# EYE POSITION BASED WHEEL CHAIR CONTROL

**A PROJECT REPORT**
***Submitted by***


**ABIRAMI.M** **Reg. No.:13BEC004**
**AKSHAYA.S** **Reg. No.:13BEC010**
**GAYATHRI.A** **Reg. No.:13BEC043**
**JEEVA.B** **Reg. No.:13BEC057**


***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**ELECTRONICS AND COMMUNICATION**

**ENGINEERING**


**KUMARAGURU COLLEGE OF TECHNOLOGY**

**COIMBATORE-641049**

**(An Autonomous Institution Affiliated to Anna University, Chennai)**


**APRIL 2017**

# KUMARAGURU COLLEGE OF TECHNOLOGY

# COIMBATORE – 641049

## BONAFIDE CERTIFICATE

Certified that this project report titled **"EYE POSITION BASED WHEELCHAIR CONTROL"** is the bonafide work of "**ABIRAMI.M [13BEC004], AKSHAYA.S [13BEC010], GAYATHRI.A [13BEC043], JEEVA.B [13BEC057]"** 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 part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**
Mr.R. Darwin ,M.E.(Ph.D).,
Assistant Professor/E.C.E
Kumaraguru College of Technology
Coimbatore.

**SIGNATURE**
Dr.K. Malarvizhi ,M.E.Ph.D.,
**HEAD OF THE DEPARTMENT**
Electronics & Communication
Engineering
Kumaraguru College of Technology
Coimbatore.

The candidates with Register numbers 13BEC004, 13BEC010, 13BEC043 and 13BEC057 are examined by us in the project viva-voce examination held on ……

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We express our sincere thanks to the Management of Kumaraguru College of Technology and Joint Correspondent **Shri. ShankarVanavarayar** for the kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks to our beloved Principal **Dr.R.S. Kumar, Ph.D.,** Kumaraguru College of Technology, who encouraged us in each and every step of the project.

We would like to thank **Dr.K. Malarvizhi, M.E.Ph.D.,** Head of the Department, Electronics and Communication Engineering, for her kind support and for providing necessary facilities to carry out the project work.

We wish to thank with everlasting gratitude to our Project Coordinator **Prof.Mr.S. Govindaraju M.E.,** Department of Electronics and Communication Engineering for his consistent support throughout the course of this project work.
We are greatly privileged to express our deep sense of gratitude and heartfelt thanks to our Project Guide **Mr.R.Darwin M.E.(Ph.D).,** Department of Electronics and Communication Engineering for his expert counselling and guidance to make this project to a great deal of success and also we wish to convey our regards to all teaching and non-teaching staff of ECE Department for their help and cooperation.

Finally, we thank our parents and our family members for giving us the moral support and abundant blessings in all of our activities and our dear friends who helped us to endure our difficult times with their unfailing support and warm wishes.

## ABSTRACT

A powered wheel chair is a mobility-aided device for persons with moderate/severe physical disabilities or chronic diseases as well as the elderly. In order to take care for different disabilities, various kinds of interfaces have been developed for powered wheelchair control such as joystick control, head control and sip-puff control. Many people with disabilities do not have the ability to control powered wheel chair using the above mentioned interfaces. The proposed model is a possible alternative. In this project, we have considered mainly the patients with the problem of Quadriplegia. These patients have paralysis below neck region. In this, we use the optical-type eye tracking system to control powered wheel chair. User‘s eye movement are given as input in the form of a image to MATLAB software. When user looks at appropriate angle, software will provide command based on the angle of position of pupil i.e., when user moves his eyes balls, left (move left), right (move right), up (move forward) and down (stop or reverse). In all other cases wheel chair will proceed straight. Once the image has been processed it moves onto the second part, Arduino. These will take the output from the laptop via a Bluetooth device and convert the signal into command signals that will be sent to the wheelchair motor circuit for movement. This project mainly focuses on wheel chair that process based on the movement of the eye. Adding to this, the wheel chair can also be used to people who has finger movements by using the touch control via mobile.

## TABLE OF CONTENTS

## LIST OF FIGURES

# LIST OF TABLES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| **ALS** | **A**myotrophic **L**ateral **S**clerosis |
| **IDE** | **I**ntegrated **D**evelopment **E**nvironment |
| **DC** | **D**irect **C**urrent |
| **USB** | **U**niversal **S**erial **B**us |
| **MATLAB** | **MAT**rix **LAB**oratory |

# 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 locomotors organ such as arm, leg and other parts. 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.This project is mainly focused on the Quadriplegia patients. It is a paralysis caused by illness or injury to the human that results in the partial or total loss of their limbs and torso. So basically they cannot move any part of their body except their neck and head. So they would have to depend on other people for their travel.

In order to avoid that situations, we have built a prototype for the wheel chair based on the eye movement. We have also built the model which can be accustomed for the movement controlled by touch using the touch screen mobile phones.These are useful for the patients who can enable finger movements.


**Figure 1.1: Stephen Hawking**

## 1.1 PROBLEM STATEMENT:

Thus, we can summarize our project as follows: the main of this project is to design a vision based wheelchair system. Using the camera to acquire user images and analysing user intent using head and eye gestures.

## 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.

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 Arduino circuitry driving the motors attached to the wheel chair.

