A Project Report on

Autonomous Path Planning Bot

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

MECHATRONICS ENGINEERING

Submitted By
Shail Shah (13012041110)
Mayur Kagathara (13012041040)
Dhruv Bhavsar (13012041009)

Guided By

Prof T.R.SHAH
Assistant Professor

Mechatronics Department
U. V. PATEL COLLEGE OF ENGINEERING
2016-2017

Department of Mechatronics Engineering
U. V. Patel College of Engineering
Ganpat University

CERTIFICATE

This is to certify that the work of User Defined Project entitled **"Autonomous Sweeper Bot"** has been carried out by **Shail Shah *(*13012041110*)*** in regards to the partial fulfillment for the requirement for the award of the degree of Bachelor of Technology in Mechatronics Engineering during the academic session 2016-2017 is a record of bona fide work carried out by him at the Department of Mechatronics Engineering, U. V. Patel College of Engineering, Ganpat University under my guidance and supervision. The work done here is a reflection of originality and transparency. The work incorporated in this thesis has not been, to the best of my knowledge, submitted to any other University or Institute for the award of any degree or diploma.

Prof. T.R.SHAH,
Assistant Professor
Mechatronics Department
U V Patel College of Engineering

Prof. J.P.Patel,
Head of Department,
Mechatronics Department
U V Patel College of Engineering

Internal Examiner: ____________________

External Examiner: ____________________

Department of Mechatronics Engineering

U. V. Patel College of Engineering

Ganpat University

CERTIFICATE

This is to certify that the work of User Defined Project entitled **"Autonomous Sweeper Bot"** has been carried out by **Mayur Kagathara (13012041040)** in regards to the partial fulfillment for the requirement for the award of the degree of Bachelor of Technology in Mechatronics Engineering during the academic session 2016-2017 is a record of bona fide work carried out by him at the Department of Mechatronics Engineering, U. V. Patel College of Engineering, Ganpat University under my guidance and supervision. The work done here is a reflection of originality and transparency. The work incorporated in this thesis has not been, to the best of my knowledge, submitted to any other University or Institute for the award of any degree or diploma.

Prof. T.R.SHAH,
Assistant Professor
Mechatronics Department
U V Patel College of Engineering

Prof. J.P.Patel,
Head of Department,
Mechatronics Department
U V Patel College of Engineering

Internal Examiner: ____________________

External Examiner: ____________________

Department of Mechatronics Engineering
U. V. Patel College of Engineering
Ganpat University

CERTIFICATE

This is to certify that the work of User Defined Project entitled **"Autonomous Sweeper Bot"** has been carried out by **Dhruv Bhavsar (13012041009)**in regards to the partial fulfillment for the requirement for the award of the degree of Bachelor of Technology in Mechatronics Engineering during the academic session 2016-2017 is a record of bona fide work carried out by him at the Department of Mechatronics Engineering, U. V. Patel College of Engineering, Ganpat University under my guidance and supervision. The work done here is a reflection of originality and transparency. The work incorporated in this thesis has not been, to the best of my knowledge, submitted to any other University or Institute for the award of any degree or diploma.

Prof. T.R.SHAH,
Assistant Professor
Mechatronics Department
U V Patel College of Engineering

Prof. J.P.Patel,
Head of Department,
Mechatronics Department
U V Patel College of Engineering

Internal Examiner: ____________________

External Examiner: ____________________

ABSTRACT

In this era of robots, the world leads to replacement of humans with robots. Robots can do work in better way than humans in the context of accuracy, precision, repeatability, efficiency et cetera. Robots can be static or dynamic. When bot is steady there is no issue of guiding it to and fro from one place to another but in case of moving robot the major issue is guiding the bot to specific location. Guidance is currently implemented by using sensors to interact with working environment or by predefined path on which robot can commute. Another alternative of guidance can be image processing, in this case the information of the destination and obstacles is extracted from the image captured in real time. Latest research is all about combining sensor data and image processing. This project is all about guidance system using digital image processing without use of any sensors. Using this obstacle avoidance path planning the project aims at sweeping the floor efficiently and with utmost accuracy.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Overview of Project

Guiding a robot is a tedious task especially when there is no predefined path and bot has to be moved from one arbitrary point to another. The use of sensors is feasible when bot is guided by any predefined physical mark leading to its destination. In this project the bot position and orientation is determined by image processing. Knowing position and orientation the bot is then guided to its destination. The project is divided into,

(1) Machine Vision
(2) Electronics Circuit

Machine vision is prime component of the project. Here the image processing is done using MATLAB software. Machine vision is used to determine the position and orientation of bot as well as real time location with respect to destination.

