_{i=0}^{n-1} w_i\, x_i = W_0 X_O + W_1 X_1 + W_{B_{11}} * B_{11} \qquad (17)$$

We can get the values for w0 and w1 from clf.coefs_ like this:
w0 = clf.coefs_[0][0][0] and w1 = clf.coefs_[0][1][0] and so on.
The main reason, why we train a classifier is to predict results for new samples. We can do this with the predict method. The method returns a predicted class for a sample, in our case a “0” or a “1” or a “2” or a”3” or a”4” or a”5” or a”6” or a”7” or ”8”. Instead of just looking at the class results, we can also use the predict_proba method to get the probability estimates.

```
(' number of iterations the solver: ', 38)
('num of layers: ', 4)
('Num of o/p: ', 9)
Accuracy:0.89%
[3]
[0 1 2 3 4 5 6 7 8]
[[  8.49610788e-03   2.08015451e-04   6.45385669e-04   9.82753473e-01
    2.44487594e-05   5.87369165e-03   1.10866244e-03   6.08322541e-04
    2.81892950e-04]]
>>>
```

Figure 4.12: Prediction value

In the Figure 4.12 shows the real predicted value which is **[3]** third head movement and there are even the probabilities of the prediction depending upon the weights. The class with higher weight would be the final predicted output.

4.4.5 Training and classification results

A. Data Acquisition Data was collected from five persons. Data for each person was collected in two periods of ten minutes, with the user being prompted to give a specified movement every 6 seconds. Each specified movement was chosen randomly from the following: forward, backward, left and right, downright, down left, upright, up left.

B. Optimisation of Network Architecture. The recorded movements of five users were randomly divided into training sets. Each set contained 100 samples of each movement. Training data was taken from the recorded movements of users 1, 2, 3, 4, 5 corresponding to 4500 samples. Different MLP neural networks with varying numbers of hidden nodes were trained to select the optimal network architecture. Nine outputs, each corresponding to one of the classes: forward, backward, left and right, downright, down left, upright, up left.

C. Head Movement Classification

1) Experiment 1: A MLP neural network classifier having two hidden nodes was trained using the training data described earlier. The performance of the trained network was tested using the first set of movement samples of 1, 2 users. The trained network classifies each type of movement and the trained network can classify all samples in the test set with an accuracy of 89%.

2) Experiment 2: More training data around 10000 data was taken from the second set of movement samples of user 1 and 2. The MLP neural network was trained using the same procedure in Experiment 1. The performance of the trained network was also tested using the live movements of user 1 and 2. The trained network can classify all samples in the test set with a success rate of 98.03%.

4.5 System accuracy

Table 4.1 shows the confusion matrix for 8 movements. The experiment is done in such a way that every individual movement was tested for about 80 times and the obtained true values are noted and for each individual movement accuracy was calculated.

Table 4.1 Confusion Matrix for 8 movements

		Predicted Classification							
	Movements	***Up***	***Down***	***Right***	***Left***	***Upright***	***Up left***	***Down right***	***Down left***
Actual Movement	***Up***	72	0	0	0	2	4	0	0
	Down	0	71	0	1	0	0	3	5
	Right	0	1	70	0	6	0	4	0
	Left	0	1	1	70	0	3	0	7
	Upright	5	1	6	0	67	0	0	0
	Up left	4	0	0	7	0	69	0	0
	Downright	0	8	6	0	0	0	66	0
	Down left	0	9	0	5	0	0	0	68
	Accuracy (%)	***90%***	***88.75%***	***87.5%***	***87.5%***	***83.75%***	***86.25%***	***82.5%***	***85%***

The table 4.2 system accuracy gives the information about the accuracy of the proposed system. The confusion matrix in the above tables gives the clear image of the number of events or experiments taken and accordingly the accuracy was calculated. The confusion matrix is true positives TP, false negative FN, false positive FP and true negative TN. There are 120 total number of events of the total system and from the confusion matric we can consider the true positive TP as 72 and the false negative as 18, false positive FP is 7. The experiment was implemented by collecting the 120 trials and the estimated value to calculate the accuracy from the confusion matrix.

Table 4.2 Accuracy of the overall system

	Experiment on Person1	Experiment Person 2
Total number of events	120	120
True Positives	89	85
False Positives	12	9
True Negatives	3	5
False Negatives	16	21
Confusion Matrix: $\begin{vmatrix} TP & FN \\ FP & TN \end{vmatrix}$	**Confusion Matrix:** $\begin{vmatrix} 89 & 16 \\ 12 & 3 \end{vmatrix}$	**Confusion Matrix:** $\begin{vmatrix} 85 & 21 \\ 9 & 5 \end{vmatrix}$
Precision: $\frac{TP}{TP+FP}$	**Precision:** $\frac{89}{89+12} \times 100 = 88.11\%$	**Precision:** $\frac{85}{85+9} \times 100 = 90.42\%$
Recall: $\frac{TP}{TP+FN}$	**Recall:** $\frac{89}{89+16} \times 100 = 84.76\ \%$	**Recall:** $\frac{85}{85+21} \times 100 = 80.18\ \%$
$\mathbf{System(Accuracy)} = 2 \times \frac{precision \times recall}{precision + recall}$	$\mathbf{System(Accuracy\ A1)} = 2 \times \frac{88.11 \times 84.76}{88.11 + 84.76} = 86.40\%$	$\mathbf{System(Accuracy\ A1)} = 2 \times \frac{90.42 \times 80.18}{90.42 + 80.18} = 85.69\%$
Overall System Accuracy =86.048%		

4.6 Graphical representation

Figure 4.13 is the graphical representation of the results obtained after using the MLP algorithm. The x-axis represents the training data, the y-axis represents the IMU data for all the nine positions and the pitch values, roll values are represented with two different colors and also the black line represents the positions. Figure 4.14 clearly shows the real difference between each position. The black line in the Figure 4.13 is the same as the graph in Figure 4.14.

