Project Report On

"CRICBOT"

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATIONS

Submitted By,
ADROJA NIKHIL EC-07
AGRAWAL HARSHVARDHAN EC-08

Under the guidance of
Prof. HARDIP K SHAH

Department of Electronics and Communication,
Faculty of Technology,
Dharamsinh Desai University,
Nadiad,Gujarat
October 2013

Certificate

This is to certify that **Mr Harshvardhan Agrawal** Roll No EC-08, Identity No. 106002 and **Mr Nikhil Adroja** Roll No EC-07, Identity No 101009 of BTech semester VII in branch of Electronics and Communications has successfully completed his project "Cricbot" as a part of the Term Project during the period of 1st July 2013 to 15th October 2013.

Prof. Hardip K. Shah
Project Supervisor,
Dept. of Electronics & Communication,
Dharamsinh Desai University, Nadiad

Dr. Nikhil J. Kothari
Head of Department,
Dept. of Electronics & Communication,
Dharamsinh Desai University, Nadiad

Acknowledgments:

I hereby express my sincere thanks to respected **Hardip Shah** Sir for giving me the chance to be the part of this project.

I also take an opportunity to express my deep sense gratitude to the EC Department, Dharamsinh Desai Institute Of Technology for providing all the required facilities and inspiration in bringing out this project work.

I express my thanks to **Mr. Mitesh Limachiya and Mr. Yogesh Meghrajani** for their keen advice and valuable cooperation during completion of this project.

ABSTRACT

This report presents a ball collecting robot. In this system, a single camera system is used as a sensor to perceive the trajectory of the ball. The environment is first captured as an image using a webcam which is mounted on a laptop. Image processing methods are then performed to identify the location of the ball within the environment. Program is written in MATLAB. It calculates the centroid of the identified color intensity and according to that it sends the control commands over the serial port to the controller of robot via USB port. The basic objective is to catch the specific colored ball in the specified region. The proposed method does not make use of any other type of sensor other than the webcam.

Applications:

Imagine an earthquake occurs and many buildings collapse and get unstable. A family might be hurt during an earthquake and be unable to come out of the building. But because the building is unstable and might collapse, it is too dangerous for human rescuer to help. This is a good example where these robots might be used. The robots might autonomously find persons in the building and bring them securely out of it. And if the building collapses, the robot is replaced/repaired easier than a human.

Also this robot can be used to clear garbage and other dirt present in stadiums or on roads and as a result avoid the use of human labor.

Tools Used:

- Software:
    1. Matlab
    2. Arduino Bootloader Software
- Hardware:
    1. L293D driver IC
    2. Arduino Kit for Atmega8-8L
    3. DC Motors
    4. Serial Cable
    5. Working Webcam mounted on a laptop

INDEX……

CHAPTER 1
INTRODICTION

Image processing is a form of signal processing where the input signals are images such as photographs or video frames. The output could be a transformed version of the input image or a set of characteristics or parameters related to the image. The computer revolution that has taken place over the last 20 years has led to great advancements in the field of digital image processing. This has in turn, opened up a multitude of applications in various fields, in which the technology could be utilized.

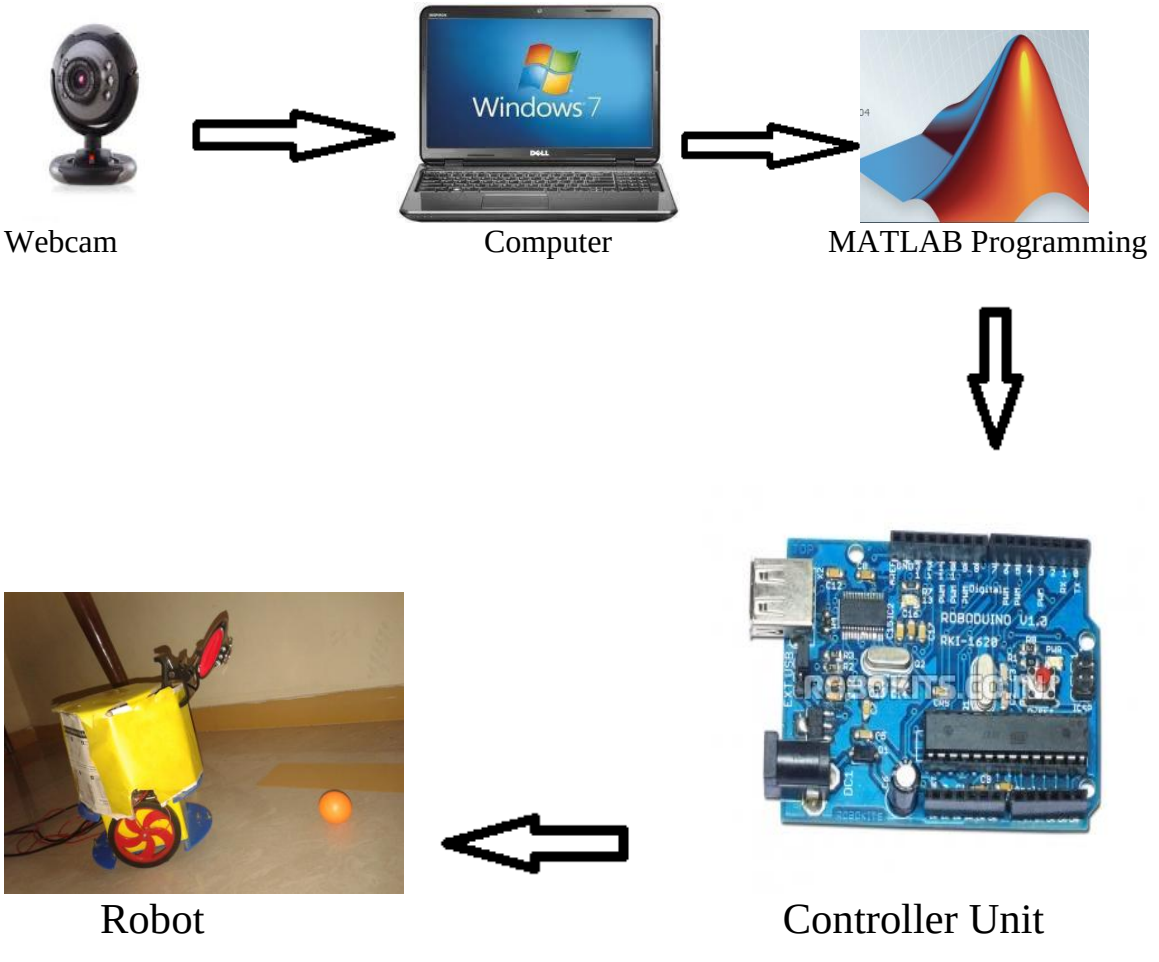
The aim of this project is to present a method for visual servo control using only visual images from a webcam. Visual servo is the use of image data in closed loop control of a robot. Without doubt, today, the use of vision in robotic applications is rapidly increasing. This is due to the fact that vision based sensors such as webcams are falling in price more rapidly than any other sensor. It is also a richer sensor than traditional ranging devices, particularly since a camera captures much more data simultaneously. Images can be captured by camera, and subsequently, processed using MATLAB

