



BRAC UNIVERSITY

# An eye-controlled system

---

An eye control system using OpenCV

**By**

**Onindita Afrin(09101030)**

**Mahabub Hassan(10210016)**

**Mohona Gazi Meem(08310040)**

**Supervisor**

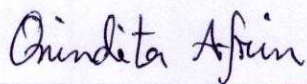
**Dr. Md. Khalilur Rhaman**

**12/4/2012**

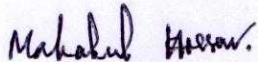
## DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole nor in part, has been previously submitted for any degree.

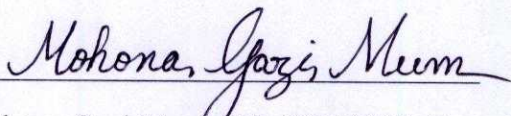
Signature of Author



Onindita Afrin (ID:09101030, Dept: CSE)



Md. Mahabub Hassan (ID: 10210016, Dept. ECE)



Mohona Gazi Meem (ID:08310040, Dept: ECE)

Signature of Supervisor



Dr. Md. Khalilur Rhaman.

Doctor of Information Engineering  
Assistant Professor,

BRAC University, Dhaka.

Phone: [+880-17-5204-2223](tel:+880-17-5204-2223)



## **Abstract**

An eye controlled system is a system which can be controlled by just looking at the system's GUI (Graphical User Interface) which figures out the position of our Iris respect to eye using webcam via image processing. A robot expected to connect with this program via serial communication and moves with our moving eyeball in the way we want to move it by hovering our eye ball over the GUI.



## **ACKNOWLEDGEMENTS**

It's an honor for us to thank from deeply heart to those people who help us from their positions to make the project possible. We owe our deepest gratitude to our supervisor, Dr. Md. Khalilur Rhaman, whose guidance, encouragement and support from the initial level till the end, help us to complete the thesis project and as well as inspired us for further research.

We would also like to show our gratitude to Mr. Matin Saad Abdulla, Assistant Professor, CSE dept. Mr. Farazul Haque Bhuiyan, Lecturer II, CSE dept and Dr. Mohammed Belal Hossain Bhuian, Assistant Professor, EEE dept. Without their continuous support it would be harder to complete the project.

We would like to give special thanks to our faculties, friends for their continuous support.

Last but not at least, thanks to the Almighty for helping me in every steps of this Thesis work.



## ABSTRACT

## ACKNOWLEDGEMENTS

## CONTENTS

Introduction: .....	5
System Overview:.....	6
First approach with neural network: .....	7
Block diagram:.....	7
Sigmoid function:.....	8
Neuroph Training:.....	8
Input vector: .....	11
Search for Eyes: .....	12
Result of Neuroph: .....	12
Second approach with OpenCV: .....	13
Eye detection:.....	13
Iris detection:.....	14
Implementation: .....	15
Driving with eye: .....	15
Electronics: .....	16
Serial port:.....	16
Mechanical:.....	17
Future Proposal:.....	18
Discussions: .....	19
References.....	20



## Introduction:

An eye controlled system is supposed to be controlled by looking at the system's monitor at its GUI. We divided the screen in six grids which is given below:

1	2	3
4	5	6
7	8	9

The quadrant we will look at it will be highlighted and the robot would perform certain tasks according to that i.e. if we look at second quadrant it will be highlighted and the robot will move forward and if we look at the eighth quadrant it will just move backward.



## System Overview:

We have integrated three parts in this project. First one is the software part which consists of Artificial intelligence, OpenCV and webcam and serial port API along with a .NET GUI and then comes the electronics part where we used microcontroller after programming it with C and we interfaced a car along with it.

