Lab 2: Odometry

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 Lab Submission Instructions on MyCourses.

Design objectives

1. Design and implement an Odometry system that provides a robot's position and orientation, allowing it to autonomously navigate a field.
2. Evaluate the design and determine the accuracy of the implemented Odometry system.

Design requirements

The following design requirements must be met by your robot:

Odometer:
- Must determine the robot's **X**, **Y** and **θ** orientation.
- Must display the **X**, **Y** and **θ** on the LCD display.
- **X** and **Y** must be in cm.
- **X** and **Y** can be negative.
- **θ** must be in degrees.
- **θ** must range from $[0°, 359.9°]$:
  - When the value increases past $360°$, it should return to $0°$.
  - When the value decreases past $0°$, it should wrap to $359.9°$

- The zero values for (**X**, **Y**), i.e. (0,0), must respect the convention shown in Figure 3.

Demonstration (30 points)

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

Note: For this lab, you will only have 3 chances to demonstrate your work to the TA.

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.

Float Motors (10 points)

The TA will check whether the **X**, **Y** and **θ** values are updated correctly on the robot's LCD screen by floating the robot's motors/wheels.

All three axes (**X**, **Y**, **θ**) are checked:

- **X** & **Y** values work   → **5 points**
- **θ** values work   → **5 points**

E.g. if the (**X**, **Y**, **θ**) convention is set as in Figure 1, then:

- Moving both wheels **forward** should increase **Y**.
- Moving both wheels **backward** should decrease **Y**.
- Moving the right wheel **backward** and the left wheel **forward** simultaneously should increase **θ**.
- Moving the right wheel **forward** and the left wheel **backward** simultaneously should decrease **θ**.

Figure 1: Robot faces north at 0º.

E.g. if the (**X**, **Y**, **θ**) convention is set as in Figure 2, then:

- Moving both wheels **forward** should increase **X**.
- Moving both wheels **backward** should decrease **X**.
- Moving the right wheel **backward** and the left wheel **forward** simultaneously should increase **θ**.
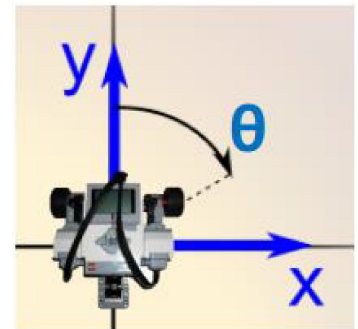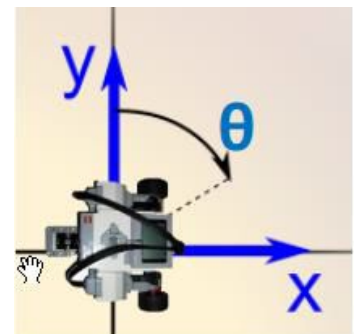- Moving the right wheel **forward** and the left wheel **backward** simultaneously should decrease **θ**.

Figure 2: Robot faces east at 90º.

Odometry Check (10 points)

The TA will ask you to run your robot off the center of a tile, as shown by **S** in Figure 3. The robot should then follow the 3-by-3 tile square trajectory using `SquareDriver`. The robot should work using odometry. Throughout the demo, the TA will observe the reported (**X**, **Y**, **θ**) values on the robot's LCD screen. When the robot stops at the final position (**X**F,**Y**F) near **S**, the final readings on the LCD screen (**X**, **Y**, **θ**) are used to evaluate the odometer's accuracy and calculate the **error distance ϵ** as:

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

Note that the **error ϵ** is calculated as the Euclidean distance between:

- The odometer's readings (**X**,**Y**), which signifies where the robot thinks it is with respect to the origin **(0,0)**, and
- The final actual position (**X**F,**Y**F), which ideally should be the point **S** where the robot started the 3-by-3 tile square trajectory.
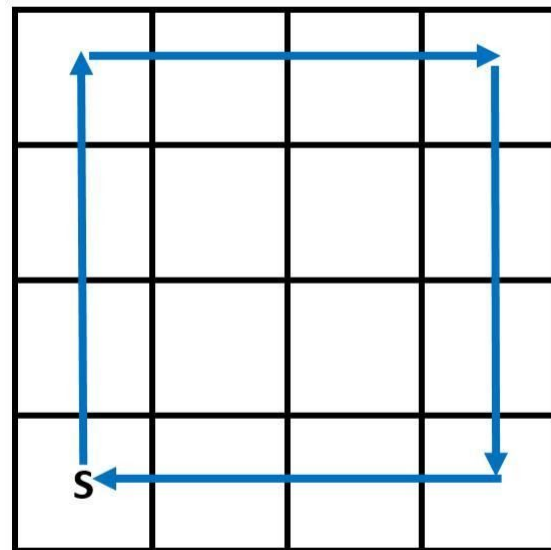
This means that it is **not an issue** if your robot does not return to the *exact* starting point **S**, as long as the odometer reports a position that **matches** its real-world location.