When the trajectory of the object is known, the object motion can be calculated. Common approaches to utilizing vision information for controlling robotic manipulators are the (i) position (velocity) control and the (ii) image feature control (color).
The goal of our work is to visually track a ball thrown in a region and send the data to the robot to move for catching the ball at the expected location. The most important problem that had to be addressed is that of selecting the feature to use for segmenting the ball. Using approach where to detect the ball by its shape is not an efficient approach, in this approach we have require to set our background static or stable because here tracking of moving object is obtained by computing the difference between the actual image (new frame) and some reference image i.e. background image which is static. This is very difficult when we used our system in real time constraints. Instead we using color of the object in which we just take care about the lightning conditions we never bother about setting the threshold value of background with the original object.

In this report we present a ball collecting robot, which significantly differs in two aspects. First, we've developed a robot with commercially available parts with a low cost. This is challenging for me to build a robot with such a low cost because the hardware and motors of the should be able to reach to the ball fast enough and in addition it has to be light weight, so that the impact of the weight doesn't effects the motors. Moreover, it is also challenging to implement the algorithms for the movements, because all the work here is to be carried out for the real time results so we need such algorithms which are more efficient and are able to work on the setup. The second new aspect is that, we used a webcam that captured 10 frames per trigger to determine the ball and to calculate its trajectory. The image processing algorithm is used here responsible for tracking ball and determines location of ball in real time. So using these simple steps I am here able to calculate the trajectory of ball and move up to the ball.

CHAPTER 2
EXPERIMENTAL SETUP

Predicting the ball with stereo vision camera is extremely expensive as we need Giga interface cards and more powerful programming language to interface. We used a webcam to implement our requirment. Computer captures the video from the camera. A program is written in MATLAB which calculates the location of the ball in real time. These details are sent to the controller of the robot via USB port at a baudrate of 9600 bps. The microcontroller used for riving the actuators is Atmel's ATMEGA8-8L which runs at clock frequency of 16 kHz. The actuators used are dc motor running at 250 rpm.



2.1 Experimental Setup

CHAPTER 3
THE CONTROL UNIT

3.1 Arduino Atmega8-8L:

Fig 3.1.1 Arduino bootloader for ATMEGA8-8L

The 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 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

