MAJOR PROJECT REPORT
(FOURTH REVIEW)

ON

IRIS AND VOICE BASED MOTORIZED WHEELCHAIR
(vision chariot)

PROJECT GUIDE:

Dr. RAM NIWASH MAHIA

Submitted By

JOYDEEP MUKHERJEE	RA1511005030028
AJISHAN JOSE	RA1511005030029
HIMANSHU GOEL	RA1511005030037

(15EE496L)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
FACULTY OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report titled "**IRIS AND VOICE BASED MOTORIZED WHEELCHAIR (VISION CHARIOT)**" is the bonafide work of "**AJISHAN JOSE [Reg. No:RA1511005030029], HIMANSHU GOEL [Reg No:RA1511005030037], JOYDEEP MUKHERJEE [Reg. No:RA1511005030028]**", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGN

DR. RAM NIWAS MAHIA

Asst. Professor, Project Guide

Dept. of Electrical & Electronics

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project coordinator, Dr. Ram Niwas Mahia for his valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department Dr.Himanshu Sharma for encouraging and allowing us to present the project on the topic "**IRIS AND VOICE BASED MOTORIZED WHEELCHAIR (VISION CHARIOT)"** at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all teaching staff of EEE Department that helped us in successfully completing our project work.

We heartily thank our friends for their help and suggestions during this project work.

Ajishan Jose

Himanshu Goel

Joydeep Mukherjee

INDEX

ABSTRACT

The system includes a method for the movement of wheelchair designed for disabled persons. The wheelchair is built to assist and assist the lives of people by reducing the input mechanical energy exerted by them for the movement of wheelchair. The systems presently in use are either expensive and complex or require too much assistance for the movement. The wheelchair reduces the energy exerted by the patient by moving the patient with the help of eye movement. The system is most beneficial to the patients who are paralyzed or can perform restricted ambulatory motions. The system is also cheap with most parts easy to get and getting replaced if needed. This project aims to achieve a sense of independence for the people with restricted ambulatory motions. Precautionary measures are also enabled which allows a feedback to a specified set of individuals in case of an emergency.

1. Introduction

The number of persons who are paralyzed and therefore dependent on others due to loss of self-mobility is growing with the population. The development of the wheelchair for paralyzed users is surprisingly recent starting with the conventional manually powered wheelchairs and advancing to electrical wheelchairs. Conventional wheelchair use tends to focus exclusively on manual use which use which assumes users still able to use their hands which excludes those unable to do so. Diseases or accidents injuring the nervous system also frequently because people lose their ability to move their voluntary muscle. Because voluntary muscle is the main actuator enabling people to move their body, paralysis may cause a person not move their locomotor organ such as arm, leg and others. Paralysis may be local, global, or follow specific patterns. Most paralysis are constant, however there are other forms such as periodic paralysis (caused by genetic diseases), caused by various other factors. Scientist Stephen W. Hawking is perhaps the most well-known victim of major paralysis – Hawking was diagnosed with incurable Amyotrophic Lateral Sclerosis (ALS) in 1962, thereafter using a wheelchair to move. Many of those suffering close to or complete paralysis usually however still can control their eye movement which inspired us to develop an eye-controlled electric wheelchair.

1.1 Problem Statement

Thus, we can summarize our project as follows: the main of this project is to design a wheelchair system which can be controlled using either by eye movement or by voice depending upon the user's comfort.

1.2 Range of Solutions

We wanted to come up with the system that is not expensive and thus can be afforded by all. The main task in the design was to accurately detect the eye movements. Since, the system is for human use we have to take an extra care about the safety of the system.

For the eye detection, we came up with multiple strategies. One was to detect the eye moments using the infrared light. The idea behind this method was to concentrate the infrared light onto the eyes from the forehead. The voltage rating of the infrared light is under the permissible levels that can be used to concentrate in the eyes. The infrared light concentrated through the forehead of the user will be collimated through the pupil of the eye and we can use this collimated light to obtain a voltage drop across the photodiode which will drive the motors.

We built the photo detector circuit and tested this method on one of the group members. During the testing of the circuitry, we found that we were not able to obtain accurate results and the voltage drop across the photo diode was not constant and kept on varying. Thus, the issue with this idea was that the voltage drop obtained across the diode was just 0.7 Volts and it was not sufficient to detect accurately. Thus, we had to come up with an alternative in order to obtain an accurate method to detect the eye as that is the most crucial part of the system and needs to be done with great precision.

The second alternative that we came up with was to connect the web camera to the RaspberyPi directly which would then process the snapshots, eliminating the need to have the camera attached to the laptop. But, we found out that the time taken to understand the new controller board is a lot. Also, normally these days such users have a laptop somehow attached to their wheel chair and thus we decided to rather go ahead with a microcontroller board that we already understand.

The alternative design which we finalized, captures the images using a webcam that will be attached to the laptop placed on the wheelchair of the user. These captured images will be used to detect the

eyes and hence detect the movements using a MATLAB script running on the laptop, which then sends serial commands to the microcontroller circuitry driving the motors attached to the wheel chair. For the above mentioned reasons, we finalized to work with this idea. A complementary system which allows voice based control of the system is also embedded along with a mechanism of turning the wheelchair into a stretcher upon the input of a certain command through voice.

1.3 Product Design Overview

We decided to use the web camera to detect the eye movements which will be further processed to drive the motors. For the simplicity and to make a prototype, we are going to design a small, motorized, wooden platform and we will attach the web camera on the helmet. We will use serial communication to communicate between the web camera and the microcontroller. The microcontroller will be placed on the wheel chair which will be connected to the motors, driving the wheel chair in the direction the person sitting on the chair desires to move in. There is also a voice controlled system which on taking the input from the mike sends the input to the microcontroller through a voice module which can also do the task of movement according to the patient's ease.

2. High Level Design

There are two major components from the system design standpoint -

a) Eye-Detection.

b) Arduino Uno controlled Wheel Chair Assembly.

c)Voice controlled Wheel Chair Assembly.

2.1Eye-Detection and Motion Tracking

A webcam is mounted on a cap, continuously staring at the user's eyes. The webcam wired to the patient's laptop, is running a MATLAB application designed to monitor and react to eye movements. Based on a series of snapshots taken and thereafter processed, the motion of the user's eyes are detected, decision to move the Wheel Chair in a particular direction is taken and communicated serially to Arduino Uno microcontroller. MATLAB 2013 has an image processing toolbox which we utilized for the eye detection.

We used the 'CascadeObjectDetector' capable of detecting eye-shaped objects based on their shape and size. It uses the Viola Jones Algorithm for the same. A description of the Algorithm is given in the software section of the report. Continuous snapshots of every 25th frame are taken and feature points extracted are saved i.e. we capture approximately 1 snapshot every second and process it. Based on the position of the feature points in previous snapshot and current snapshot, a movement is detected and this is communicated to the wheelchair assembly via the serial port.