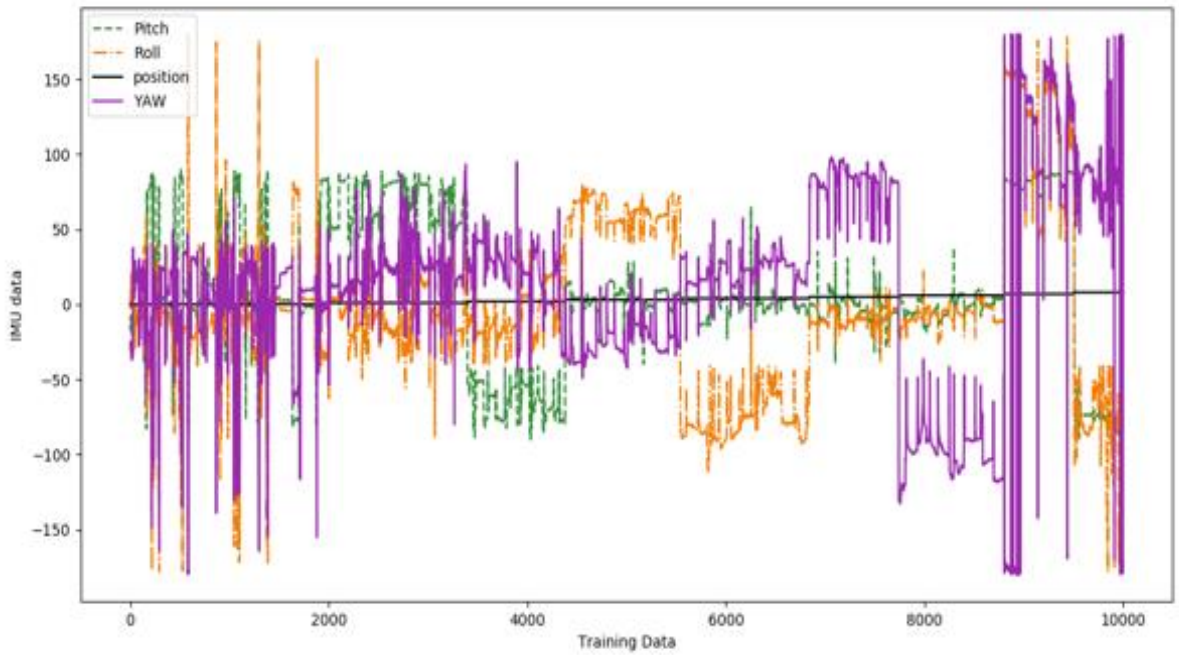


Figure 4.13: Results with MLP trained data

The Figure 4.14 is the graphical representation of the X, Y, Z axes of IMU when the head is in a stable position. The Figure 4.15 is the graphical representation of the X, Y, Z axes of IMU when the head moves upwards. The Figure 4.16 is the graphical representation of the X, Y, Z axes of IMU when the head moves downwards. The Figure 4.17 is the graphical representation of the X, Y, Z axes of IMU when the head moves left side. The Figure 4.18 is the graphical representation of the X, Y, Z axes of IMU when the head moves right side. The Figure 4.19 is the graphical representation of the X, Y, Z axes of IMU when the head moves up left. The Figure 4.20 is the graphical representation of the X, Y, Z axes of IMU when the head moves up right. The Figure 4.21 is the graphical representation of the X, Y, Z axes of IMU when the head moves down left. The Figure 4.22 is the graphical representation of the X, Y, Z axes of IMU when the head moves downright.