3.2 The Arduino Bootloader Development:

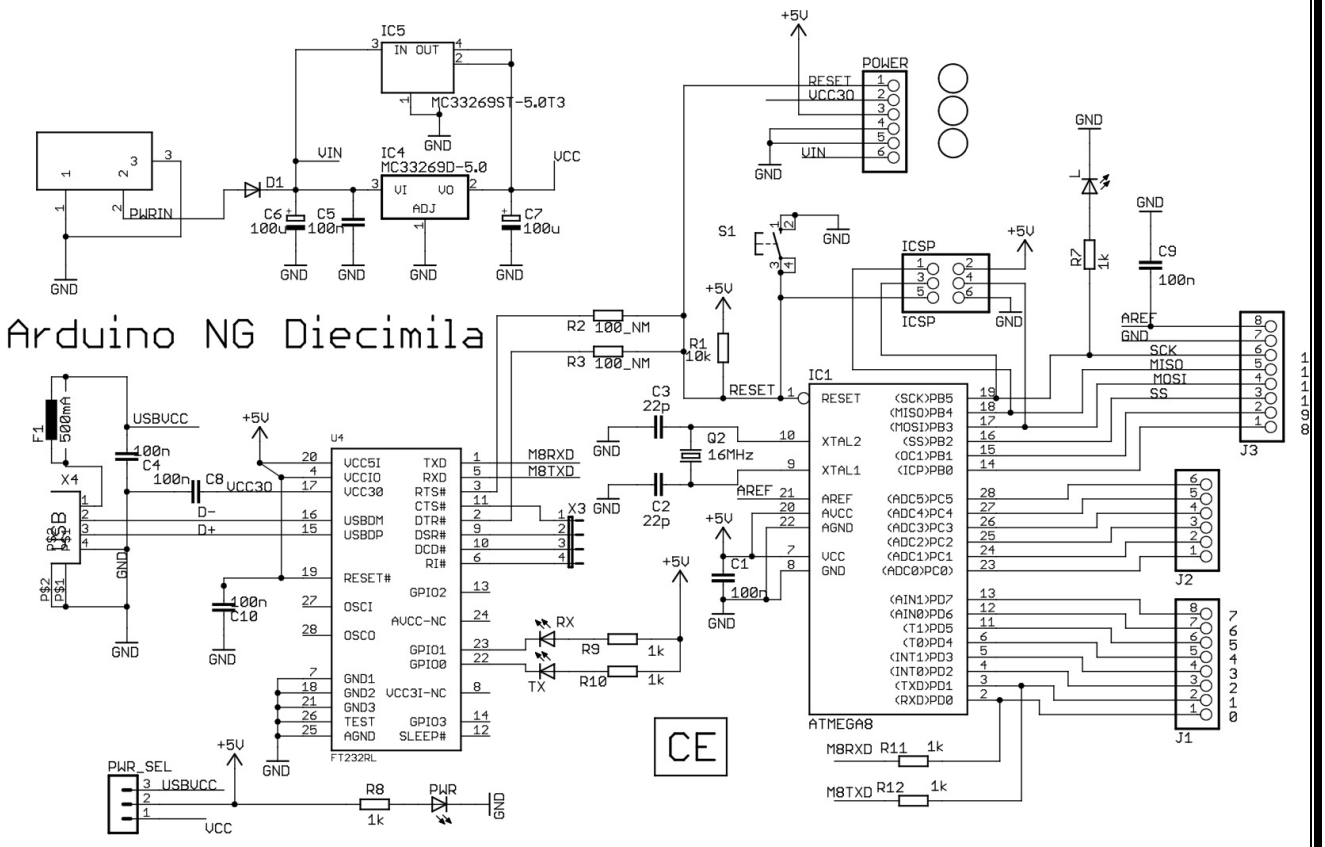
The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Software written using Arduino are called **sketches**. These sketches are written in the text editor. Sketches are saved with the file extension .ino. It 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 environment including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar

buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

The ATmega8 bootloader only takes up 1 KB of flash. It does not timeout when it receives invalid data, you need to make sure that no data is sent to the board during the 6-8 seconds when the bootloader is running.

Some ancient versions of the bootloader run at 9600 baud (instead of 19200). In order to successfully upload sketches to boards with this bootloader, you'll need to change the serial.download_rate in your preferences file to 9600.



3.2.1 Circuit Diagram for the Arduino Kit

CHAPTER 4
THE DRIVER UNIT

MOTOR DRIVER

4.1 INTRODUCTION

This circuit , shown below for the motor driver , is basically use for operating the motors of robot. We are using L293D driver IC for that purpose. As shown in the fig. 7.1.1.

