Contents

1. Introduction

1.1 Problem Statement

1.2 Range of Solutions

1.3 Product Design Overview

2. High Level Design

2.1 Eye Detection and Motion Tracking

2.2 AtMega8 Controlled Motor Assembly

3. Software Design

3.1 MATLAB Component

3.2 Firmware Design

4. Hardware Design

4.1 Motor Circuit & PCB Layout

4.2 Component Description

5. Mechanical Design

6. Testing and Results

6.1 Testing Strategy

6.2 Accuracy

7. Conclusions

7.1 Performance

7.2 Future Modifications

8. User Manual

9. MATLAB Code

10. Motor drive Code

11. References

**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 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.

We precisely aim at targeting the movements of the eye and the head. The idea is to create a eye monitored wheelchair system where a camera constantly stares at the person’s eyes and based on the combined movement of eye and head, decide to move the wheelchair in the direction the person desires to move in.

The camera is wired to the person’s laptop which has a MATLAB script running which constantly captures snapshots and processes them. The script based on the processing determines whether the person wants to move in a particular direction, and communicates this information using serial communication to a microcontroller which drives motors of the wheelchair is the desired direction.

**1.1 Problem Statement**

The Aim of this project is to design a vision based wheelchair system. Using the camera to acquire user images and analyzing user intent using head 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.

we 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.

**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, platform and we will attach the web camera with Laptop. 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.

**2. High Level Design**

There are two major components from the system design standpoint –

a) Eye-Detection and motion tracking.

b) ATMega8 controlled Wheel Chair Assembly.

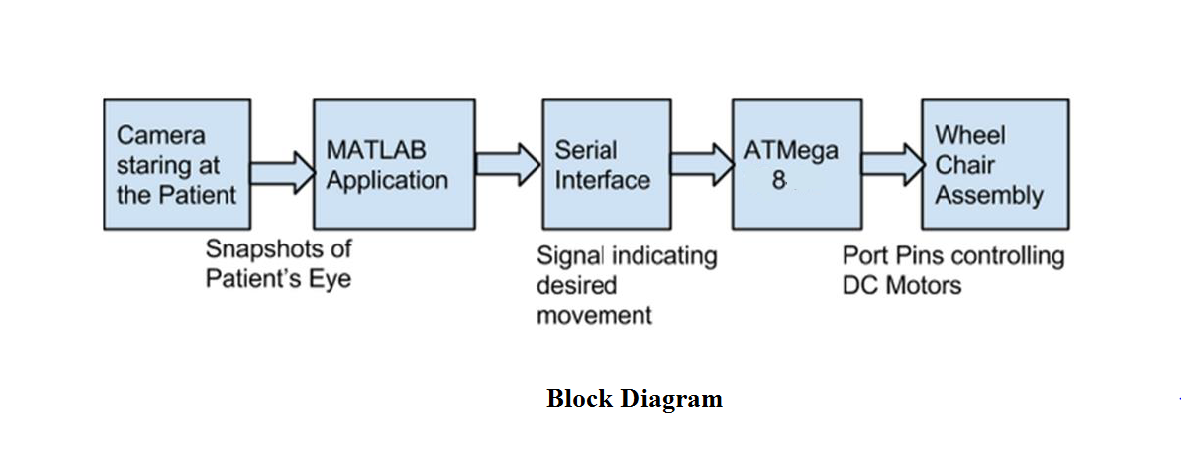
**2.1 Eye-Detection and Motion Tracking**

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 ATMega8 microcontroller. MATLAB 2016 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 are taken and feature points extracted are saved i.e. we capture snapshot every second and process it. Based on the position of center of eyeball , a movement is detected and this is communicated to the wheelchair assembly via the serial port.

**2.2 ATMega8 controlled Wheel Chair Assembly**

A decision based on the processing done by the MATLAB application is communicated and received by the ATMega8. The controller on reception forces the port pin high on which the motors have been connected for desired motion of the Wheel Chair.



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 gray 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. **Estimatio*n***: Now, after working on the each frame we try to detect the eyes. This we do by detector1 object and Eyepair Argumet.

4. **Detectio*n***: Now, in this step we actually detect the eye movements. The idea is to find center of eyeball and find it location in rectangle it is in left part or right part or in the mid part. 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, In which if there are two faces in the frame then it will detect the bigger face present in the frame and crop it.

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.  **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. After that, the patient will turn right to again look forward. Thus, the wheelchair will move in the Left direction.

b**. 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. After that, the patient will turn left to again look forward. Thus, the wheelchair will move in Right direction.

c. **Straigh**t: The signal to go straight is when a person looks straight. This will be detected as to go straight.

