ZHANNA KLIMANOVA AND SANSITHA PANCHADSARAM
GROUP 33

LAB 2: MEASURING THE DISTANCE TRAVELED BY A TWO-WHEELED ROBOT

ECSE 211 DESIGN PRINCIPLES AND METHODS
FALL 2020

Section 1: Design Evaluation

- **Present your workflow**

Initially, the hardware design and the software functionality of the robot were implemented independently by each lab partner in order to thoroughly comprehend the lab criteria and individually gain an understanding of the design process. If, at any point during the lab one of the partners encountered difficulties, collaborative problem solving would take place. When one of the lab partners successfully met a design criterion, their hardware and software design were used to complete the lab requirements.

- **An overview of the hardware design**

The hardware design for this laboratory is very similar to the one used for the first laboratory. The only difference is that the mounting parts of the ultrasonic sensor were changed to provide better support. The ultrasonic sensor was kept because it will be used in future laboratories.

- **An overview of the software functionality**

As shown in Fig.1, the Main class starts three threads (main, odometer, and square driver). It performs a few physics steps before starting the threads to assure that everything is initialized. After starting the threads, the program executes the Webots simulation until it is stopped or closed.

The Odometer class sets the starting position of the robot and increments the current x, y, and theta values using the delta changes calculated with motor techtometer counts.

As shown in Fig. 2, the Resources class provides all the constants needed for program execution.

The SquareDriver class drives the robot in a square of a specified Length x Width grid distance. The class implements this movement with a method that commands the robot to drive straight, turn 90 degrees at corners, and complete a full lap around the world before returning to its starting position.