**Figure 1.2: Real life model of eye position based wheel chair**

## 2. FLOW CHART:

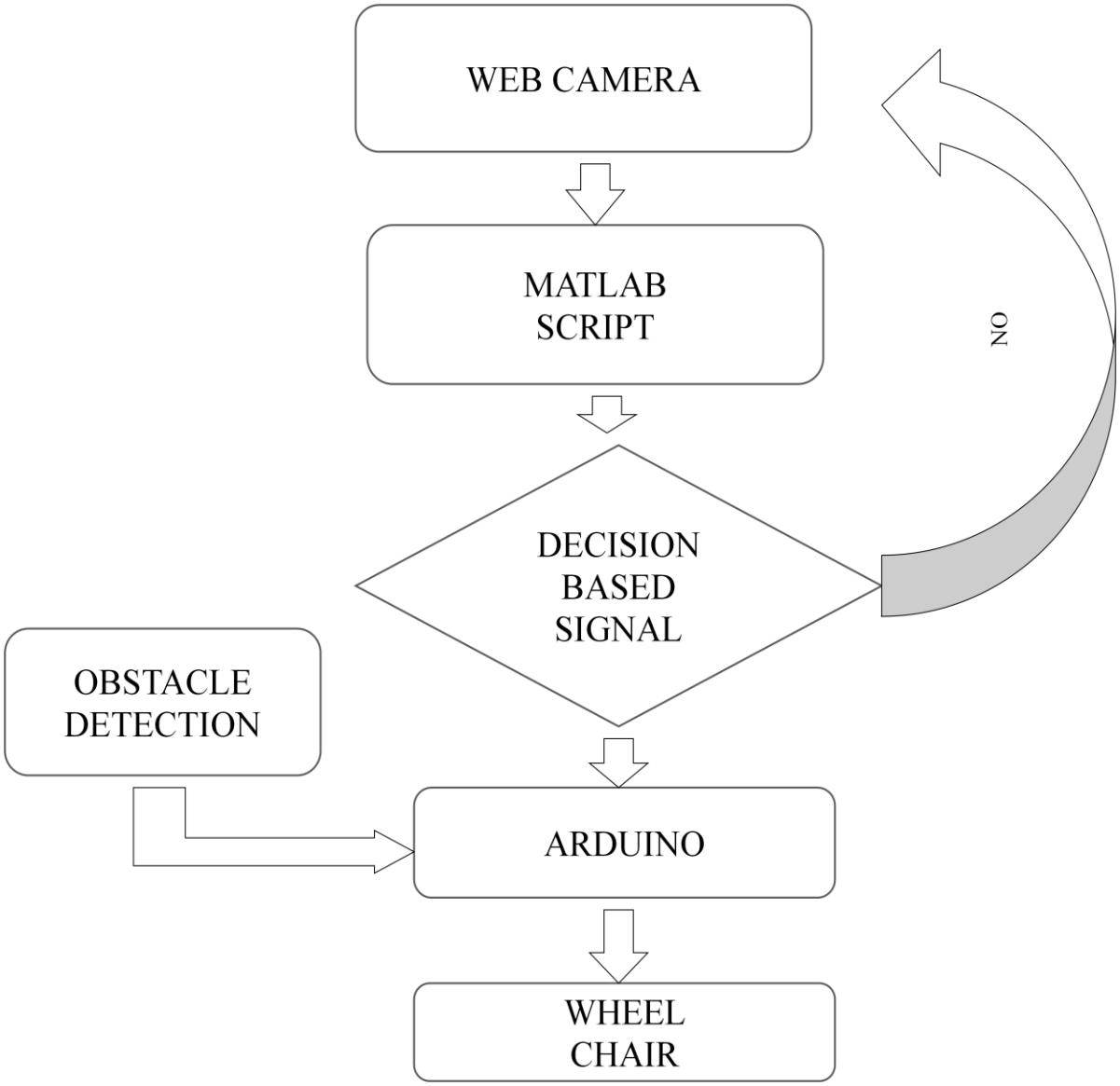



**Figure 2.1: Flowchart of the operation**

# 3. HARDWARE COMPONENTS:

## 3.1 ARDUINO:

Arduino is an open source, computer hardware and software company, project, and user community that designs and manufactures microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. Arduino board designs use a variety of microprocessors and controllers. Arduino is the major functional hardware of this project. The arduino board used is duemilanove which means two thousand nine in Italian. The main purpose of arduino is running a single program again and again.

### 3.1.1 ARDUINO DUEMILANOVE

### SUMMARY:

| | |
|---|---|
| Microcontroller | ATmega328 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB of which 2 KB used by bootloader |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Clock Speed | 16 MHz |


**Figure 3.1 ARDUINO DUEMILANOVE**

## 3.2 DC MOTOR:

DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances. Larger DC motors are used in propulsion of electric vehicles, elevator and hoists, or in drives for steel rolling mills.

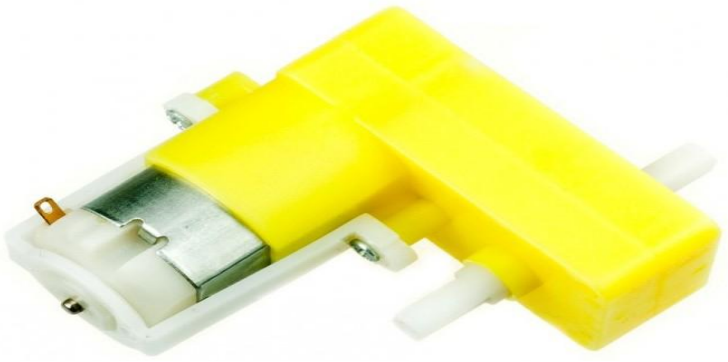


**Figure 3.2: DC Motor**

A simple DC motor has a stationary set of magnets. In the stator and an armature with one or more windings of insulated wire wrapped around a soft iron core that concentrates the magnetic field. The windings usually have multiple turns around the core, and in large motors there can be several parallel current paths. The ends of the wire winding are connected to a commutator. The commutator allows each armature coil to be energized in turn and connects the rotating coils with the external power supply through brushes. The sequence of turning a particular coil on or off dictates what direction the effective electromagnetic fields are pointed. By turning on and off coils in sequence a rotating magnetic field can be created. These rotating magnetic fields interact with the magnetic fields of the magnets in the stationary part of the motor to create a force on the armature which causes it to rotate.

DC motors can operate directly from rechargeable batteries, providing the motive power for the first electric vehicles and today's hybrid cars and electric cars as well as driving a host of cordless tools.

## 3.3 BLUETOOTH MODULE:

The commands which are generated in the simulation are communicated to the arduino using a Bluetooth module HC-05. This Bluetooth module provides wireless connection to between the laptop and the hardware. The ultrasonic sensor sends the distance from the obstacles which are located in front of them. According to the values received, the arduino board avoids the obstacle and moves along the path. The commands are received in digital mode and they are processed and sent to the H-bridge, which operates the motors. The arduino receives power supply from the voltage regulator.

**Figure 3.3: Bluetooth Module**

## 3.4 POWER:

The Arduino Duemilanove can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. The recommended range is 7 to 12 volts. Here we use 12v power supply from the voltage regulator which converts the sum of supplies from the two 9v batteries i.e.(18v) into 12v.

The power pins are as follows:

- VIN: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- 5V: The regulated power supply used to power the microcontroller and other components on the board.

This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

- 3V 3A 3.3 volt supply generated by the on-board FTDI chip. Maximum current draw is 50 mA.

- GND. Ground pins.

### 3.4.1 POWER SUPPLY:

Here we use either a constant 18v DC power supply or two 9v batteries which is sufficient for operating the hardware.


**Figure 3.4: 9v Battery**

## 3.5 VOLTAGE REGULATOR:

A voltage regulator is designed to automatically maintain a constant voltage level. A voltage regulator may be a simple feed-forward design or may include negative feedback control loops. It may use an electromechanical mechanism, or electronic components. Depending on the design, it may be used to regulate one or more AC or DC voltages. We use 7812 voltage regulator to convert 18v power supply from two batteries into 12v. This 12v is supplied to arduino board.


**Figure 3.5: 7812C Regulator**

## 3.6 ULTRASONIC SENSOR:

The ultrasonic sensor used is HC-SR04 which is widely used for detecting the objects and measuring the distance from the object. It is mainly based on transceiver principle. It has two parts namely transmitter and receiver. The transmitter converts the AC into ultrasound and transmits the signal. The signal is reflected back by the object and it is received by the receiver. The receiver performs reverse operation of converting the ultrasound signal into AC signal. This operation helps us identify the distance of the object from the point of transmission.

**Figure 3.6: Ultrasonic Sensor**

# 4. MECHANICAL DESIGN:

For the mechanical part, we have considered a prototype which is built by using a metallic plate and two wheels. Each wheel is anchored with a motor. For the image processing part, the laptop camera is used for capturing the instantaneous video. Instead of the camera of the laptop we can also use the web camera or we can also use the mobile handsets with a better picture quality. This can be done by connecting the networks of both laptop and the capture device with a same wireless network.

