Lab: Navigation & Obstacle Avoidance

Submission instructions: Students are expected to work in their assigned lab groups. The lab consists of three components, demonstration, lab report, and code submission. Instructions on the demonstration can be found in this handout. Lab reports and code submission must follow the guidelines established in this handout and for the course. For more information, see the ECSE211SubmissionInstructions.pdf on MyCourses.

Design objectives

1. Design a system that allows the robot to drive to a specified set of coordinates, known as waypoints, while avoiding any obstacles in its path.
2. Implement the design using previous implementations of the Odometer.
3. Evaluate the design and ensure it can navigate effectively around a field.

Design requirements

The following design requirements must be met:

- The robot must navigate through a series of specified points using the minimal distance.
- An array of waypoints must be provided in one location in the code.
- The robot must use the grid system found on the play floor, where the origin is the starting corner of a tile.
- Waypoints will be given with respect to the tile grid system. For example, (1,0) will be located 30.48 cm to the right of the origin (0,0). *Note: waypoints can contain negative integers.*
- When turning to a waypoint, the robot must use the minimal angle needed to turn to it.
- In encountering an obstacle, the robot must avoid it and resume driving to the next waypoint.
- In avoiding an obstacle, the robot must continue to detect if there is another obstacle along its way and react accordingly.

Demonstration (30 points)

The design must satisfy the requirements by completing the demonstration outlined below.

Design presentation (10 points)

Before demoing the design, your group will be asked some questions for less than 5 minutes. You will present your design and answer questions designed to test your individual understanding of the lab concepts. Each person will be graded individually.

You must present your workflow, an overview of the hardware design, and an overview of the software functionality. Visualizing software with graphics such as flow charts is valuable.

Simple Navigation (10 points)

Starting from the (0,0) origin point on the 4x4 tile grid, the robot must travel through 5 waypoints. The TA will specify the waypoints during the demo. You will add them to your code and upload your solution to the robot. An example is shown below in **Figure 1** using a **minimal angle** needed to turn at each waypoint. A **maximal angle turn** should be avoided (check **FAQ**).

- **5 points** are given for turning using a **minimal angle** at all the waypoints. **0 points** are given if the robot turns using a **maximal** (not **minimal**) angle at one or more waypoints.
- **5 points** are given for reaching the last waypoint within an error tolerance of **3 cm** using a *Euclidean distance measure*. The *Euclidean distance error* **ϵ** is defined by the formula shown below, where **X** and **Y** represent the destination waypoint, and X_F and Y_F represent the final position of the robot. A penalty of **-1 point per cm** is used.

$$\epsilon = \sqrt{(X - X_F)^2 + (Y - Y_F)^2}$$

Hence, the following point grid is used:

- **[0, 3] cm → 5 points**
- **(3, 4] cm → 4 points**
- **(4, 5] cm → 3 points**
- **(5, 6] cm → 2 points**
- **(6, 7] cm → 1 point**
- **(7,∞) cm → 0 points**

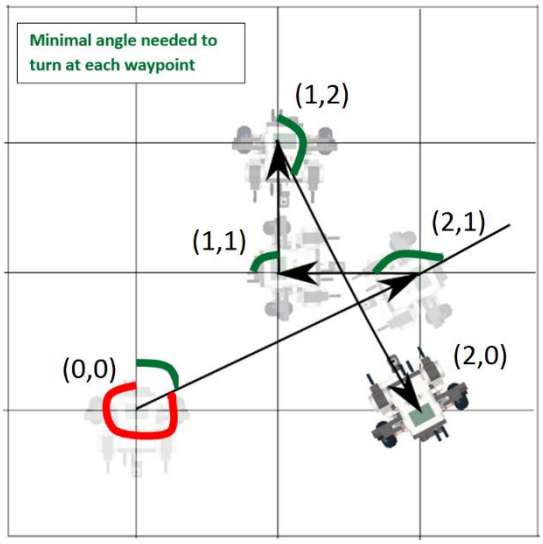

Figure 1. Example robot path 1 (labelled).

Navigation with Obstacle Avoidance (10 points)

Starting from the (0,0) origin point on the 4x4 tile grid, the robot must travel through the provided waypoints from the previous section. *Note that you are not allowed to reupload your code between the two demos!* At most two obstacles will be placed **ANYWHERE** along the robot's trajectory, but not on a waypoint. The robot must avoid the obstacle during navigation without touching it.

- **5 points** are awarded for completely avoiding obstacle(s).
- **5 points** are awarded for reaching the final waypoint within an error tolerance of **3 cm** using a *Euclidean distance measure* as in the previous demo. Hence, the following table is used:
  - **[0, 3] cm → 5 points**
  - **(3, 4] cm → 4 points**
  - **(4, 5] cm → 3 points**
  - **(5, 6] cm → 2 points**
  - **(6, 7] cm → 1 point**
  - **(7, ∞) cm → 0 points**

Provided materials

Sample code

No sample code is provided for this lab. Instead, follow the guidelines given below. Create a `Navigation` class that has the following methods:

- `void travelTo(double x, double y)`
  This method causes the robot to travel to the absolute field location (x, y), specified in tile points. This method should continuously call `turnTo(double theta)` and then set the motor speed to forward(straight). This will make sure that your heading is updated until you reach your exact goal. This method will poll the odometer for information.

- `void turnTo(double theta)`
  This method causes the robot to turn (on point) to the absolute heading theta. This method should turn a *MINIMAL* angle to its target.

- `boolean isNavigating()`
  This method returns true if another thread has called `travelTo()` or `turnTo()` and the method has yet to return; false otherwise.

Physical material

In the lab, there will be floors with the grid available, and blocks that will make up the obstacles.