7. **Serial Communication*:*** Now according to the detected command, the MATLAB application will transmit L, R or F 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 –

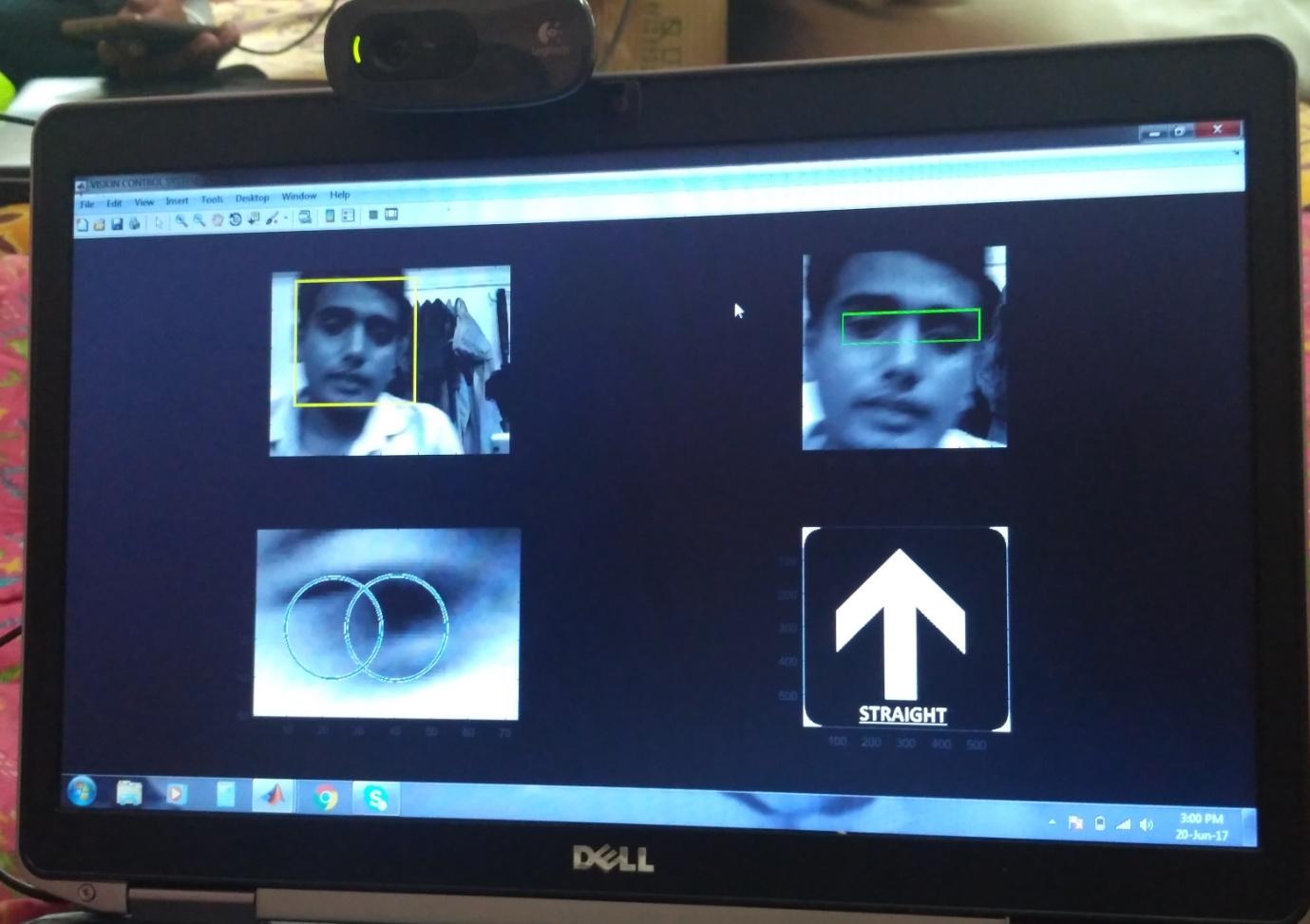


Figure: MATLAB Application

**(i) initialization of variables and setting up serial communication –**

MATLAB 2016 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 4800. 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 AtMega8, we reset the variables needed in the course of the program to their initial valuables.

**(ii) Image Capture and Eye Detection -**

MATLAB 2016 equips an Image Processing Toolbox, which we have used majorly in this section of the Software Design. We use a Logitech HD Webcam C270 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. . So, a snapshot 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[5] 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.

he ‘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 face in the frame have been detected or not. If the face is detected then it is cropped by the image by imcrop() command and further the cropped image is processed and eye-pair is detected in it. The eye-pair is further processed and only left eye is cropped for the eye ball movement detection .