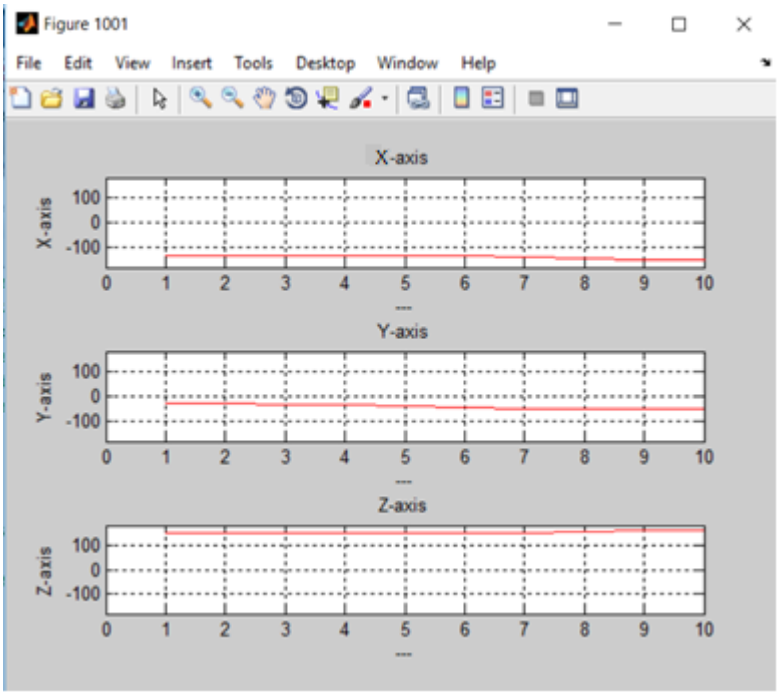


Figure 4.14: X Y Z axis data of IMU when head is in stable position

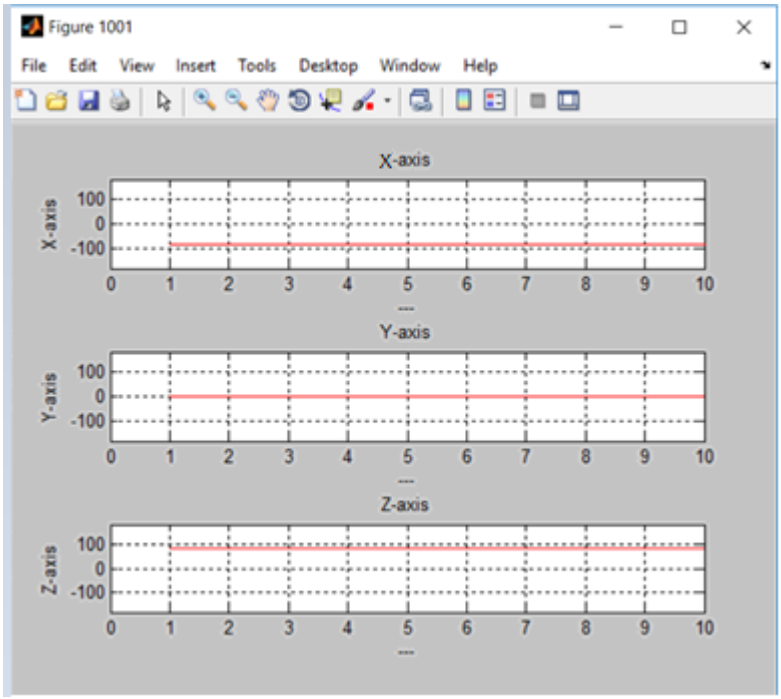


Figure 4.15: X Y Z axis data of IMU when the head moves upwards

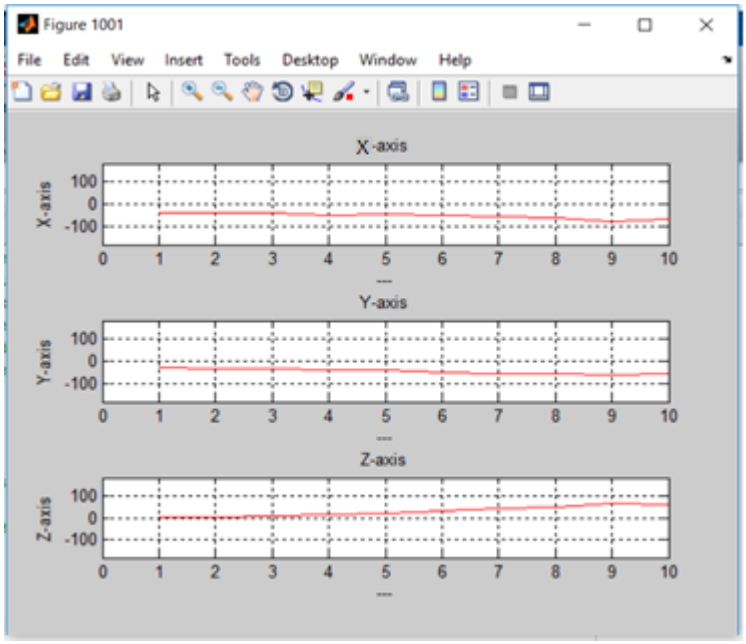


Figure 4.16: X Y Z axis data of IMU when the head moves downwards

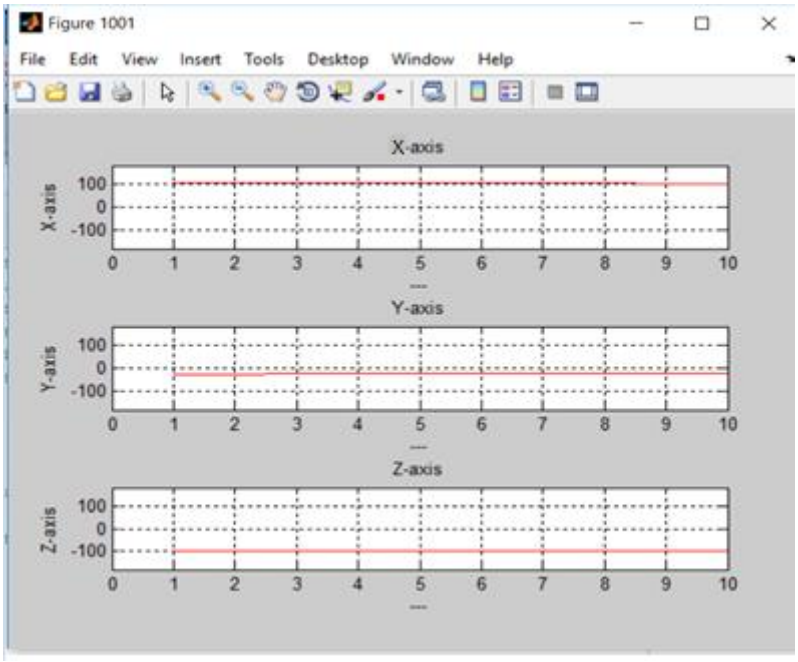


Figure 4.17: X Y Z axis data of IMU when the head moves left side

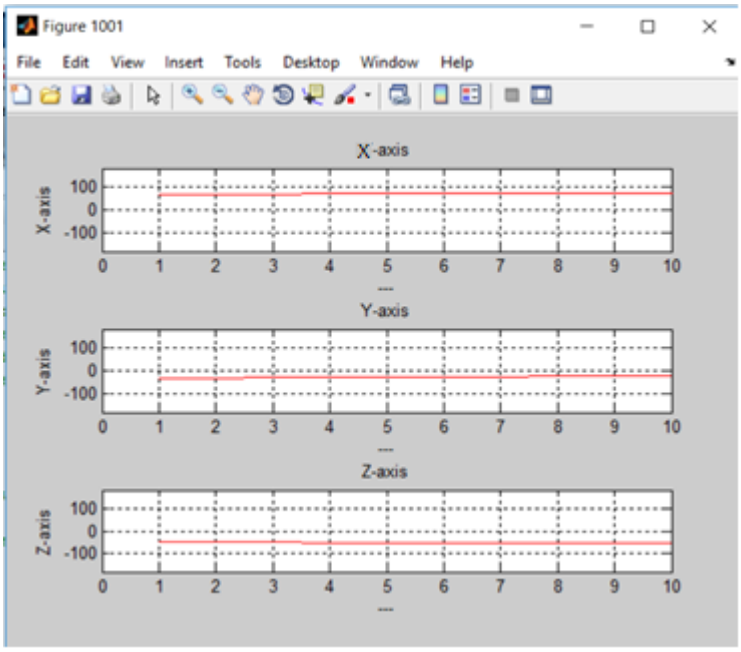


Figure 4.18: X Y Z axis data of IMU when the head moves right side

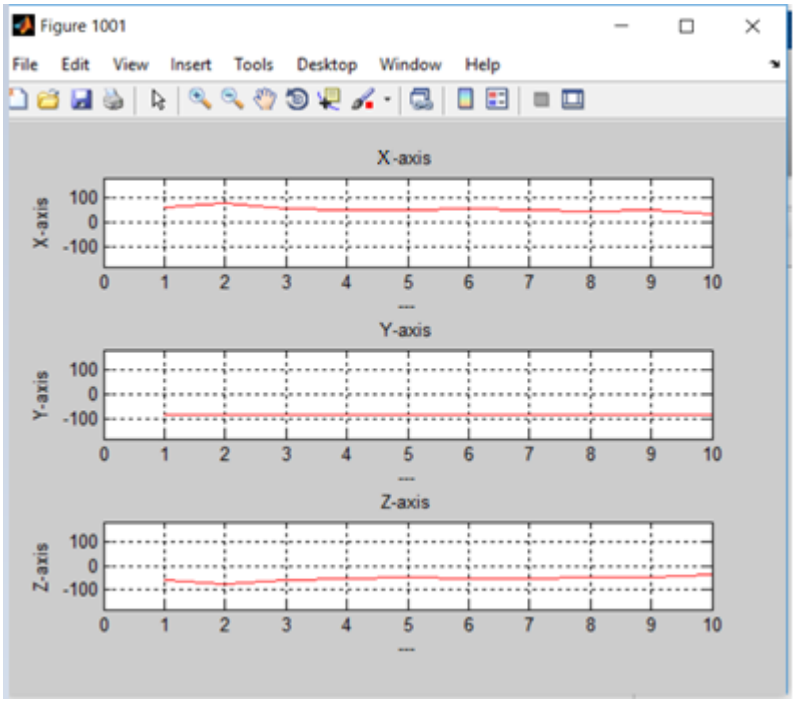


Figure 4.19: X Y Z axis data of IMU when the head moves up left

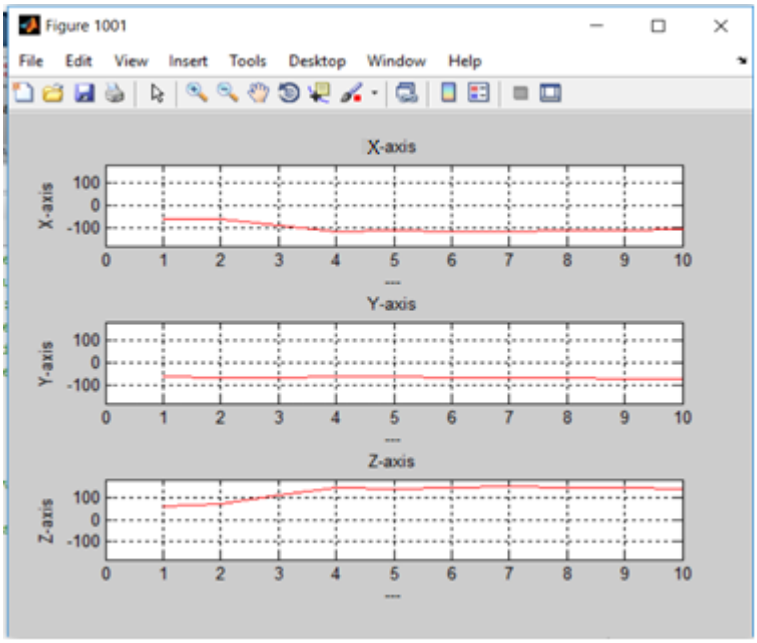


Figure 4.20: X Y Z axis data of IMU when the head moves upright

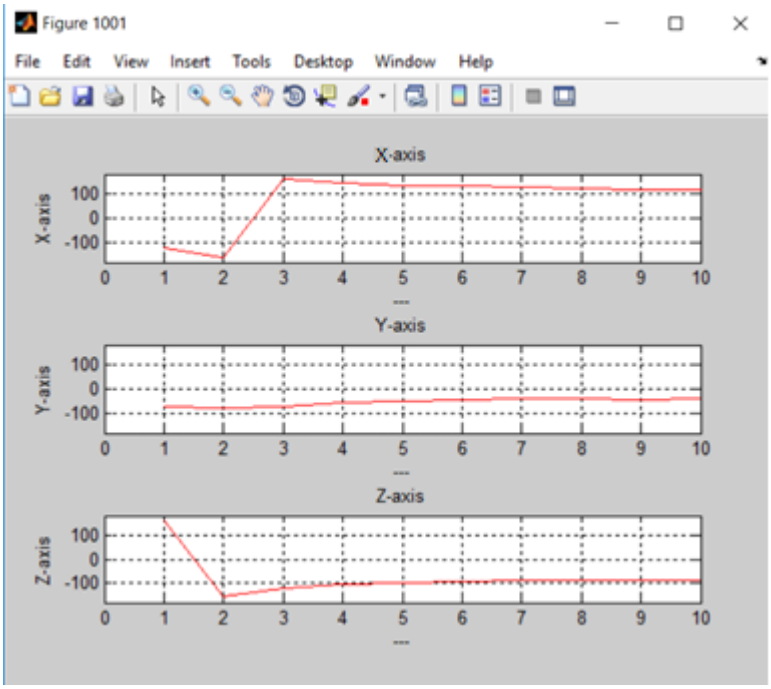


Figure 4:21 X Y Z axis data of IMU when the head moves down left

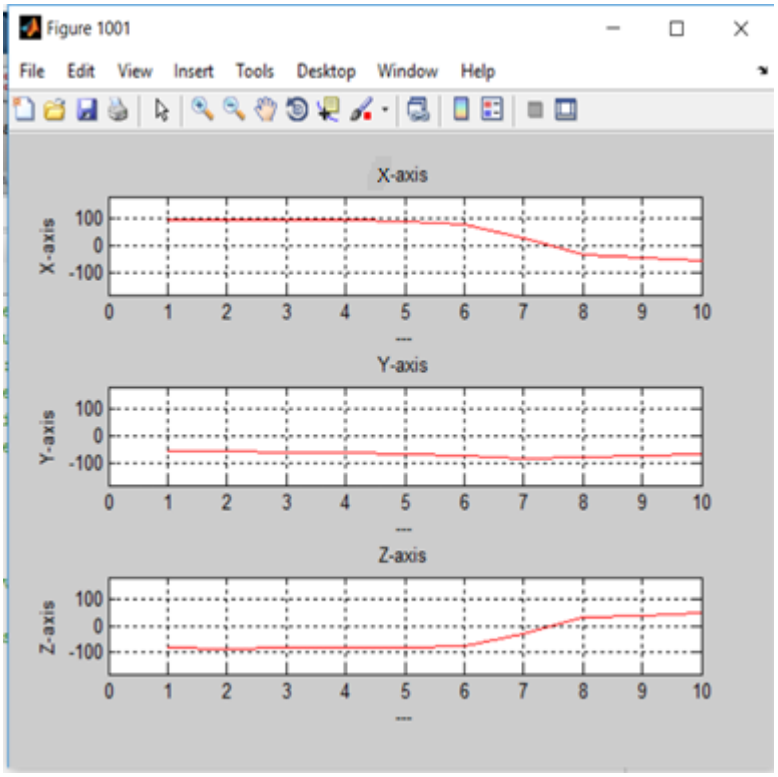


Figure 4.22: X Y Z axis data of IMU when the head moves downright

CHAPTER 5
CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

With the proposed concept, the wearable headband was designed in this thesis. It is an auto-calibrated head orientation based headband. The main aim of this system is to help the quadriplegic, paralyzed patients to communicate with the other people for their basic necessities by using their simple head movements. The system is of multi-input and output control system, which includes a GSM module, Xbee module, speakers, and relay. It is a