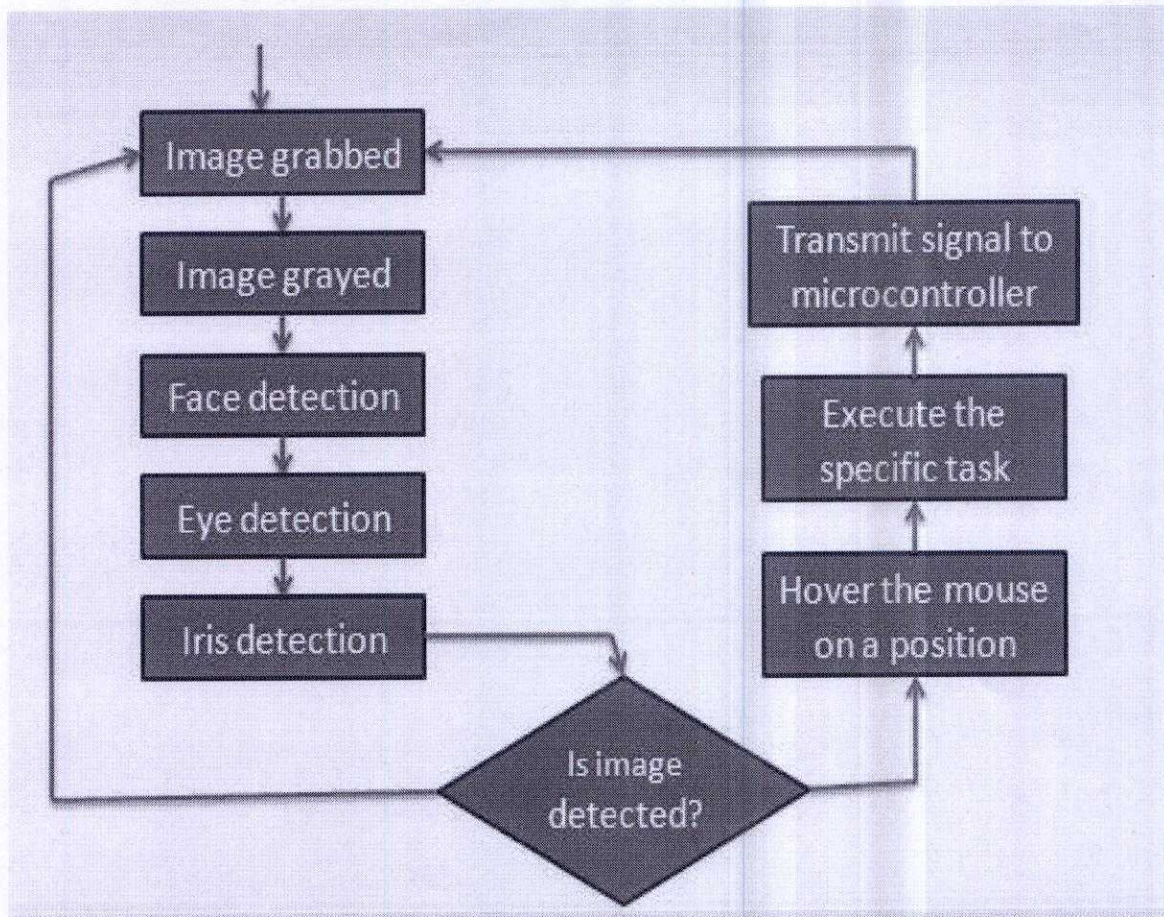


Figure1: Work-flow through the project



## First approach with neural network:

We used Java Neural Network Framework Neuroph to detect where we are looking at.