2.2Voice Controlled Wheelchair Assembly

This is a complementary system added to the design to improve the wheelchair design and ease of use for the user. The system is quite easy to operate. In case of the need of switching to an alternative method of control the user can change the method through a button specifically provided for the aforementioned purpose. The system will then take on voice commands through a mike from a designated previously stored voice to control the movement. The input commands are sent to the voice module which similarly, sends the commands to the Arduino Uno microcontroller.

2.3Arduino Uno controlled Wheelchair Assembly

A decision based on the processing done by the MATLAB application is communicated and received by the Arduino Uno. The controller on reception forces the port pin high on which the motors have been connected for desired motion of the Wheel Chair. 9 Figure 2: Block diagram.

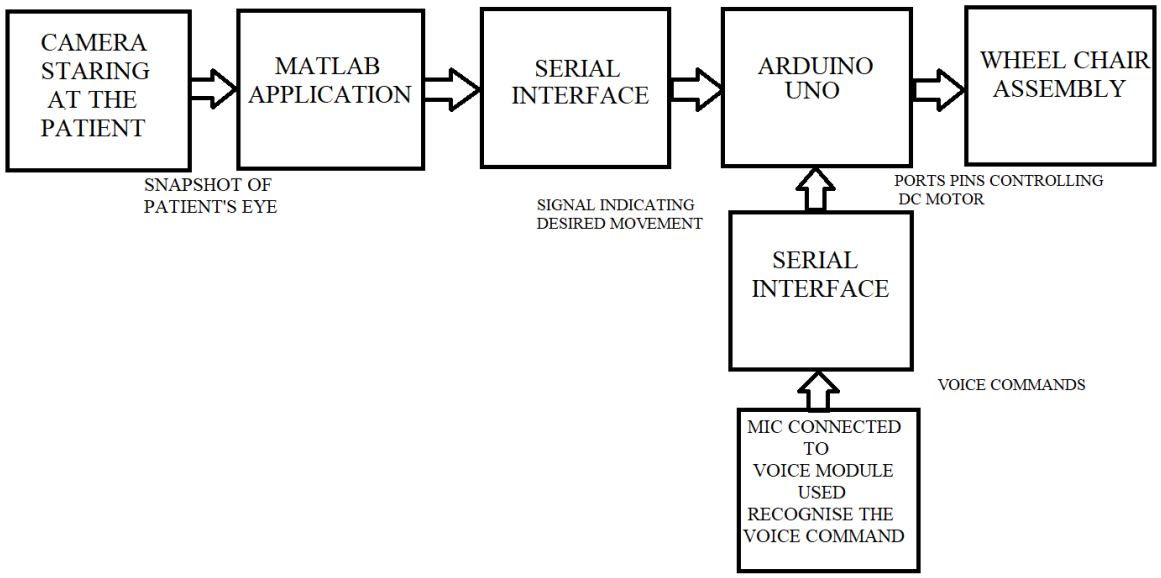

Figure 1: Block Diagram

So, now we will have a look at the overall code structure of our algorithm and the logic behind the decision making. There are two parts in the code structure. The first part is to detect the eye movements and the other part is to drive the motors. The code structure can be explained in the following steps:

1. Initialization: Initially we set up the serial communication that will be used later for the interface between MATLAB and the controller, the video capture and the program variables.

2. Image and Video Processing: We then take continuous video frames and sample the input and save it as the screen shots. Each frame is then converted into the black and white frames. For the accurate results, we perform contrast stretching on each frames to make the dark region darker and bright region brighter. This will enable the detection of the eyes better.

3. Estimation: Now, after working on the each frame we try to detect the eyes. This we do by estimating the position of left as well as the right eye. Thus, we set the threshold and detect the position of the eyes which can be used for the further processing.

4. Detection: Now, in this step we actually detect the eye movements. The idea is to compare the current position of the eye with the previous position. Thus, the difference in the coordinates will help us to predict the motion in the particular eye. But sometimes, it may be possible that only one of the either eye will be detected. In that case, we will give preference to the eye that is detected currently.

5. Error Handling: To avoid detection errors, we incorporated an error handling mechanism, which specifies a threshold for the height and width of a valid eye, by calibrating it for the user. If the detection results give a height and width value lesser or greater than the threshold, the value is voided and not considered for the decision making.

6. Motion: Now after detecting the eye movements, we have to come up with a decision algorithm that will help the controller to drive the motors accordingly:

a. Valid Left: The decision to turn left will be considered as valid if the eye turns left and stays there for a cycle. This action will be detected as a left turn request. If the patient moves its eye before completion of the cycle then this signal should be considered as void.

b. Valid Right: Similarly, the decision to turn right will be considered as valid if the eye turns right and stays there for a cycle. This action will be detected as a right turn request. If the patient moves its eye before completion of the cycle then this signal should be considered as void.

c. Valid Straight: The signal to go straight is when a person looks straight upright. This will be detected as to go straight.

d. Valid Backward: The signal to go backward is when a person looks down. This action will be detected as a backward movement request.

7. Safety Considerations: Given the application of the system, we incorporated a safety mechanism, wherein based on the command of the user the wheel chair halts. If the user wants to halt the wheel chair in case of an emergency, he can say the command ”STOP”, causing the wheel chair to halt. There is also an accelerometer present to detect a fall. Then a GSM Module sends a message to the authorised personnel along with the location of the user through a GPS Module.

8. Serial Communication: Now according to the detected command, the MATLAB application will transmit 0, 1 or 2 for left, right and straight respectively to the controller which will drive the motors.

3. Software Design

There are multiple aspects to the software design of this project. Since majority of computational work is done in software, a lot of our time went in software design and testing. The MATLAB component is responsible for capture of regular snapshots, processing of those snapshots, determining the movement of eyes, algorithm for movement of wheelchair and serial transmission of the decision to move. The firmware component deals with receiving the serial signal, based on which drive the motor connected to the port pins, forcing the wheelchair in the direction it is supposed to move in.

3.1 MATLAB Component

The MATLAB design can be structured into many small sub-parts each of which is described below –

(i) Initialization of variables and setting up serial communication - MATLAB 2013 can easily be configured to serially transmitting data on the Port mentioned in the code. Initially all the already set up serial ports are disabled. After which, we need to mention the current port, by checking the ‘Device Manager’ which indicates the port in use. The baud rate of communication is set to 9600. The communication is set to have no flow control and parity check is disabled.

After setting up the serial communication to enable the data link between MATLAB and Arduino Uno, we reset the variables needed in the course of the program to their initial valuables.