Afterwards, to detect the eye ball or IRIS, we detect black circle in the final cropped image and we extract the center position of the circle and to detect the eye movement we calculate the distance of the circle center form the Right edge of the circle.

We set the thresholds for right and left movement detection .

If the eye ball moves in the right direction the distance from right edge of the bounding box reduces if the reduced distance satisfies the threshold then signal for the right movement is transmitted and if the eye ball moves in the left direction the distance from right edge of the bounding box increases, if the increased distance satisfies the threshold then signal for the left movement is transmitted.

**(iv) Movement Detection –**

To detect the eye ball movement, we first detect the eye ball or IRIS by detecting black circle in the final cropped image and we extract the center position of the circle and to detect the eye movement we calculate the distance of the circle center form the Right edge of the circle.

We set the thresholds for right and left movement detection .

If the eye ball moves in the right direction the distance from right edge of the bounding box reduces if the reduced distance satisfies the threshold then signal for the right movement is transmitted and if the eye ball moves in the left direction the distance from right edge of the bounding box increases, if the increased distance satisfies the threshold then signal for the left movement is transmitted.

After detecting the eye movements, we can proceed to determining and sending serial signals to the micro-controller.

**(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 tilting face 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.

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.

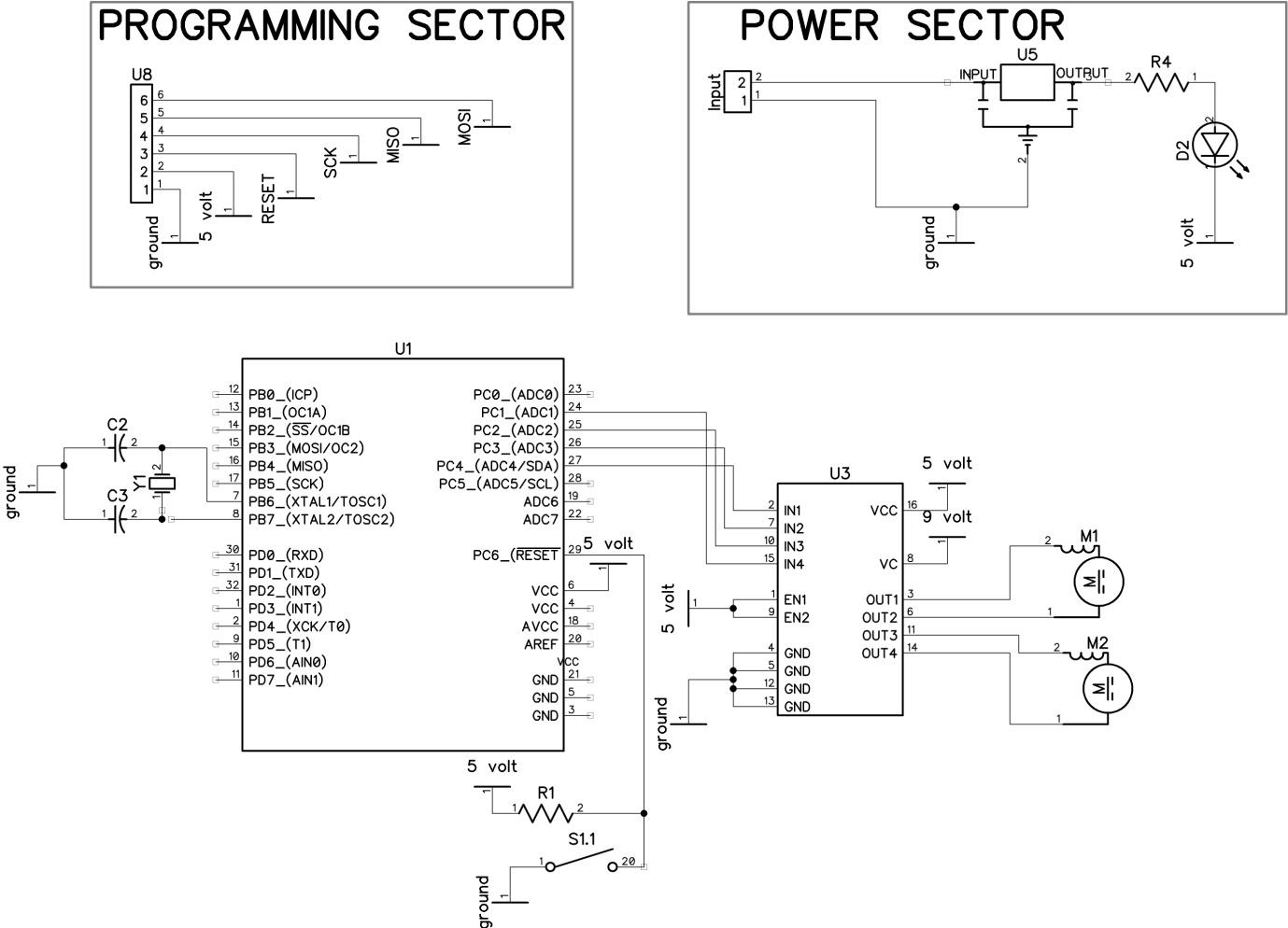
As far as the firmware design is concerned, it is fairly simple. All that is required is take in the serial input, move in a direction, give a delay and keep doing this repeatedly.