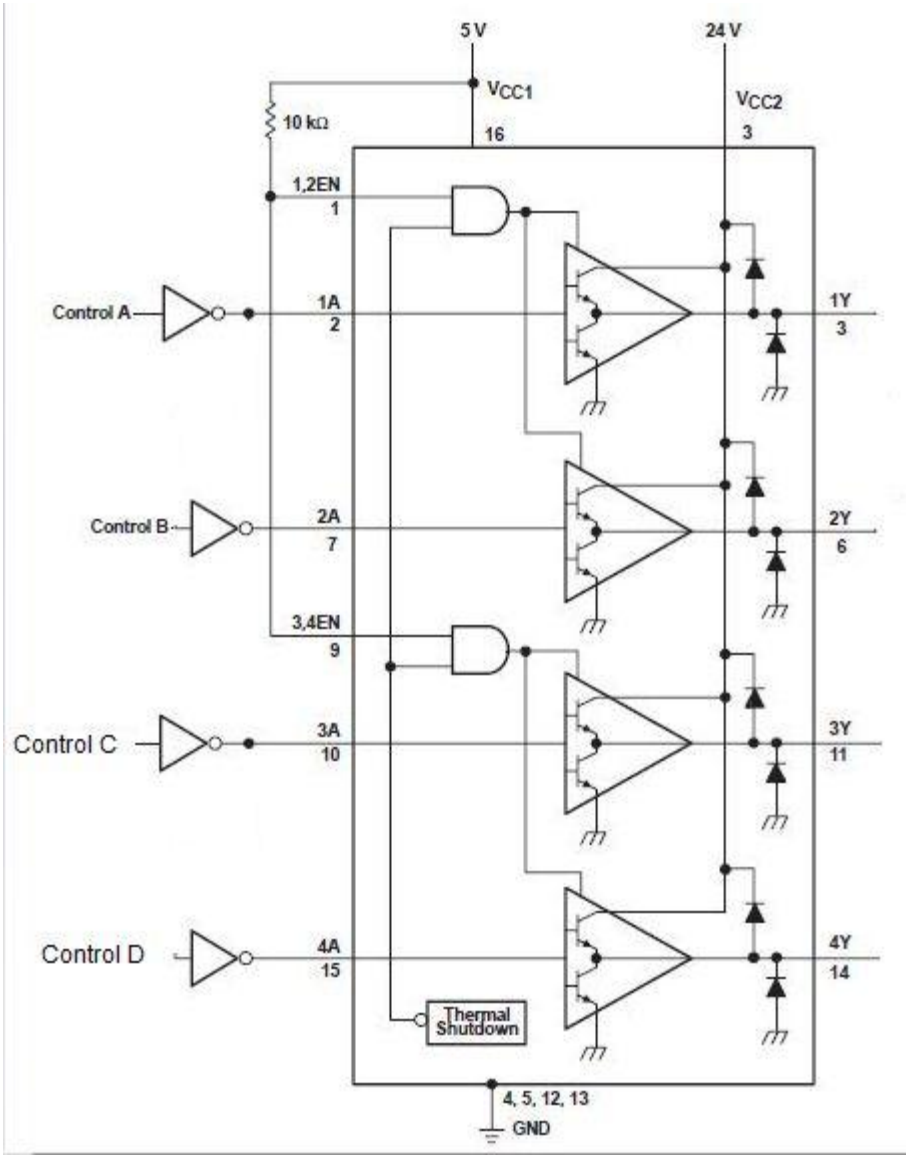


Fig. 4.1.1 Logic symbol of IC L293D

Here in this circuit we are giving input as the binary code, this IC will increase the current so that we can drive the motor. The L293D is a quadruple high-current half-H driver designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5V to 36 V. It is designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL-compatible. Each output is a complete totem-pole drive circuit with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. External high-speed output clamp diodes should be used for inductive transient suppression. When the enable input is low, those drivers are disabled, and their outputs are off and in a high-impedance state. With the proper data inputs, each pair of drivers form a full-H (or bridge) reversible drive suitable for solenoid or motor applications. A VCC1 terminal, separate from VCC2, is provided for the logic inputs to minimize device power dissipation.

CHAPTER 5
ACTUATORS

5.1 DC Motors:-

- The motors which we are using are a low power DC motor, which uses for special robotics purpose. It is shown in fig. 6.1 as below.

Fig. 5.1.1 DC Motor

5.2 WORKING:-

The name "H-Bridge" is derived from the actual shape of the switching circuit which controls the motion of the motor. It is also known as "Full Bridge". Basically there are four switching elements in the H-Bridge as shown in the figure below.

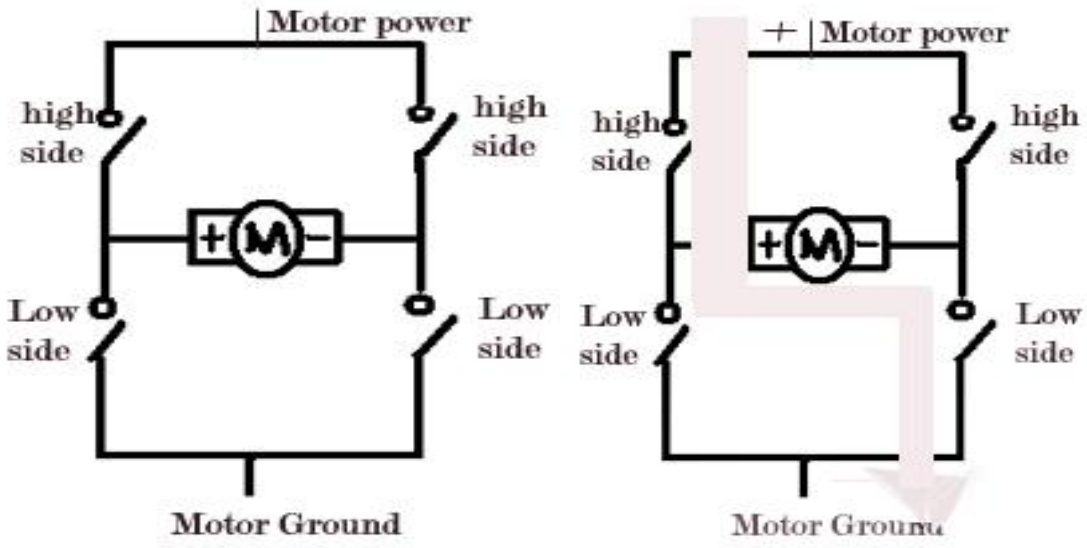


FIG 5.2.1 MOTOR WOKING

As you can see in the figure above there are four switching elements named as "High side left", "High side right", "Low side right", "Low side left". When these switches are turned on in pairs motor changes its direction accordingly. Like, if we switch on High side left and Low side right then motor rotate in forward direction, as current flows from Power supply through the motor coil goes to ground via switch low side right. This is shown in the figure below.

Similarly, when you switch on low side left and high side right, the current flows in opposite direction and motor rotates in backward direction. This is the basic working of H-Bridge. We can also make a small truth table according to the switching of H-Bridge explained.

