TITLE

The title of our project is, “Line Maze Solver”.

ACKNOWLEDGEMENT

We owe a great many thanks to a great many people who helped and supported us during this project.

Our deepest thanks to Department of Electronics and Computer Engineering, the department of Kathmandu Engineering College for guiding and correcting various mistakes of ours with attention and care. The department members have taken pain to go through the project and make necessary correction as and when needed.

We express our thanks to the Locus 2011 organizing committee of, Institute of Engineering, for selecting and providing us a platform to show our project entitled “LINE MAZE SOLVER”.

Our deep sense of gratitude to Toshak Raj Uprety (Project coordinator), Kathmandu Engineering College for support and guidance. Thanks and appreciation to the helpful people at Institute of Engineering, for their support.

We would also thank our colleague and lecturers without whom this project would have been a distant reality. We also extend our heartfelt thanks to our family and well wishers.

TABLE OF CONTENTS

ABSTRACT

This project will undertake the construction and implementation of a two-wheeled robot that is capable of solving a maze constructed as a line. The structural, mechanical, and electronic components of the bot will be assembled in a manner that will make the bot possible to solve the maze as quickly as possible.

The wheels of the robot are capable of independent rotation in two directions, driven by a motor IC circuit. Information about tracks, the dead ends and turns will be obtained from sensors on the device. The line of tracks will be determined by sending a Infrared signal to the track and photo-transistors will be used to sense the infrared signals. Information from the sensors will be fed back to the Arduino Duemilanove 328 having a ATMEGA 328P-PU micro-controller, which will convert them to digital values using ADC of the micro-controller, compare the result and generate output to the motor to keep it in track. To take the correct turns and to avoid dead ends, next time robot is operated, the memory of the micro-controller will also be used.

INTRODUCTION

A line maze is usually a black line on a white background. It could also be a white line on a black background. Each line maze has a Start point and a Finish point. There are a number of dead-end paths in the maze.

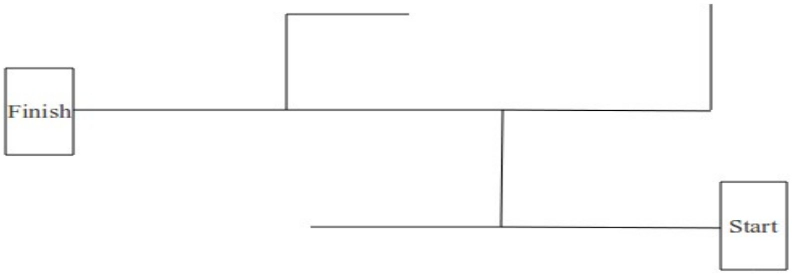


Fig: A Line Maze

A line maze solver is a robot that can solve the maze in fastest time possible. As the line maze contains many dead ends, the robot typically cannot traverse the maze without first taking a number of wrong turns.

.

OBJECTIVE

Autonomous robotics is a field with wide-reaching applications. From bombing robots to autonomous devices for finding humans in wreckage to home automation, many people are interested in low-power, high speed, reliable solutions. There are many problems facing such designs: unfamiliar terrain and Inaccurate sensing continues to plague designers. Automatic vehicles and other instruments always saves human effort and our time and if we build it properly. The systems will be controlled when we are out of reach if we build automatic vehicles or any other systems.

Sensing a line, maneuvering the robot to stay on course and solving a maze, while constantly correcting wrong moves using feedback mechanism forms a simple yet effective closed loop system and an effective autonomous robotics. As a programmer we get an opportunity to ‘teach’ the robot how to solve the maze thus giving it a human-like property of responding to stimuli.