Electronics Circuit is based on open source environment Arduino Uno which is interfaced with Bluetooth module and servo motor. Entire bot's mobility and sweeping function using servo motor is controlled by programming done in the arduino IDE.

1.2 Scope of Project

Scope of Guidance system can be vast. When the bot is working in an environment without any physical guidance the handling is almost impossible, hardcode coding of sensors and programming board is required. In this situation solution is nothing but the guidance system developed in this project. Though this project can't be used directly in any real time practical problem as it is in development phase and requires system to avoid obstacles. This project is the base of guidance system development using image processing.

1.3 Purpose of Project

The purpose of the project is to develop a guidance system that effectively guides the bot from its position to destination with assumption of static environment in path.

Chapter 2 System Design

2.1 Objective

The prime objective is to develop a guidance system for bot to guide it from its position to destination. To achieve this, we have number of objectives to achieve which are:

1. Fabricate the bot
2. Communication between Matlab and Arduino
3. Develop program of matlab as well as arduino

2.2 Block Diagram

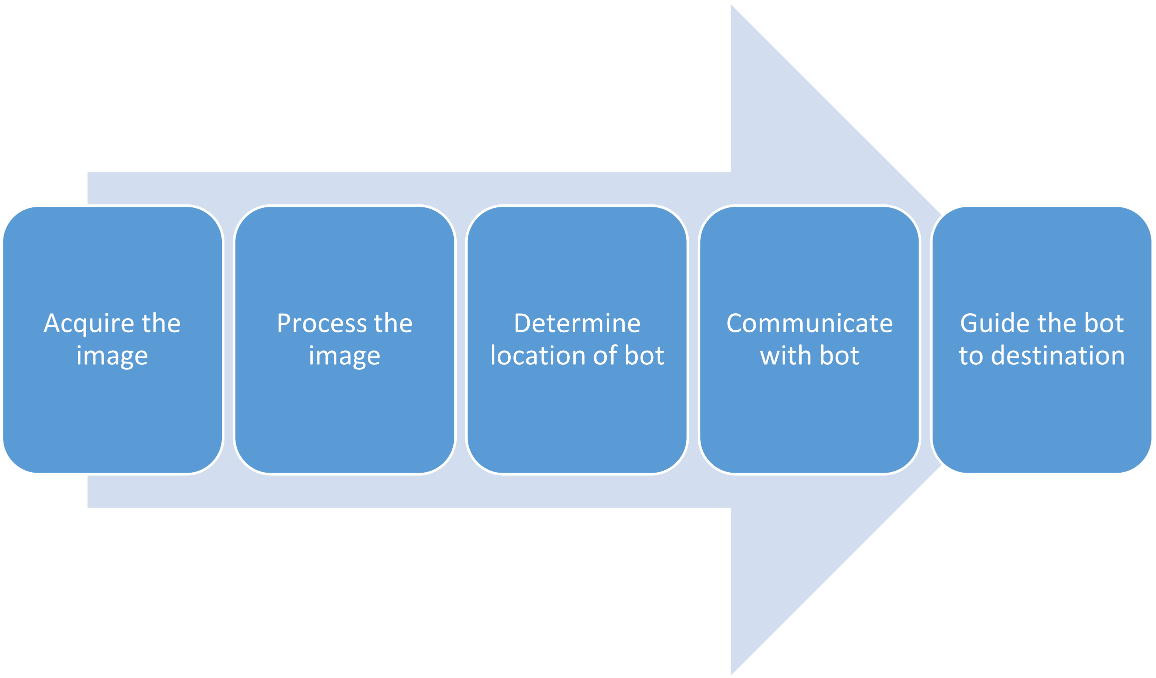


Figure 2.1 Block Diagram

2.3 Working

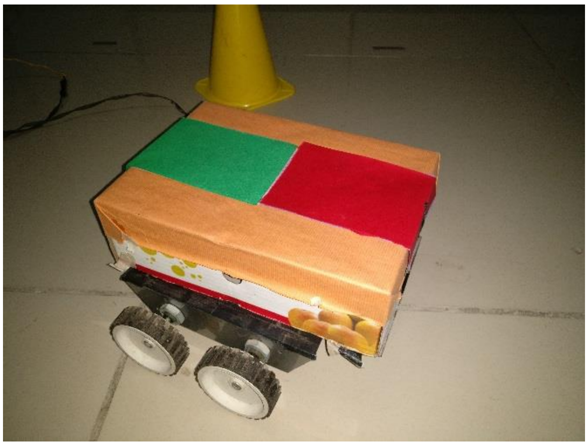
The entire working depends on two major parts of the project,
(1) Image processing
(2) Bot handling

The image processing comprise of several tasks. First of all it determines the position and orientation of bot as well as of destination. Then these are correlated to determine the bot position and orientation with respect to destination. Knowing this the program takes decisions regarding guidance of bot. Bot is first rotated towards the destination and then moved to it. During it also check the orientation. If it is not acceptable, it corrects it by rotation and then continue moving.

Bot is handled by codes sent to it via Bluetooth. Bot has controller which is interfaced to matlab. Matlab gives number of commands to bot to go to target point.

2.4 Working environment

Arena consists of uniform background in context of colour as well as structure. Bot has two colour marked body and destination has different colour. Camera is mounted above the arena not on the bot. As a camera IP cam or wired USB cam is used.