Truth Table				
High Left	High Right	Low Left	Low Right	Description
On	Off	Off	On	Motor runs clockwise
Off	On	On	Off	Motor runs anti-clockwise
On	On	Off	Off	Motor stops or decelerates
Off	Off	On	On	Motor stops or decelerates

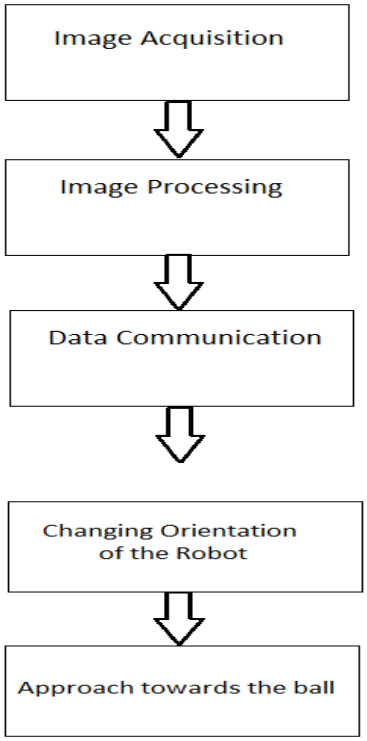
Fig 5.2.2 Truth Table

CHAPTER 6
SOFTWARE IMPLEMENTATION

6.1 Image Processing Algorithm:

The project uses a single camera system for the purpose of image processing. The camera is mounted such that the camera feed takes a 640x480 size region. This video feed is captured in the computer. A program is written in MATLAB to detect the location of ball in real time and predict the trajectory of the ball. The program also calculates the angle required to send to dc motors. The program is written in MATLAB using standard Image Processing functions. MATLAB is a program that implements many algorithms commonly used in the field of signal and image processing. Images as they are stored on computers are large, unstructured, two dimensional arrays of pixels. Processing techniques can also be applied to videos, which are stored as sequences of images. MATLAB provides functions that can be used for tasks such as locating faces in an image, recognizing predefined objects and shapes and detecting movement in a video.

6.2 BLOCK DIAGRAM OF THE ALGORITHM:



The software implementation consists of the following steps:

- Image Acquisition using the camera.
- Image Processing using Matlab.
- Data Communication.
- Building an acceleration module.

The project is divided into the following challenges:

1) Image Acquisition

Acquiring an image in real time from a web cam or any camera source is quite simple. it can be done with few lines of simple code. It's interesting to know about you're camera before you use it and it's required image actualization hardware information.

Fig 6.1.1 Image taken by camera

>> imaqhwinfo

InstalledAdaptors: {'winvideo'}

MATLABVersion: '7.0 (R14)'

ToolboxName: 'Image Acquisition Toolbox'

ToolboxVersion: '1.5 (R14)'

Get to know about the adaptor installed in device. In my case its ‘winvideo’. Winvideo should be replaced with adaptor name. Most of us will be having winvideo as the installed adaptor.
For few more details about cam, check out with this command

>> imaqhwinfo
InstalledAdaptors: {'coreco' 'winvideo'}
MATLABVersion: '7.6 (R2008a)'
ToolboxName: 'Image Acquisition Toolbox'
ToolboxVersion: '3.1 (R2008a)'

>> imaqhwinfo (’winvideo’, 1);
DefaultFormat: 'YUY2_160x120'
DeviceFileSupported: 0
DeviceName: 'Acer CrystalEye webcam'
DeviceID: 1
ObjectConstructor: 'videoinput('winvideo', 1)'
SupportedFormats: {1x5 cell}

>> vid=videoinput('winvideo',1)
Summary of Video Input Object Using 'Acer CrystalEye webcam'.
Acquisition Source(s): input1 is available.
Acquisition Parameters: 'input1' is the current selected source.

10 frames per trigger using the selected source. 'YUY2_160x120' video data to be logged upon START. Grabbing first of every 1 frame(s). Log data to 'memory' on trigger.
Trigger Parameters: 1 'immediate' trigger(s) on START. 0 frames acquired since starting.
0 frames available for GETDATA. Now its time to adjust the lens at proper angle and it depends on cameras vertical and horizontal view angles.

>>rgb_image = getsnapshot (vid);

To get snap shot of a video object that is all with Image Acquisition. Now its time to process this image.

2) Image Processing

Processing this acquired image is the crucial and important phase in developing a vision system. The processing method varies from person to person, its like how you are looking at image and what is that you are looking for in the image. It all depends on how the frame is. Whether you are looking for some colors in the frame or want to find out obstacles in the frame and how to go about with lot of variations in lighting.

Let's start off this phase with few important functions.

Two commands to view image which has been captured.

*** imview('I')**

*** imshow('I')**

If u want to display two or more image using imshow then command syntax would be

* **figure, imshow('I2')**

We can read an image already stored in computer with

*** I=imread('imgname.ext');**

1) With imview you can see the value of each pixel in the frame and it's a powerful tool for image analysis which helps us to decide our algorithm.

2)Processing time for imview() is few milliseconds more than imshow() so better to use imshow(). If you want to display some image in the process of execution.

Run time or processing time is a very important factor to be considered in image processing. Even a fraction of millisecond reduction in runtime would be an add-on to your code. You can check your codes runtime by using tic and toc commands at the beginning and end of your infinite loop. In task based processing proper analysis of image gets you an effective code. This analysis also depends on type of operation we are gonna adopt… i.e.