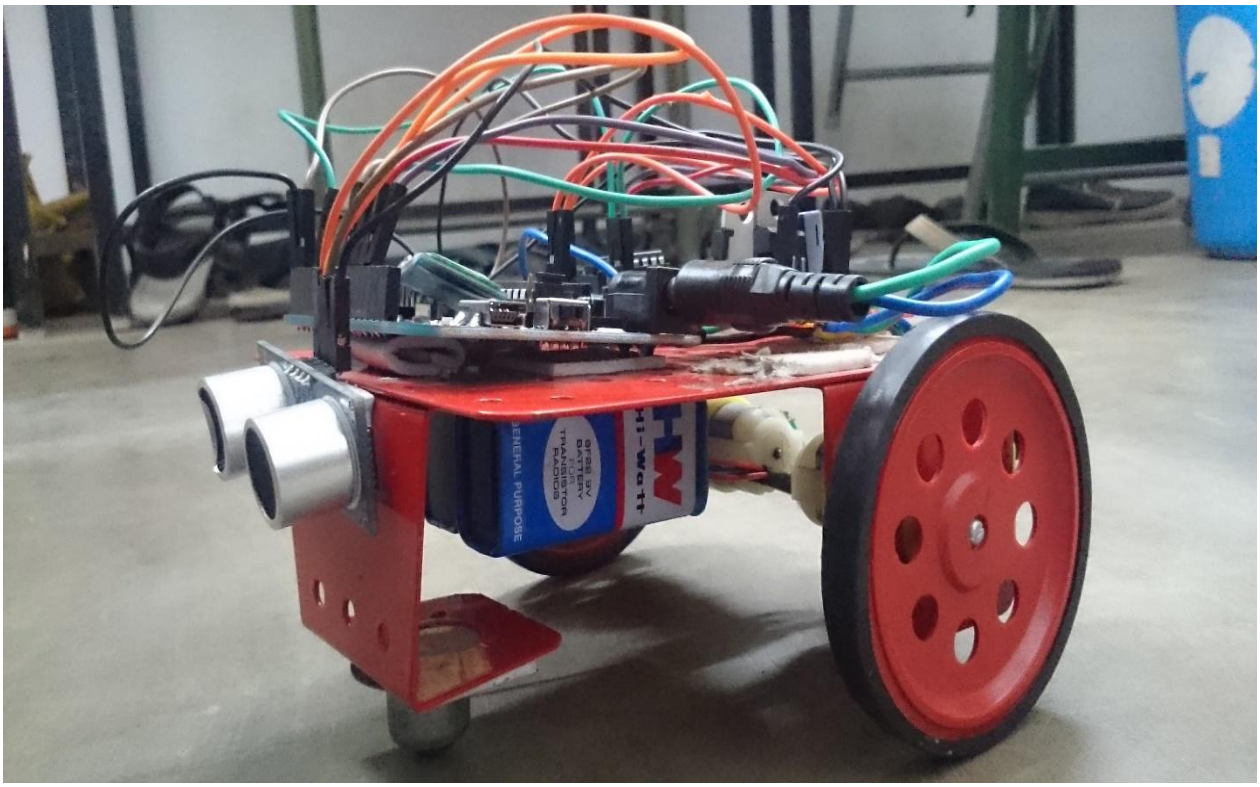



**Figure 4.1: Mechanical Setup of the Prototype**

# 5. SOFTWARE TOOLS:

## 5.1 MATLAB:

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**.**

### 5.1.1 IMAGE CAPTURE AND FACE DETECTION:

MATLAB 2013 equips an Image Processing Toolbox, which we have used majorly in this section of the Software Design.We have used 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 requirement of our design was to continuously look at different frames, based on which determine motion. It is ractically 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.

### 5.1.2 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. A series of 3 such blinks command the motors to freeze, halting the wheel chair.

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.

### 5.1.3 EYE DETECTION:

The movement detection is done with a very basic principle. We take in the feature points for both left and right eyes and save it. Thereafter take the difference in pixels of the left eye position and right eye position in the current snapshot from the previous snapshot. We define the threshold for the minimum movement of the eye required to be qualified as a valid attempt. In each snapshot the difference is evaluated, and if this difference above the threshold in any direction left or right, the flags indicating left movement or right movement are set. If the difference is not above the threshold, the flag which says that no movement has occurred is set.

Sometimes due to non-linearities, both the eyes are not detected. At such instances while evaluating the difference for detecting movement, we would give a bias to the eye which was detected in the previous snapshot. After detecting the eye movements, we can proceed to determining and sending serial signals to the micro-controller.

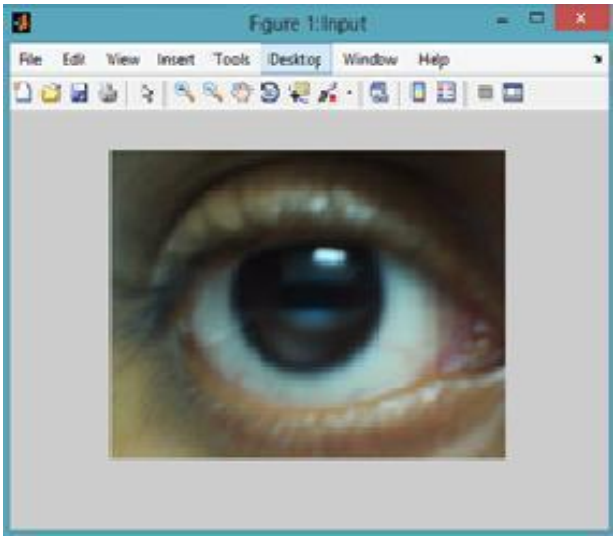



**Figure 5.1: Input for the straight looking eye**

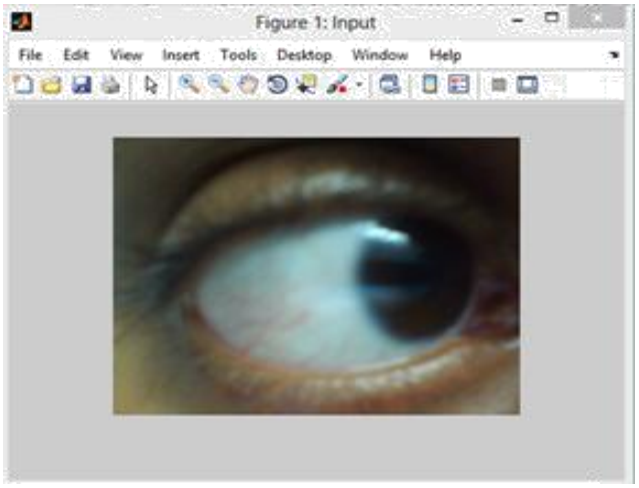



**Figure 5.2: Input for Left looking eye (as per user) and the image looks Right (as per computer)**

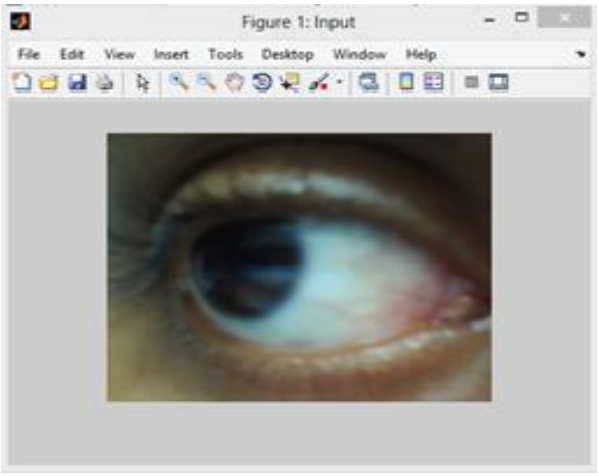



**Figure 5.3: Input for Right looking eye (as per user) and the image looks Left (as per computer)**

## 5.1.4 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 towards the right side and stay there for a second, after which the wheel chair starts moving. But the person's eye is still tilted on the right. If the person now tries to go back to the initial position by moving 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.

### 5.1.5 WHEELCHAIR MOVEMENT:

In this MATLAB output is used to control the motor in desired 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.

## 5.2 ARDUINO SOFTWARE (IDE):

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors.

The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

 **Verify -**

Checks your code for errors compiling it.

 ***Upload -***

Compiles your code and uploads it to the configured board.

 ***New -***

Create a new sketch.

 ***Open -***

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

 ***Save -***

Saves your sketch.