SYSTEM BLOCK DIAGRAM

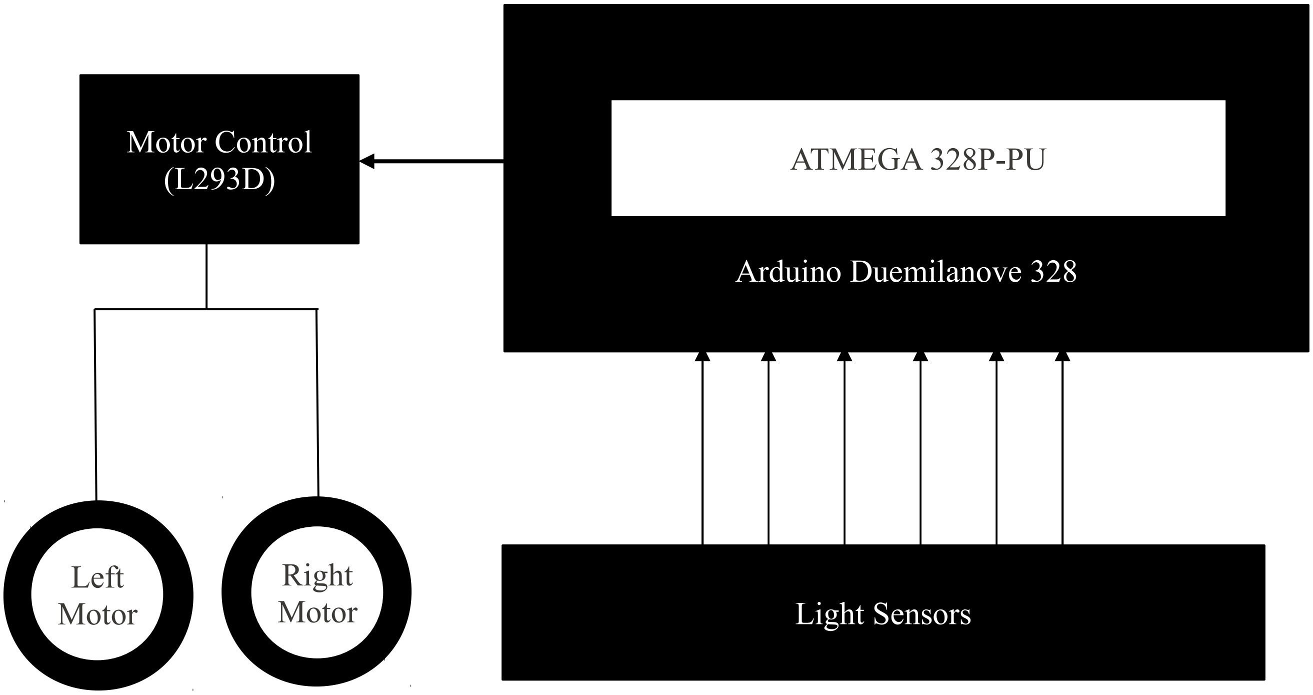


Fig: Hardware System Block Diagram