2.5 Circuit Diagram

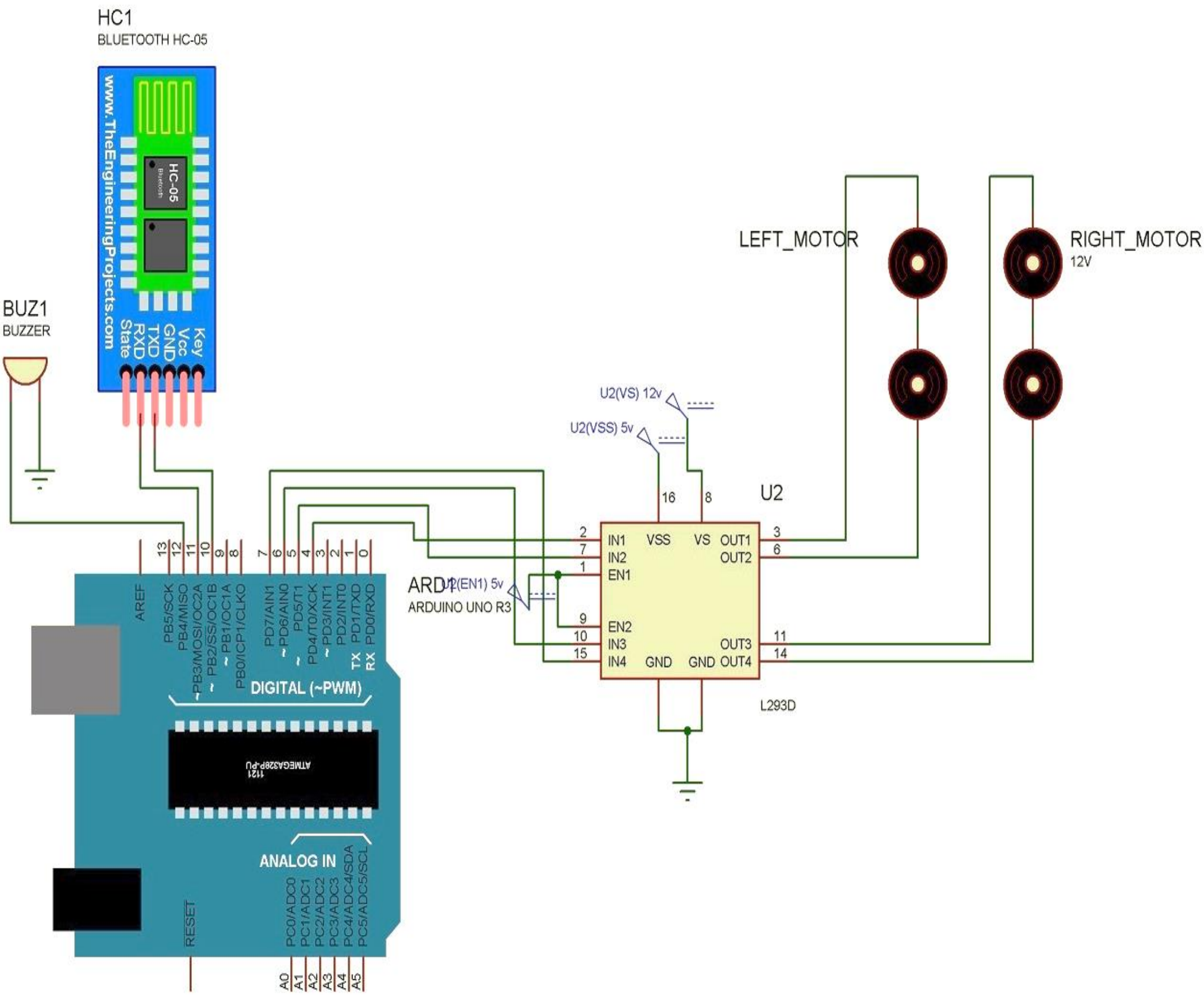


Figure 2.2 Circuit diagram

Chapter 3 Research and Experiments

3.1 Overview

The map is passed as information in form of image which is considered to occupy binary value at each grid. Value “1” indicates region of obstacle whereas value “0” indicates free space. In image form, black color indicates free space whereas white indicates region of obstacle. The obstacle can be of any shape. The first step of process is extracting nodes for all obstacles. For any shape of obstacle, the bounding box is generated first whose four indices may serve as nodes or via points. These nodes will also enable us to avoid obstacle while generating waveguide path. Fig. 1 illustrates result of nodes extraction. White four dots around circle are nodes created for circular obstacle. Similarly, four dots surrounding rectangular obstacle shall be observed. Next step involves database generation in which distance between each node to other node has been stored. These data will serve as primary inputs for all three optimization technique. The Euclidean distance between any two nodes is given by,

$$d(n1,n2)= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Where, d is distance between nodes n1 and n2 and tuples (xi, yi) indicates their co-ordinates from upper-left corner.

In addition to via points generated as nodes user is free to select source and destination nodes anywhere in an image map. Thus, the database will contain distance from one node to all other nodes. Total number of nodes will be given by,

$$n_n = 4n_o + 2$$

Where n_n is number of nodes generated and no is number of static obstacles present in map. Table 1 shows part of database created.

		Node j				
		1	2	3	4	5