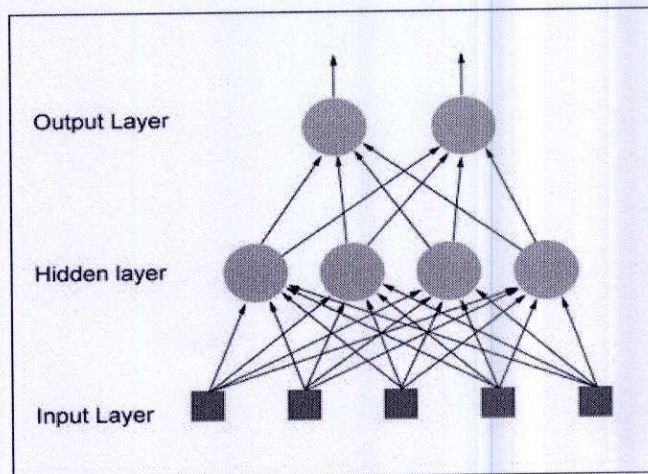


Figure2: Sample Neural Network

## Block diagram:

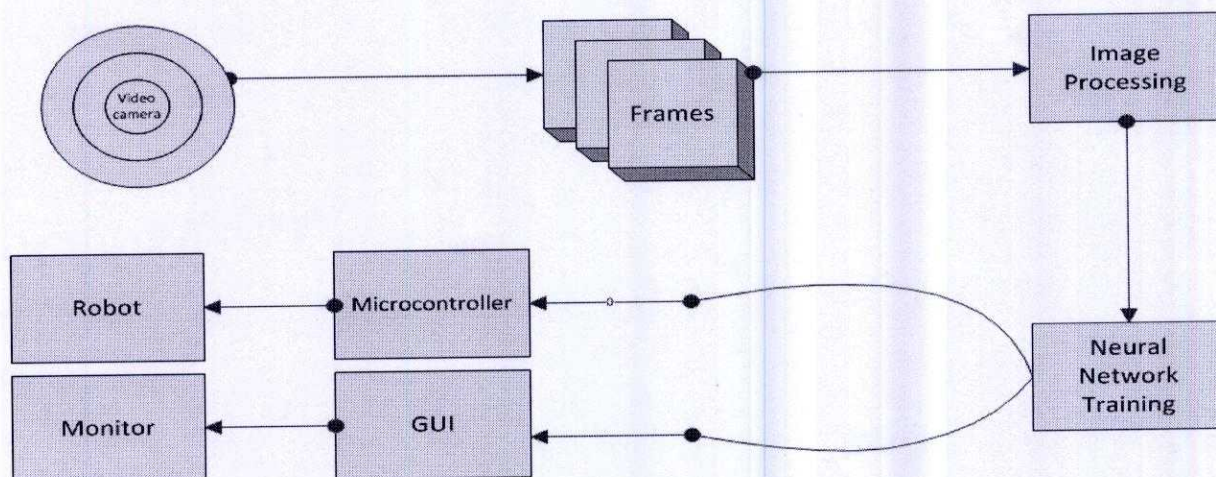


Figure3: Block diagram of eye detection process using neural network



We used webcam as our input media and we took the frames of the video stream for further processing where we used image processing and neural network. We used a Java Framework named Neuroph to train and test our Neural system. We used its built in class of image recognition with the efficient architecture “Multi Perceptron” and “Back propagation” learning algorithm with a sigmoid function.

### Sigmoid function:

$$output = \frac{1}{1 + e^{-a/p}}$$

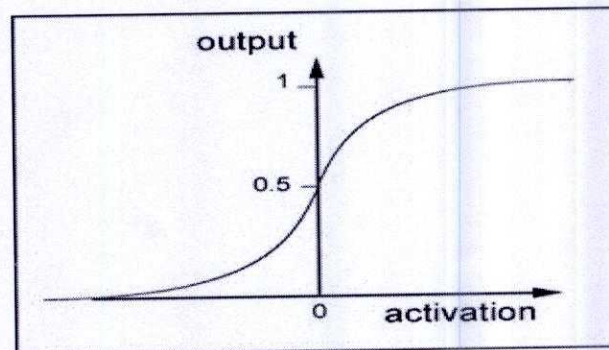


Figure4: graph of sigmoid function

It will look for eyes first in the frames using either OpenCV eye detection classifier or a Neuroph trained system. After the recognition of eye it is trained to figure out the Irish's position to determine the quadrant the user is looking at.