Figure 1: odometry robot execution flowchart

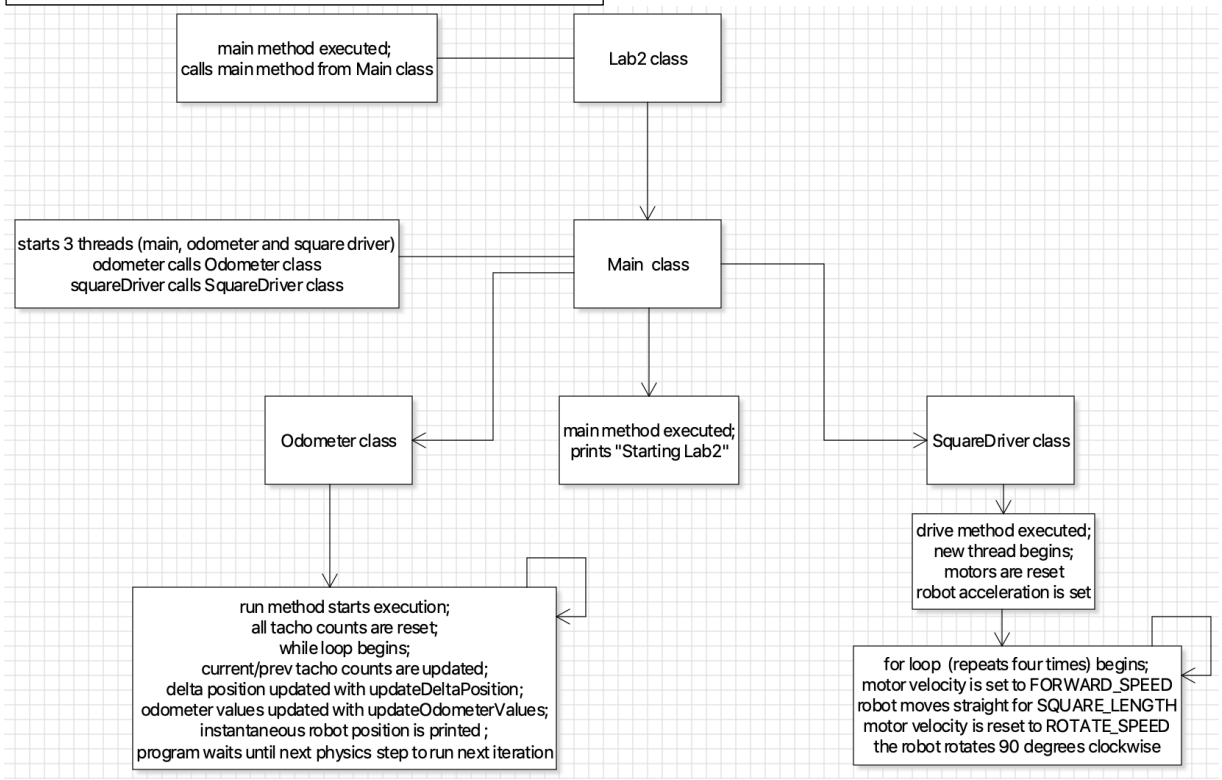
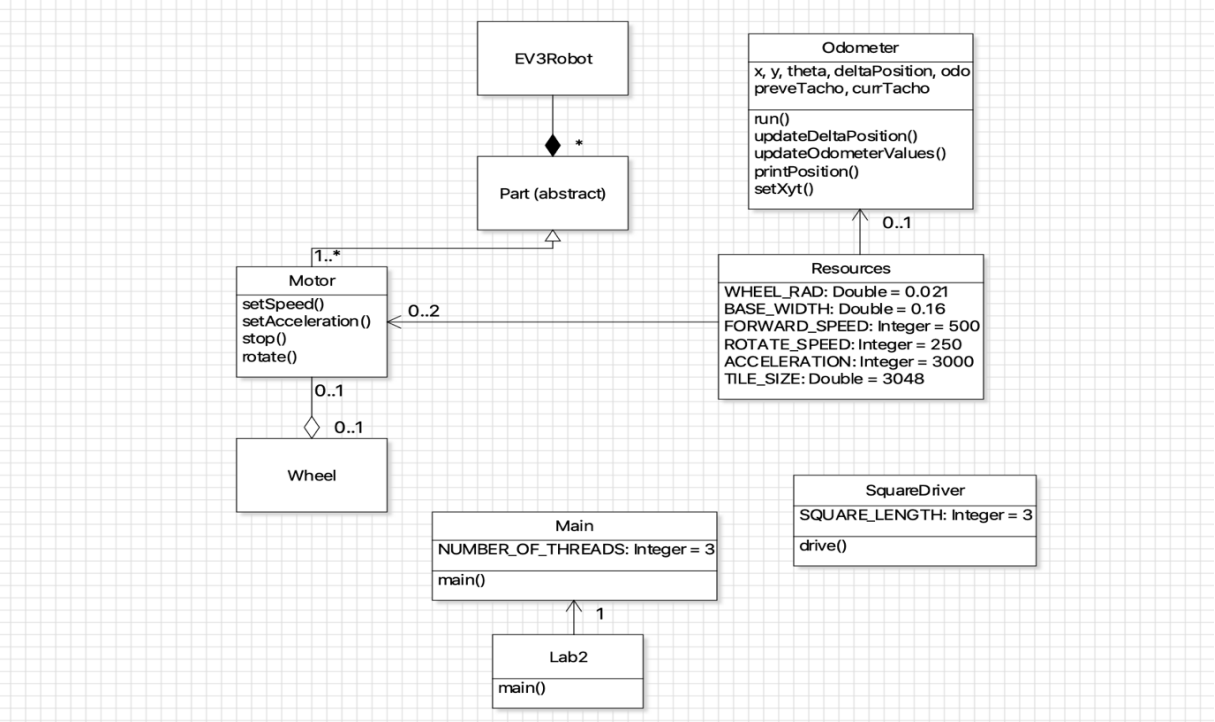


Figure 2: odometry robot class diagram



Section 2 and 3: Test Data and Analysis

Odometer test (5 independent trials): trials were conducted by varying the base width parameter of the robot. The table below displays the numerical outcome of each trial and the observations made. Important: x-axis in Webots is y-axis in real world; z-axis in Webots is x-axis in real world.

Base Width in m	Start Position (X) in m	Start Position (Y) in m	End Position (X_F) in m	End Position (Y_F) in m	Odometer (X_F) in m	Odometer (Y_F) in m	Euclidian Distance Error	Notes
0.145	0.172	0.153	0.444	0.049	0	0	0.291	Wall collision. Diagonal movement.
0.150	0.172	0.153	0.297	0.075	0	0	0.147	Tilted. No collision.
0.155	0.172	0.153	0.209	0.126	0	0	0.046	No collision w/ wall.
0.160	0.172	0.153	0.112	0.218	0	0	0.089	Tilted. No collision.
0.165	0.172	0.153	0.092	0.256	0	0	0.132	Diagonal movement. No collision Robot moved back.