Node i	1	Infinite	158.71	Infinite	176.55	39.56
	2	158.71	Infinite	73	335.25	197.14
	3	Infinite	73	Infinite	Infinite	Infinite
	4	176.55	335.25	Infinite	Infinite	139
	5	39.56	197.14	Infinite	139	Infinite

Table 3.1 Distance between nodes

Distance value for nodes which cannot be connected by line due to obstacle is considered infinite to show that those nodes cannot be connected directly. Similarly, considering the fact that current and next node cannot be same, distance between same nodes is also marked „infinite". This value causes cost function or heuristics function value very high which results in proper guidance for path planning. It shall also be observed that upper triangle and lower triangle of matrix are same which means that matrix is diagonally symmetric matrix.

After development of database matrix, next step involves development of Nearest Neighbor, Breadth First Search and Ant Colony Inspired Optimization algorithm for path planning. All three algorithms are briefly discussed here.

3.1.1 Breadth first search technique

Breadth first search technique is non-heuristic technique which searches for less number of nodes than for lesser distance. In this technique, all the possible solutions are explored simultaneously from given current nodes. If any of the explored nodes matches with destination node then process of exploring next level nodes stops and path is regained. If none of the explored nodes match with destination node then these nodes are explored simultaneously again. This process continues till either all solutions are checked or destination node is found. As it explores simultaneous solutions, there is guaranteed lesser number of stages if it exists. But, having lesser number of via nodes cannot guarantee optimal path.