**4. Hardware Design**

The hardware part consists a microcontroller atmega8 and a moter driver ic L293D to control the motor. The motor controller circuit is given below.

**4.1 Motor Control**

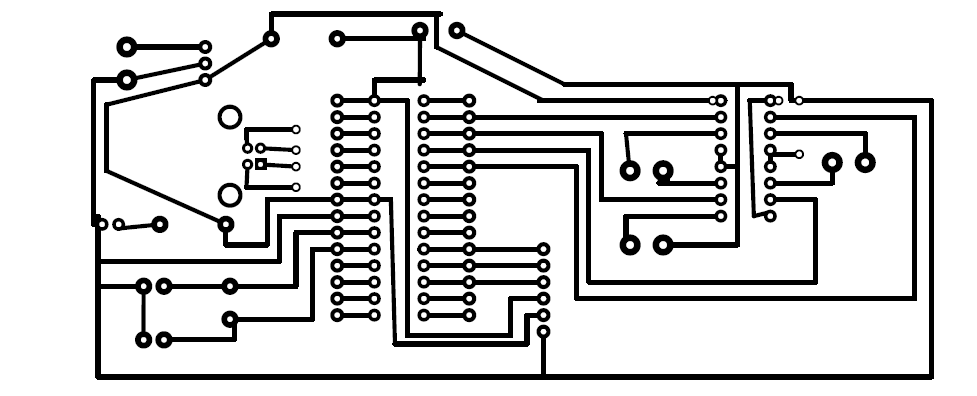
The signal that we obtain from the MATLAB script, we then use that to drive the motors. The MATLAB program does all the decision making for the motor to run in which direction. So, the script sends the bit L,R and F for left, right and straight direction. The circuit shown below is



**Figure 4: Motor control circuit**

fairly safe and used to drive the motors. We are using L293D motor deive IC for control motor.

**4.2 PCB LAYOUT**



**4.3 Component Description**

1. **Logitech C270 hd webcam**

****

| **Basic Requirement** | **HD Requirement** |
| --- | --- |
| **System Requirements** | 1GHz = CPU Minimum  1.6GHz = CPU Recommended  1gb = RAM Minimum  2gb = RAM Recommended | 2.4 GHz Intel Core 2 Duo = CPU Recommended 2 GB RAM |
| **Software Support (at release)** | Logitech Webcam Software 2.0 (LWS) | |
| **OS Support (at release)** | Windows XP, Windows Vista, Windows 7 | |

| Camera Specifications: |  |
| --- | --- |
| **Available Image(s)** | [[Left Side Image]](https://secure.logitech.com/assets/31706/c260-left-side-image.jpg) |
| **Connection Type** | Corded USB |
| **USB Type** | High Speed USB 2.0 |
| **USB VID\_PID** | VID\_046D&PID\_081A |
| **Microphone** | Built-in, Noise Supression |
| **Lens and Sensor Type** | Plastic |
| **Focus Type** | Fixed |
| **Field of View (FOV)** | 60° |
| **Focal Length** | 4.0 mm |
| **Optical Resolution (True)** | 1280 x 960 1.2MP |
| **Image Capture (4:3 SD)** | 320x240, 640x480 1.2 MP, 3.0 MP |
| **Image Capture (16:9 W)** | 360p, 480p, 720p |
| **Video Capture (4:3 SD)** | 320x240, 640x480, 800x600 |
| **Video Capture (16:9 W)** | 360p, 480p, 720p, |
| **Frame Rate (max)** | 30fps @ 640x480 |
| **Video Effects (VFX)** | N/A |
| **Right Light** | Right Light 2 |
| **Buttons** | Other NA |
| **Indicator Lights (LED)** | Activity/Power |
| **Privacy Shade** | No |
| **Clip Size (max)** | 0 to infinity |
| **Cable Length** | 5 Feet or 1.5 Meters |

| Additional Hardware in Package: |  |
| --- | --- |
| **Headset** | Not Included |
| **Desktop Stand** | Not Included |