prototype device which is a wireless band this band is placed on the patient's head. The patient can fully depend on the device for any requirement and this device also acts as the home automation as it can be used to operate the electronic appliances like fan and light. The device is tested and works satisfactorily with the minimum assistance to the person suffering from quadriplegia. It has a good response with IMU 9dof sensor placed on the head. The standard gesture patterns were generated by the motion of the head. The patient would not be strained while using the headband as it involves very simple and convenient head movements for the patient. The battery connected to the IMU was the 9v battery and it is easily dischargeable therefore must be replaced frequently it is one of the limitations of the system. The most advantageous of this thesis is to use the IMU with the three sensors fused in which it gives the most accurate head movement recognition and also the module is small in size and can easily mount on the headband with low power consumption and less in cost.
The results obtained shows that artificial neural networks using Multi-layer perceptron can be used to classify head movement accurately. The use of two hidden nodes is an optimal choice for the network architecture as it and the fast training algorithm. This number of hidden nodes guarantees the best generalization of the trained network. When the MLP was trained using head movement data with maximum accuracy with a significant improvement.

5.2 Recommendation for the future work

The recommendations for the future work is discussed here

- In the system we can add any device to operate therefore the proposed system can be further extended in the home automation applications.
- In order to monitor the patient's health conditions, there are many biomedical sensors such as GSR to measure stress, blood pressure sensor and temperature sensor, pulse sensors which can be added.
- The system can be extended by using the GSM module that can sense GPS co-ordinates in case of any emergency alert is given, the doctor who is attending the particular patient can get the location of the patient precisely, and that would make the doctor's job easy and also immediate attention can be given to the patient.

REFERENCES

[1] E. G. Widerström-Noga and D. C. Turk, "Types and effectiveness of treatments used by people with chronic pain associated with spinal cord injuries: influence of pain and psychosocial characteristics," *Spinal Cord*, vol. 41, no. 11, pp. 600–609, 2003.

[2] S. Mitra, and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 3, pp. 311-324, 2007.

[3] T. M. Nirmal, "Wheelchair for physically and mentally disabled persons," *International Journal of Electrical and Electronics Research*, vol. 2, pp. 112-118. 2017

[4] R. Xu, S. Zhou, and W. J. Li, "MEMS accelerometer based nonspecific-user hand gesture recognition," *2012, IEEE sensors journal*, pp. 1166-1173, 2012.