### Neuroph Training:



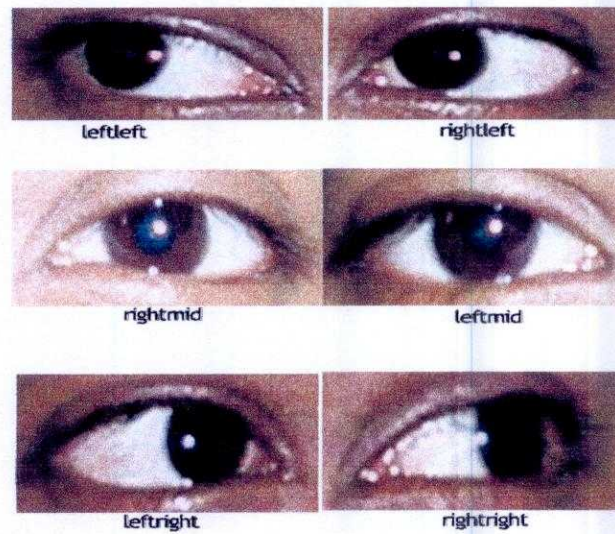


Figure5: Sample Irish image

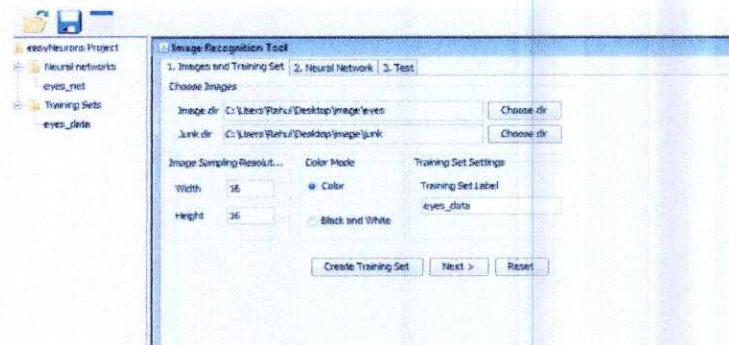


Figure6: Neuroph image training set



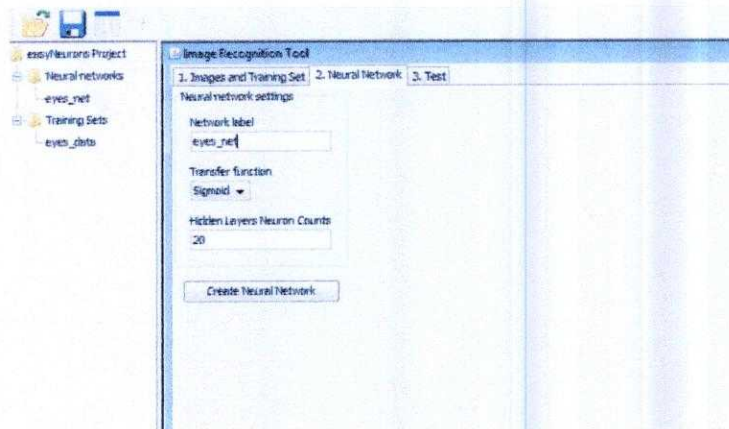


Figure7: Neuroph Neural network configuration

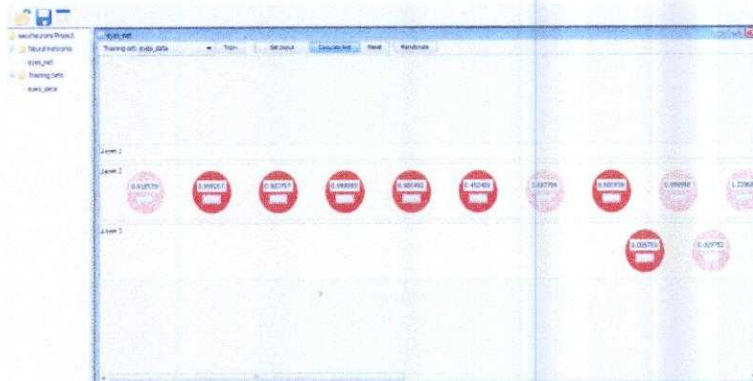


Figure8: Neuron training set

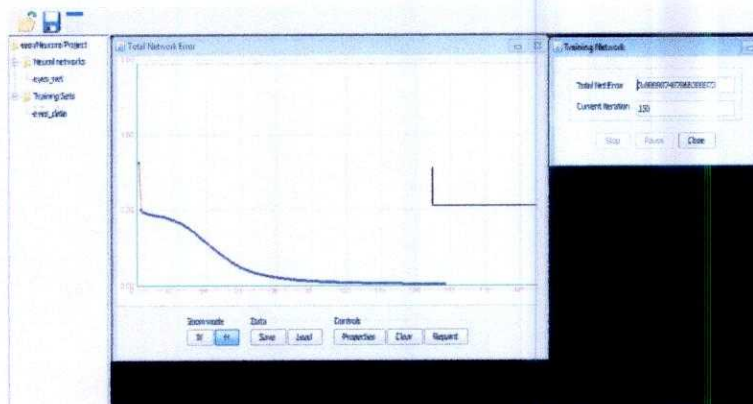


Figure9: Error detection graph



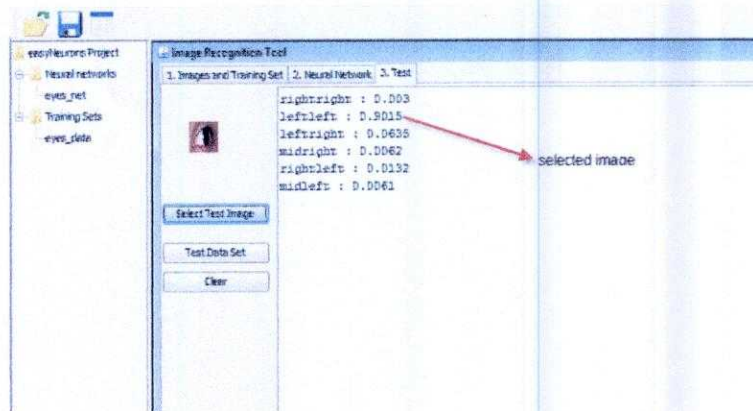


Figure10: Image detection with highest configuration

### Input vector:

To represent some image in a RGB system we can use three two-dimensional arrays, one for each color component, where every element corresponds to one image pixel.

```
int [ ][ ] redValues
int [ ][ ] greenValues
int [ ][ ] blueValues
```

For example, if pixel at location [20, 10] has color RGB[33, 66, 181] we have

```
redValues[10][20] = 33;
greenValues[10][20] = 66;
blueValues[10][20] = 181;
```

The dimensions of each of these arrays are [imageHeight][imageWidth]

We can merge these three arrays into a single one-dimensional array so it contains all red values, then all green and at the end all blue values. That's how we create **flattenedRgbValues[]**

The dimension of this array is [imageHeight \* imageWidth \* 3]

Now we can use this one-dimensional array as input for neural network, and to train neural network to recognize or classify them. Multi-layer perceptions are type of neural networks suitable for these tasks.



### **Search for Eyes:**

First we tried to detect eyes searching through the whole image. The computer cannot process this in real time. Then we made two windows where the user can put his/her eyes and then the system will detect whether there is an eye or not. If there is an eye, the system will detect in which grid the user is looking at. We used about five hundred of images to train the neural network for an eye and almost two hundred of images for each grid detection.