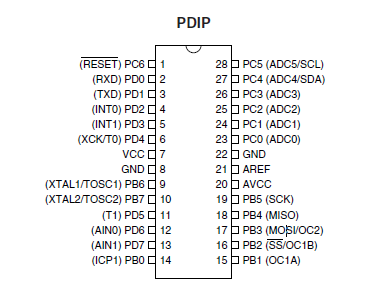
1. **USB to TTL**

****

We do lots of converting from TTL serial to USB (using the FTDI and Silicon Labs ICs). But what do you do if you've got a laptop and need TTL serial? This is a high quality, low cost, small, and easy to use USB to RS232 converter. Device uses the Prolific IC CP2102 and supports baud rates up to 1Mbps. Driver support for Windows, Mac, and Linux RedHat.   
  
This converter plugs directly into your USB port on your computer and even comes with a small extension cable and CD for drivers and serial port Bluetooth. 

1. **ATmega8**

**Pin diagram**

****

The Atmel®AVR® ATmega8 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8 achieves throughputs approaching 1MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

**USART -**The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly-flexible serial communication device. The main features are:

• **Full Duplex Operation (Independent Serial Receive and Transmit Registers)**

• **Asynchronous or Synchronous Operation**

• **Master or Slave Clocked Synchronous Operation**

• **High Resolution Baud Rate Generator**

• **Supports Serial Frames with 5, 6, 7, 8, or 9 Databits and 1 or 2 Stop Bits**

• **Odd or Even Parity Generation and Parity Check Supported by Hardware**

• **Data OverRun Detection**

• **Framing Error Detection**

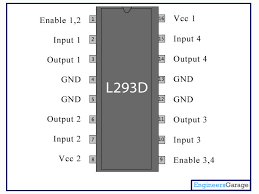
• **Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter**

• **Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete**

• **Multi-processor Communication Mode**

• **Double Speed Asynchronous Communication Mode**

1. **L293D Motor driving IC**

****

**L293D Description**

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC. Dual H-bridge *Motor Driver integrated circuit* (*IC*).

The l293d can drive small and quiet big motors as well, check the Voltage Specification at the end of this page for more info.

Concept:

It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As you know voltage need to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, Hence H-bridge IC are ideal for driving a DC motor.

In a single L293D chip there are two h-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors. Given below is the pin diagram of a L293D motor controller.

There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left H-bridge you need to enable pin 1 to high. And for right H-Bridge you need to make the pin 9 to high.

If anyone of the either pin1 or pin9 goes low then the motor in the corresponding section will suspend working. It’s like a switch.

Working of L293D

There are 4 input pins for l293d, pin 2,7 on the left and pin 15 ,10 on the right as shown on the pin diagram. Left input pins will regulate the rotation of motor connected across left side and right input for motor on the right hand side. The motors are rotated on the basis of the inputs provided across the input pins as LOGIC 0 or LOGIC 1.

In simple you need to provide Logic 0 or 1 across the input pins for rotating the motor.

L293D Logic Table.

Lets consider a Motor connected on left side output pins (pin 3,6). For rotating the motor in clockwise direction the input pins has to be provided with Logic 1 and Logic 0.

• **Pin 2** = **Logic 1** and **Pin 7**= **Logic 0** | Clockwise Direction  
• **Pin 2** = **Logic 0**and **Pin 7**= **Logic 1** | Anticlockwise Direction  
•**Pin 2**= **Logic 0** and **Pin 7** = **Logic 0** | Idle [No rotation] [Hi-Impedance state]  
• **Pin 2**= **Logic 1** and **Pin 7** = **Logic 1** | Idle [No rotation]

In a very similar way the motor can also operate across input pin 15,10 for motor on the right hand side.

**Voltage Specification**

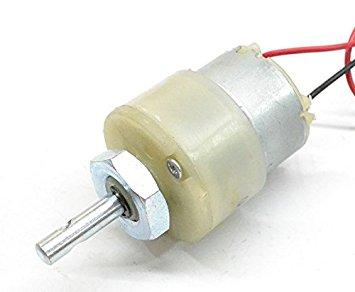
VCC is the voltage that it needs for its own internal operation 5v; L293D will not use this voltage for driving the motor. For driving the motors it has a separate provision to provide motor supply VSS (V supply).  L293d will use this to drive the motor. It means if you want to operate a motor at 9V then you need to provide a Supply of 9V across VSS Motor supply.