[5] C. M. Rahul, and N. C. Iyer, "Voice and Accelerometer Controlled Wheelchair," *International Journal of Engineering Research in Electronics and Communication Engineering*, vol 2, no.11, 2015.

[6] J. Takahashi, S. Suezawa, Y. Hasegawa, and Y. Sankai, "Tongue motion-based operation of support system for paralyzed patients," *2011 IEEE International Conference on Rehabilitation Robotics (ICORR)*, pp. 1-6, 2011.

[7] Y. S. Desai, "Natural Eye Movement & its application for paralyzed patients," *2013 International Journal of Engineering Trends and Technology (IJETT)*, pp. 679-686, 2013.

[8] A. Al-Rahayfeh and M. Faezipour, "Eye Tracking and Head Movement Detection: A State-of-Art Survey," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 1, pp. 2100212–2100212, 2013.

[9] D. Katz, L. B. Hafsia, O. Salem, and A. Mehaoua, "Intelligent remote control of smart home devices using physiological parameters," *2015 IEEE 17th International Conference on* E-*health Networking, Application & Services (HealthCom)*, pp. 280-285, 2015

[10] E. J. Rechy-Ramirez, H. Hu, and K. Mcdonald-Maier, "Head movements based control of an intelligent wheelchair in an indoor environment," *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012.

[11] V. Anbarasu, and T. Ravi, "Head Mounted Input Device Using MEMS Sensors," *2012 Indian Journal of Computer Science and Engineering (IJCSE)*, 2012.

[12] B. Fang, N. D. Lane, M. Zhang, and F. Kawsar, "Headscan: A wearable system for radio-based sensing of head and mouth-related activities," *2016, In Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, 2016.

[13] S. Spinsante and E. Gambi, "Home automation systems control by head tracking in AAL applications," *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, 2012.

[14] S. T. Nguyen, H. T. Nguyen, P. B. Taylor, and J. Middleton, "Improved head direction command classification using an optimised bayesian neural network," *2006. EMBS'06. 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5679-5682, 2006.

[15] L. King, H. Nguyen, and P. Taylor, "Hands-free Head-movement Gesture Recognition using Artificial Neural Networks and the Magnified Gradient Function," *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005.

[16] N. Rudigkeit, M. Gebhard, and A. Graser, "Evaluation of control modes for head motion-based control with motion sensors," *2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) Proceedings*, 2015.

[17] M. F. Ruzaij, S. Neubert, N. Stoll, and K. Thurow, "Auto calibrated head orientation controller for robotic-wheelchair using MEMS sensors and embedded technologies," *2016 IEEE Sensors Applications Symposium (SAS)*, 2016.

[18] Y. Zhao, and H. Yan, "Head orientation estimation using neural network," *2011 International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 2075-2078, 2011.

[19] T. Siriteerakul, Y. Sato, and V. Boonjing, "Estimating change in head pose from low resolution video using LBP-based tracking," *2011 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*, 2011.

[20] Z. Zhao, Y. Wang, and S. Fu, "Head movement recognition based on Lucas-Kanade algorithm," *2012 IEEE International Conference on Computer Science & Service System (CSSS)*, pp. 2303-2306, 2012.

[21] K. M. Vamsikrishna, D. P. Dogra, and H. Bhaskar, "Classification of head movement patterns to aid patients undergoing home-based cervical spine rehabilitation," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[22] "Multilayer Perceptron," *Multilayer Perceptron — DeepLearning 0.1 documentation*. [Online]. Available: http://deeplearning.net/tutorial/mlp.html. [Accessed: 04-Jan-2018].

[23] "Introduction to Multi-Layer Perceptrons," *Introduction to Multi-Layer Perceptrons (Feedforward Neural Networks) — Notes de cours IFT6266 Hiver 2012*. [Online].Available: http://www.iro.umontreal.ca/~bengioy/ift6266/H12/html.old/mlp_en.html.[Accessed: 04-Jan-2018].

[24] "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," *Machine LearningMastery*,05-Jul-2017.[Online].Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Accessed: 04-Jan-2018].