Figure11: Window procedure

### **Result of Neuroph:**

It lacks in precision and has a very poor performance in real time. But the precision can be improved by building an enriched database of eyes and a well-trained neural network.



## Second approach with OpenCV:

OpenCV can detect eyes and face in real time with 80% precision with its Viola-Jones object detection algorithm which uses haar-cascade classifier. We were able to detect eyes in real time with good precision using openCV java wrapper JavaCV. We tried to train our own haar-cascade classifier for iris detection. But it did not work that well.



Figure12: Face and Eye detection

### Eye detection:

First we detected the face, and then we detected the eyes setting the region of interest's value where the upper left corner is  $x = \text{face\_x} + \text{face\_width} / 5$  and  $y = \text{face\_y} + \text{face\_height} / 5$  and the width and height is the width of face and height of face/3 respectively.



## **Iris detection:**

Then we detected the iris selecting the ROI of the eyes. At first we converted the image into gray scale, then we filtered it with Gaussian blur and then we applied Canny-edge detection algorithm. At last, we used openCV's Hough circle detection to detect the iris. It works very well in real time with good precision.

### **Steps:**

- Image



- RGB to gray scale conversion



- Filtering with Gaussian filter



- Canny-edge detection



- Circle detection





## Implementation:

We used two ways to control our robot. The first one uses eye detection to hover the mouse cursor and the second one uses iris to hove the mouse cursor.