The maximum voltage for VSS motor supply is 36V. It can supply a max current of 600mA per channel.Since it can drive motors Up to 36v hence you can drive pretty big motors with this l293d.

VCC pin 16 is the voltage for its own internal Operation. The maximum voltage ranges from 5v and upto 36v.

TIP: Don’t Exceed the Vmax Voltage of 36 volts or it will cause damage.

1. **DC motor 100rpm**

****

100RPM Centre Shaft Economy Series DC Motor is high quality low cost DC geared motor. It has steel gears and pinions to ensure longer life and better wear and tear properties. The gears are fixed on hardened steel spindles polished to a mirror finish. The output shaft rotates in a plastic bushing. The whole assembly is covered with a plastic ring. Gearbox is sealed and lubricated with lithium grease and require no maintenance. The motor is screwed to the gear box from inside.  
Although motor gives 100 RPM at 12V but motor runs smoothly from 4V to 12V and gives wide range of RPM, and torque. Tables below gives fairly good idea of the motor’s performance in terms of RPM and no load current as a function of voltage and stall torque, stall current as a function of voltage.

**Specifications**

* DC supply: 4 to 12V
* RPM: 100 at 12V
* Total length: 46mm
* Motor diameter: 36mm
* Motor length: 25mm
* Brush type: Precious metal
* Gear head diameter: 37mm
* Gear head length: 21mm
* Output shaft: Centred
* Shaft diameter: 6mm
* Shaft length: 22mm
* Gear assembly: Spur
* Motor weight: 100gms