(ii) Image Capture and Eye Detection –

MATLAB 2013 equips an Image Processing Toolbox, which we have used majorly in this section of the Software Design. We use a Microsoft LifeCam HD-5000 web camera which is connected via a USB cable to the Computer on which the MATLAB script is running. We can stream continuous video

signals on MATLAB coming from the camera using the video processing toolbox available. The function ‘imaqhwinfo’ is used to recognize all video capture adaptors. Identifying the correct device and then using it to stream the video signal is the next step.

The requirement of our design was to continuously look at different frames, based on which determine motion. It is practically impossible to do a lot of processing on a per frame basis. That is why we try to sample every 25th frame. So, a snapshot of every 25th frame is captured and processed. We used the ‘getsnapshot’ command to capture these snapshots.

The image is then converted to grayscale image, as we do not need color information to detect eye feature points. The conversion infact makes the detection easier. The ‘imadjust’ command is used then to contrast stretch the image to make darker sections even darker, enhancing the eye feature points useful for the application.

This pre-processing of the image makes the image easier to process and extract the eyes from. After the initial pre-processing, we move towards the eye detection. The Eye Detection is done using the Viola-Jones Object Detection Algorithm. Primarily this algorithm was designated for face detection though it is used for all sorts of object detections. The algorithm is designed to work on sum of pixels in a rectangular area. Viola-Jones algorithm says that face can be detected by looking for rectangle. And then the large rectangle is made up of many such smaller rectangles, which are fundamentally feature points on a human face.

The ‘cascadeobjectdetector’ on MATLAB, utilized this algorithm to extract and detect the eyes of the person. We then show the detected eye by plotting the rectangle at the appropriate position of the eye.

(iii) **Image Processing -** Initially, all we do is monitor if any eye feature points have been detected or not. If not set a flag and display it on the debug screen. To increase the detection accuracy, we wanted to neglect all other points on the screen except the actual eye of the person. The reason being, if anyone except quadriplegic person comes in front of the camera, the person should not affect the system. Also certain things seemingly looking like eyes should be rejected as well. The way we incorporated this is taking into account the height and length of the eye. After repeated testing, we decide a length and height of a valid eye, sets a range around the threshold and reject everything which is outside it.

The blink detection section is not compute intensive. We use a flag which is set each time no valid eyes are detected. If in corresponding frames the flag value sets, it indicates a blink.

What is assumed is that the position of the camera is fixed, relative to which the left and the right eye approximate positions can be estimated. Using this, we try to distinguish and store left and right eyes in different matrices. This helps getting a clear discrimination between both the eyes, helping in easy movement detection.

(iv) **Movement Detection -** The movement detection is done with a very basic principle. The input is taken from either the eye detection or through the voice module. If and only if the input gets processed a then the serial monitor connected to the Arduino Uno would be displaying the output. That output will be displayed on the LCD.

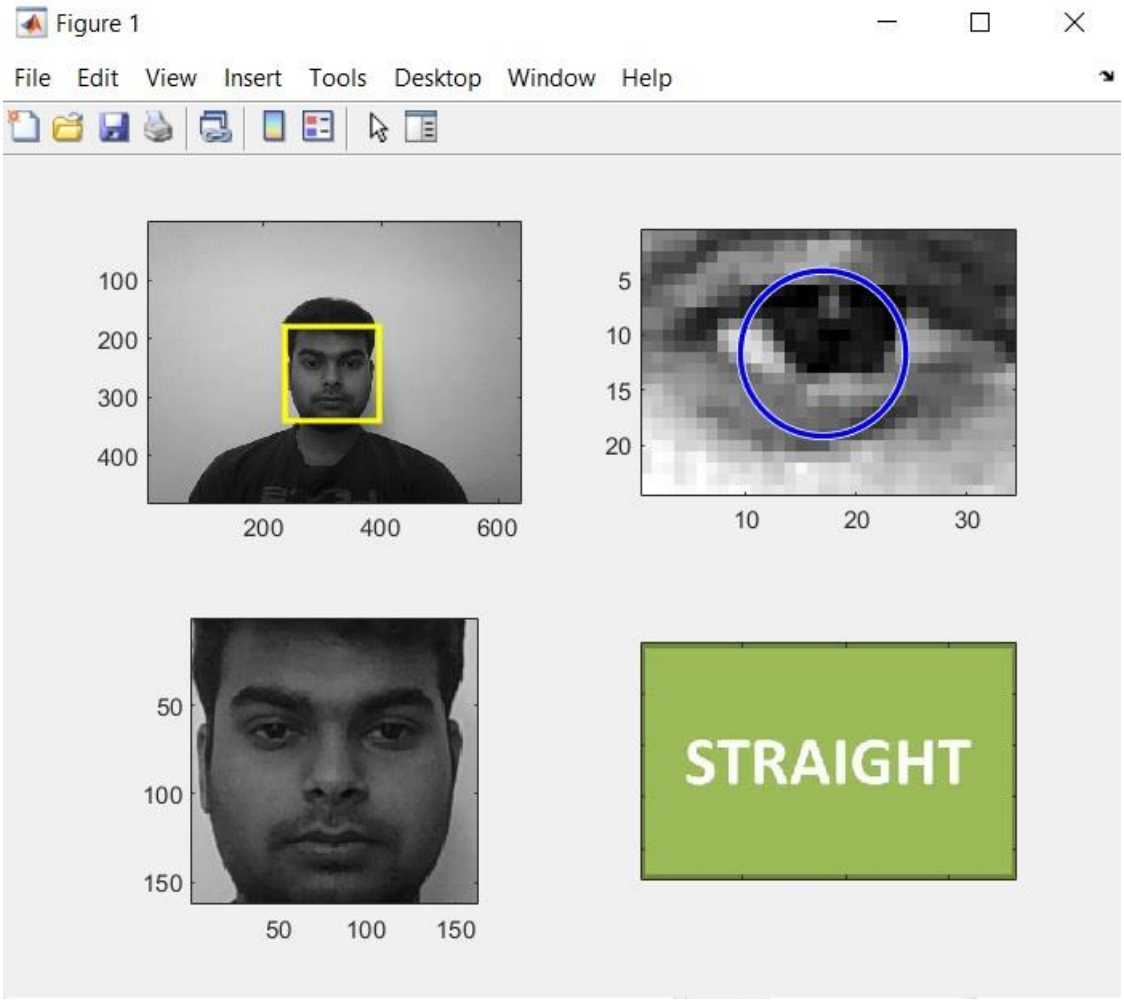

Figure 2: Eye Detection

(v) **Motor signals -** The way we qualify a valid right, left and straight attempts to move, we need to incorporate many factors. The way a valid right is recognized is by moving eye on the right side and stay there for a second, after which the wheel chair starts moving. But the person's face is still tilted on the right. If the person now tries to go back to the initial position by tilting left, the system will detect it and lead to an otherwise invalid left movement of the chair. This has to be avoided.