### Driving with eye:

The camera we used gives an image of 640x480 px. We made an interface with C# of form size 640x480 px where we have 9 grids to drive the car forward, backward, left, right, left-forward, right-forward, left-backward, right-backward and stop. In which position the eye would be detected, the cursor will move on that grid. In this manner, we can drive our robotic car robustly with any eyes.

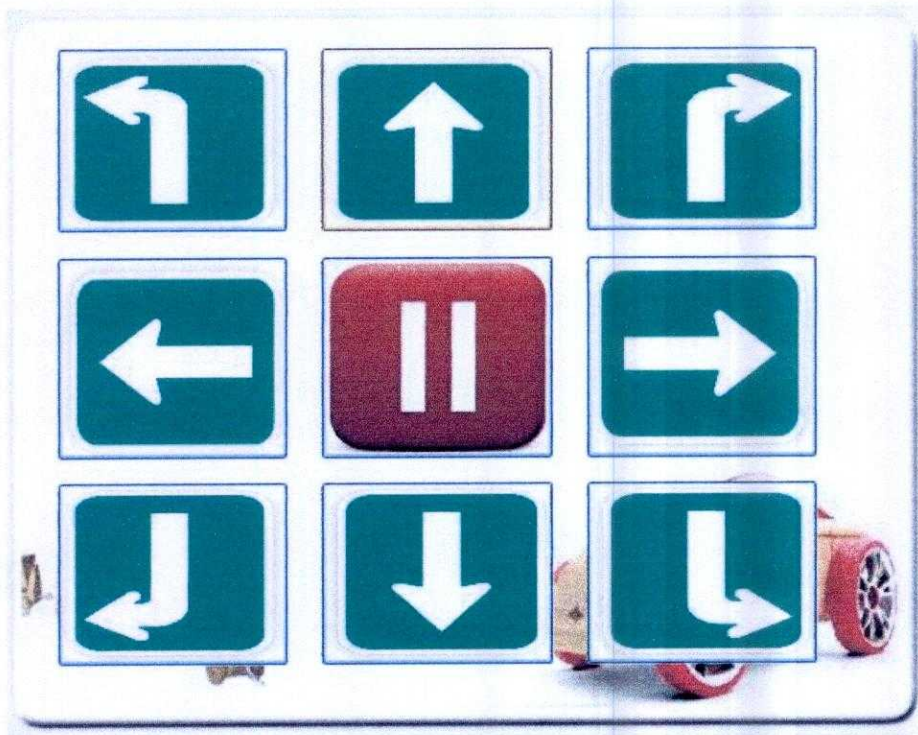


Figure13: Interface



## Electronics:

The heart of our circuit is a micro-controller. We used a micro-controller of microchip brand (PIC16F877A). A crystal is required to drive the micro-controller. In our case, we used a crystal of 8 MHz. Then we used some NMOS's for switching (IRF3205). We used a circuit of RFIC communication and we controller our car via this. When we set the output high of micro-controller, it will short the drain and source of NMOS and will drive the car.