* Point

* Local

* Global

- Point- The output value at a specific coordinate is dependent only on the input value at that same coordinate.
- Local-The output value at a specific coordinate is dependent on the input values in theneighborhood of that same coordinate.
- Global-The output value at a specific coordinate is dependent on all the values in the input image.

We will stick on with local operation for now. So our region of interest (ROI) in a frame (i.e. red ball), should be properly analyzed for RGB values and tabulate the values for 10 different frames. This RGB values can be noted from pixel info at bottom left corner of the frame. Ex: In the above frame, pixel at point (165,151) has [R G B] values [188 60 137] respectively

- **Ball Follower Algorithm:**

This is the virtual frame in which (x, y) is the center of the ball. We consider entire blue box as origin and 'upper blue U' is positive Y axis and 'lower blue U' as negative Y axis. Now the logic is, we will find the quadrant in which the center of the ball is there and act accordingly.

3) Data Communication

Communicating with robot, this is interesting to get on with it. MATLAB provides good support and ease to access any of these ports for communicating with robot

1. Parallel port
2. Serial port
3. USB

Personally if like to use parallel port because of its good support with MATLAB and its access time is less compared with other ports. But serial communication is quite useful in many projects. it's just with couple of lines we can access this Serial port. some times it happens that that you need to close to your MATLAB and then reopen or even restart your system sometimes when you are using serial port in continuous loop.

Parallel Port:

parport=digitalio('parallel','LPT1?);

This command creates a digital I/O object… once you are done with this command you can access your parallel port.

Line1= addline(parport,0:3,'out');

Now you need to tell Matlab which lines of parallel port should be used as output or input pins. This command configures first 4 data pins of parallel port as output pins.

Line2= addline(parport,4:7, 'in');

Suppose you want to send 4 bits to parallel port according to requirement, store the values in an array.

output=[0 0 0 0;0 1 0 1;1 0 0 1;0 1 1 0;1 0 1 0];

pval=output(1,:);

Note: This output (1,:) lists all the values in the 1st row of output array.

putvalue(parport,pval);

Note: It's the last step, to send the output values to parallel port.

Serial Port:

Fig 6.1 USB cable A and B

It is used for serial transmission of the data to the controller.

ser=serial('com1?);

Note: This creates a serial object.

fopen(ser)

Note: This open the object in both read and write mode…

fprintf(ser,'data');

Note: writes the data to the object… i.e. sends data to serial port

fclose(ser)

Note: It's important to close the object at the end of communication.

4) Important Concepts of Image Processing.

1. Image adjustment
2. Image conversions
3. Edge detection

1.Image adjustment:

You can adjust the value of pixels in a frame to some fixed range if light conditions are varying. Imadjust() is the command to be used for adjusting the values of the pixel to fixed range. you can find more details at this link.

2. Image conversions:

- RGB to Gray Scale

Converting RGB to gray scale is basically depends on image color…In our case we have used orange color ball so our values for orange color should be 0(maximum intensity white color).

For this we use thresholding of image.

We use

rgb_image(: , : , 1); for red

rgb_image(: , : , 2); for green

rgb_image(: , : , 3); for blue

Red colour is between 222 to 255(intensity level)

Green colour is between 95 to 127(intensity level)

Blue colour is between 42 to 77(intensity level)

So final image ‘I’ becomes…

I=((fR>=222)&(fR<=255)&(fG>=95)&(fG<=127)&(fB>=42)&(fB<=77));