We set flags for left and right movements each time the wheel chair moves, avoiding precisely this unwarranted behavior of the system. The way a valid straight movement is detected is titling in corresponding frames in left and right directions. Over here as well, the effect of the offset coming into picture are avoided in the same fashion with the help of flags.

As already mentioned, three corresponding blinks should halt the motors. Along with the motion command, a halt command is also transmitted to the microcontroller assembly which thereby halts the motors as per the user's desire.

After determining which direction the wheel chair has to be moved in, the decision is transmitted to the micro-controller via the serial port. The only thing sent is a one digit decision, saying right, left or straight movement.

3.2 Firmware Design

The firmware design is fairly straight forward as all the computation has been done on MATLAB and the only thing which the micro-controller has to do is control the motors to move in a particular direction.

The firmware constantly monitoring the serial input. The firmware turns ON the port pin based on this received signal. After turning ON the port pin, a small delay is given, giving the chair time to move for a fixed time in the desired direction.

As far as the firmware design is concerned, it is fairly simple. All that is required is take in the serial input, move in a direction, give a delay and keep doing this repeatedly.

3.3 GPS and GSM Module

This particular part is controlled by a separate micro-controller known as Arduino Nano. A sensor is connected to detect a 'fall' called as accelerometer. When the chair falls this sensor sends signal to GPS module which first encodes the latitude and longitude then further it is sent to GSM module where the SIM connected to it will send a location encoded message to the registered number.

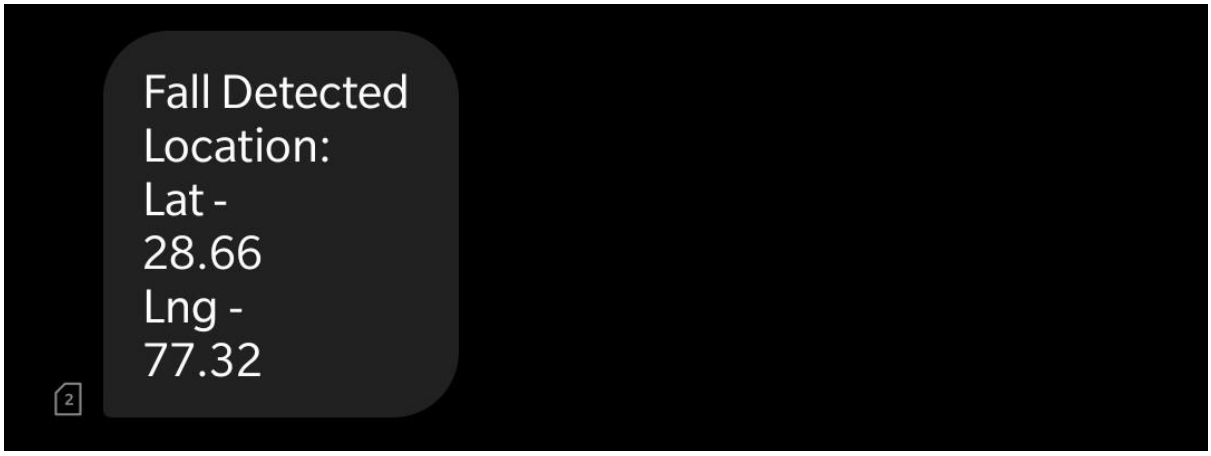

Figure 3: Message Sent Displaying Patient's Location

4. Hardware Design

The hardware part consists only to control the motor and to provide power to the system. The motor controller circuit is given below.