 ***Serial Monitor –***

Opens the serial monitor.

Additional commands are found within the five menus: **File**, **Edit**, **Sketch**, **Tools**, **Help**. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

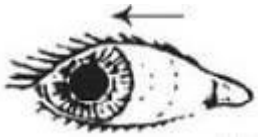
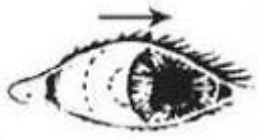
## PROGRAMMING:

The Arduino Duemilanove can be programmed with the Arduino software (download). Select "Arduino Diecimila or Duemilanove w/ ATmega168" or "Arduino Duemilanove w/ ATmega328" from the Tools > Board menu (according to the microcontroller on your board).

The ATmega168 or ATmega328 on the Arduino Duemilanove comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

## 6. COMMAND GENERATION:

| POSITION OF EYE | PRESENT STATE | NEXT STATE |
|---|---|---|
|  | MOTOR OFF | START THE MOTOR |
| | MOTOR ON/SPEED LEVEL 2 | SPEED LEVEL 1 |
| | SPEED LEVEL 1 | SPEED LEVEL 2 |
|  | ANY OTHER STATE | STOP THE MOTOR |
| | MOTOR IN STOP POSITION | REVERSE THE MOTOR |
|  | ANY POSITION EXCEPT STOP | TURN THE MOTOR LEFT |
|  | ANY POSITION EXCEPT STOP | TURN THE MOTOR RIGHT |

| | | |
|---|---|---|
|  | ANY POSITION EXCEPT STOP | STRAIGHT |
|  | NO EYE POSITION TOWARDS CAMERA | NO FACE |

**Table 6.1: Table of command operations**

- For the right side movement of wheelchair right side motor stops and left side motor works .
- For the left side movement of wheelchair left side motor stops and right side motor works.
- For the straight movement of wheelchair left side motor and right side motor works.
- For the reverse movement of wheelchair left side motor and right side motor works.
- When the command is stop both the motors stops working.

## 7. SIMULATION RESULTS:

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 ‘move right’ or ‘move left’ or ‘move straight’ or ‘move forward’ or ‘stop’. Different pupils where used for testing and the best results were gained by pupils directly from the tester, which was not really surprising. Obtaining them is not that simple though. An algorithm from Hugh Transform operator was used, which requires too much calculation time to be used in real-time environments, but is fast enough for getting the first pupils.

The output of the simulation is as follows. The input and the output figures and the commands are given below.