APPENDIX

```
###import
import serial
import time
import lcd
import math
import string
import RPi.GPIO as GPIO
import gsm
import os
from sklearn.externals import joblib
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
import pandas as pd
##### INITIALIZATION
gsm.Gsminit()
lcd.lcd_init()
lcd.stringlcd(0x80,"Welcome")
time.sleep(1)
default_port='/dev/ttyUSB0'
 # Check your COM port and baud rate
#ser = serial.Serial(port=default_port,baudrate=9600,
timeout=1)
ser1 = serial.Serial('/dev/ttyUSB0', 9600)
number = ""##GSM NUMBER TO TO THE GURDIAN
count =0
time.sleep(0)
sleep=0
#configure IO ports
act1=7
act2=5
act3=3
act4=11
act5=13
act6=15
act7=19
act8=21
buz=29
GPIO.setup(act1,GPIO.IN,pull_up_down=
GPIO.PUD_UP)
GPIO.setup(act2,GPIO.IN,pull_up_down=
GPIO.PUD_UP)
GPIO.setup(act3,GPIO.IN,pull_up_down=
GPIO.PUD_UP)
GPIO.setup(act4,GPIO.IN,pull_up_down=
GPIO.PUD_UP)
GPIO.setup(act5,GPIO.IN,pull_up_down=
GPIO.PUD_UP)
GPIO.setup(act6,GPIO.IN,pull_up_down=
GPIO.PUD_UP)
GPIO.setup(act7,GPIO.IN,pull_up_down=
GPIO.PUD_UP)
GPIO.setup(act8,GPIO.IN,pull_up_down=
GPIO.PUD_UP)
GPIO.setup(act1,GPIO.OUT)
GPIO.setup(act2,GPIO.OUT)
GPIO.setup(act3,GPIO.OUT)
GPIO.setup(act4,GPIO.OUT)
GPIO.setup(act5,GPIO.OUT)
GPIO.setup(act6,GPIO.OUT)
GPIO.setup(act7,GPIO.OUT)
GPIO.setup(act8,GPIO.OUT)
GPIO.setup(buz,GPIO.OUT)
GPIO.output(buz,False)
GPIO.output(act1,False)
GPIO.output(act2,False)
GPIO.output(act3,False)
GPIO.output(act4,False)
GPIO.output(act5,False)
GPIO.output(act6,False)
GPIO.output(act7,False)
GPIO.output(act8,False)
def action1():
    if GPIO.input(act1):
        GPIO.output(act1,False)
        gsm.Sendmsg(number,"act1  Fan OFF")
        os.system('mpg321 FanOFF.mp3 &')
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"FAN OFF")
        print ("act1 false")
        ser1.flushInput()
        main()

    elif act1 != 1:
        GPIO.output(act1,True)
        gsm.Sendmsg(number,"act1 Fan ON")
        os.system('mpg321 FanON.mp3 &')
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"FAN ON")
        print("act1 true")
        ser1.flushInput()
        main()

def action2():
    if GPIO.input(act2):
        GPIO.output(act2,False)
        gsm.Sendmsg(number,"act2 Light OFF")
        os.system('mpg321 LightOFF.mp3 &')
        print ("act2 Light false")
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"LIGHT OFF")
        ser1.flushInput()
        main()

    elif act2 != 1:
        GPIO.output(act2,True)
        gsm.Sendmsg(number,"act2 Light ON")
        os.system('mpg321 LightON.mp3 &')
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"LIGHT ON" )
        print("act2 Light true")
        ser1.flushInput()
        main()

def action3():
    if GPIO.input(act3):
        GPIO.output(act3,False)
        gsm.Sendmsg(number,"Water")
        os.system('mpg321 water.mp3 &')
        print ("act3 false")
        ser1.flushInput()

    elif act3 != 1:
        GPIO.output(act3,True)
        gsm.Sendmsg(number,"Water")
        print("act3 true")
        os.system('mpg321 water.mp3 &')
        ser1.flushInput()

def action4():
    if GPIO.input(act4):
        GPIO.output(act4,False)
        gsm.Sendmsg(number,"Food")
        print ("act4 false")
        os.system('mpg321 Food.mp3 &')
        ser1.flushInput()
        main()

    elif act4 != 1:
        GPIO.output(act4,True)
        gsm.Sendmsg(number,"Food")
        print("act4 true")
        os.system('mpg321 Food.mp3 &')
        ser1.flushInput()
        main()def action5():
    if GPIO.input(act5):
       GPIO.output(act5,False)
        gsm.Sendmsg(number,"Toilet")
```

```
        os.system('mpg321 Toilet.mp3 &')
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"Toilet")
        print ("act5 false")
        ser1.flushInput()
        main()

    elif act5 != 1:
        GPIO.output(act5,True)
        gsm.Sendmsg(number,"Toilet")
        os.system('mpg321 Toilet.mp3 &')
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"Toilet")
        print("act5 true")
        ser1.flushInput()
        main()

def action6():
    if GPIO.input(act6):
        GPIO.output(act6,False)
        gsm.Sendmsg(number,"Medicine")
        print ("act6 Medicine false")
        os.system('mpg321 Medicine.mp3 &')
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"Medicine")
        ser1.flushInput()
        main()

    elif act6 != 1:
        GPIO.output(act6,True)
        gsm.Sendmsg(number,"Medicine")
        os.system('mpg321 Medicine.mp3 &')
        lcd.lcd_init()
        lcd.stringlcd(0xC0,"Medicine" )
        print("act6 Medicine true")
        ser1.flushInput()
        main()

def action7():
    if GPIO.input(act7):
        GPIO.output(act7,False)
        gsm.Sendmsg(number,"Doctor")
        print ("act7 false")
        os.system('mpg321 Doctor.mp3 &')
        ser1.flushInput()
        main()

    elif act7 != 1:
        GPIO.output(act7,True)
        gsm.Sendmsg(number,"Doctor")
        print("act7 true")
        os.system('mpg321 Doctor.mp3 &')
        ser1.flushInput()
        main()

def action8():
    if GPIO.input(act8):
        GPIO.output(act8,False)
        gsm.Sendmsg(number,"Emergency")
        print ("act8 false")
        os.system('mpg321 Emergency.mp3 &')
        ser1.flushInput()
        main()
    elif act8 != 1:
        GPIO.output(act8,True)
        gsm.Sendmsg(number,"Emergency")
        print("act8 true")
        os.system('mpg321 Emergency.mp3 &')
        ser1.flushInput()
        main()'''Training File and Features and Label
selecion'''
df=pd.read_csv('ppr_500.txt')#Filenamedf1
=df.replace(np.nan, '', regex=True)
```

```
print    (df)#---------------------------------->print    traning
data
#define X & Y
X = np.array(df1.drop(['position'], 1))
y = np.asarray(df1['position'] )#dtype = '|S6'
X_train,        X_test,        y_train,        y_test        =
train_test_split(X,y,test_size=0.2, random_state = 5)
'''MLP Realtime live training'''
clf=MLPClassifier(activation='tanh',solver='adam',max_i