## Serial port:

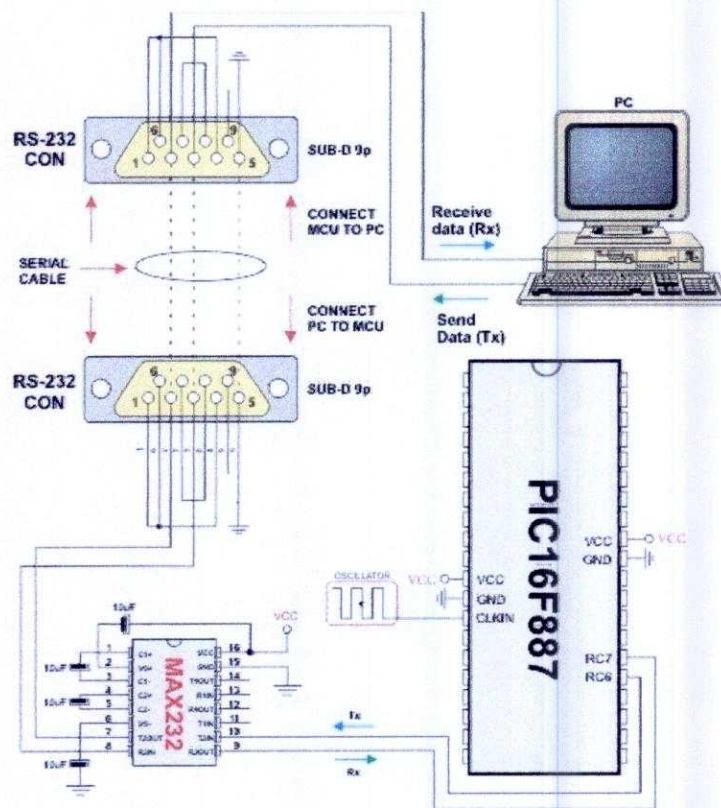


Figure14: serial port diagram

We used C# library to interface between our computer and micro-controller.

**Mechanical:**

We have bought a car and interfaced it with our circuit.



## **Future Proposal:**

- Iris detection using other methodology
- Getting precision for any environment usage
- Eye blink detection for clicking
- Implementing it in full length for full screen



## **Discussions:**

At the very beginning of the project our primary goal was to detect eye which we accomplished by neural network but for more precision we moved on to OpenCV. We got satisfying output on detecting our Iris's position. This system can be used anywhere i.e. we can use it like a mouse, in other robotic project.



## References

### Internet

**Nuuroph, Neural Network Based Framework**<<http://neuroph.sourceforge.net/>>

**NeatBeans Zone**<<http://netbeans.dzone.com/articles/neuroph-smart-java-apps-neural-1>>

**Image Recognition**<[http://neuroph.sourceforge.net/image\\_recognition.html](http://neuroph.sourceforge.net/image_recognition.html)>

**OpenCV**<<http://ubaa.net/shared/processing/opencv/>>

**OpenCV WIKI** <<http://opencv.willowgarage.com/wiki/>>

### Image Processing Introduction

<[http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref\\_cv.htm](http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm)>

### Image Processing with

**OpenCV**<[http://www.discover.uottawa.ca/~qchen/my\\_presentations/A%20Basic%20Introduction%20to%20OpenCV%20for%20Image%20Processing.pdf](http://www.discover.uottawa.ca/~qchen/my_presentations/A%20Basic%20Introduction%20to%20OpenCV%20for%20Image%20Processing.pdf)>

**OpenCV Eye Detection**<[http://nashruddin.com/OpenCV\\_Eye\\_Detection](http://nashruddin.com/OpenCV_Eye_Detection)>

**OpenCV Eye Detection with sub-regions of the face**  
<<http://eyedetectionopencv.codeplex.com/>>

**Introduction to programming with OpenCV**<<http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/>>

**IRIS detection**<<http://codingbox.net/jinstall/>>

## **Books**

Raul Rojas (1996), **Neural Networks: A Systematic Introduction**

Jeff Heaton (2008), **Introduction to Neural Networks for Java**

Cornelius T. Leondes (1998), **Image Processing and Pattern Recognition**

Gary Bradski & Adrian Kaebler (2008), **Learning OpenCV Computer Vision With OpenCV Library**



Robin Hewitt (2007), **Seeing with OpenCV – A Computer Vision Library**

Robert Laganier (2011), **OpenCV 2 Computer Vision Application Programming Cookbook**

## **Papers**

Kohei Arai & Ronny Mardiyanto (2011), **Autonomous Control of Eye Based Electric Wheel Chair with Obstacle Avoidance and Shortest Path Findings Based on DijkstraAlgorith**

MatjažDivjak& Horst Bischof, **Eye blink based fatigue detection for prevention of Computer Vision Syndrome**

Jon Parris (2008), **Face and Eye Detection on Hard Datasets**

R.B.J van Brakel& P. Hanckmann (2008), **EYE TRACKING Additional component computer graphics**

ZafarSavas (2005), **REAL-TIME DETECTION AND TRACKING OF HUMAN EYES IN VIDEO SEQUENCES**