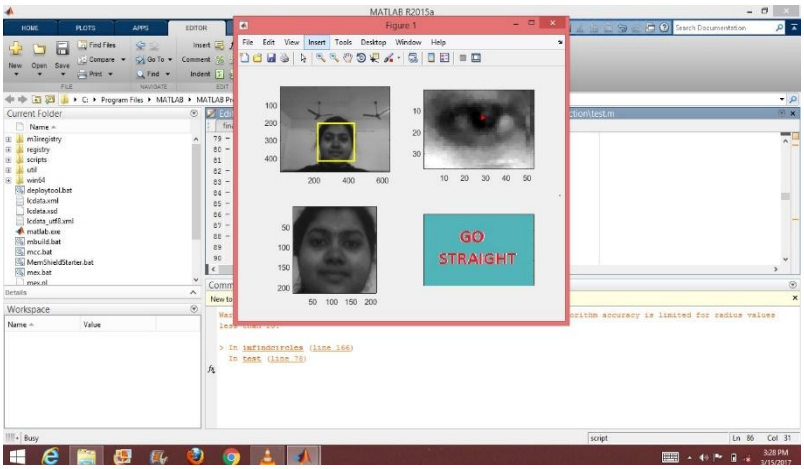



**Figure 7.1**

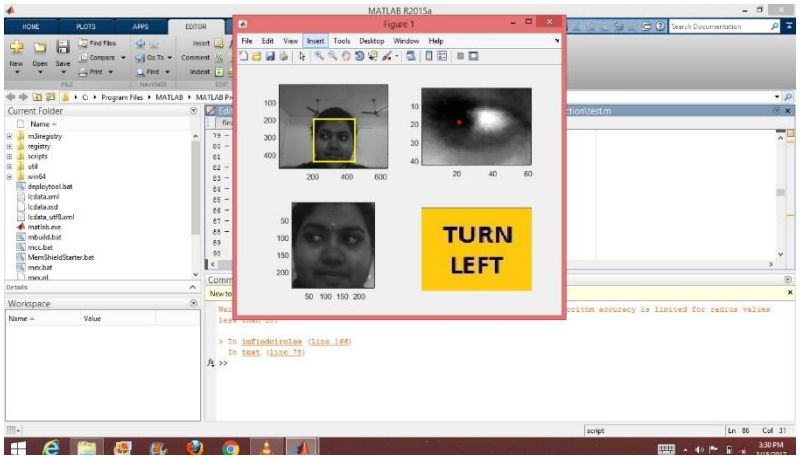



**Figure 7.2**

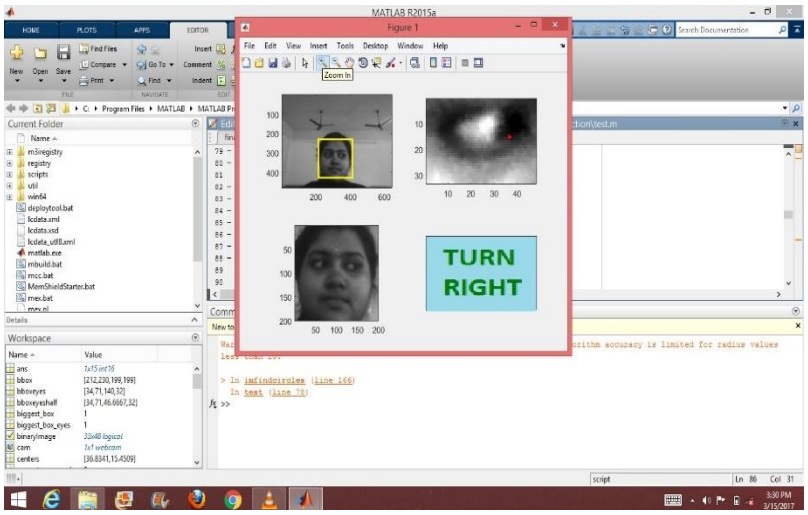



**Figure 7.3**

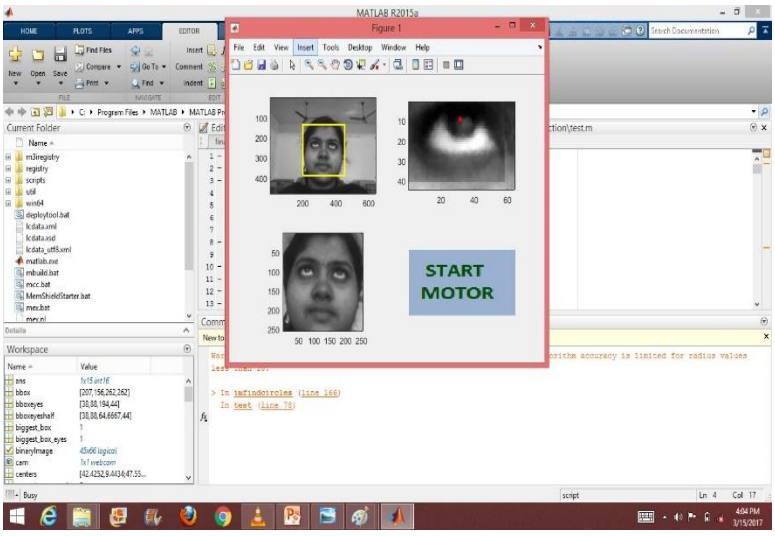



**Figure 7.4**

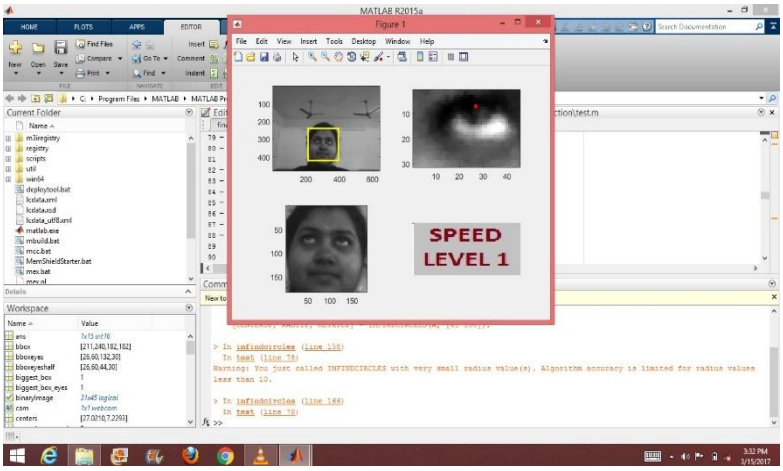



**Figure 7.5**

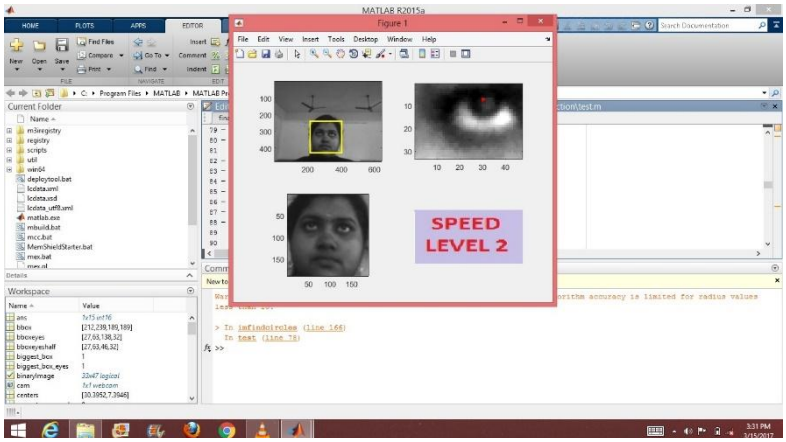



**Figure 7.6**

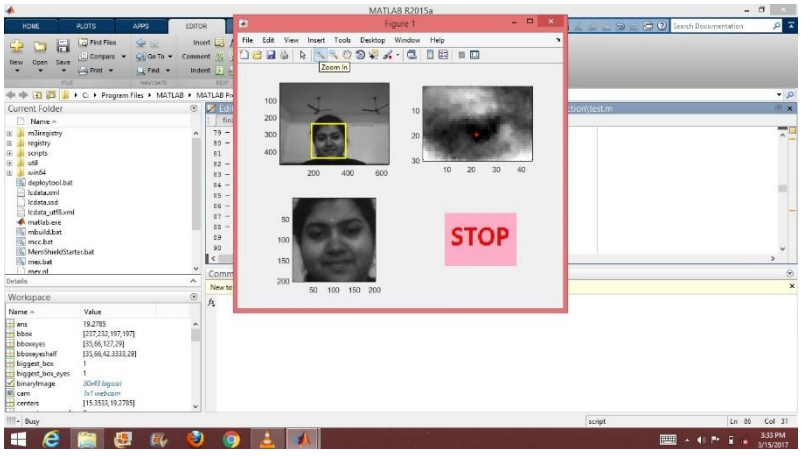



**Figure 7.7**

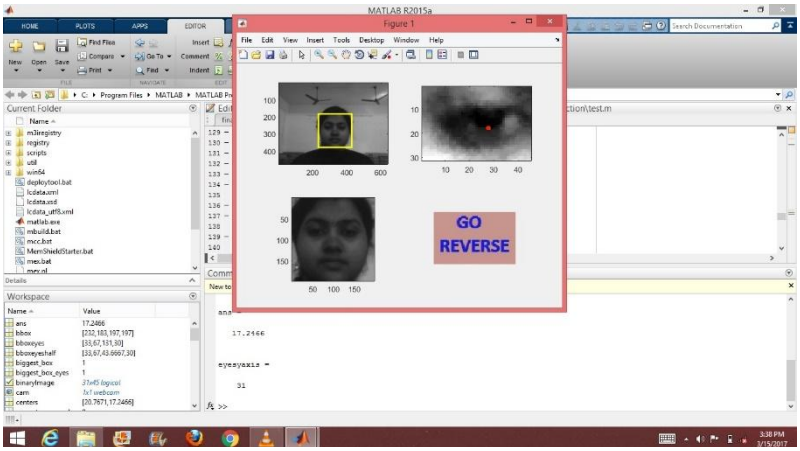



**Figure 7.8**

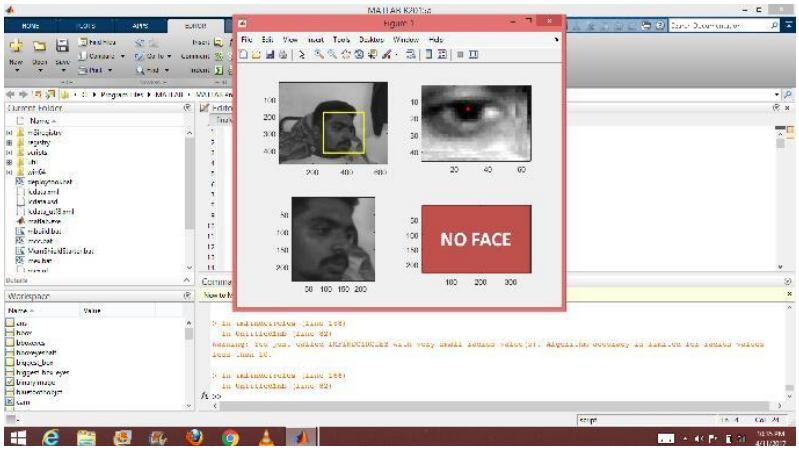



**Figure7 .9**

**Figure 7: Command generated for various eye positions**

## 8.TOUCH CONTROL BASED WHEEL CHAIR:

The wheel chair has also been built based on the touch control so that it can be used by normal patients who do not suffer from Quadriplegia and can be able to move fingers. This is implemented via mobile phones with a touch control or command driven control. It is done by a software, Bluetooth to Arduino in mobile and it has two modes.