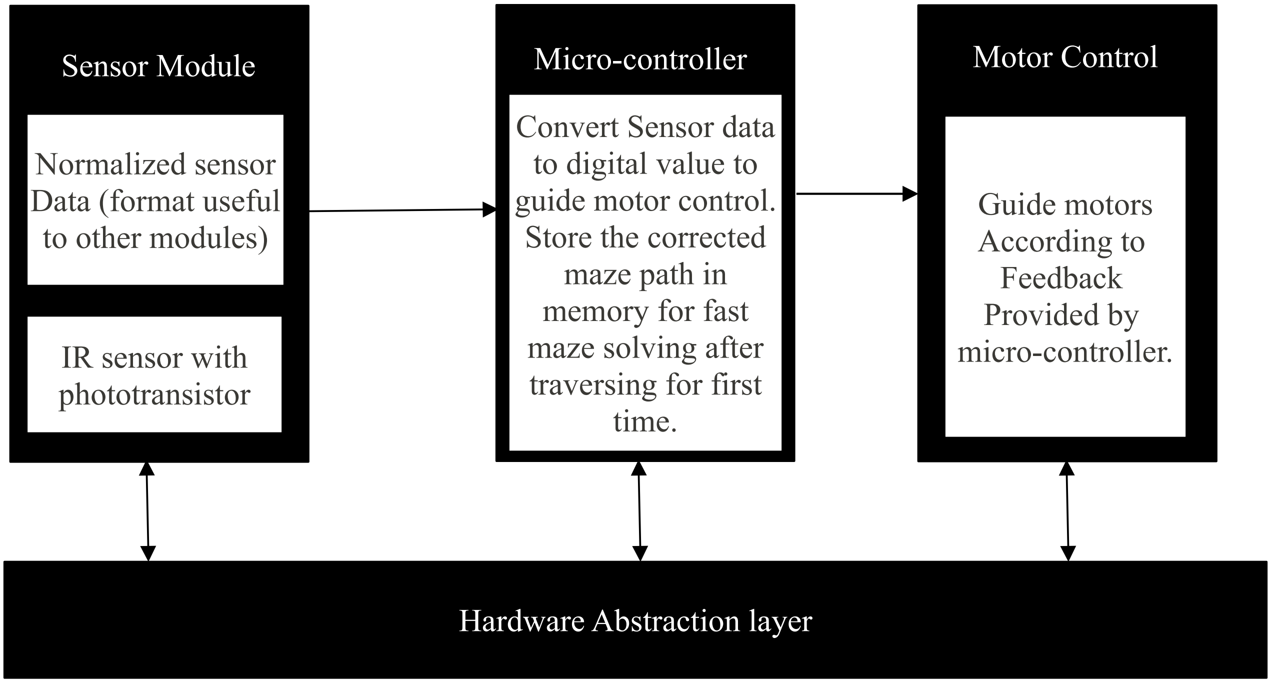
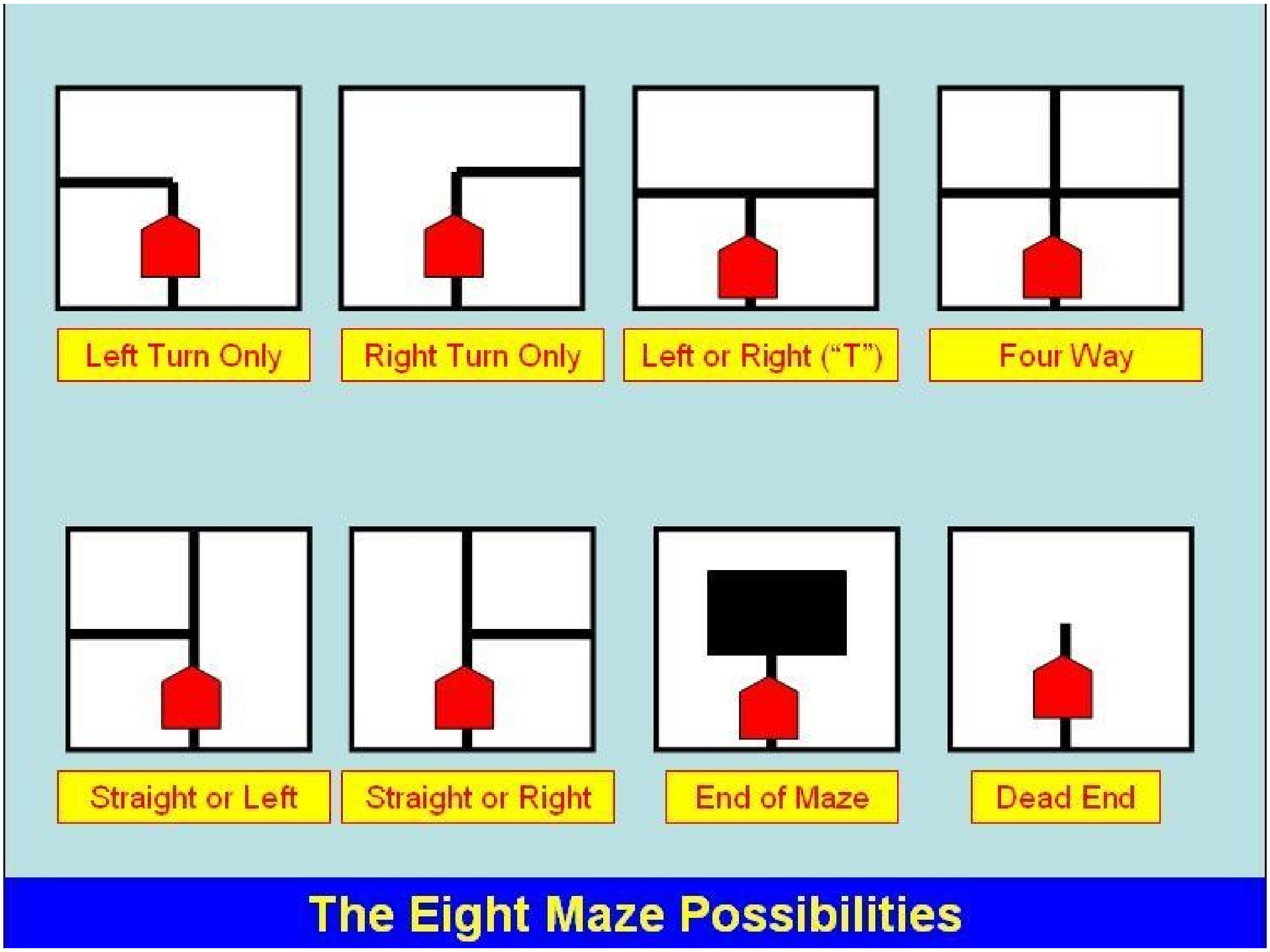