3.1.2 Ant Colony Inspired Optimization algorithm

As already discussed, many researchers have used Ant Colony Optimization algorithm for mobile robots, logistics, and circuit testing etc. applications. Convergence of algorithm depends on many parameters like Ant move, pheromone update, varying control parameters etc. [2]. To overcome limitation of this parametric method, new non-parametric method is suggested in this literature. Effects of parameters are diminished by taking number of solutions in account. Based on probability theory, the number of solutions which may exist is given by,

$$n_p = \sum_{i=0}^{n} \frac{n!}{(n-i)!}$$

Where np indicates total number of paths, n is number of stages or number of via nodes that may be allowed. The number of ants are needed will be equal to number of paths. Assuming that at none of the instance, number of ants will be less than predefined value; ants are allowed to move in different direction in order simultaneously. Number of ants that will be sent to particular node from current node will be equal in all the directions which mean that probability of number ants‟ distribution will be equal for all possible next nodes. If any ant has reached to the goal position, then it will not travel further and wait for ants to complete their journey. Finally as all ants complete their journey, maximum up to predefined „n‟ stages, all ants will report distance travelled by each ant. The ant or group of ants which have reached to destination node and distance travelled by those ants is minimum compared to all other ants which have also reached to destination node, is considered winner. The path travelled by winner ants is optimum.

3.1.3 A^* Algorithm

A* is a computer algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently directed path between multiple points, called nodes. It enjoys widespread use due to its performance and accuracy. However, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, although other work has found A* to be superior to other approaches.

It solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance travelled, shortest time, etc.), and among these paths it first considers the ones that appear to lead most quickly to the solution. It is formulated in terms of weighted graphs: starting from a specific node of a graph, it constructs a tree of paths starting from that node, expanding paths one step at a time, until one of its paths ends at the predetermined goal node.

At each iteration of its main loop, A* needs to determine which of its partial paths to expand into one or more longer paths. It does so based on an estimate of the cost (total weight) still to go to the goal node. Specifically, A* selects the path that minimizes

$$\mathbf{f(n)=g(n)+h(n)}$$

where n is the last node on the path, g(n) is the cost of the path from the start node to n, and h(n) is a heuristic that estimates the cost of the cheapest path from n to the goal. The heuristic is problem-specific. For the algorithm to find the actual shortest path, the heuristic function must be admissible, meaning that it never overestimates the actual cost to get to the nearest goal node.

3.2 Comparison between algorithms

No.	BFS Dist	BFS Time(sec)	ACIO Dist	ACIO Time(sec)	A^* Dist	A^* Time(sec)	Min Dist	Min Time
1	583	4.4	634	56	521	1.8	A*	A*
2	654	4.4	576	45.6	616	1.3	ACIO	A*
3	288	5	396	64	290	1.5	BFS	A*

Table 3.2 Comparison between algorithms

The computational time depends on the processor of processing unit and may vary from one system to another.

Chapter 4 Conclusion & Future Work

4.1 Conclusion

The system works completely as expected. The image from IP cam is effortlessly acquired in matlab environment without any noticeable lag by wireless communication. The image is then processed in matlab with the algorithm which detects the bot position and displays its position and orientation. The resulting data such as rotation and movement is transferred to bot via Bluetooth communication. Robot moves according to the commands it received.

4.2 Applications

- Server room and data centers
- Go-downs
- Inventories
- Multi story building

4.3 Future Expansion

- Path planning with obstacle avoidance
- Guidance system for swarm robotics
- Inculcation with smart car

Appendix

ARDUINO UNO

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The **Uno** is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform.