**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.

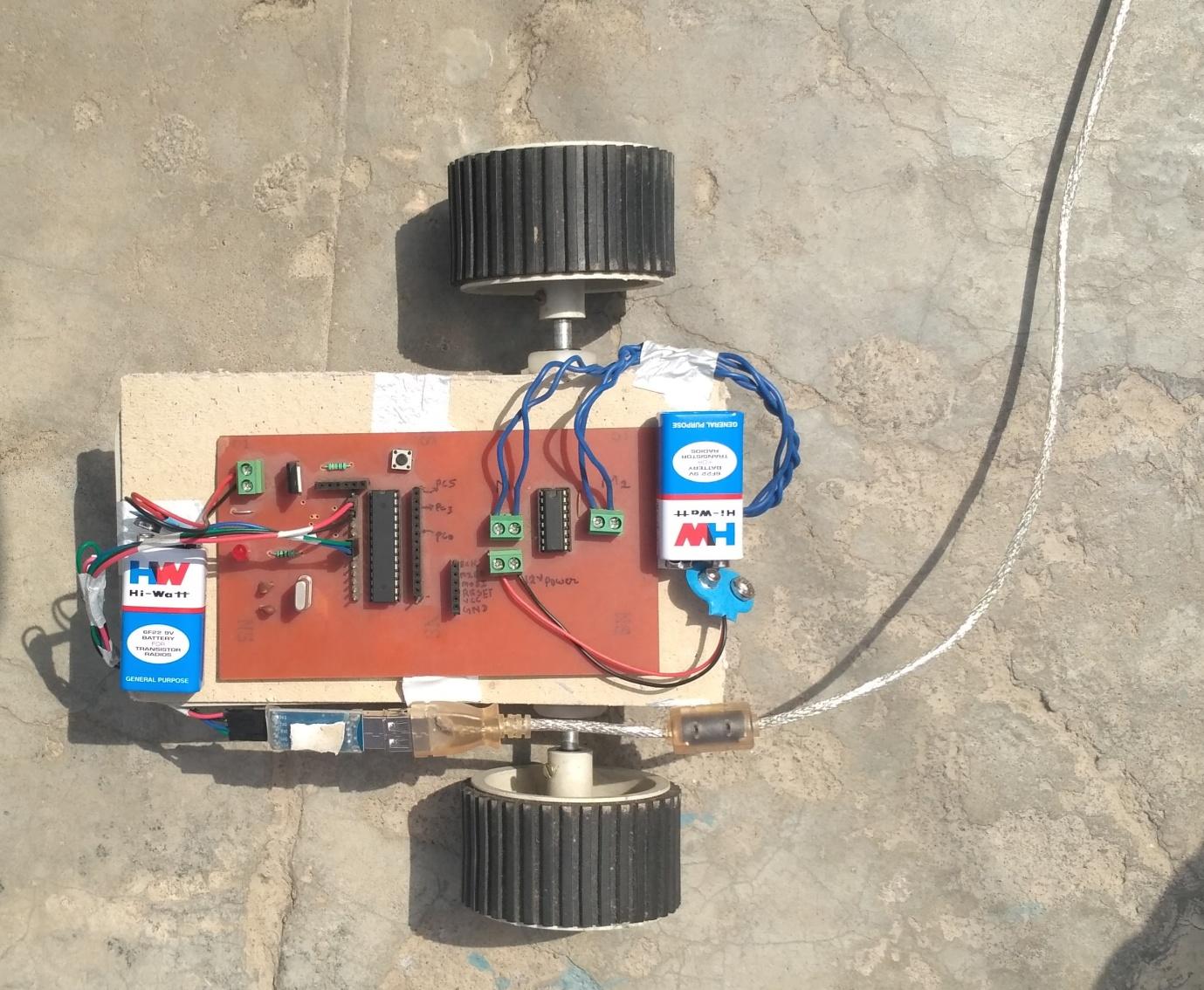


Figure: Wheelchair Assembly

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 Laptop 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.

**6. Testing and Results**

**6.1 Testing Strategy**

We used command window as a debug screen. And It 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.

It also indicates if the serial port connection is established or not.

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 Accuracy**

The project performs satisfactory with performance accuracy of around 60-70%. The results after testing it for 80 to 100 attempts to move in a random direction were made by the project members.

**7. Conclusions**

**7.1 Performance**

The system functions with an accuracy rate of 60-70 % 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 quite 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.

To stop the system a proper mechanism can be implemented by Eye Blink detection and Closed Eye Detection.

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.

**8. User Manual**

**STEP 1:** Install the webcam that is mounted on the Laptop.

**STEP 2:** Connect the hardware circuit (motors connected to the controller) to the 9 volt supply and also power the micro controller.

**STEP 3:** 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 4:** Make the eye ball movements that will be tracked which thereby will control the motors to move in the expected direction.

**9.MATLAB Code**

**Project.m**

clear all

clf('reset');

fig=figure('Name','VISION CONTROL SYSTEM','NumberTitle','off','color','k');

axis off;

text(0.2,1,'EYE-BALL','color','w','fontsize',50);

text(0.2,0.8,'CONTROL','color','w','fontsize',50);

text(0.2,0.6,'SYSTEM','color','w','fontsize',50);

text(0.2,0.2,'Developed by:','color','w','fontsize',20);

text(0.2,0.1,'Anurag,Banne singh,Chandan,Deepanshu','color','w','fontsize',10);

pause(5);

close(fig);

right=imread('right2.png');

left=imread('left2.png');

noface=imread('noface1.png');

straight1=imread('straight2.png');

detector = vision.CascadeObjectDetector(); % Create a det

%ector for face using Viola-Jones

detector1 = vision.CascadeObjectDetector('EyePairSmall'); %create detector for eyepair

fig1=figure('Name','VISION CONTROL SYSTEM','NumberTitle','off','color','k');

cam=webcam(1);

axis off;

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(2,2,2),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

for i=1:size(bboxeyes,1) %draw all the regions that contain eyes

rectangle('position', bboxeyes(i, :), 'lineWidth', 1, 'edgeColor', 'g');

end

bboxeyeshalf=[bboxeyes(biggest\_box\_eyes,1),bboxeyes(biggest\_box\_eyes,2),bboxeyes(biggest\_box\_eyes,3)/3,bboxeyes(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(2,2,3),subimage(eyesImage); hold on;

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

if ~isempty(centers)

pupil\_x=centers(1);

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

subplot(2,2,4);

if disL>disR+16

imshow(right);

right1;

display('right');

% fprintf(s,'%s',char(R));

else if disR>disL

subimage(left);

left1;

display('Left');

% fprintf(s,'%s',char(L));

else

subimage(straight1);

display('Straight');

straight;

%fprintf(s,'%s',char(F));

end

end

end

end

else

subplot(2,2,4);

subimage(noface);

% fprintf(s,'%s',char(mov0));

end

hold off;

end

**Left.m**

function [] = left1()

disp('Setting up serial communication...');

% Determine which COM port is for microcontroller and change

ard = serial('COM5','Parity','none','FlowControl','none','BaudRate',4800);

% Open serial COM port for communication

left1=imread('left2.png');

%set(ard,'Timeout',10);

fig2=figure('Name','VISION CONTROL SYSTEM','NumberTitle','off','color','k');

axis off;