Sensor Module
Normalized sensor Data (format useful to other modules)
IR sensor with phototransistor
Micro-controller
Convert Sensor data to digital value to guide motor control. Store the corrected maze path in memory for fast maze solving after traversing for first time.
Motor Control
Guide motors According to Feedback Provided by micro-controller.
Hardware Abstraction layer

Fig: Software System Block Diagram

METHODOLOGY

The wall follower, the best-known rule for traversing mazes, is also known as either the *left-hand rule* or the *right-hand rule*. If the maze is *simply connected*, that is, all its walls are connected together or to the maze's outer boundary, then by keeping one hand in contact with one wall of the maze the player is guaranteed not to get lost and will reach a different exit if there is one; otherwise, he or she will return to the entrance.

The wall follower technique that we are going to implement is left-hand rule. As our maze contains no loop. The robot can encounter only eight different possibilities of situations:



As Arduino Duemilanove 328 has six analog input pins. We shall be using six sensors to detect the lines. The sensors arrangement will be:

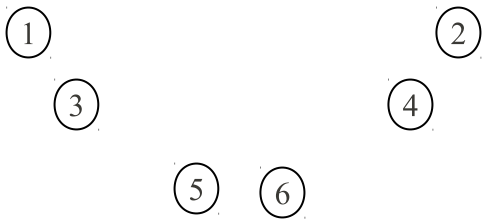


Fig: Sensor Placement

In the above figure we can see that the two sensors #1 and #2 give us an advance warning about any right or left turns we are going to encounter through this we can make out the correct decision timely. Our robot will always be in track with the remaining sensors.

As we have six sensors in placement, we can make 2^6=64 different combinations out of this combination with advance warning which helps to solve the maze very fast.

The robot, most of the time, will be involved in one of the following behaviors:

1. Following the line, looking for the next intersection.
2. At an intersection, deciding what type of intersection it is.
3. At an intersection, making a turn.

These steps continue looping over and over until the robot senses the end of the maze.

From the above figure of the eight situations, there is only one possible action.
This distinction needs to be made because later some of these will need to be stored by the robot and some won't.

- In the first two cases of left turn only and right turn only, robot has no choice but to make a 90 degree turn. As the robot has no other options available, these turns need not be stored in memory.

- In the case of dead end, robot has no other choice than to make a 180 degree turn or a U-turn. Since reaching a dead end means robot has made a bad move, this type of move should be stored in memory to correct it at next round.

- In the case of left turn only or, straight or left turn the robot should move a small inch forward and determine the correct move. Similar is in the case of right turn only or, straight or, right turn. This type of turn should be stored in memory.

- In the case of T turn, Four way and end the robot should go a inch forward and check the path. This type of turn should be stored in memory.

