Lab 3

November 28th, 2022

This task introduces you to the available sensors of the robots and the corresponding interfaces of the LeJOS API. All sensors help your robot navigating through the maze during the final challenge, so make sure you know how to use them.

Bonus points

- For this lab, you can earn up to **2 bonus points** for a solution that fulfills the required **functionality** and has a good **code quality**.
- For evaluation criteria of the code quality, please refer also to the **Java style guide** (available through StudIP).
- We will only evaluate the solution developed to solve **Task** 3.3.
- Make sure your solution **works in the simulator**, as we will use it to test and evaluate your solutions on the `3x4_1.png` map.
- We will notify you via StudIP once you can find the evaluation feedback in your repository.
- **Submission Deadline:** Upload your solution to GitLab until **December 7th, 23:59h**

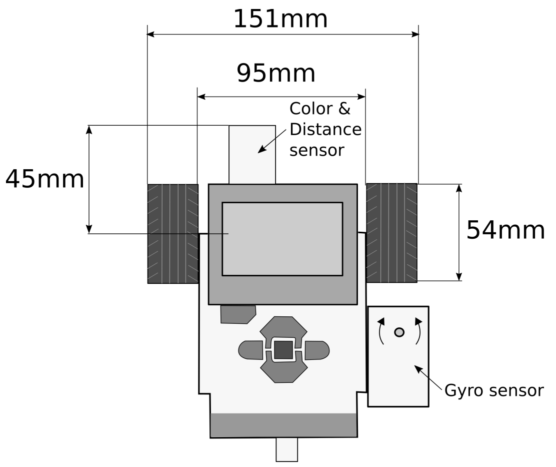
Change Simulator Map. In the `mazes/` directory you can find a selection of ready-to-use maps and a `README.txt` with instructions how to change the maze at your preference. However, the bonus task solution is required to run with the following configuration:

```
MazebotSimulation sim = new MazebotSimulation("mazes/3x4_1.png", 1.4, 1.05);
GuiMazeVisualization gui = new GuiMazeVisualization(1.4, sim.getStateAccessor());
sim.scaleSpeed(1);
sim.setRobotPosition(0.5, 0.5, 90);
```

Task 3.1: Color Sensor

Sensor principle. The robot is equipped with a color sensor at the front center. It can function as a light sensor measuring reflected light, measuring ambient light and measuring colors. We primarily use the latter function mode, where the sensor quickly reads the individual reflected light intensity for red, green, blue and the backlight, to determine the color on the object in front of it.

Program description. Write a program that continuously determines the color of the surface in front of the sensor. Use the color ID to print the detected color as string on the LCD. Please note that the color IDs in the documentation of the `Ev3ColorSensor` class are not correct. Use the IDs of the `Color` class instead. Examine the behavior of the sensor with differently colored walls – which colors are detected robustly, which might cause problems? Alter the distance between sensor and wall – what is the distance range for correct readings? Write this value down, since you need it for further tasks.



(a) Overview of robot and important measurements.

Functionality	Port
Color identification	`SensorPort.S1`
Gyroscope	`SensorPort.S3`
Ultrasonic (distance)	`SensorPort.S4`
Left wheel	`Motor.B`
Right wheel	`Motor.C`

(b) Wiring of attached motors and sensors.

Figure 1: Details of the robots used in the simulator.

Hints. The class `EV3ColorSensor` provides access to the capabilities of the sensor. Instead of providing the values directly, the color sensor uses the class `SensorMode`. For color identification use the lines:

```
EV3ColorSensor colSens = new EV3ColorSensor(SensorPort.S1);
SensorMode colorId = colSens.getColorIDMode();
```

To obtain sensor readings, use the method `fetchSample()` of the sensor mode class. You have to create a floating point array first to use it as parameter for `fetchSample()`. Keep in mind that you have to choose the right size for the array. The sensor mode provides the length of its samples by the method `sampleSize()`. For further capabilities, you are referred to the API.

Task 3.2 : Ultrasonic Sensor

Sensor principle. The ultrasonic sensor enables distance measurement and thus allows you to implement obstacle detection and avoidance algorithms.

Program description. Write a program that reads the distance provided by the ultrasonic sensor continuously. Print the distance in cm to the LCD. Keep in mind, that the distance sensor is displaced 45 cm from the robot's wheel axis center (see Figure Figure 1).

Check the detection range of the sensor. Try what happens if you place objects very closely.

Since the sensor uses directed speakers and microphones, its "field-of-view" is cone-shaped (as illustrated in Fig. 2). Determine the angle α empirically by moving objects at a constant distance laterally in front of the ultrasonic sensor.

Hints. LeJOS provides the class `EV3UltrasonicSensor` for communication with the sensor. Again, values are not provided directly but through class `SampleProvider`. Use the lines

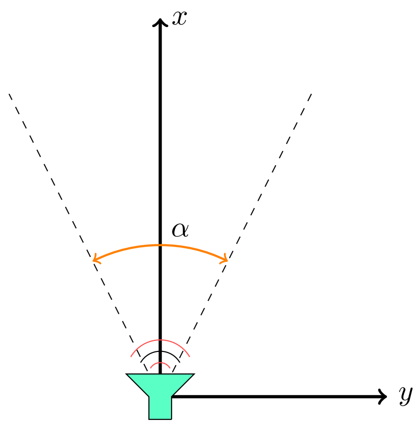


Figure 2: Field-of-view of the ultrasonic sensor; z-dimension is omitted for simplicity.

```
EV3UltrasonicSensor distSens = new EV3UltrasonicSensor(SensorPort.S4);
SampleProvider dist = distSens.getDistanceMode();
```

for initializing the distance sensor.

Task 3.3 : Approaching a wall (bonus point assignment)

Your task is to write a program that combines the reading of the color sensor, the distance sensor and driving of the wheels.

- Once the program is executed the robot should start moving forward in a straight line until an obstacle is detected. Make sure that you approach the obstacle until you reach a distance at which you can determine the color of the obstacle accurately.
- Display the current distance (in cm) to the obstacle to the LCD while moving, and
- print the color on the LCD after the robot has stopped in front of an obstacle.

Task 3.4 : Gyro sensor

Sensor principle. On top of your robot, you find a gyroscope which provides readings about changes in angular velocity while the robot is moving. It can thus be used to determine the turning angle of your robot during movement.

Program description. Write a program that queries the sensor continuously and outputs the current turning angle to the display. Drive one wheel motor to turn the robot on the spot. Calculate the number of full turns, i.e. 360° and output it to the LCD.

Hints. Use the class `EV3GyroSensor` of LeJOS for communication with the sensor. To obtain the accumulated angle since last reset of the sensor, use the method `getAngleMode()` which returns a `SampleProvider`. You can use the same methods as in previous tasks to fetch the sensor readings.