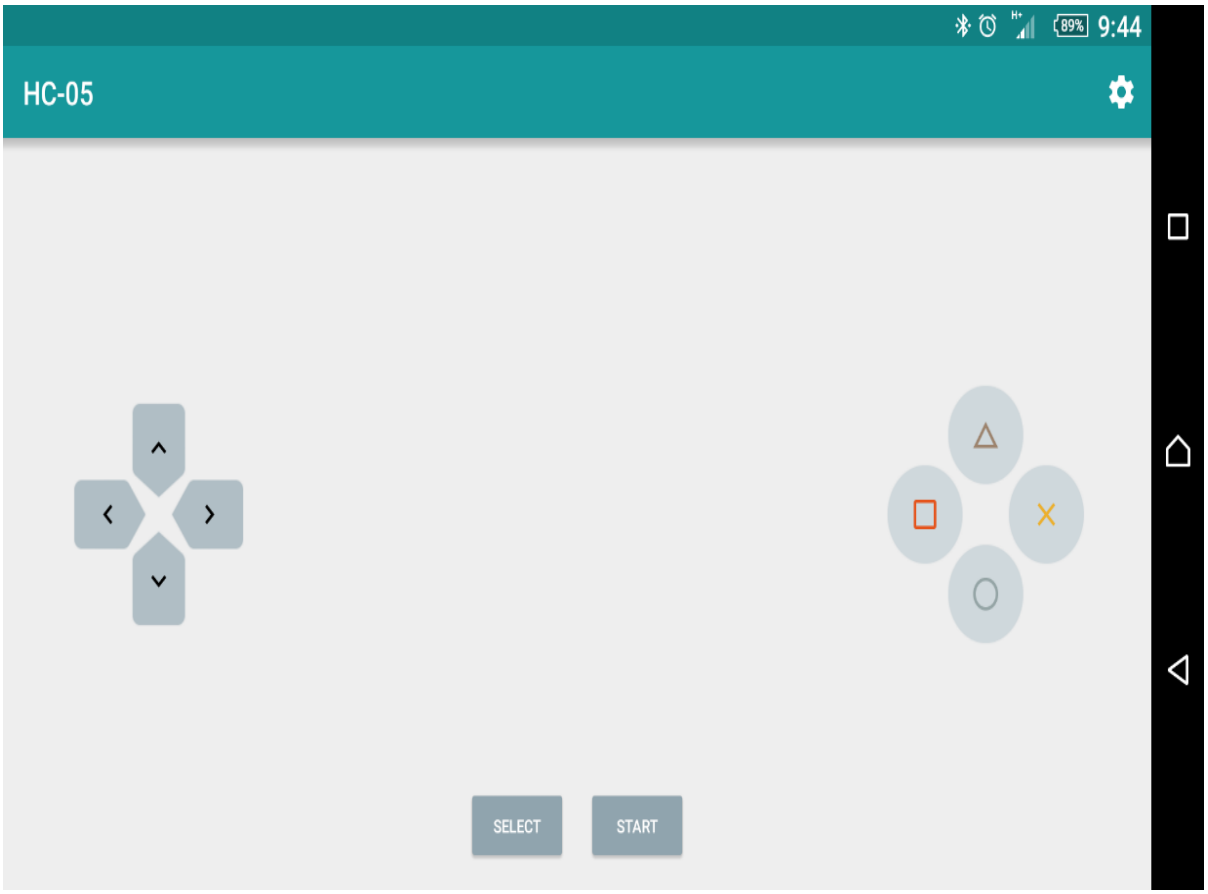



**Figure 8.1: Controller mode**

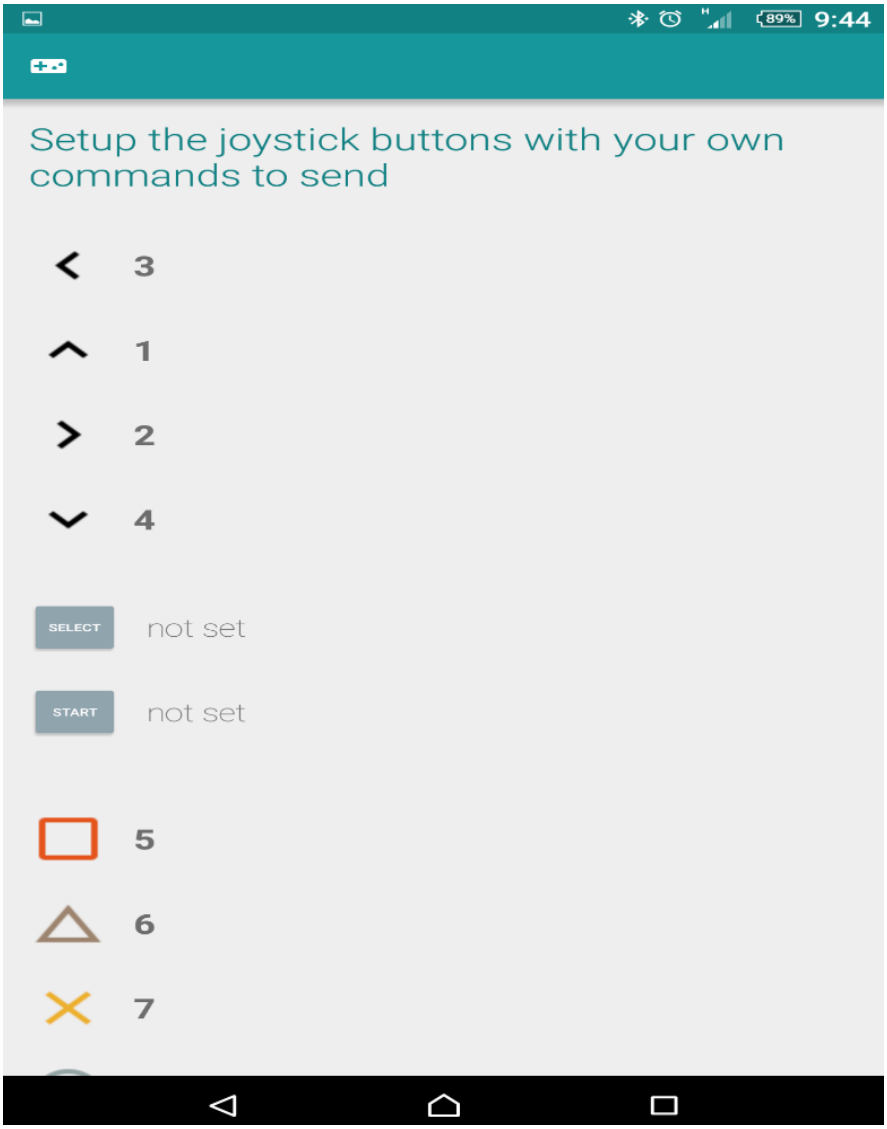



**Figure 8.2: Setting commands in controller mode**

## 8.1 TERMINAL MODE:

This mode provides a terminal like interface in which user can type and send an individual or multiple commands which will execute sequentially.Since ARDUINO serial port receive one byte at a time ,if the user types string in the input field,application divides the string into characters and send them one by one to ARDUINO.