Point grid based on **error ϵ**:
   [0, 3] cm → **5 points**
   (3, 6] cm → **2.5 points**
   (6, ∞) cm → **0 points**

Point grid based on the difference between the displayed **θ** and actual **θ**:
   [0, 15] ° → **5 points**
   (15, 30] ° → **2.5 points**
   (30, ∞) ° → **0 points**

(0, 0)
Figure 3. 3-by-3 tile trajectory using
`SquareDriver`.

Provided materials

Physical material
In the lab, tiles with black grid lines are provided. These make up the competition floor, where the robot will operate. Grid lines are separated by a distance of 30.48cm.

Sample code
A package of sample code is provided that contains the following:

- **Display.java**
  - Provides a display mechanism for the Odometer.
  - Runs in a thread

- **Main.java**
  - The main class that runs the robot.
  - Starts Odometer thread and Odometer correction thread.
  - Drives the robot using `SquareDriver.java`.

- **Odometer.java**
  - A skeleton class for building an odometer.
  - Runs in a thread.
  - Provides methods for manipulation of volatile odometer variables, such as X and Y position.

- **Resources.java**
  - Defines the ports used by motors and sensors.

- **SquareDriver.java**
  - Runs the robot in a 3-by-3 tile square, where one tile is 30.48cm.

Implementation instructions

1. In `Odometer.java`, implement your odometer design in the `run()` method of the Odometer class. This class is threaded and will run continuously when your robot is working.
2. In `Resources.java`, tweak the values of `WHEEL_RAD` and `BASE_WIDTH` so that your robot drives in a square pattern when calling `SquareDriver.drive()`.

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 the Lab Submission Instructions. Always provide justifications and explanations for all your answers.

Section 1: Design Evaluation

You should concisely explain the overall design of your software and hardware. You must present your workflow, an overview of the hardware design, and an overview of the software functionality. You must briefly talk about your design choices before arriving at your final design. Visualizing hardware and software with graphics (i.e. flowcharts, class diagrams) must be shown. The design evaluation section is expected to be within **half a page** (excluding graphics).

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.

Odometer test (*10 independent trials*)
1. Note the starting position **S** of the robot's center and consider it to be **(0,0)** for this trial.
2. Run the robot in a 3-by-3 square using `SquareDriver.java`.
3. Measure its resulting signed X_F and Y_F position with respect to its starting position **S**.
4. Note the reported values of **X** and **Y** shown for the odometer.

Section 3: Test Analysis

Present the following analysis in a table in your report
1. Compute the Euclidean error distance ϵ of the position for each test.
2. Compute the mean and standard deviation for **X**, **Y**, and ϵ. That means, you need to perform 3 mean and 3 standard deviation calculations in total. Use the sample standard deviation formula. Show one sample calculation for both mean and standard deviation formulas.
.

Answer the following questions in your report

1. What does the standard deviation of X, Y and ϵ tell you about the accuracy of the odometer? What causes changes in standard deviation?

2. What is the sampling frequency of your odometer (i.e. the frequency at which the tacho count is measured)? What is the tradeoff of having a high sampling frequency versus a low sampling frequency?

Section 4: Observations and Conclusions

- Is the error you observed in the odometer tolerable for larger distances? What happens if the robot travels 5 times the 3-by-3 grid's distance?
- Do you expect the odometer's error to grow linearly with respect to travel distance? Why?

Section 5: Further Improvements

- Propose a means of reducing the slip of the robot's wheels using software.
- Propose a means of improving the accuracy of the odometer using one or more light sensors.

Frequently asked questions (FAQ)

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 partial 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.

2. **Are partial points awarded for Float Motors?**
    No partial points are awarded. Possible demo points: {0, 5, 10}.

3. **What is the length of each square tile?**
    Each tile is 30.48 cm (1 ft) long.

4. **Do the displayed values of θ have to be in ° (degrees)?**
    Yes.

5. **Do I have to follow the same (X, Y, θ) convention as in Figure 1 and Figure 2?**
    Yes.

6. **In the Float Motors part, what will be the initial conditions of the robot?**
    The robot's initial condition is based on your (X, Y, θ) convention - you must inform the TA about your initial orientation. Once the float motors option is selected on the robot, the TA will place your robot on a table. Assuming the same convention as in Figure 1, the TA will forcefully rotate both the wheels forward to test the Y values. Not that the robot will not move automatically, but rather the TA will rotate both wheels using his/her hands. The TA wants to notice whether the Y-value will increase or not, without affecting the X and θ values by much (since they could be affected due to experimental error). Therefore, the odometer's accuracy is irrelevant here. The main idea is to observe whether the odometer functions well using a fixed convention. A similar wheel-rotation test is also performed for checking decreasing values as well as for other variables.

7. **How accurate should the (X, Y, θ) values be during the floating motor demo? Should the values only increase and decrease properly without considering the errors?**
    Accuracy of (X, Y, θ) values in floating motor demo is irrelevant. However, let us consider an example when a robot that is oriented along the +Y-axis and both wheels are moved forward. If the reported Y-values are negative, this means that the Y-axis does not work.