Design Project: Maze Solver Robot

1 PROJECT DESCRIPTION

Developmet of an autonomous mobile robot to participate in a maze solving race

The first phase is the learning of the maze, where robots need to follow a black track to learn the maze and find the shortest path. The second phase is a timed event where robots need to travel through the same maze on the shortest path.

To do this you will be required to use Matlab and Simulink for both simulation and solution implementation.

1.1 PHASE 1- LEARNING THE MAZE

In the maze learning phase, the robot must start at the push of an accessible button and indicate with an LED the mode of operation. The robot must then stop and indicate once it has learnt the maze. A solid black box indicates the end of the maze.

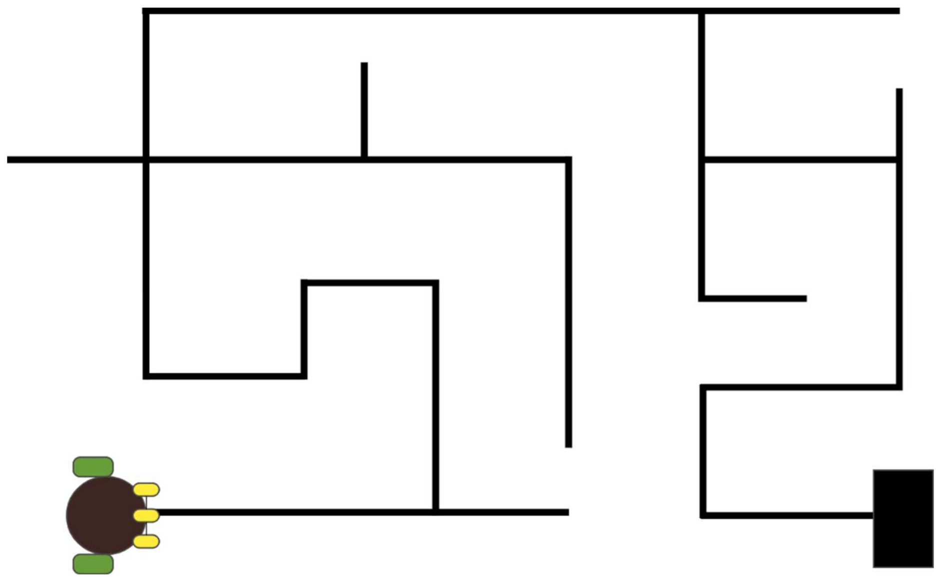

Figure 1 Sample Maze.

Specifications

- The robot must start with a button push and indicate(LED) that it is busy learning the maze.
- The robot must sense and follow a line
- The robot must implement a maze learning algorithm
- The robot must stop once maze learning completed
- The robot must indicate once it has arrived at the end with an indication (LED)
- The robot must sense
    - Dead-end
    - End of maze
    - Left T junction
    - Right T junction
    - Cross
    - T junction
    - Right Turn
    - Left Turn

1.2 PHASE 2 - OPTIMISE AND SOLVE THE MAZE

Finally, the robot must indicate with a blinking LED that it has found the shortest path from the beginning to the end of the maze.

Given the map, the robot needs to optimise the path and find the shortest path to the end of the maze. The robot needs to be prompted to optimise the path, and then once it is ready, it needs to indicate that it can now solve the maze using the optimised path.