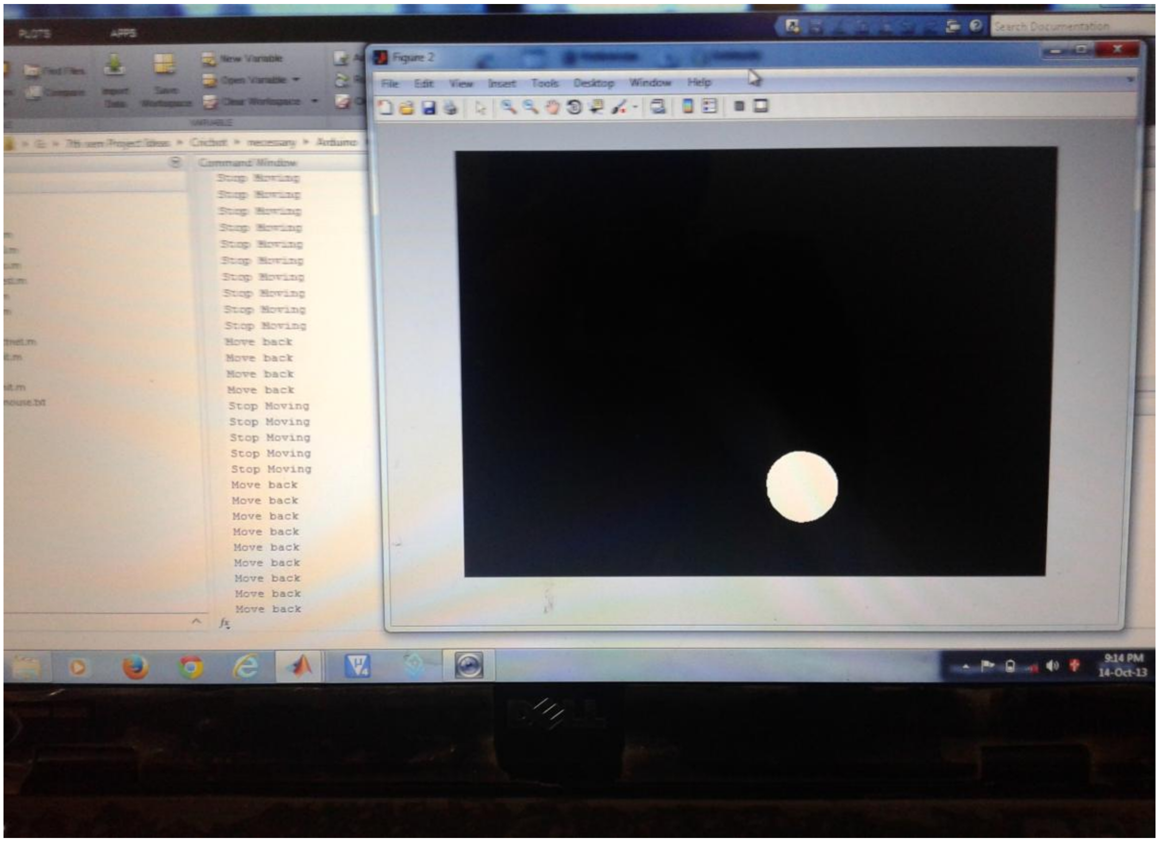


Fig 6.4.1 Image acquisition and command transmission

5) Logic of the Program.

Now our logic of the moving robot right left forward backword is as following.

Cbar=horizontal mean of white spot

rbar=verticle mean of white spot

x1=x/2-120;

x2=x/2+120;

y1=y/2-30;

y2=y/2+30;

e=(((cbar>=x1)*2*2*2) + ((cbar<=x2)*2*2) + ((rbar>=y1)*2) + (rbar<=y2)); % Converting to decimal number

If the case is 5 than it ll move left and than our case will become 13. So programming is shown below.

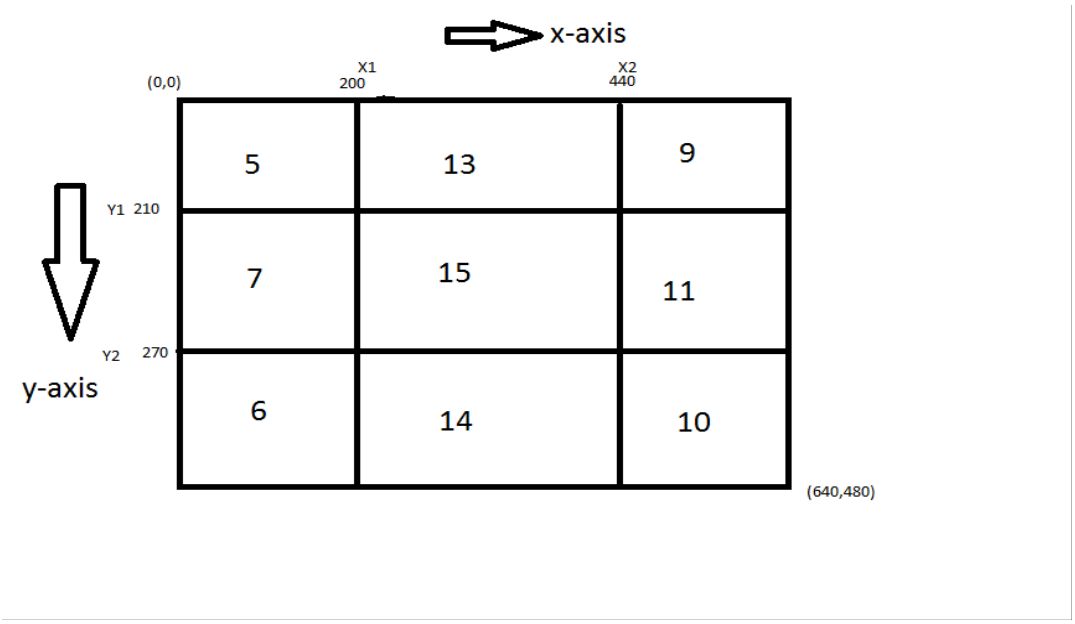


Fig 6.2.1 Centroid Logic

```
switch (e)
case 5
disp('Move left');
%fwrite(ser,'L');
case 6
disp('Move left');
%fwrite(ser,'L');
case 7
disp('Move left');
%fwrite(ser,'L');
case 9
disp('Move right');
%fwrite(ser,'R');
case 10
disp('Move right');
%fwrite(ser,'R');
case 11
disp('Move right');
%fwrite(ser,'R');
```

```
case 13
disp('Move forward');
%fwrite(ser,'F');
case 14
disp('Move back');
%fwrite(ser,'B');
otherwise
disp('Stop Moving');
%fwrite(ser,'S');
```