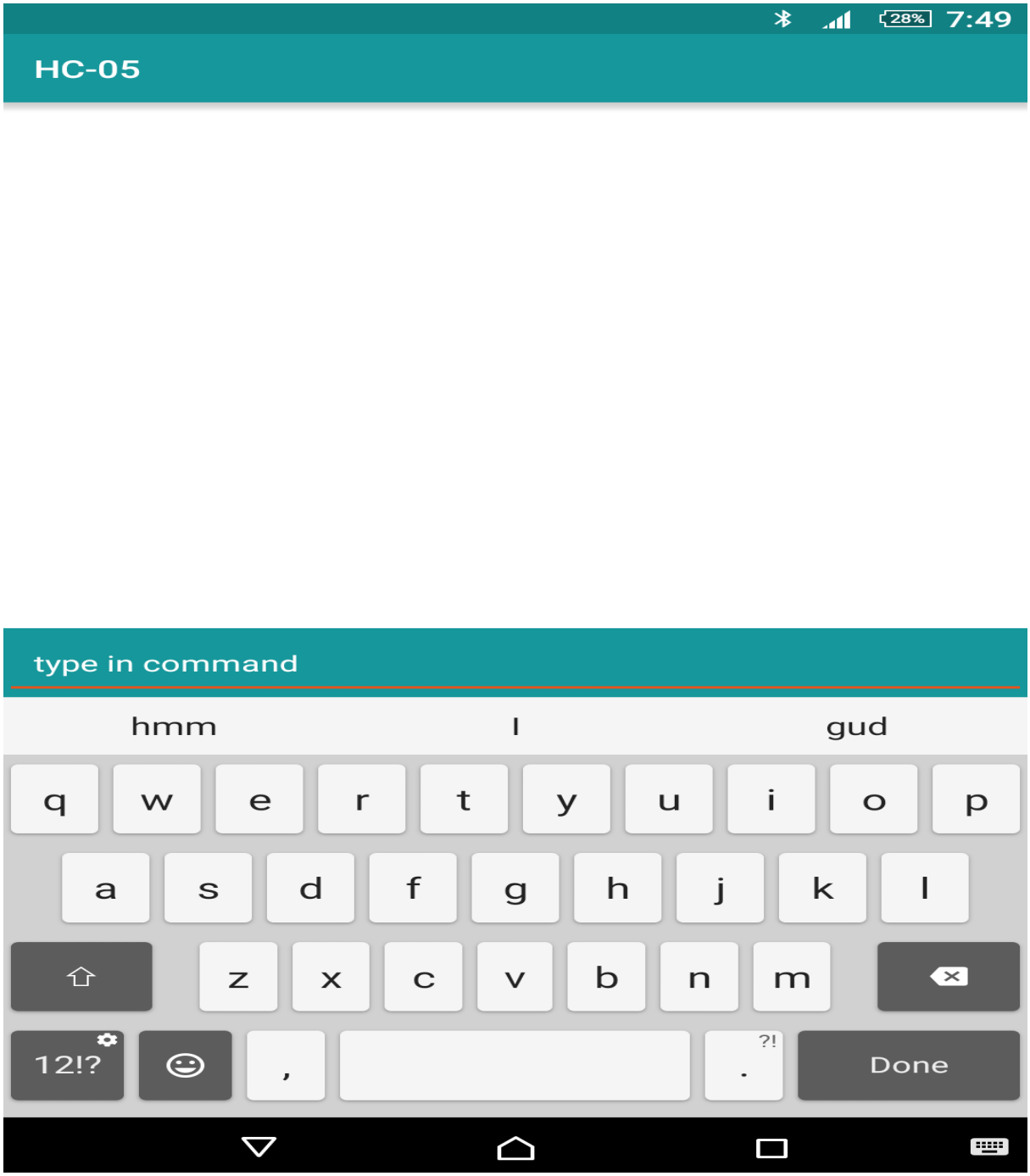



**Figure 8.3: Terminal mode**

# 9.PROGRAM CODE:

## 9.1 MATLAB CODE

```
clear all;
clf('reset');
bluetoothobjct = instrfindall;
delete(bluetoothobjct);
%cam = ipcam('http://192.168.1.7:4747/mjpegfeed?640x480');
%cam2 = ipcam('http://192.168.43.216:4747/mjpegfeed?640x480');
cam=webcam();   %create webcam object
%preview(cam);
up=imread('START MOTOR.jpg');
down=imread('Down.jpg');
reverse=imread('GO REVERSE.jpg');
right=imread('TURN RIGHT.jpg');
stopit=imread('STOP MOTOR.jpg');
left=imread('TURN LEFT.jpg');
engineon=imread('START MOTOR.jpg');
engineoff=imread('STOP MOTOR.jpg');
gear1=imread('SPEED LEVEL 1.jpg');
gear2=imread('SPEED LEVEL 2.jpg');
noface=imread('no_face.jpg');
straight=imread('Go straight.jpg');
int16 tmp;
int16 firstgearonce;
int16 gears;
int16 maxgear;
int16 mingear;
```

```
int16 currentcommand;
int16 previouscommand;
maxgear=0;
tmp=0;
UPP=0;
stop=0;
rrevrse=0;
gears=0;
onengine=0;
mingear=0;
previouscommand=0;
firstgearonce=0;
currentcommand=0;
detector = vision.CascadeObjectDetector();
detector1 = vision.CascadeObjectDetector('EyePairSmall');
b = Bluetooth('HC-05',1)
fopen(b)
set(gcf,'currentchar',' ')
while get(gcf,'currentchar')==' '
vid=snapshot(cam);
vid = rgb2gray(vid);
img = flip(vid, 2);
bbox = step(detector, img);
      if ~ isempty(bbox)
      biggest_box=1;
      faceImage = imcrop(img,bbox(biggest_box,:));
      bboxeyes = step(detector1, faceImage);
      subplot(2,2,1),subimage(img); hold on;
      for i=1:size(bbox,1)
```

```
        rectangle('position', bbox(i, :), 'lineWidth', 2, 'edgeColor', 'y');
        end
subplot(2,2,3),subimage(faceImage);
        if ~ isempty(bboxeyes)
        biggest_box_eyes=1;
bboxeyeshalf=[bboxeyes(biggest_box_eyes,1),bboxeyes(biggest_box_eyes,2),b
boxeyes(biggest_box_eyes,3)/3,bboxeyes(biggest_box_eyes,4)];
        eyesImage = imcrop(faceImage,bboxeyeshalf(1,:));
        eyesImage = imadjust(eyesImage);
        eyesimagesize=size(eyesImage);
        eyesxaxis=eyesimagesize(2);
        eyesyaxis=eyesimagesize(1);
        binaryImage = im2bw(eyesImage);
r = bboxeyeshalf(1,4)/4;
[centers, radii, metric] = imfindcircles(eyesImage, [floor(r-r/2) floor(r+r/2)],
'ObjectPolarity','dark', 'Sensitivity', 0.93); % Hough Transform
        [M,I] = sort(radii, 'descend');
        eyesPositions = centers;
        leftthreshld=.30*eyesxaxis;
        rightthreshold=.65*eyesxaxis
        upperthreshold=.20*eyesyaxis;
        downthreshold=.60*eyesyaxis;
 subplot(2,2,2),subimage(eyesImage);
        if ~isempty(centers)
        hold on;
  plot(centers(1,1),centers(1,2), 'r+', 'MarkerSize', 1, 'LineWidth', 5);
        pupil_x=centers(1);
        disX=abs(0-pupil_x);
  subplot(2,2,4);
```

```
        if centers(1,2)<upperthreshold && UPP<=3
subplot(2,2,4),subimage(up);
        UPP=UPP+1;
        stop=0;
        tmp=3;
        if UPP==1
subplot(2,2,4),subimage(engineon);
        onengine=1;
fprintf('STARTENGINE');
        else if UPP==2
subplot(2,2,4),subimage(gear1);
fprintf('GEAR1');
fprintf(b,'5');
        else
subplot(2,2,4),subimage(gear2);
fprintf('GEAR2');
fprintf(b,'6');
        UPP=1;
        end
        end
        else if centers(1,2)>downthreshold && (onengine~=0)
        tmp=3;
        if stop==0
subplot(2,2,4),subimage(stopit);
fprintf('STOP');
fprintf(b,'7');
        stop=1;
        else
subplot(2,2,4),subimage(reverse);
```

```
 fprintf('REVERSE');
 fprintf(b,'4');
       stop=0;
       end
        else if disX<leftthreshld && (tmp~=3) && (stop~=1) && (onengine~=0)
 subplot(2,2,4),subimage(left);
       tmp=2;
fprintf('LEFT');
fprintf(b,'3');
       else if disX>rightthreshold && (tmp~=3) && (stop~=1) &&
(onengine~=0)
subplot(2,2,4),subimage(right);
       tmp=1;
fprintf('RIGHT');
fprintf(b,'2');
       else if stop~=1 && (onengine~=0)
       tmp=4;
subimage(straight);
fprintf(b,'1');
fprintf('STRAIGHT');
       end
       end
       end
       end
       end
       end
       else
       end
set(gca,'XtickLabel',[],'YtickLabel',[]);
```

```
        hold off;
        end
        else
subplot(2,2,4);
subimage(noface);
        end
        end
```