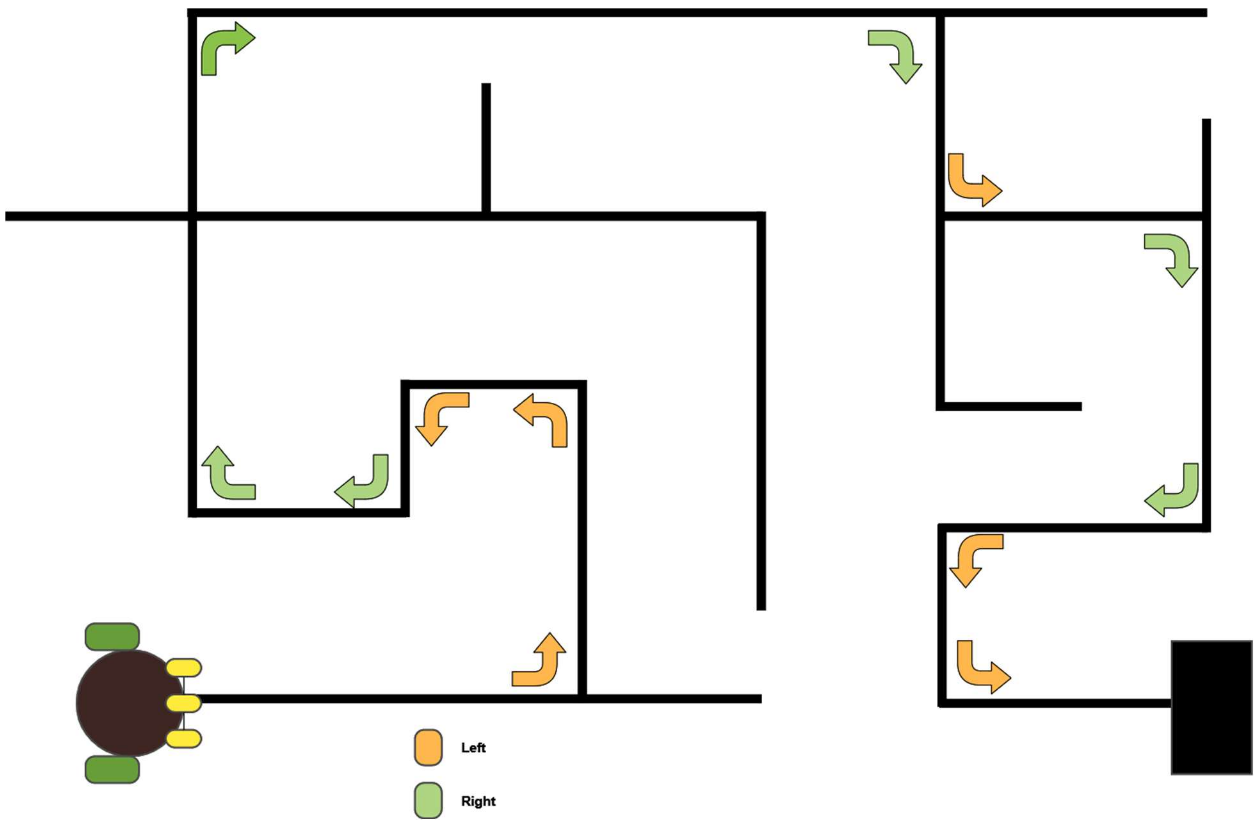

Figure 2 Optimised Path

Specifications

- The robot must optimise the path on a button push and indicate(LED) once it's ready to move through the shortest path.
- The robot must race through the maze using the optimised path and indicate(LED) after completing the timed race.

1.3 CONSTRAINTS

- A self-powered differential drive robot will be provided with the following
    - Romeo V2 (Compatible with Arduino) behaves like Arduino Leonardo based on the ATmega32u4 chip. https://www.dfrobot.com/product-844.html

- 1 x Turtle: 2WD Mobile Robot Platform. https://wiki.dfrobot.com/2WD_Mobile_Platform_for_Arduino__SKU_ROB0005_
    - Axle length: 13.6 cm
    - Wheel Diameter: 6.2 cm
- 4 x Gravity: Digital Line Tracking(Following) Sensor. https://wiki.dfrobot.com/Line_Tracking_Sensor_for_Arduino_V4_SKU_SEN0017
- 2 x Gravity: TT Motor Encoders Kit. https://www.dfrobot.com/wiki/index.php/Wheel_Encoders_for_DFRobot_3PA_and_4WD_Rovers_(SKU:SEN0038)
    - 20 ticks per rotation

- MATLAB and Simulink should be used for both simulation and real-world implementation.
- Arduino Support from Simulink (https://www.mathworks.com/hardware-support/arduino-simulink.html), should we used to test and run your models on the Romeo V2.
- To simulate the robot, the following tool kit should be used https://www.mathworks.com/matlabcentral/fileexchange/62966-student-competition-mobile-robotics-training

1. Project Milestone 1 – Motion Control Model (25%)

2 MATLAB AND SIMULINK SETUP

You may install Matlab on your personal computer (see http://www.icts.uct.ac.za/matlab). The university has a campus-wide license that makes this possible. You may also use Matlab online (see: https://www.mathworks.com/products/matlab-online.html).

Toolboxes you need on your local machine:

- MATLAB,
- Simulink
- Control System Toolbox
- Image Processing Toolbox
- Simulink Control Design
- System Identification Toolbox
- Image Processing Toolbox
- Robotics System Toolbox

3 MILESTONE2 - M OTION ONTROL

For the robot to complete both phases, it requires the ability to drive and steer. To do this, you need to understand the underlying robot kinematics equations that relate the speed of each wheel to the robot's velocity and direction.

To show your understanding of robot motion, you must demonstrate the following in both simulation and on the robot platform:

- The model can move the robot a given distance (i.e. 1 meter) and stop
- The model can rotate the robot given angles of 90, 180, 270 degrees and stop

You should firstly simulate the robot using the already available Mobile Robotic Training Library in Simulink. The base file for this will be provided. The library includes a Robot Simulator block, Encoder block, Motor block, Line sensor and other blocks you can use to create the simulation. The Motor block requires you to model the real-world motor, which is provided.

Once your robot can be controlled in simulation, you can use the same model to control the robotic platform. The robotic platform makes use of the RomeoV2, which is an equivalent of the Arduino Leonardo. Therefore, you can use the Simulink Support Package for Arduino Hardware Library to control the robotic platform.

Note that the start files will be provided (see the following link https://drive.matlab.com/sharing/ca84415f-3542-4352-ac3f-2d0865ab11ff).

6.1 RESOURCES

Differential drive: http://planning.cs.uiuc.edu/node659.html

Mobile Robotics Training: https://www.mathworks.com/videos/series/student-competition-mobile-robotics-training.html

Encoder Ticks: http://faculty.salina.k-state.edu/tim/robot_prog/MobileBot/Pose/encoders.html

RomeoV2: https://wiki.dfrobot.com/Romeo_V2-All_in_one_Controller__R3___SKU_DFR0225_

Dead reckoning lab:
http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/s07/labs/NXTLabs/Lab%203.html

Encoders:
https://www.youtube.com/watch?v=oLBYHbLO8W0&ab_channel=SparkFunElectronics