ter=400,
hidden_layer_sizes=(900,90),learning_rate='adaptive',
verbose=True, random_state=1)
clf.fit(X_train, y_train)
joblib.dump(clf, 'final.sav')
'''Trained Model'''
clf = joblib.load('final.sav')
'''Print Accuracy'''
accuracy = clf.score(X_test, y_test)*100
print ("Accuracy:"+str(accuracy)+"%")#----------------------
>print acurracy
'''Stable the sensor'''
lcd.stringlcd(0x80,"Stable the sensor")
time.sleep(3)
while(1):
    x = ser1.read(1)
    if(x == '*'):
        y = ser1.read(1)
        lcd.stringlcd(0x80,"L:"+str(y))
        time.sleep(0.5)
        if(y == '0'):
            count = count + 1
        else:
            count = 0
        if(count > 10):
            count = 0
            break
        ser1.flushInput()
lcd.stringlcd(0x80,"Stabled")
time.sleep(3)
ser1.flushInput()
 ######MAIN FUNCTIONS#######
def main():
    dar = 0
    dal = 0
    darl = 0
    daroll = 0
    dap = 0
    grad2rad = 3.141592/180.0
    GPIO.output(buz,True)
    time.sleep(1.5)
    GPIO.output(buz,False)
    global clf
    while (1):
        line = ser1.readline()
        line = line.replace("#YPR=","")
        line  =  line.replace("#YPRAMG=","")          # Delete
"#YPR="
                                                      #    Write    to
the output log file
words = string.split(line[4:],",")
        if len(words) > 2:
                try:
                    dap = -float(words[1])*grad2rad
 daroll = -float(words[2])*grad2rad
                except Exception as e:
                    print e
        print daroll
        if (daroll > 0.5):
            dar += 1
        else:


       dar = 0
```

```
      if (dar > 5):
          Activities()
          print ("STABLED")
       print dar
def Activities():
   p1=0
   p2=0
   p3=0
   p4=0
   p5=0
   p6=0
   p7=0
   p8=0
   global clf
   action3()
   sleep = 0
   while(1):
x = ser1.read(1)
       if(x == '*'):
 line = ser1.readline()
          print (line)
          line = line.replace("#YPR=","")

          line                                                    =
line.replace("#YPRAMG=","").replace("\n","")
          # Delete "#YPR="
          line = line.rstrip()
 raw_ypr = line.split(",")
          if len(raw_ypr)>2:
                try:
                   raw_y = raw_ypr[0]
                   raw_p = raw_ypr[1]
                   raw_r = raw_ypr[2]
                except Exception as e:
                   print e
          print
('yaw='+str(raw_y)+'\t'+'pitch='+str(raw_p)+'\t'+'roll='+st
r(raw_r))
          live_data = np.asarray([raw_p,raw_r],dtype =
'float')
          live_data = live_data.reshape(1, 2)
          prediction = clf.predict(live_data)
          print (prediction)
          y =''.join(map(str, prediction))
          #y= (str(prediction))
                   print (y)
          if(y == '0'):
             if(sleep == 0):
                lcd.stringlcd(0x80,"Normal")
          else:
             if(sleep == 0):
                lcd.stringlcd(0x80,"Position: "+str(y))
                            time.sleep(1)
          if(y == '1'):
             p1 += 1
          else:
             p1 = 0
          if(y == '2'):
             p2 += 1
          else:
             x2 = 0
##           if(y == '3.0'):
##              p3 += 1
##           else:
##     x3 = 0
          if(y == '4'):
             p4 += 1
          else:
             x4 = 0
          if(y == '5'):
             p5 += 1
          else:
            x5 = 0
          if(y == '6'):
             p6 += 1
          else:
             x6 = 0
          if(y == '7'):
             p7 += 1
          else:
             x7 = 0
          if(y == '8'):
             p8 += 1
          else:
             x8 = 0


#####_____________ACTIVITIES_______############
###
          #######_____activity1
          if(p1 >= 5):
             p1 = 0
             sleep = 0
             action1()
             lcd.stringlcd(0x80,"Act1")
             time.sleep(2)
             ser1.flushInput()
          if(sleep == 1):
             continue
          ########_____activity2
          if(p2 >= 5):
             p2 = 0
             sleep = 0
             action2()
             lcd.stringlcd(0x80,"Act2")
             time.sleep(2)
             ser1.flushInput()
          if(sleep == 1):
              continue
          ########_____activity3
          if(p3 >= 5):
             p3 = 0
             sleep = 0
             action3()
             lcd.stringlcd(0x80,"Act3")
             time.sleep(2)
             ser1.flushInput()
          if(sleep == 1):
             continue
          #######_____activity4
          if(p4 >= 5):
             p4 = 0
             sleep = 0
             action4()
             lcd.stringlcd(0x80,"Act14")
             time.sleep(2)
             ser1.flushInput()
          if(sleep == 1):
             continue
         ########_____activity5
          if(p5 >= 5):
             p5 = 0
             sleep = 0
             action5()
             lcd.stringlcd(0x80,"Act5")
             time.sleep(2)
             ser1.flushInput()
          if(sleep == 1):
             continue
        ########_____activity6
          if(p6 >= 5):
             p6 = 0
             sleep = 0
             action6()
            lcd.stringlcd(0x80,"Act6")
             time.sleep(2)
             ser1.flushInput()
          if(sleep == 1):
```

```
            continue
        ########_____activity7
          if(p7 >= 5):

            p7 = 0
            sleep = 0
            action7()
            lcd.stringlcd(0x80,"Act7")
            time.sleep(2)
            ser1.flushInput()
          if(sleep == 1):

            continue
       ########_____activity8
          if(p8 >= 5):
            p8 = 0
            sleep = 0
            action8()
            lcd.stringlcd(0x80,"Act8")
            time.sleep(2)
            ser1.flushInput()
          if(sleep == 1):
            continue
          ser1.flushInput()
ser1.flushInput()

__init__=main()
```