% text(0.2,1,'STRAIGHT DETECTED','color','w','fontsize',50);

mov = 'L';

fopen(ard);

fprintf(ard,'%s',char(mov));

while true

imshow(left1);

a = fscanf(ard,'%s');

disp(a);

if a == 'L'

break;

end

end

fclose(ard);

close(fig2);

end

**Right.m**

function [] = right1()

disp('Setting up serial communication...');

% Determine which COM port is for microcontroller and change

ard = serial('COM5','Parity','none','FlowControl','none','BaudRate',4800);

% Open serial COM port for communication

sright2=imread('right2.png');

%set(ard,'Timeout',10);

fig2=figure('Name','VISION CONTROL SYSTEM','NumberTitle','off','color','k');

axis off;

% text(0.2,1,'STRAIGHT DETECTED','color','w','fontsize',50);

mov = 'R';

fopen(ard);

fprintf(ard,'%s',char(mov));

while true

imshow(sright2);

a = fscanf(ard,'%s');

disp(a);

if a == 'R'

break;

end

end

fclose(ard);

close(fig2);

end

**Straight.m**

function [] = straight()

disp('Setting up serial communication...');

% Determine which COM port is for microcontroller and change

ard = serial('COM5','Parity','none','FlowControl','none','BaudRate',4800);

% Open serial COM port for communication

straight2=imread('straight2.png');

%set(ard,'Timeout',10);

fig2=figure('Name','VISION CONTROL SYSTEM','NumberTitle','off','color','k');

axis off;

%subplot(2,1,1);

%text(0.2,1,'STRAIGHT DETECTED','color','w','fontsize',20);

mov = 'F';

fopen(ard);

fprintf(ard,'%s',char(mov));

while true

%subplot(2,1,2);

imshow(straight2);

a = fscanf(ard,'%s');

disp(a);

if a == 'F'

break;

end

end

fclose(ard);

close(fig2);

end

**10. Motor driver code**

**#include <avr/io.h>**

**#include <util/delay.h> // includes delay header file**

**#include <avr/interrupt.h>**

**//This function is used to initialize the USART**

**//at a given UBRR value**

**char buffer[10];**

**void USARTInit(uint16\_t ubrr\_value)**

**{**

**//Set Baud rate**

**UBRRL = ubrr\_value;**

**UBRRH = (ubrr\_value>>8);**

**UCSRC=(1<<URSEL)|(3<<UCSZ0);**

**//Enable The receiver and transmitter**

**UCSRB=(1<<RXEN)|(1<<TXEN);**

**}**

**//This function is used to read the available data**

**//from USART. This function will wait untill data is**

**//available.**

**char USARTReadChar()**

**{**

**//Wait untill a data is available**

**while(!(UCSRA & (1<<RXC)))**

**{**

**//Do nothing**

**}**

**//Now USART has got data from host**

**//and is available is buffer**

**return UDR;**

**}**

**//This fuction writes the given "data" to**

**//the USART which then transmit it via TX line**

**void USARTWriteChar(char data)**

**{**

**//Wait untill the transmitter is ready**

**while(!(UCSRA & (1<<UDRE)))**

**{**

**//Do nothing**

**}**

**//Now write the data to USART buffer**

**UDR=data;**

**}**

**int main()**

**{**

**char data;**

**USARTInit(103);**

**DDRC=0b1111111;**

**while(1)**

**{**

**data=USARTReadChar();**

**switch(data)**

**{**

**case 'F':**

**PORTC=0b00010100;**

**\_delay\_ms(5000);**

**PORTC =0x00;**

**USARTWriteChar(data);**

**break;**

**case 'R':**

**PORTC=0b00000100;**

**\_delay\_ms(1200);**

**PORTC=0b00010100;**

**\_delay\_ms(2000);**

**PORTC =0x00;**

**USARTWriteChar(data);**

**break;**

**case 'L':**

**PORTC=0b00010000;**

**\_delay\_ms(1200);**

**PORTC=0b00010100;**

**\_delay\_ms(2000);**

**PORTC =0x00;**

**USARTWriteChar(data);**

**break;**

**case 'B':**

**PORTC=0b00010000;**

**\_delay\_ms(5000);**

**PORTC =0x00;**

**USARTWriteChar(data);**

**break;**

**default:**

**PORTC=0b0000000;**

**break;**

**}**

**}**

**return 0;**

**}**

**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] Voila Jones Algorithm - http://en.wikipedia.org/wiki/Viola%E2%80%93Jones\_object\_detection\_framework

[3] Cascade Object Detector –

http://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html