EEE3099S Design Project 2019: Maze Solver

1 Project Description

In groups of two, students must develop an autonomous mobile robot to participate in a maze solving operation.

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.

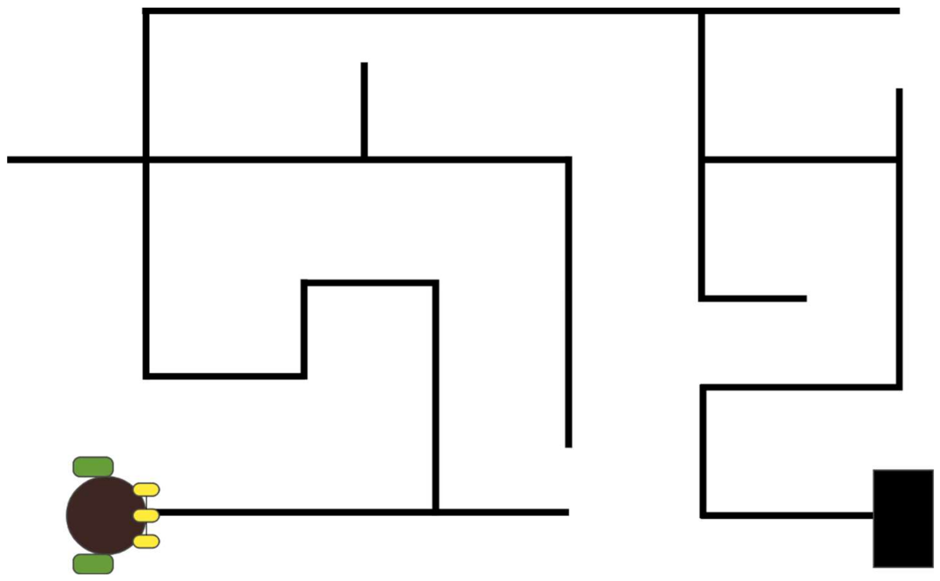


Figure 1 Sample Maze.

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.