Specifications:-

Arduino Microcontroller:-

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog I/O Pins	6
EEPROM	1 KB
DC Current per I/O Pins	40 mA on I/O Pins; 50 mA on 3.3 V Pin

General

Input Voltage	7-12 V
Digital I/O Pins	20 (of which 6 provide PWM output)
PWM Output	6
PCB Size	53.4 x 68.6 mm
Weight	25 g
Product Code	A000066 (TH); A000073 (SMD**)**

Bluetooth (HC05):-

It is a class-2 Bluetooth module with Serial Port Profile, which can configure as either Master or slave. A Drop-in replacement for wired serial connections, transparent usage. You can use it simply for a serial port replacement to establish connection between MCU, PC to your embedded project and etc.

HC-05 Specification:

- Bluetooth protocol: Bluetooth Specification v2.0+EDR
- Frequency: 2.4GHz ISM band
- Modulation: GFSK(Gaussian Frequency Shift Keying)
- Emission power: ≤4dBm, Class 2
- Sensitivity: ≤-84dBm at 0.1% BER
- Speed: Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps
- Security: Authentication and encryption
- Profiles: Bluetooth serial port
- Power supply: +3.3VDC 50mA
- Working temperature: -20 ~ +75Centigrade
- Dimension: 26.9mm x 13mm x 2.2 mm

Figure 4.1 HC-05

Application:

- Computer and peripheral devices
- GPS receiver
- Industrial control
- MCU projects

L293D Motor Driver

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.

L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

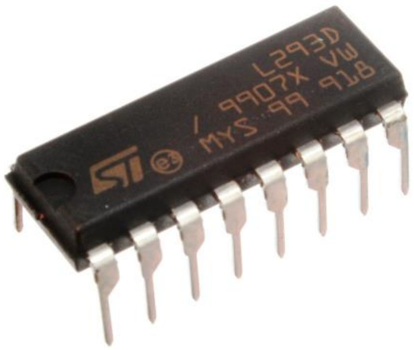


Figure 4.2 L293D

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

Pin Diagram:-

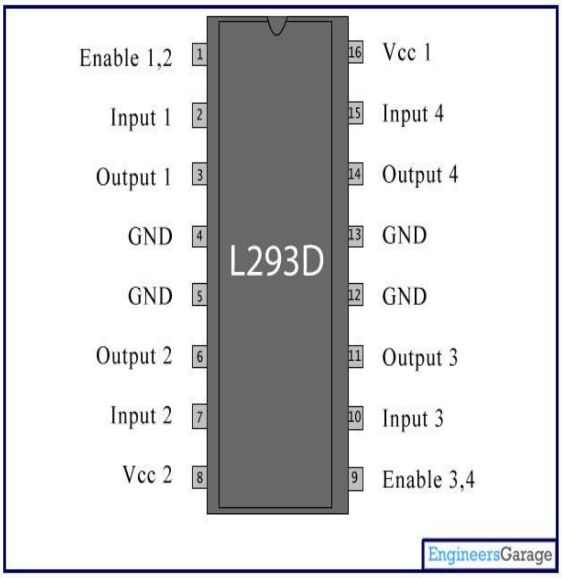


Figure 4.3 Pin Diagram of L293D

Pin Description:

Pin No	Function	Name
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	Vcc2
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V (up to 36V)	Vcc1

Table 4.1 Pin Description

Programs:

MATLAB parent program:

```
clear all
clc
global bt url
bt=Bluetooth('HC-05',1);
url = 'http://192.168.1.106:8080/shot.jpg';
roomclickimage = imread(url);
roomimage = im2bw(roomclickimage,0.1);
fclose(bt);
fopen(bt);
disp('bluetooth connected');

apoints = scanpoints(roomimage);

for i=1:apoints

    movebot(apoints(i,1),apoints(i,2));

end
fclose(bt);
```