Sample Calculations

Mean Xf is the mean for the DPM-Markers values

$$Mean\ X_F = \frac{0.444 + 0.297 + 0.209 + 0.112 + 0.092}{5} = 0.231\ meters$$

Mean X is the mean for the Odometer output values which were all 0's for the 5 trials

$$Mean\ X = \frac{0 + 0 + 0 + 0 + 0}{5} = 0\ meter$$

Standard Deviation Xf is for the DPM-Markers values

$$Standard\ Deviation\ X_F = \sqrt{\frac{(0.444-0.231)^2 + (0.297-0.231)^2 + (0.209-0.231)^2 + (0.112-0.231)^2 + (0.0924-0.231)^2}{4}} = 0.145\ \text{meters}$$

Standard Deviation X is for the Odometer output values which were all 0's for all 5 trials

$$Standard\ Deviation\ X = \sqrt{\frac{(0-0)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2}{4}} = 0\ meter$$

Mean Y_F = 0.145 m (mean for the DPM-Markers values)
Mean Y = 0 m (mean for the Odometer output values which were all 0's)
Standard Deviation Y_F = 0.090 m (SD for the DPM-Markers values)
Standard Deviation Y = 0 m
Mean Euclidian Distance Error = 0.141 m (varied robot base width)
Mean Euclidian Distance Error2 = 0 m (robot base kept at 16 cm; unvaried)
Standard Deviation Euclidian Distance Error = 0.093 m (varied robot base width)
Standard Deviation Euclidian Distance Error2 = 0 m (robot base kept at 16 cm; unvaried)

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

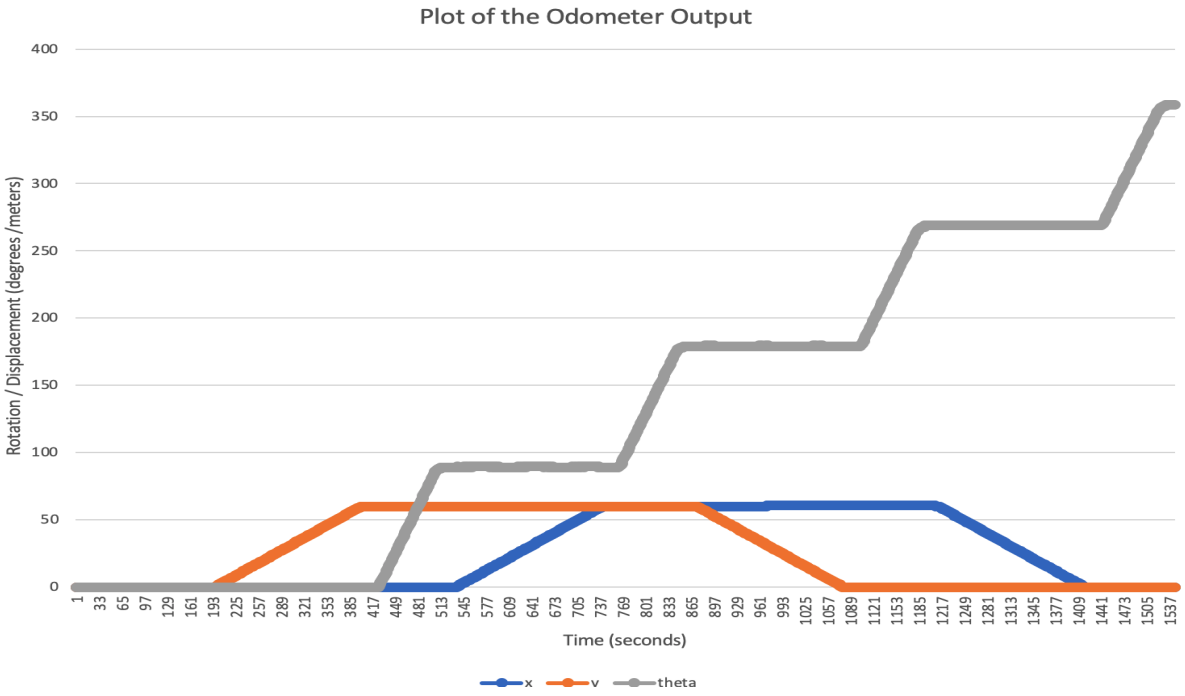
When the 5 trials are performed while keeping all other parameters constant, the standard deviation of X, Y, and ε is 0 m. This result occurs because the robot starts and ends at the same position throughout the trials. However, when the 5 trials are performed while varying the robot base width parameter, the standard deviation increases to 0.144 m for X_F, 0.0897 for Y_F, and 0.093 for ε_F; where X_F, Y_F, and ε_F stand for the final position of the robot when the base width parameter was varied. Standard deviation for X_F is greater compared to Y_F and ε_F, which means that the final X position of the robot was different from its initial X position. Standard deviation is the measure of how far each observed value is from the mean; in the case of X_F, Y_F, and ε_F, they all end up being slightly off from the mean at the end of each trial.