## 9.2 ARDUINO CODE:

```
int pwm = 12;
int pot = A0;
int t1 = 0;
int t2 = 0;
void setup()
{
        pinMode(pwm, OUTPUT);
        pinMode(pot, INPUT);
}
void loop()
{
         t2= analogRead(pot);
         t1= 1000-t2;
         digitalWrite(pwm, HIGH);
         delayMicroseconds(t1);
         digitalWrite(pwm, LOW);
         delayMicroseconds(t2);
}
const int trigPin = 9;
```

```
const int echoPin = 10;
long duration;
int distance;
int pwm1 = 12;
int pwm2 = 11;
int pwm3 = 4;
int pwm4 = 8;
int t1 = 0;
int t2 = 0;
char data = 0;
void setup()
{
        pinMode(trigPin, OUTPUT);
        pinMode(echoPin, INPUT);
        Serial.begin(9600);
        pinMode(pwm1, OUTPUT);
        pinMode(pwm2, OUTPUT);
        pinMode(pwm3, OUTPUT);
        pinMode(pwm4, OUTPUT);
        pinMode(4, OUTPUT);
        digitalWrite(4, HIGH);
        pinMode(3, OUTPUT);
```

```
        digitalWrite(3, HIGH);
}
void loop()
{
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH);
        distance= duration*0.034/2;
        Serial.print("Distance: ");
        Serial.println(distance);
        if(distance>=30)
{
        if(Serial.available() > 0)
{
        data = Serial.read();
        if(data=='1')   //FWD
{
       Serial.println(data);
       for(int i =0;i<1000;i++)
```

```
 {
        t2= 800;
        t1= 1000-t2;
        digitalWrite(pwm1, HIGH);
        digitalWrite(pwm3, HIGH);
        delayMicroseconds(t1);
        digitalWrite(pwm1, LOW);
        digitalWrite(pwm3, LOW);
        delayMicroseconds(t2);
        digitalWrite(pwm2, LOW);
        digitalWrite(pwm4, LOW);
}}
        if(data=='2')    //RIGHT
{
        for(int i =0;i<1000;i++)
{
        t2= 10;
        t1= 1000-t2;
        digitalWrite(pwm1, HIGH);
        delayMicroseconds(t1);
        digitalWrite(pwm1, LOW);
        delayMicroseconds(t2);
```

```
        digitalWrite(pwm2, LOW);
        digitalWrite(pwm4, LOW);
        digitalWrite(pwm3, LOW);
}}
        if(data=='3')     //LEFT
{
        for(int i =0;i<1000;i++)
{
        t2= 10;
        t1= 1000-t2;
        digitalWrite(pwm3, HIGH);
        delayMicroseconds(t1);
        digitalWrite(pwm3, LOW);
        delayMicroseconds(t2);
        digitalWrite(pwm2, LOW);
        digitalWrite(pwm4, LOW);
        digitalWrite(pwm1, LOW);
}}
        if(data=='4')     //REV
{
        for(int i =0;i<1000;i++)
{
```

```
        t2= 700;
        t1= 1000-t2;
        digitalWrite(pwm2, HIGH);
        digitalWrite(pwm4, HIGH);
        delayMicroseconds(t1);
        digitalWrite(pwm2, LOW);
        digitalWrite(pwm4, LOW);
        delayMicroseconds(t2);
        digitalWrite(pwm1, LOW);
        digitalWrite(pwm3, LOW);
}}
        if(data=='5')     //SPEED
{
        for(int i =0;i<1000;i++)
{
        t2= 400;
        t1= 1000-t2;
        digitalWrite(pwm1, HIGH);
        digitalWrite(pwm3, HIGH);
        delayMicroseconds(t1);
        digitalWrite(pwm1, LOW);
        digitalWrite(pwm3, LOW);
```

```
        delayMicroseconds(t2);
        digitalWrite(pwm2, LOW);
        digitalWrite(pwm4, LOW);
}
}
        if(data=='6')     //HISPEED
{
        for(int i =0;i<1000;i++)
{
        t2= 100;
        t1= 1000-t2;
        digitalWrite(pwm1, HIGH);
        digitalWrite(pwm3, HIGH);
        delayMicroseconds(t1);
        digitalWrite(pwm1, LOW);
        digitalWrite(pwm3, LOW);
        delayMicroseconds(t2);
        digitalWrite(pwm2, LOW);
        digitalWrite(pwm4, LOW);
}
}
        if(data=='7')     //HISPEED
```

```
{
        for(int i =0;i<1000;i++)
{
        t2= 100;
        t1= 1000-t2;
        digitalWrite(pwm1, LOW);
        digitalWrite(pwm3, LOW);
        delayMicroseconds(t1);
        digitalWrite(pwm1, LOW);
        digitalWrite(pwm3, LOW);
        delayMicroseconds(t2);
        digitalWrite(pwm2, LOW);
        digitalWrite(pwm4, LOW);
}}
 }}
        else
{
        for (int i=0;i<1000;i++)
{
        t2= 10;
        t1= 1000-t2;
        digitalWrite(pwm1, HIGH);
```

```
        delayMicroseconds(t1);
        digitalWrite(pwm1, LOW);
        delayMicroseconds(t2);
        digitalWrite(pwm2, LOW);
        digitalWrite(pwm4, LOW);
        digitalWrite(pwm3, LOW);
}}}
```

# 10. CONCLUSION:

The idea of eye controls of great use to not only the future of natural input but more importantly the handicapped and disabled. One of the main goals for Eye Movement controlled wheelchair is to enable completely paralyzed patients to make their life more accessible and to provide them opportunity of independence and movement. This type of wheelchair is controlled by eye movement , the camera capture the image and focus on eye in image and the center position of pupil will be found then the different value of X,Y coordinates will be set for different commands like Right, Left ,Forward ,reverse and stop.Here we have provision to switch between two speed levels.Then signals pass to the motor driver. It controls speed and direction of DC motor. DC motor move Right, Left, Forward ,reverse and stop. The system functions with an accuracy rate of 70-90 % which was above our expectations. Adding to this, the wheel chair can also be used to people who has finger movements by using the touch control via mobile .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.

## 10.1 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. 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.

## 11. REFERENCES:

[1] S. Tameemsultana and N. Kali Saranya, "Implementation of Head and Finger Movement Based Automatic Wheel Chair", *Bonfring International Journal of Power Systems and Integrated Circuits*, vol. 1, Special Issue, pp 48-51, December 2011.

[2]Kohei Arai and Ronny Mardiyanto, "Eyes based electrical Wheelchair control system",*International Journal of Advanced Computer Science and Applications*,vol.2,No.12,2011

[3] Tabasum Shaikh, Naseem Farheen Sayyed, Shaheen Pathan, "Review of Multilevel Controlled Wheelchair", 4th National Conference On Electronic Technologies, pp. 275-279, April 2013.

[4] Motor Controller circuitry from ECE 4760 web page for lab 4 http://people.ece.cornell.edu/land/courses/ece4760/labs/f2013/lab4.html

[5]Voila Jones Algorithm - http://en.wikipedia.org/wiki/Viola%E2%80%93Jones-object-detection-framework

[6] Cascade Object Detector – http://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html