Implementation instructions

1. Adjust the parameters of your controller, if any, to minimize the distance between the desired destination and the final position of the robot. Simultaneously, you should aim to minimize the number of oscillations made by the robot around its destination before stopping**.**
2. Write a program to travel through a set of waypoints. For example, **Figure 2** shown below follows (2,1), (1,1), (1,2), and (2,0) in order.
3. Modify your code to use the ultrasonic sensor to detect an obstacle (i.e. a block) and avoid collisions with it. You may mount the ultrasonic sensor as you wish, and may use the wall follower code if desired. Additionally, you may create another class to perform this function.
4. Modify the program to avoid any obstacles encountered along the robot's path. For example, if the robot was travelling from (0,2) to (2,0), it should avoid the obstacle along the path and still reach (2,0) as shown in **Figure 3** below.

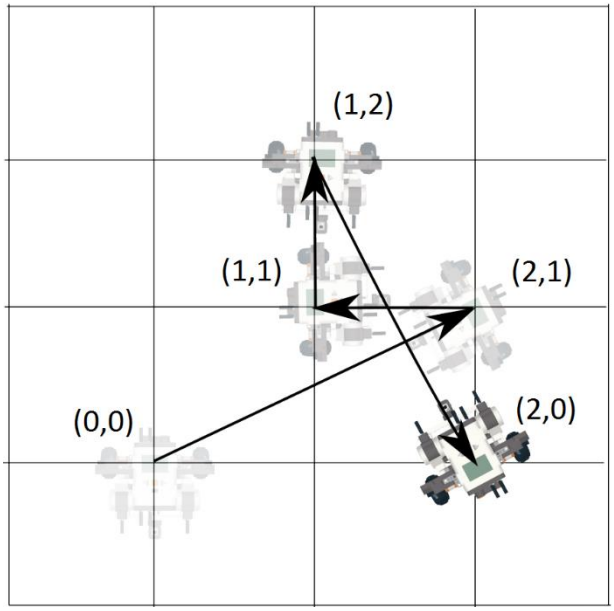

Figure 2. Example robot path 1.

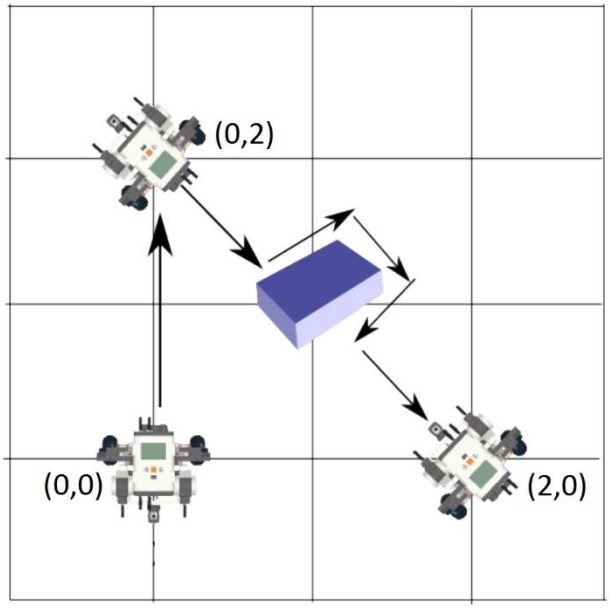

Figure 3. Example robot path 2.

Report Requirements

The following sections must be included in your report. Answer all questions in the lab report and copy them into your report. For more information, refer to ECSE211SubmissionInstructions.pdf.

Section 1: Design Evaluation

You should concisely explain the overall design of both controllers (software) and your robot (hardware). You must present your workflow, an overview of the hardware design, and an overview of the software functionality. Visualizing software with graphics is valuable.

Section 2: Test Data

This section describes what data must be collected to evaluate your design requirements. Collect the data using the methodology described below and present it in your report.

Navigation test (*10 independent trials*)

1. Program the robot to travel to waypoints (2,1), (1,1), (1,2), and (2,0) as in **Figure 2**.
2. Place the robot on the origin of a grid tile.
3. Ensure there are no obstacles in the robot's path.
4. Run the program to drive the robot through the waypoints specified in Step 1.
5. Record the final position of the robot:
    a. (**X**,**Y**), as reported by the odometer.
    b. (X_F,Y_F), as measured on the field.
6. Compute the error ϵ between the two measures.

Section 3: Test Analysis

1. Compute the mean and standard deviation for the errors measured in the **Navigation test**. Be sure to show general formulas and sample calculations.
2. Are the errors present because of the odometer or the navigator? Give reasons to back up your claim.

Section 4: Observations and Conclusions

- In three to four sentences, explain the operation of your controller for navigation.
- How accurately does it move the robot it its destination?
- How quickly does it settle (i.e. stop oscillating) on its destination?
- How would increasing the speed of the robot affect the accuracy of your navigation?
- What is the main source of error in navigation?

Section 5: Further Improvements

- What steps can be taken to reduce the errors in navigation and Odometry? In four to six sentences, identify at least one hardware and one software solutions and provide an explanation as to why they would work.

Frequently asked questions (FAQ)

1. **What is a minimal angle?**
   Referring to the (0,0) waypoint in **Figure 1**, its **minimal turning angle** is approximately 60° clockwise, as opposed to a **maximal turning angle** of 300° counterclockwise.

1. **What is meant by "design presentation"?**
   Before a lab demo, you and your partner will briefly present your design. This can include a basic visualization of how your code functions, such as a flow chart. You will then be asked a series of questions. These could be related to the lab tutorial and the initial lab code. For this part, a grade of 10 signifies full understanding, 5 signifies satisfactory understanding, while 0 shows no understanding at all. Note that memorized answers are discouraged and both partners should be responsible for understanding the design and their associated code, even if one of them did not write all of it.