- **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?**

The sampling frequency of the odometer is the measurement of how fast the tachometer retrieves readings from its instantaneous wheel orientation. These samples are used to determine the displacement between the two wheels by finding the difference between the last and current tachometer readings. The tradeoff of having a high versus low sampling frequency is that higher sampling frequencies allow for much smaller displacement values which can be set equal to the arc length in the trigonometric calculations. The

arc length is approximated to be the difference between wheel displacements, and if the difference is too large, the arc length cannot be considered approximately to equal to this value. Higher sampling frequency is also especially useful in very jagged trajectories, where the higher sampling rate permits the robot to traverse angled areas more accurately.

- **BONUS: Plot the values of x, y, and theta and discuss these plots (errors and their potential causes, and contrast with how you think an actual robot would perform).**



The plot is a good replication of the robot movement in simulation:

1. When the y is increasing, the x is constant.
2. When x is increasing, the y is constant.
3. The theta is always increasing as the robot turns by 90 degrees four times.

Although the plot of x, y, and theta is accurate with respect to the simulation, it cannot be representation of reality due to several factors. For example, in the physical world there is slip between the robot wheels and floor, which will introduce error into the tachometer reading process. Moreover, the slip may also be different between each wheel; as the distance from the starting point to the ending point gets longer, the more the odometer readings will drift off from the reality. The other reason why the plot is not a good representation of reality is because the x, y, and theta values are calculated using double and float arithmetic. In the physical world, the EV3 does not have enough processing power to work with more accurate values. Therefore, the plot that will result from the physical world x, y, theta values will be more rounded off and less precise compared to that of the simulation.

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? (Change the DPM-Floor dimensions to the appropriate values to answer this.) Do you expect the odometer's error to grow linearly with respect to travel distance? Why?**

When the DPM-Floor dimensions are changed to a 15x15 grid distance, the robot does not compete its trajectory and runs into a wall at a location significantly far off from the starting point. Although the algorithm works correctly (the robot goes forward and turns 90 degrees when it is supposed to), the outcome of this trial shows that the dimensions appropriate for a 3x3 grid distance are not applicable for a 15x15 grid distance and must be adjusted accordingly. Therefore, the error observed throughout the 3x3 grid distance trials is not applicable to the 15x15 grid distance trial; the odometer's error does grow with respect to travel distance, but it does not necessarily have a linear trend.

- **What is the value in writing helper methods?**

Helper methods allows programmers to reuse a method in other methods or other parts of a program. The use of helper methods reduces the chance of having errors or bugs in code. They also increase the readability and decrease the overall complexity of the program by shortening lines of code that are used frequently.

- **How did you test your logic for the odometer and square driver?**

When Webots printed unexpected values of x, y and theta to the console, the updateDeltaPosition and updateOdometerValues methods in the Odometer class were reviewed because those were the main methods that calculated and updated the x, y and theta values. The other methods were getters and setters, which did not have an influence on the accuracy of the program.

When the robot was not turning at 90 degrees or moving straight, the moveStraightFor, turnBy, convertDistance, and convertAngle methods were modified accordingly. The convertDistance and convertAngle methods were more important to check for bugs since they performed the calculations to convert the distance and the angle into wheel rotations. The moveStraightFor and turnBy methods used these helper methods to move the robot straight and to turn it by a specific angle. Therefore, convertDistance and convertAngle had a significant influence on the robot movement and needed correct implementation.

- **In which order did you implement your classes and why?**

The SquareDriver class was implemented first because this class drives the robot on the demo floor. If the robot does not move, the Odometer class would not be able to update the robot's position and it would continuously print the starting position of the robot. After successful implementation of SquareDriver and verification that the robot was capable of completing consistent laps around the perimeter of the grid distance, the Odometer class was implemented.

Section 5: Further Improvements

• Propose a means of improving the accuracy of the odometer using one or more light sensors. Briefly discuss both hardware and software aspects.

To improve the accuracy of the odometer, two light sensors would be placed at an equal distance at the front of the robot for the purposes of detecting grid lines while the robot is in motion. A light sensor detects a line by observing a transition from bright to dark. To verify light sensor accuracy, the sampling rate must be at least one sample for every W units, where W represents the width of the grid line. Hence, when one of the light sensors finds a grid line, the distance traveled by the robot would return the wheel rotations until the second light sensor finds a grid line.

Light sensors would give the robot another perspective to interpret better its surrounding reality. The inclusion of light sensors would increase the complexity of the algorithm as there would need to be another thread sampling as the grid line coloration changes, however light sensors' use in the model would improve the overall accuracy of the robot. With light sensors, the robot would be able to locate exactly when it needs to turn 90 degrees and as well detect any anomalies in its movement. For example, moving diagonally feeds in a different patter of bright and dark transitions.