In order to solve the maze, the robot needs to traverse the maze twice. In the first run, it goes down some number of dead-ends, but records these as “bad” paths so that they can be avoided on the second run.

CIRCUIT DIAGRAMS

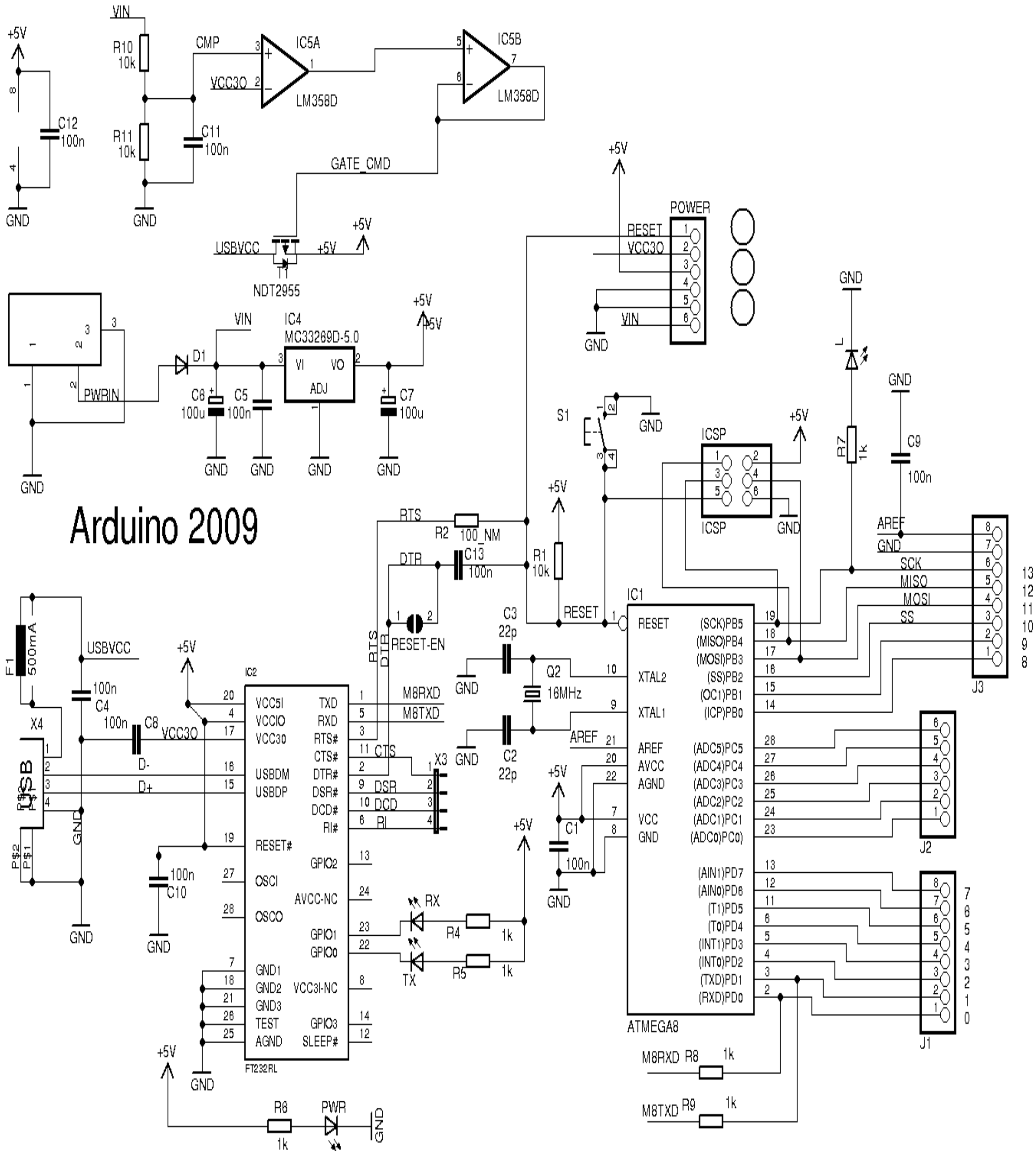


Fig: Arduino Duemilanove

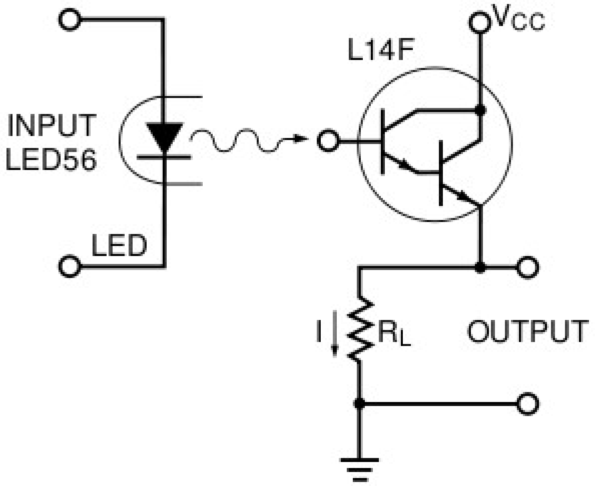


Fig: L14F1 Photo Darlington transistor

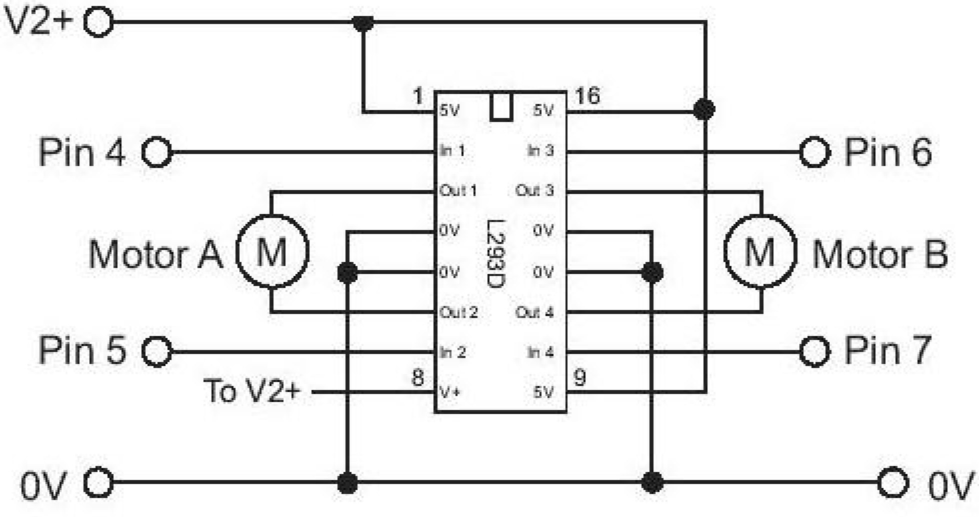


Fig: L293D motor Driver Circuit

APPLICATION

- To develop automatic robots
- To use robots in bombing
- To find humans in wreckage and to save them
- To extract radioactive elements from mines

COMPONENTS AND COST ESTIMATION

Components	Qty.	Cost(NRs.)
Arduino Duemilanove 328	1	3000
L293D IC	1	300
IR Transmitter	6	400
L14F1 Photo-transistor	6	1200
Gear Motors	2	800
Wheels	2	200
Robot Base	1	150
Jumper Wire	30	300
Resistor	20	200
Header	10	250
Connecting Wires	4 meter	50
Tracks	2 meter	100
LCD	1	1000
Miscellaneous		1500
	Total:	9450

RESULTS

The maze solver turns out to meet the specifications set by the sub goals. It is able to solve a maze of which it has no more information than that the track is in black and the background is white. In solving the maze using the technique of always following the left side of the track, it stores the essential information to find the exit the second time without detour. Hence the maze solver will find the accurate track and completes the maze of any type.

REFERENCES

http://en.wikipedia.org/
http://www.arduino.cc/
http://www.freewebs.com/isuru-c/motor_driver.htm
*courses.cit.cornell.edu/ee476/FinalProjects/s2006/nrs27th257/.../**L14F1**.pdf*
www.google.com
and many other reference books and materials.