CHAPTER 7

MATLAB PROGRAM

```
clear all;
clc;                                          % Clearing Matlab desktop
vid=videoinput('winvideo',2,'YUY2_640x480');       % Defining the video input object
set(vid,'FramesPerTrigger',1);
set(vid, 'ReturnedColorspace', 'rgb');            % Setting frames per trigger
preview(vid);                                  %Showing the video of the moving Ball
 pause(5);                % Waiting for a certain time for the system to get initialised
rgb_image = getsnapshot(vid);          % Storing Image in an array variable
[a b c]= size(rgb_image);        % Determining the size of the captured frame.
y=a;
x=b;
% Defining Boundaries
x1=x/2-120;
x2=x/2+120;
y1=y/2-30;
y2=y/2+30;
  ser=serial('COM3');            % Defining the specified COM Port to be used
  pause(5);
  fopen(ser);                     % starting serial Communication, opening serial port


while(1)
pause(1);
image = getsnapshot(vid);      % storing image in an array variable
flushdata(vid);                          %Flushing the buffer
rbar=0;
cbar=0;
e=0;
k=imsubtract(image(:,:,1),rgb2gray(image));
```

```
diff_im = medfilt2(k, [3 3]);      %Use a median filter to filter out noise
 % Convert the resulting grayscale image into a binary image.
   diff_im = im2bw(diff_im,0.18);%if illumination level>0.18 ==>1, otherwise..

diff_im = bwareaopen(diff_im,300);      % Remove all those pixels less than 300px

    % Label all the connected components in the image.
    I = bwlabel(diff_im, 8);% 4 or 8 connectivity

% Following are the steps For Detecting the red ball
se=strel('disk',5);%structuring element, shape=disk, radius=5
B=imopen(I,se);          %morphological open operation erosion followed by dilation
final=imclose(B,se);    %morphological close operation dilation followed by erosion
[L,n]=bwlabel(final);
imshow(L);

for k=1:n
[r,c]=find(L==k);
rbar=mean(r);
cbar=mean(c);
% Converting to decimal number
e=(((cbar>=x1)*2*2*2) + ((cbar<=x2)*2*2) + ((rbar>=y1)*2) + (rbar<=y2));
end
% Decision Making Conditions
switch (e)
case 5
disp('Move left');
 fwrite(ser,'L');
case 6
disp('Move left');
```

```
 fwrite(ser,'L');
case 7
disp('Move left');
 fwrite(ser,'L');
case 9
disp('Move right');
 fwrite(ser,'R');
case 10
disp('Move right');
 fwrite(ser,'R');
case 11
disp('Move right');
 fwrite(ser,'R');
case 13
disp('Move forward');
 fwrite(ser,'F');
case 14
disp('Move back');
 fwrite(ser,'B');
otherwise
disp('Stop Moving');
 fwrite(ser,'S');
end
end
fclose(ser);                % closing
delete(ser);                %delete object file
clear ser;                  % clearing s matrix from the workspace
```

CHAPTER 8

ARDUINO BOOTLOADER PROGRAM

```
int inbyte = 0; // incoming serial byte
void setup()
{
        // start serial port at 9600 bps;
        Serial.begin(9600);
        pinMode(13,OUTPUT);
        pinMode(12,OUTPUT);
        pinMode(11,OUTPUT);
        pinMode(10,OUTPUT);
}
void loop()
{
        //if we get a valid byte,read analog ins:
if (Serial.available() > 0)
{
        // get incoming byte:
        inbyte = Serial.read();
        if (inbyte == 'R')
        {
                digitalWrite(13,HIGH);
                digitalWrite(12,LOW);
                digitalWrite(11,LOW);
                digitalWrite(10,HIGH);
        }
        else if (inbyte == 'L')
        {
                digitalWrite(13,LOW);
                digitalWrite(12,HIGH);
```

```
                digitalWrite(11,HIGH);
                digitalWrite(10,LOW);
        }
        else if (inbyte == 'F')
        {
                digitalWrite(13,HIGH);
                digitalWrite(12,LOW);
                digitalWrite(11,HIGH);
                digitalWrite(10,LOW);
        }
        else if (inbyte == 'B')
        {
                digitalWrite(13,LOW);
                digitalWrite(12,HIGH);
                digitalWrite(11,LOW);
                digitalWrite(10,HIGH);
        }
        else if (inbyte == 'S')
        {
                digitalWrite(13,LOW);
                digitalWrite(12,LOW);
                digitalWrite(11,LOW);
                digitalWrite(10,LOW);
        }
}
}
```

CHAPTER 9

CONCLUSION

From this project we can conclude that we have used concepts of image processing and embedded systems for real time application. In this system, a single camera system is used as a sensor to perceive image and image processing methods are then performed to identify the location of the ball within the environment by calculating the centroid of the identified color intensity. With the help of only one sensor i.e. webcam we have successfully achieved our basic objective to catch the specified colored ball in the specified region.

CHAPTER 10

POSSIBLE EXTENTIONS

- Current prototype is not sufficiently big to cover larger area, so this can be rectified by using longer links provided we have good actuators to drive them.
- Using better actuators that are better than dc motor used since these have limited angular rotation and it very difficult to find a high torque motor with good performance and speed.
- Links can be fabricated using some other material which is sturdy enough and light in weight.
- The current prototype shows some performance issues while the ball is continuously moving. Actually the end effector many times lags behind the ball, some good solution to rectify this.
- The current image processing algorithm can modified to predict the future trajectory of ball.
- Same methodology can be implemented on the basis of shape detection algorithm. The robot can be used as an object sorting robot on basis of color and shape. Also the degrees of freedom can be increased for utilizing this concept in 3-D region.

BIBLIOGRAPHY:

1) Digital image processing -R. Gonzalez, R. Wood
2) www.mathwork.in/help/vision/example/ball-detection-and-tracking.html
3) en.wikipedia.com/wiki/Arduino