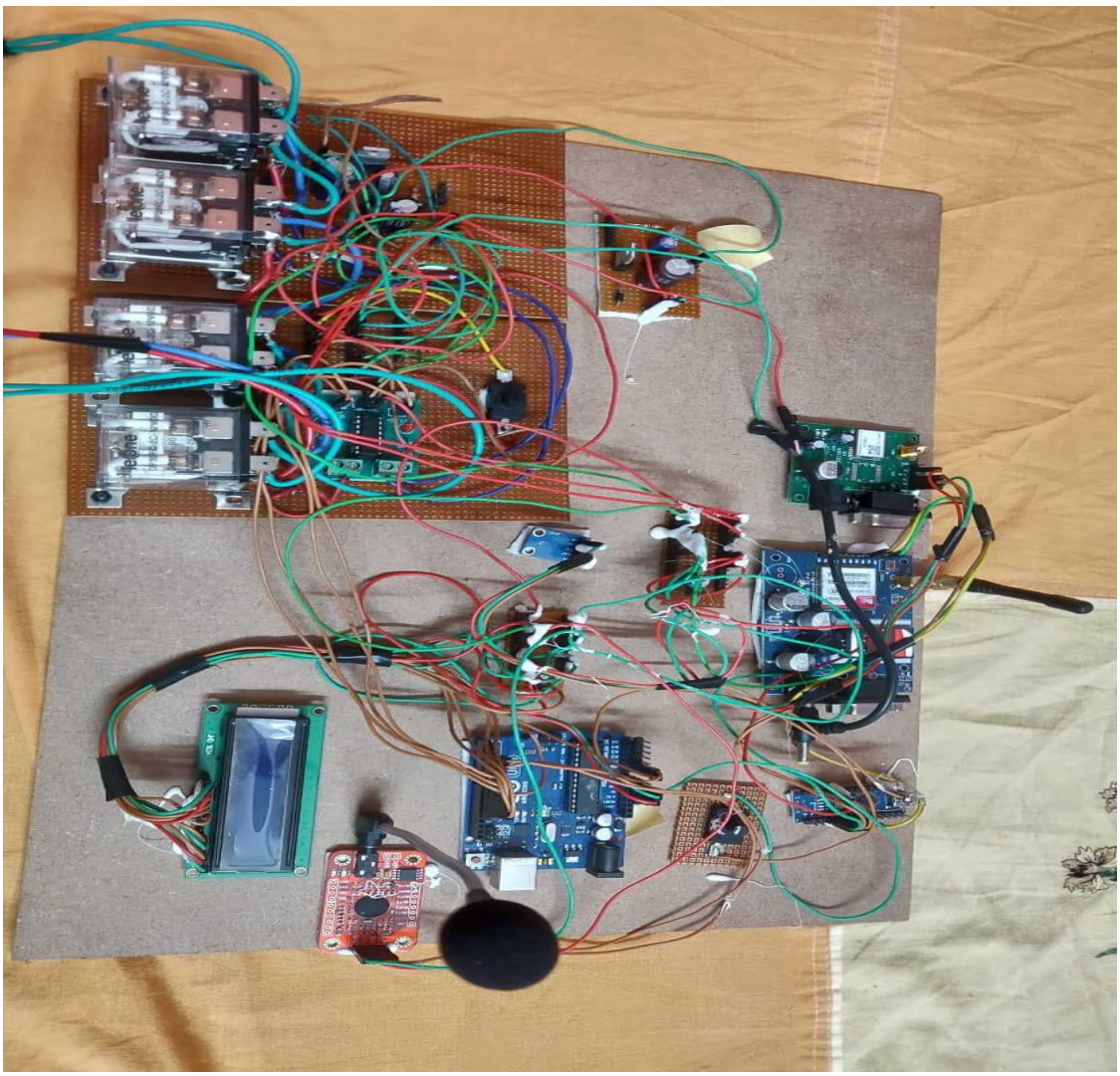

Figure 4: Hardware Circuit

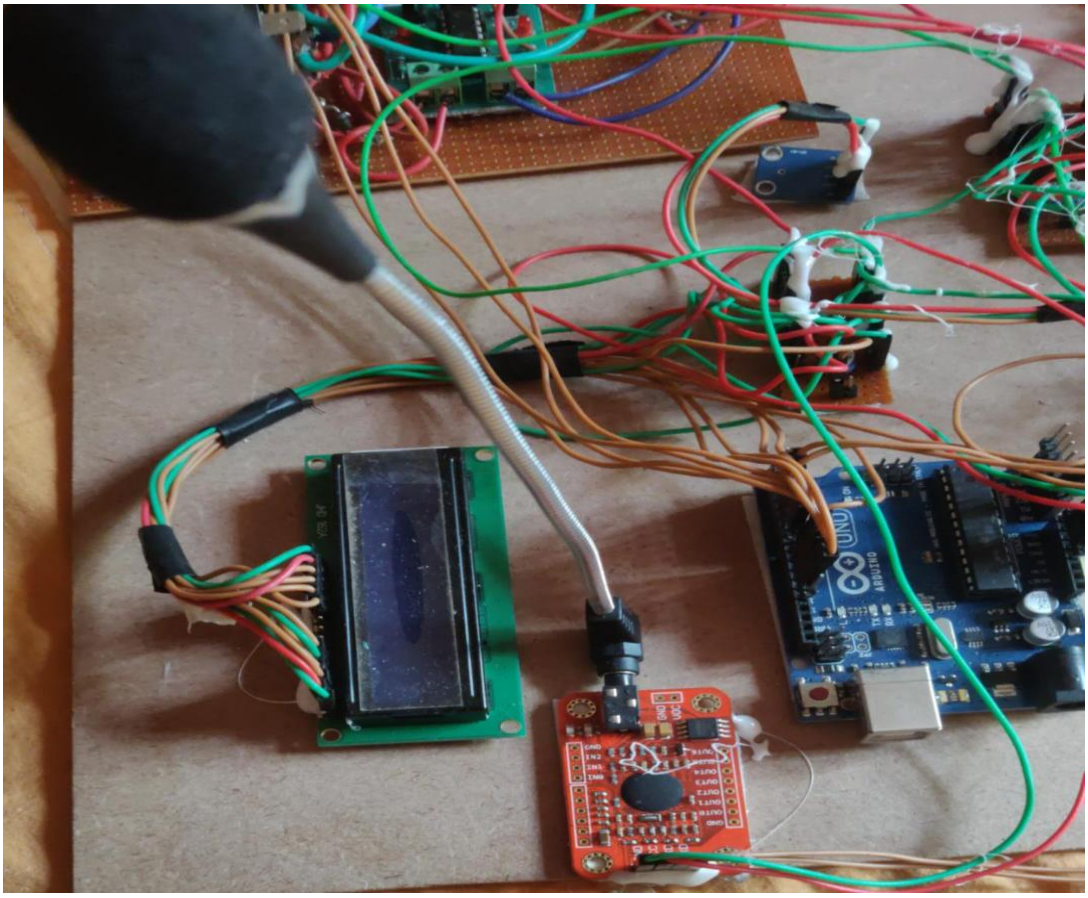

Figure 5: Voice Module Assembly

4.1 Motor Control

The signal that we obtain from the MATLAB script, we then use that to drive the motors. The MATLAB program does all the decision making for the motor to run in which direction. So, the script sends the bit 0,1 and 2 for left, right and straight direction. The circuit shown below is fairly safe and used to drive the motors. An optoisolator completely isolates the microcontroller from the motor. The diode placed across the motor shorts out spikes when the motor is turned off. The resistor grounding the base of the phototransistor is set for best fall time, probably around 1Mohm. The motor capacitor should start around 0.1uf. The pinout of the 4N35 optoisolator and BUZ73 are also shown. Note that the bandwidth of the 4N35 is very small, so we use a low PWM frequency, perhaps about 1000 Hz.

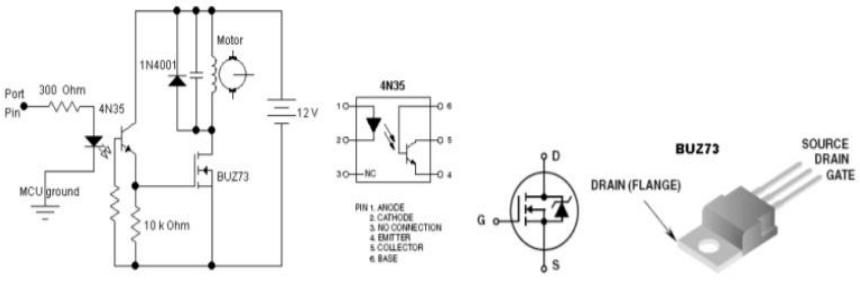

Figure 6: Circuit Diagram

5. **Mechanical Design**

For the mechanical part, we worked on two things: the wheel chair prototype and the web camera mount. For the wheel chair mount, we have tried to build the small prototype from wood with four wheels that are connected to the motors. The photo shown below shows the wheel chair prototype.

Figure 7: Wheelchair Assembly

Now, in order to detect the eye movements, we need to have a web camera in front of the user. For simplicity, we are attaching the web camera onto the head mount like helmet that will be used to detect the eye motion. Since, we are not making the actual wheel chair, thus, we have to use an independent head mount. Otherwise, we can use the web cam directly attach to the chair. The photo of the head mount is given below.

Figure 8: Head mount for the web camera

Figure 9: Stretcher mode for the wheelchair

6. Testing and Results

6.1 Testing Strategy

The debug screen was the most useful aspect of our testing strategy. The debug screen showed whenever nothing was detected as a valid eye by the cascade object detector. When a valid eye is seen, the length, height, x and y pixel co-ordinates are displayed on the debug screen. This is done for every snapshot. After taking the difference between two snapshots, a tilt movement is indicated by either saying ‘moved right’ or ‘moved left’.