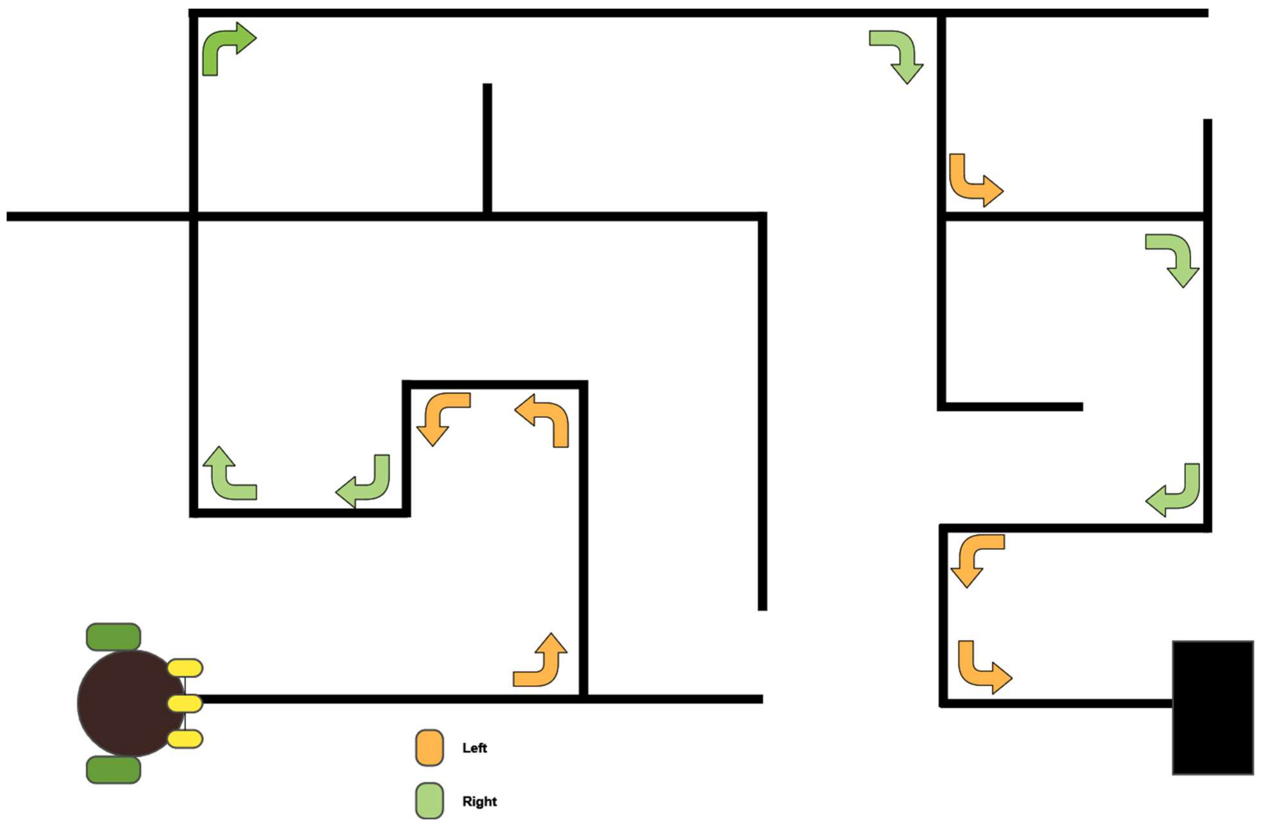


Figure 2 Optimised Path

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

2 MARKS

Calculation of the final mark for EEE3099S

1. Project Milestone 1 – Motion Control Model (25%)
2. Project Milestone 2 – Line Sensing (25%)
3. Project Milestone 3 – Final demo (25%)
4. Project Milestone 4 – Report (25%)

Simulink and Matlab Onramp are for DP only

- https://matlabacademy.mathworks.com/R2021a/portal.html?course=gettingstarted
- https://www.mathworks.com/learn/tutorials/simulink-onramp.html
- https://www.mathworks.com/learn/tutorials/stateflow-onramp.html
- https://www.mathworks.com/learn/tutorials/control-design-onramp-with-simulink.html

3 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

4 KEY DATES:

4.1 DEADLINES

- 20 August Simulink and Matlab Onramp certificate submissions
- 23 August Stateflow and Control Design Onramp with Simulink certificate submissions
- 30 August Milestone 1 Deadline
- 17 September Milestone 2 Deadline
- 18 October Milestone 3 Deadline
- 22 October Milestone 4 Deadline

5 PENALTIES

- Each day of late submission translates to a penalty of 5% (max 3 days).
- 10% will be deducted if a report includes any hand-drawn diagram or diagram that is not visible if printed.

6 MILESTONE 1 - MOTION CONTROL

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

6.2 Submission

1. A short report (7 pages maximum) containing how the different subsystems work and how you modelled them. See the rubric for details.
2. Four Simulink files per group one for each demonstration.
    a. Simulation
        i. The robot moves 1 meter (within 10% error)
        ii. The robot can turn 90, 180, 270 degrees (within 10% error)
    b. Real-world
        i. The robot moves 1 meter (within 10% error)
        ii. The robot can turn 90, 180, 270 degrees (within 10% error)

Please document the blocks/sections, and make use of subsystems where it makes sense. Annotate the units of each line (Volts, etc.)

6.3 Rubric

Group number		Mark	Subtotal
Report	Robot kinematic model and description		6
	Encoder model and description		6
	PWM explanation		6
	Distance Control Algorithm		6
	Angle Control Algorithm		6

Demo	Simulation	The robot moves 1 meter (within 10% error)		15
		The robot can turn 90, 180, 270 degrees (within 10% error)		15
	Real World	The robot moves 1 meter (within 10% error)		15
		The robot can turn 90, 180, 270 degrees (within 10% error)		15
Simulink Model Parameter correctness (Wheel Radius, Axle Length, Encoder Ticks per rotation, Sample Time)				5
Code and Model Neatness/Originality				5
Penalty				Penalty
Total				100

Table 1 Milestone 1 Rubric

7 MILESTONE 2 – LINE SENSING

Now that we can move the robot, you need to sense and follow the line describing the maze. In your design, four digital line sensors are provided to detect the presence of the black line. Furthermore, the robot is required to sense the line configurations outlined in Figure 3 Maze junctions.

