Maze solver robot

ERTS course project

Team - 18

Bhuvan (49), Varun (63), Salman (64), Asok (72)

Table of contents

1. **Introduction**.

Mazes are one of the oldest puzzles known to man. Mazes have been built with walls and rooms, with hedges, turf, corn stalks, hay bales, books, paving stones of contrasting colours or designs, bricks and turf, or in fields of crops such as corn or, indeed, maize.Finding a route through the maze from the start to finish is Maze solving. Most maze solving methods are designed to be used inside the maze by a traveller with no prior knowledge of the maze.

With the inventions of computers and robotics, man has always tried to automatize different tasks and processes, starting from computing and calculating. The programming robots to solve puzzles and play games became prominent with idea of artificial intelligence.

This has inspired us to create a bot that solves mazes fast and efficiently.

2. **Problem Statement.**

The aim of the project is to design an autonomous robot that will solve the maze using three different algorithms. The three algorithms employed are the A-star algorithm, wall-follower algorithm and the tremaux algorithm.

The maze can be either a wall maze or a white/black line maze. The bot first explores the maze and reaches the end. Then it retraces the shortest path to the start.

A simulation of the bot solving the maze is created using Player-Stage.

3. **Requirements.**

I. **Hardware Requirements**

- Spark V Robot.
- A well designed maze.

II. **Software Requirements**

- AVR studio and Bootloader.
- Player-Stage Simulator.

4. **Implementation**.

Designing the Maze. The maze must be designed so that comparison between different algorithms will be visible. The end point of the maze should be designed so that the bot can differentiate between it and the rest of the maze.

Robot Motion. The bot must be moved so that it explores the entire maze and then it, retraces the shortest path. The bot should follow the black/white line all the time and do necessary actions when an intersection or the end point of the maze is approached.

Designing the algorithm. The algorithm should be efficient and it should be fast. Heuristics for A-star algorithm should be chosen wisely.

5. **Testing Strategy and Data**.

The sensors will sent the measured white line readings. The black line algorithm maintains the bot on the line. At the end of the maze the algorithms process the whole maze stored in the program in some structure to fond the optimal path. At the end of the maze the bot beeps. The LCD screen outputs the sensor values.

In the simulation, player-stage is run and the output window appears with the bot. [Images].

6. **Discussion of System**.

a. What worked as per plan?
    - The Line follower.

      It's important that the intersections are thicker as the bot has only three sensors, and is hard to distinguish between a line and an intersection.

    - Efficiency of the algorithm.

      The efficiency and response-time of the algorithm is very important to maintain the real time following of the line.

b. What we added?
    - The Simulation.

      We have also implemented a simulation of the robot in a computer based environment using Player-Stage software.

7. **Future Work.**

Maze-Solver can be used in many projects where exploration and path-finding is required like mine sweeper, fire engine model, search and rescue and the same.

8. **Conclusion.**

Like we said above, the code is written in three ways for each algorithm. One is pseudo code for human understanding, one is code to run on robot, and one is code to run on player-stage simulator.

The code is clearly divided into hardware dependent and independent parts, so that changing the platforms won’t be a big issue..