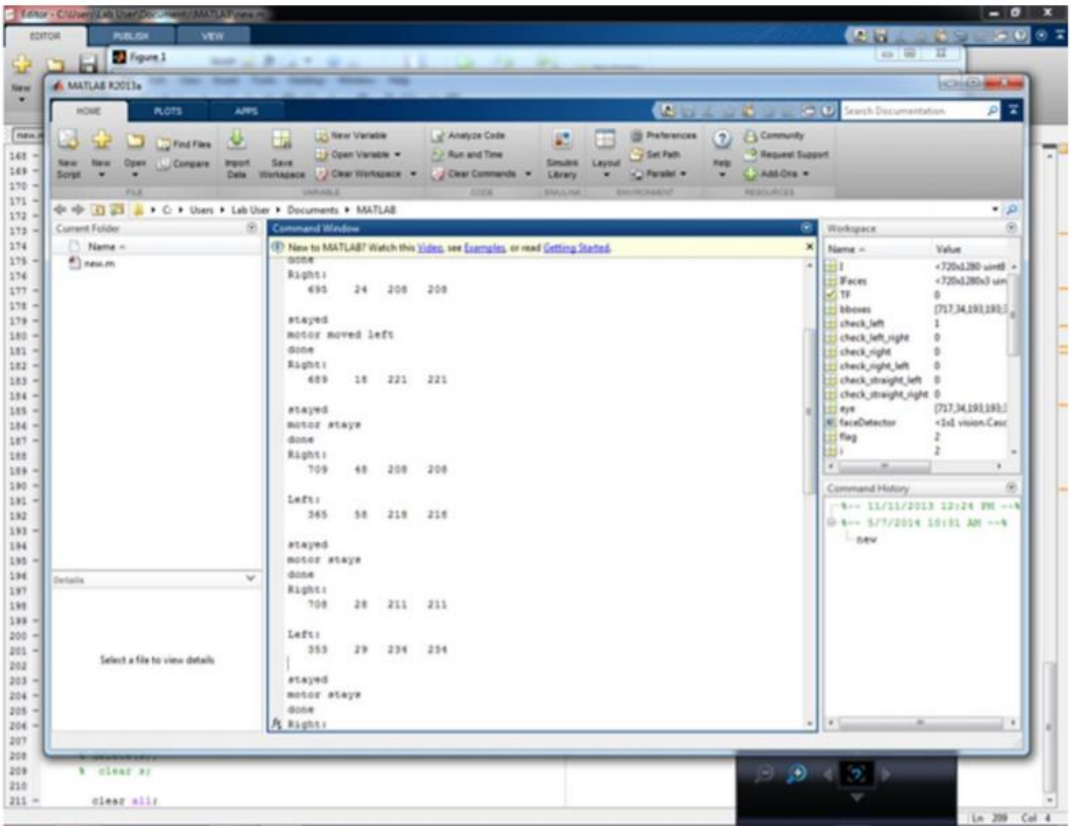

Figure 10: MATLAB Debug Screen

After determining if the motor has to be moved or not and before transmitting the serial signal, on the debug screen, we indicate the intended motion of the motor. This allows us to debug if any error. If the movement suggested on the debug screen and then movement of the wheel chair are in conjunction. This allowed us to test consistently with ease.

6.2 Speed of Execution

We were never actually limited by the speed of execution. We could go upto 15 frames a second, evaluating and determining the face tilt. But there were couple of issues because of which we limited the speed of execution. One being, we do not want the person to get a jerk whenever he wants to move. Hence we compute and drive the motor at a slow pace, as compared to the ability we can escalate the speed to. Also, we want to keep the system secure by taking a while to evaluate the movement and thereafter making a decision, given the criticality of the application.

6.3 Accuracy

The project performs satisfactory with performance accuracy of around 70-90%. The results after testing it for 100 to 200 attempts to move in a random direction were made by both the project members. The results were tabulated as shown.

NAME	ATTEMPTS	SUCCESSFUL ATTTEMPTS	ACCURACY
AJISHAN	20	17	85%
JOYDEEP	10	8	80%

Figure 11: Accuracy Results

Better lighting can improve the accuracy by providing brighter snapshots to process. The initial pre-processing contrast stretches the image around the mean, which helps in improving the accuracy by making the detection more accurate. Successful attempts were counted as all those attempts which resulted in movement of the wheel chair in the desired direction. For the system to be accurate, each time a system is configured for a person, alter the height and length of the specified eye so that the system recognizes the eye of the person with high precision.

6.4 Safety Features

The safety features incorporated in the system were –

a) Controlled speed of detection and wheel chair drive.

b) Eye height and width threshold.

c) Controlled movement in either direction for limited time period.

d) In case of mishap message sent to authorized personnel.

e) Location of the patient is also sent to the authorised personnel.

f) Stretcher mode in case of mishap.

7. Conclusions

7.1 Performance

The system functions with an accuracy rate of 70-90 % which was above our expectations. The image capture, eye movement detection and the algorithm for validating movement attempts perform very reliably as our results suggest.

The aim of this project is to contribute to the society in our small way by setting out an idea for a system which could actually better the lives of millions of people across the globe. We believe we have done great justice to the idea, and ended up getting more than satisfying results.

7.2 Future Modifications

Though our prototype performs satisfactorily, but a lot of work needs to be done before making the product commercially viable. Some sort of sequence of events should trigger start of detection, because we do not want the wheel chair to move when the person is just casually glaring in different directions. Similarly, we can incorporate certain sequence for turning ON and OFF electrical devices, or door locks.

Also since the criticality of the application is so high, lot of safety precautions need to be incorporated. It needs to be made sure that the system is not fatal to the health of the person. A lot of testing needs to be done before making such a product a reality.

Appendix A - User Manual

STEP 1: Install the webcam that is mounted on the helmet.

STEP 2: Connect the hardware circuit (motors connected to the controller) to the 5 volt supply and also power the micro controller.

STEP 3: Wear the helmet on the head that will have the camera attached to it.

STEP 4: Program the microcontroller with the code given and also run the MATLAB script. This will initiate the serial communication between the MATLAB script and the microcontroller.

STEP 5: Make the head movements that will be tracked which thereby will control the motors to move in the expected direction.