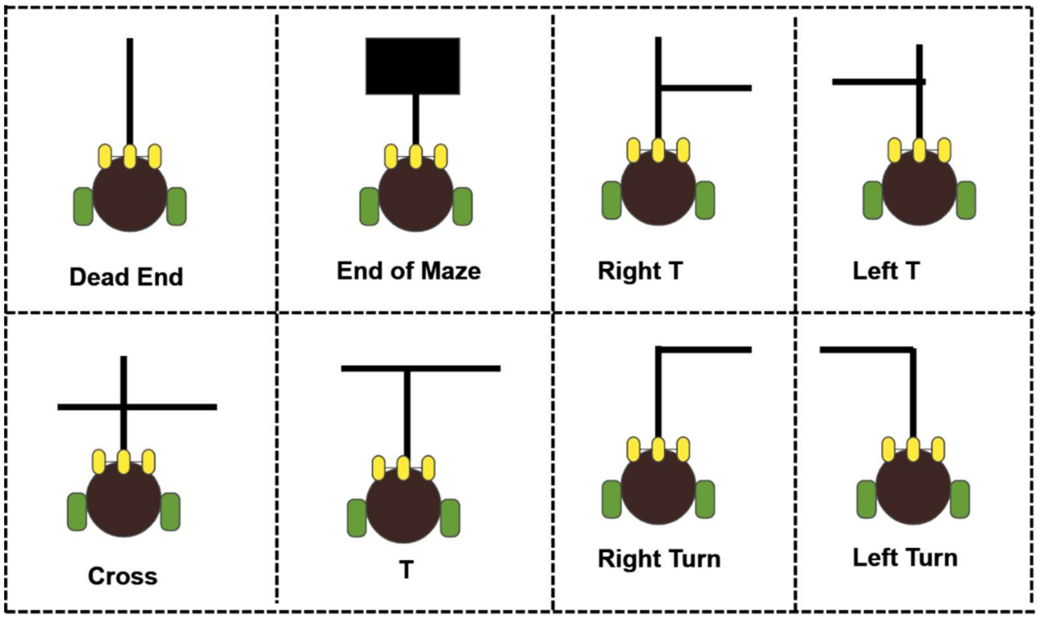


Figure 3 Maze junctions

You will be provided with 4 sensors positioned at (x: 64.3mm, y: 35.4mm), (x: 67.5mm, y: 5mm), (x: 67.5mm, y: (-)5mm) and (x: 64.3mm, y: (-)35.4mm) (see Figure 4 Differential-drive robot axisfor robot axis).

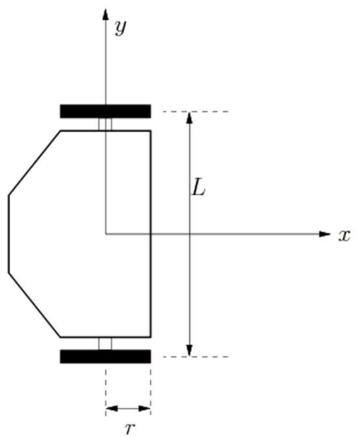
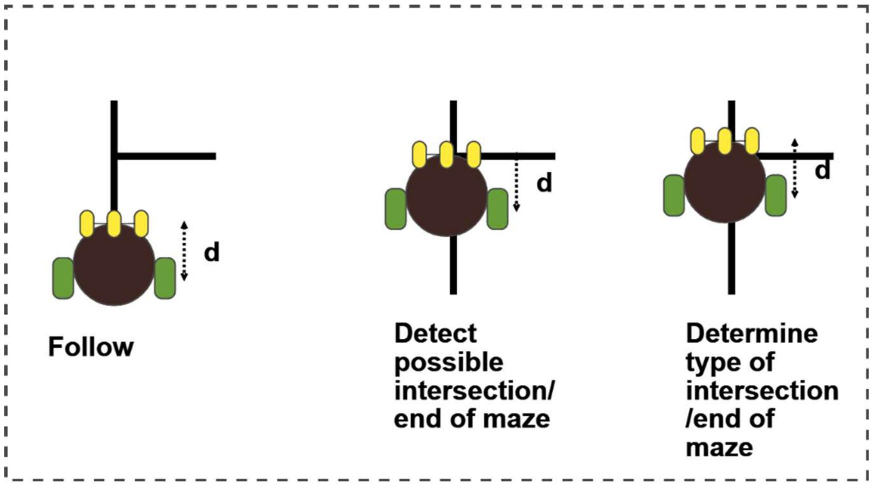


Figure 4 Differential-drive robot axis

To demonstrate that your robot can sense the different line configurations and follow a line, your robot is required to detect all the eight configurations in simulation and real-world. Move along an **18mm** line and stop when it detects one of the configurations (see Figure 5 Line detection). When the robot stops, it needs to indicate which configuration is detected.



d
Follow
d
Detect possible intersection/ end of maze
d
Determine type of intersection /end of maze

Figure 5 Line detection

Furthermore, on the real-world implementation, the robot needs to indicate the mode of operation on digital pin 13 LED and start following the line only when switch S1 on the RomeoV2 board has been pressed. The modes of operation are initial (LED off), following line (LED flashes at 1hz), possible intersection detected (LED flashes at 5hz), and line configuration detected (LED on). **Note that the start files will be provided (see the following link https://drive.matlab.com/sharing/49d4c7f0-bdfa-4173-8d7a-ee6da5d66596)**

7.1 Resources

Line Tracking in Simulink:
https://www.mathworks.com/help/supportpkg/legomindstormsev3/examples/line-tracking.html

Line Follower Application for Arduino® Robot:
https://www.mathworks.com/help/supportpkg/arduino/examples/arduino-robot-line-follower-application.html