Functions:

```
function mbot = movebot(xd,yd)
global bt url
timg = imread(url);
hold on
cgd = [xd yd];

flag=0;
k=0;
while 1
timg = imread(url);
ir = timg(:,:,1);
ig = timg(:,:,2);
ib = timg(:,:,3);
imb = ir>165 & ir<190 & ig>185 & ig<205 & ib>205 & ib<225 ;
imp = ir>200 & ir<225 & ig>140 & ig<155 & ib>180 & ib<200 ;
imp = bwconvhull(imp);
statsp = regionprops(imp);

imb = bwconvhull(imb);
statsb = regionprops(imb);


subplot 211
imshow(timg);
hold on
```

```
plot(cgd(1),cgd(2),'-y');
subplot 212
imshow(imb+imp+imd);
fwrite(bt,'5');
if size(statsp)>0 & size(statsb)>0
cgp=statsp(1).Centroid;
cgb=statsb(1).Centroid;
botpos = (cgp+cgb)/2;
a = cgb(1)-cgp(1);
b = cgp(2)-cgb(2);
inclin = atan2d(b,a);
if inclin<0
   inclin=abs(inclin)+180;
end

hold on
plot([statsp(1).Centroid(1),statsb(1).Centroid(1)],[statsp(1).Centroid(2),statsb(1).Centroid(2)],'-
y');

plot(cgp(1),cgp(2),'*r')
plot(cgb(1),cgb(2),'*b')

plot(botpos(1),botpos(2),'*k');
x=[botpos(1) cgd(1)];
y=[botpos(2) cgd(2)];
plot(x,y);
a1= cgd(1)-botpos(1);
b1 = botpos(2)-cgd(2);
inclin1 = atan2d(b1,a1);
if inclin1<0
   inclin1=abs(inclin1)+180;
end
dbetcg = abs(a1)+abs(b1);

plot([cgp(1),cgb(1)],[cgp(2),cgb(2)],'-y');

   if k~=1
      pause(0.2);
      k=k+1;
   else
      if flag==0
         if inclin<inclin1-5
           fwrite(bt,'4');
                 disp('anti CW');
           elseif inclin>inclin1+5
              fwrite(bt,'6');
                 disp('CW');
           else
              fwrite(bt,'5');
              flag=1;
         end
      end
```

```
        if flag==1
            if dbetcg>40 && abs(inclin-inclin1)<20
                fwrite(bt,'8');
                disp('forward');
            else
                flag=0;
                   if inclin<inclin1-5
                        fwrite(bt,'4');
                        disp('anti CW');
                   elseif inclin>inclin1+5
                       fwrite(bt,'6');
                       disp('CW');
                   else
                       fwrite(bt,'5');
                       flag=1;
                   end
            end
        end

       if dbetcg<45
            fwrite(bt,'5');
            break;
       end

    end
    pause(0.05);
end
end
end

function points = scanpoints(immg)

im1 = immg;
im1=rgb2gray(im1);
im1=im2bw(im1);
im1=1-im1;
[m,n]=size(im1);

[L, num] = bwlabel(im1, 4);
b=zeros(m,n);
im2=b;
im5=b;

for i=1:num
    [x,y]=find(L==i);
    x1=min(x)-5;
    y1=min(y)-5;
    x2=max(x)+5;
    y2=max(y)+5;

    im5(x1+5:x2-5,y1+5:y2-5)=1;
```

```
    im2(x1,y1)=1;
    im2(x2,y2)=1;
    im2(x1,y2)=1;
    im2(x2,y1)=1;
end
%
% [c1,r1,p1] = impixel(im1);
% [c2,r2,p2] = impixel(im1);
%
% % For manual source and destination selection.
c1=261;
r1=49;
c2=821;
r2=222;
cd=c2;
rd=r2;
close all

[im2L,num] = bwlabel(im2,4);
ns=num+2;

co_or=zeros(ns,2);
co_or(1,1)=r1;
co_or(1,2)=c1;
co_or(ns,1)=r2;
co_or(ns,2)=c2;

for i=2:ns-1;
    [x,y]=find(im2L==i-1);
    co_or(i,:)=[x,y];
end

dist=zeros(1,ns);
dist(:,:)=Inf;
im3=im1;
co_orr = co_or;
i=1;
r1=co_orr(i,1);
c1=co_orr(i,2);
k=2;
 next_coor(1,1)=co_orr(1,2);
 next_coor(1,2)=co_orr(1,1);
 co_orr(1,:)=[];
for l=1:ns-1
        dist1=inf;
for j=1:length(co_orr)
        im3(:,:)=0;

            r2=co_orr(j,1);
            c2=co_orr(j,2);

            m=(r2-r1)./(c2-c1);
```

```
            if abs(m)~=Inf
            co=r2-m*c2;
            if(abs(m)<=1)
            for j1=c1:sign(c2-c1):c2
                i1=round(m*j1+co+0.5);
                im3(i1-2:i1+2,j1-2:j1+2)=ones(5,5);
            end
            else
            for i1=r1:sign(r2-r1):r2
                j1=round(((i1-co)/m)+0.5);
                im3(i1-2:i1+2,j1-2:j1+2)=ones(5,5);
            end
            end
            im4 = im3 & im5;

            count1=sum(sum(im4));
            end
            if(count1==0)
                dist(i,j)=sqrt((rd-r2).^2 + (cd-c2).^2);
                if(dist(i,j)<dist1)
                    dist1=dist(i,j);
                    min_dist=dist(i,j);
                   rn=r2;
                   cn=c2;
                   jmin2=j;
                end
            end
            end
               next_coor(k,1)=cn;
               next_coor(k,2)=rn;
             if rn==rd && cn==cd
                 break;
             end
               co_orr(jmin2,:)=[]
              k=k+1;
              r1=rn;
              c1=cn;

end

imshow(im1)
hold on
for i=1:length(next_coor)-1
    plot([next_coor(i,1),next_coor(i+1,1)],[next_coor(i,2),next_coor(i+1,2)],'-r');
end

points = next_coor;
end
```