Appendix B - MATLAB script

```
clear all
clf('reset');
webcamlist;

cam=webcam('HP Wide Vision HD');   %create webcam object

right=imread('RIGHT.jpg');
left=imread('LEFT.jpg');
noface=imread('no_face.jpg');
straight=imread('STRAIGHT.jpg');
back=imread('BACK.jpg');

detector = vision.CascadeObjectDetector(); % Create a detector for face using Viola-Jones
detector1 = vision.CascadeObjectDetector('EyePairSmall'); %create detector for eyepair

while true % Infinite loop to continuously detect the face

   vid=snapshot(cam);  %get a snapshot of webcam
   vid = rgb2gray(vid);    %convert to grayscale
   img = flip(vid, 2); % Flips the image horizontally

    bbox = step(detector, img); % Creating bounding box using detector

    if ~ isempty(bbox)  %if face exists
```

```
biggest_box=1;
for i=1:rank(bbox) %find the biggest face
  if bbox(i,3)>bbox(biggest_box,3)
    biggest_box=i;
  end
end
faceImage = imcrop(img,bbox(biggest_box,:)); % extract the face from the image
bboxeyes = step(detector1, faceImage); % locations of the eyepair using detector

subplot(2,2,1),subimage(img); hold on; % Displays full image
for i=1:size(bbox,1)    %draw all the regions that contain face
  rectangle('position', bbox(i, :), 'lineWidth', 2, 'edgeColor', 'y');
end

subplot(2,2,3),subimage(faceImage);     %display face image

if ~ isempty(bboxeyes)  %check it eyepair is available

  biggest_box_eyes=1;
  for i=1:rank(bboxeyes) %find the biggest eyepair
    if bboxeyes(i,3)>bboxeyes(biggest_box_eyes,3)
      biggest_box_eyes=i;
    end
  end

bboxeyeshalf=[bboxeyes(biggest_box_eyes,1),bboxeyes(biggest_box_eyes,2),bboxeyes(biggest_box_eyes,3)/3,bbox
eyes(biggest_box_eyes,4)];   %resize the eyepair width in half

  eyesImage = imcrop(faceImage,bboxeyeshalf(1,:));    %extract the half eyepair from the face image
  eyesImage = imadjust(eyesImage);    %adjust contrast

  r = bboxeyeshalf(1,4)/4;
```

```
[centers, radii, metric] = imfindcircles(eyesImage, [floor(r-r/4) floor(r+r/2)], 'ObjectPolarity','dark', 'Sensitivity', 0.93); % Hough Transform

[M,I] = sort(radii, 'descend');


eyesPositions = centers;


subplot(2,2,2),subimage(eyesImage); hold on;


viscircles(centers, radii,'EdgeColor','b');


if ~isempty(centers)
  pupil_x=centers(1);
  pupil_y=centers(2);
  disL=abs(0-pupil_x);    %distance from left edge to center point
  disR=abs(bboxeyes(1,3)/3-pupil_x);%distance from right edge to center point
   disU=abs(0-pupil_y);    %distance from up to center point
  disD=abs(bboxeyes(1,2)/3-pupil_y);%distance from down edge to center point
  disp(disU)
  %disp(disD)
  subplot(2,2,4);
  if disL>disR+16
     subimage(right);
  else if disR>disL
     subimage(left);
     end
  end
   if disU<=14
        subimage(straight);
    else if disU>=14
        subimage(back);
       end
```

```
                pause(1)
            end

        end
    end
  else
    subplot(2,2,4);
    subimage(noface);
  end
  set(gca,'XtickLabel',[],'YtickLabel',[]);

 hold off;
end
```

Appendix C – Voice Module script

```
#include <SoftwareSerial.h>
#include "VoiceRecognitionV3.h"
#include <LiquidCrystal.h>
char r;
#include <Servo.h>
int i =0;
Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

int flag = 0;
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(A0, 8, 9, 10, 11, 12);
int sense = 0;
VR myVR(2,3);    // 2:RX 3:TX, you can choose your favourite pins.

uint8_t records[7]; // save record
uint8_t buf[64];



#define sleep    (0)
#define forward   (1)
#define back    (2)
#define left   (3)
```

```
#define right    (4)


/**
  @brief   Print signature, if the character is invisible,
           print hexible value instead.
  @param   buf     --> command length
           len     --> number of parameters
*/
void printSignature(uint8_t *buf, int len)
{
  int i;
  for(i=0; i<len; i++){
    if(buf[i]>0x19 && buf[i]<0x7F){
      Serial.write(buf[i]);
    }
    else{
      Serial.print("[");
      Serial.print(buf[i], HEX);
      Serial.print("]");
    }
  }
}


/**
  @brief   Print signature, if the character is invisible,
           print hexible value instead.
  @param   buf  -->  VR module return value when voice is recognized.
             buf[0]  -->  Group mode(FF: None Group, 0x8n: User, 0x0n:System
             buf[1]  -->  number of record which is recognized.
             buf[2]  -->  Recognizer index(position) value of the recognized record.
             buf[3]  -->  Signature length
             buf[4]~buf[n] --> Signature
```

```
*/
void printVR(uint8_t *buf)
{
  Serial.println("VR Index\tGroup\tRecordNum\tSignature");

  Serial.print(buf[2], DEC);
  Serial.print("\t\t");

  if(buf[0] == 0xFF){
    Serial.print("NONE");
  }
  else if(buf[0]&0x80){
    Serial.print("UG ");
    Serial.print(buf[0]&(~0x80), DEC);
  }
  else{
    Serial.print("SG ");
    Serial.print(buf[0], DEC);
  }
  Serial.print("\t");

  Serial.print(buf[1], DEC);
  Serial.print("\t\t");
  if(buf[3]>0){
    printSignature(buf+4, buf[3]);
  }
  else{
    Serial.print("NONE");
  }
  Serial.println("\r\n");
}

void setup()
```

```
{
  /** initialize */
  myVR.begin(9600);
   lcd.begin(16, 2);
    lcd.setCursor(0,0);
   lcd.print("Voice Controlled");
      lcd.setCursor(0,1);
   lcd.print("Wheel Chair, By:");
  // delay(5000);
   lcd.clear();
    lcd.setCursor(0,1);
   lcd.print(" ");
   lcd.setCursor(0,2);
   lcd.print(" ");
  // delay(5000);
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Mode:");
      lcd.setCursor(0,1);
   lcd.print(" ");

  Serial.begin(115200);
  Serial.println("Elechouse Voice Recognition V3 Module\r\n");
 myservo.attach(A2);
 myservo.write(0);
  pinMode(A1, INPUT);
   digitalWrite(A1, HIGH);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
     digitalWrite(4, LOW);
      digitalWrite(5, LOW);
```

```
      digitalWrite(6, LOW);
       digitalWrite(7, LOW);
  if(myVR.clear() == 0){
    Serial.println("Recognizer cleared.");
  }else{
    Serial.println("Not find VoiceRecognitionModule.");
    Serial.println("Please check connection and restart Arduino.");
    while(1);
  }

  if(myVR.load((uint8_t)sleep) >= 0){
    Serial.println("sleep loaded");
  }

  if(myVR.load((uint8_t)forward) >= 0){
    Serial.println("forward loaded");
  }
  if(myVR.load((uint8_t)back) >= 0){
    Serial.println("back loaded");
  }
  if(myVR.load((uint8_t)left) >= 0){
    Serial.println("left loaded");
  }
  if(myVR.load((uint8_t)right) >= 0){
    Serial.println("right loaded");
  }
}

void loop()
{
   while(digitalRead(A1)==1)
 {
  if(i==0)
```

```
 {
  i=1;
  Serial.begin(9600);
  delay(500);
 }
 lcd.setCursor(0,0);
 lcd.print("Mode: Eye   ");
 if(Serial.available()>0)
  {
    r=Serial.read();
   Serial.println(r);
    if(r=='F')
    {
       lcd.setCursor(0,1);
 lcd.print("Forward   ");
       digitalWrite(4, HIGH);
    digitalWrite(5, LOW);
     digitalWrite(6, HIGH);
      digitalWrite(7, LOW);
      delay(2000);
      digitalWrite(4, LOW);
    digitalWrite(5, LOW);
     digitalWrite(6, LOW);
      digitalWrite(7, LOW);
    }
     if(r=='B')
    {
      lcd.setCursor(0,1);
 lcd.print("Back     ");
    Serial.println("back");
       digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
     digitalWrite(6, LOW);
```

```
      digitalWrite(7, HIGH);
      delay(2000);
      digitalWrite(4, LOW);
    digitalWrite(5, LOW);
     digitalWrite(6, LOW);
      digitalWrite(7, LOW);
    }

     if(r=='R')
    {
        digitalWrite(4, HIGH);
    digitalWrite(5, LOW);
     digitalWrite(6, LOW);
      digitalWrite(7, HIGH);
            lcd.setCursor(0,1);
  lcd.print("Right     ");
  delay(2000);
      digitalWrite(4, LOW);
    digitalWrite(5, LOW);
     digitalWrite(6, LOW);
      digitalWrite(7, LOW);
    }
     if(r=='L')
    {
      digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
     digitalWrite(6, HIGH);
      digitalWrite(7, LOW);
            lcd.setCursor(0,1);
  lcd.print("Left     ");
  delay(2000);
      digitalWrite(4, LOW);
    digitalWrite(5, LOW);
```

```
      digitalWrite(6, LOW);
       digitalWrite(7, LOW);
     }
   }
}
while(digitalRead(A1)==0)
{
 int ret;

  lcd.setCursor(0,0);
  lcd.print("Mode: Voice ");




 ret = myVR.recognize(buf, 50);
 if(ret>0){
   switch(buf[1]){
     case sleep:
     Serial.println("sleep");
       myservo.write(90);

       lcd.setCursor(0,1);
  lcd.print("Sleep     ");
       break;
     case forward:
      Serial.println("fwd");
      lcd.setCursor(0,1);
  lcd.print("Forward   ");
        digitalWrite(4, HIGH);
     digitalWrite(5, LOW);
      digitalWrite(6, HIGH);
       digitalWrite(7, LOW);
```

```
      delay(2000);
      digitalWrite(4, LOW);
     digitalWrite(5, LOW);
      digitalWrite(6, LOW);
      digitalWrite(7, LOW);

      break;
        case back:


              lcd.setCursor(0,1);
  lcd.print("Back     ");
    Serial.println("back");
      digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
     digitalWrite(6, LOW);
      digitalWrite(7, HIGH);
      delay(2000);
      digitalWrite(4, LOW);
    digitalWrite(5, LOW);
     digitalWrite(6, LOW);
      digitalWrite(7, LOW);
      break;
        case left:
         Serial.println("left");
     digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
     digitalWrite(6, HIGH);
      digitalWrite(7, LOW);
            lcd.setCursor(0,1);
  lcd.print("Left     ");
  delay(2000);
      digitalWrite(4, LOW);
```

```
    digitalWrite(5, LOW);
     digitalWrite(6, LOW);
      digitalWrite(7, LOW);
      break;
       case right:
Serial.println("right");
      digitalWrite(4, HIGH);
    digitalWrite(5, LOW);
     digitalWrite(6, LOW);
      digitalWrite(7, HIGH);
            lcd.setCursor(0,1);
  lcd.print("Right     ");
  delay(2000);
      digitalWrite(4, LOW);
    digitalWrite(5, LOW);
     digitalWrite(6, LOW);
      digitalWrite(7, LOW);
      break;
    default:
      Serial.println("Record function undefined");
      break;
   }
   /** voice recognized */
  // printVR(buf);
 }
}
}
```