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

7.2 Submission

1. Each group must submit a short report (7 pages maximum) describing how the different subsystems work and how you modelled them. See the rubric for details.
2. Each group must submit at least two Simulink, one for the simulation solution and the other for the real-world implementation.
    a. Simulation
        i. The robot follows the line
        ii. The robot identifies (line configuration), indicates (line configuration) and stops when it detects line configurations.
    b. Real-world
        i. The robot follows the line
        ii. The robot identifies (line configuration), indicates (line configuration) and stops when it detects line configurations.
        iii. The robot indicates the mode of operation on PIN 13 LED
        iv. The robot starts on the press of switch S1

Please document the blocks/sections, and make use of subsystems where it makes sense. Annotate the units of each line (Volts, etc.)

7.3 Rubric

Group number		Mark	Subtotal

Report	Line following algorithm			10
	Line configuration algorithm			10
Demo	Simulation	The robot follows the line		14
		The robot identifies (line configuration), indicates (line configuration) and stops when it detects line configurations.		16
	Real World	The robot follows the line		14
		The robot identifies (line configuration), indicates (line configuration) and stops when it detects line configurations.		16
		The robot indicates the mode of operation on PIN 13 LED		5
		The robot starts on the press of switch S1		5
Simulink Model Parameter correctness (Wheel Radius, Axle Length, Encoder Ticks per rotation, Sample Time)				5
Code and Model Neatness/Originality				5
Penalty				Penalty
Total				100

Table 2 Milestone 2 rubric

8 MILESTONE 3 – THE RACE

Now that you can control your robot platform and sense the operation environment, the robot can learn the maze and find and travel on the shortest path.

To learn the maze, an algorithm (wall following recommended) needs to be employed. The algorithm needs to find the end of the maze in the fastest time. Many maze learning algorithms exist, and you will need to employ one of them. Then once the robot has learned the maze, it needs to find and travel on the shortest path to the end of the maze.

Both the learning operations and travelling on shortest path operations will be timed. The time will be used to compute a mark relating to the fastest time record by any group in the class.

$$timed\ event\ mark = (t_f/t_r) * maximum\ mark$$

Where t_f is the fastest time and t_r is your recorded time. For example, if your group records 2 seconds and the fastest time is 1 second, your group will get 50% of the maximum mark possible.

Furthermore, on the real-world implementation, the robot needs to indicate the mode of operation on digital pin 13 LED and start learning the maze/start the race when switch S1/S2 is pressed. The modes of operation are described below.

Mode	**LED/Switch**
Initial	LED: OFF Switch: Start learning on the press of S1
Robot learning maze	LED: flashes at 1 Hz
Robot finished learning maze	LED: flashes at 2 Hz
Robot ready to race through the shortest path	LED: flashes at 5 Hz Switch: Start the race on the press of S2
Race complete	LED: On

Note that the start files will be provided (see the following link https://drive.matlab.com/sharing/96b05847-234e-41a2-b03d-98c68c5b45f0)

8.1 RESOURCES

- Maze Solving Algorithms: https://en.wikipedia.org/wiki/Maze-solving_algorithm
- https://medium.com/@TowardInfinity/coding-a-line-follower-robot-using-lsrb-and-finding-the-shortest-path-d906ffec71d

8.2 SUBMISSION

1. Short report (7 pages maximum) explaining the maze learning and shortest path algorithms

2. Simulink file for simulation and real-world demonstrations

8.3 RUBRIC

Group number			Mark	Subtotal
Report	Maze Learning Algorithm			5
	Shortest Path Algorithm			5
Demo	Simulation	Successfully learn the maze		10
		Stop when finished learning the maze at the end		5
		Shortest Path find and follow		10
	Real World	Successfully learn maze on the press of a button		10
		Stop when finished learning the maze at the end		5
		Shortest Path find and follow		10
		LED modes of operation		5
		The robot starts learning on the press of switch S1		5
		The robot starts the race on the press of switch S2		5

Timed Event Marks	Simulation	Timed Learning operation		5
		Timed Race through the maze		5
	Real World	Timed Learning operation		5
		Timed race through the maze		5
Code and Model Neatness/Originality				5
Penalty				Penalty
Total				100

Table 3 Milestone 3 rubric

9 MILESTONE 4 - REPORT

9.1 SUBMISSION

Group number		Mark	Subtotal
Report	Introduction		10
	Hardware description		20
	Motion control design, implementation, and results		20
	Line sensing design, implementation, and results		20
	Maze solver and shortest pathfinder algorithm design, implementation, and results		20
	Conclusion		10
Penalty			Penalty
Total			100

Table 4 Milestone 4 rubric