Arduino Program:

```
#include<SoftwareSerial.h>
char bt;
int p=100,q=100;
SoftwareSerial hc(10, 11);
void forward(void)
{
  analogWrite(3, p);
  analogWrite(5, 0);
  analogWrite(6, p);
  analogWrite(9, 0);
}
void reverse(void)
{
  analogWrite(3, 0);
  analogWrite(5, p);
  analogWrite(6, 0);
  analogWrite(9, p);
}
void stopit(void)
{
  analogWrite(3, 0);
  analogWrite(5, 0);
  analogWrite(6, 0);
  analogWrite(9, 0);
}
void left(void)
{
  analogWrite(3, q);
  analogWrite(5, 0);
  analogWrite(6, 0);
  analogWrite(9, q);
}
void softleft(void)
{
  analogWrite(3, q);
  analogWrite(5, 0);
  analogWrite(6, 0);
  analogWrite(9, 0);
}
void right(void)
{
  analogWrite(3, 0);
  analogWrite(5, q);
  analogWrite(6, q);
  analogWrite(9, 0);
}
void softright(void)
{
  analogWrite(3, 0);
```

```
  analogWrite(5, 0);
  analogWrite(6, q);
  analogWrite(9, 0);
}
void setup()
{
  Serial.begin(9600);
  hc.begin(9600);
  pinMode(3, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(9, OUTPUT);
  analogWrite(3, 0);
  analogWrite(5, 0);
  analogWrite(6, 0);
  analogWrite(9, 0);
}

void loop()
{
  bt = hc.read();
  Serial.println(bt);
  switch (bt)
  {
    case '8': forward();    break;
    case '2': reverse();    break;
    case '6': right();      break;
    case '4': left();       break;
    case '5': stopit();     break;
    case '7': softleft();   break;
    case '9': softright();  break;
  }
}
```

Bibliography:

1.Marco Dorigo, Mauro Birattari, and Thomas Stutzle (2006), “Ant colony optimization: Artificial ants as a computational intelligence technique.” IEEE Computational intelligence magazine. Pp. 28-39.

2. Joon-Woo Lee, Jeong-Jung Kim, Byoung-Suk Choi and Ju-Jang Lee (2008), “Improved ant colony optimization algorithm by potential field concept for optimal path planning”. IEEE-RAS International conference on humanoid robots. Pp. 662-667.

3.Farnood Gholami, M. J. Mahjoob (2007), “An investigation of parameters in ant colony optimization for a path optimization algorithm”. Proceedings of the IEEE International Conference on Mechatronics and Automation. Pp. 463-468.

4. Tolga Yuksel, Abdullah Sezgin. “An implementation of path planning algorithms for mobile robots on a grid based map”.