Appendix D – GPS & GSM Module script

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

static const int RXPin = 2, TXPin = 3;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);



char inchar; // Will hold the incoming character from the GSM shield
```

```
SoftwareSerial SIM900(4, 5);
 int timesTosend=1;
int count=0;
int numring=0;
int comring=3;
int onoff=0; // 0 = off, 1 = on
 int flag = 0;
int sense=0;

void setup()
{
  Serial.begin(9600);



  Serial.println("GPS Vehicle Tracking System");
  Serial.println();

  SIM900.begin(9600);

   ss.begin(GPSBaud);
  delay(100);

  analogReference(DEFAULT);
  pinMode(A0, INPUT);
}

void loop()
{
  // This sketch displays information every time a new sentence is correctly encoded.
  Serial.println(analogRead(A0));
if(analogRead(A0) <= 320 || analogRead(A1) >= 400)
{
  Serial.println(" fall");
```

```
}

  while (ss.available() > 0)
    if (gps.encode(ss.read()))
    {
      displayInfo();

if(analogRead(A0) <= 320 || analogRead(A1) >= 400)
{
  Serial.println(" fall");
}

    }

  if (millis() > 5000 || gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    //while(true);
  }

}

void displayInfo()
{
   Serial.print(F("Location: "));
  if (gps.location.isValid())
  {
  if((analogRead(A0) <= 320 || analogRead(A1) >= 400) && flag==0)
{
  flag=1;
        SIM900.begin(9600);
     count = 0;
      SIM900.println("AT+CMGF=1"); //set GSM to text mode
```

```
  delay(200);
  while(count<timesTosend){
delay(1500);
 SIM900.print("AT+CMGS=\"");
 SIM900.print("9599875718");
 SIM900.println("\"");
while ( SIM900.read()!='>');
{
  SIM900.println("Fall Detected");
   SIM900.println("Location:");
    SIM900.println("Lat -");
 SIM900.println(gps.location.lat());  //SMS body
delay(500);
SIM900.println();
SIM900.println("Lng -");
SIM900.println(gps.location.lng());  //SMS body
delay(500);
 SIM900.write(0x1A);  // sends ctrl+z end of message
    SIM900.write(0x0D);  // Carriage Return in Hex
 SIM900.write(0x0A);  // Line feed in Hex
//The 0D0A pair of characters is the signal for the end of a line and beginning of another.
delay(5000);
}
count++;
}
  }
  ss.begin(GPSBaud);
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.println(gps.location.lng(), 6);
  }
  else
  {
```

```
    Serial.println(F("INVALID"));
  }
}
```

References

https://www.instructables.com/id/Eye-controlled-wheelchair/

https://www.seminarsonly.com/electronics/eye-directive-wheelchair.php

http://ijsetr.org/wp-content/uploads/2016/01/IJSETR-VOL-5-ISSUE-1-364-368.pdf