Lab 3 – Basic Closed-Loop Control

Overview:

Two ways to categorize robot control is open and close-loop control. In open-loop control, the robot system receives no feedback or information about the environment. As it is performing its actions, it is essentially blind. The advantage of open-loop control is that the robot does not need sensors and the controller does not require much logic. In closed-loop, the robot has access to sensor information and feedback to inform its actions. Closed-loop control is theoretically more robust, but also requires more components (sensors) and programming logic.

For Lab 1 and 2, we will be focusing on open-loop control. Open-loop control requires that the environment and the correspondence between robot commands and the physical movement is known. One way to obtain the corresponded between robot commands and actual actions is to measure how the robot moves given different command values and generate a calibration curve. For Lab 3, you will begin experimenting with sensors in order to perform basic navigation.

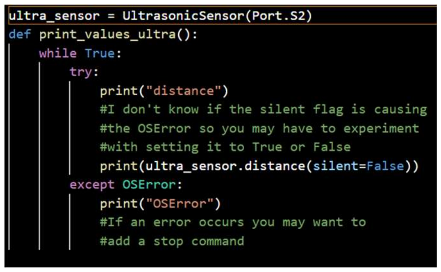
Objectives:

In this Lab, you begin working with sensors to produce closed-loop control. Additionally, you will:

- Gain familiarity with the EV3 robot sensors.
- Program simple navigation tasks such as line and wall following.

Procedure

0. Setup and Reading Sensors
    a. Assemble the driving base.
    b. Attach the ultrasonic sensor. See the Ultrasonic Sensor Driving Base instructions.
    c. Create the method shown in Figure 1 to test read the ultrasonic sensor.
    d. In my experiments, the ultra-sonic sensor was finicky and kept throwing an error which is why I added the exception handling.
    e. I also found changing the sensor port helped.
        i. Note, the sensor ports are not the same as the motor ports.
    f. **<u>For each sensor you use in this Lab, I want to see a method like the one shown in Figure 1 where you test print the values.</u>**



```
ultra_sensor = UltrasonicSensor(Port.S2)
def print_values_ultra():
    while True:
        try:
            print("distance")
            #I don't know if the silent flag is causing
            #the OSError so you may have to experiment
            #with setting it to True or False
            print(ultra_sensor.distance(silent=False))
        except OSError:
            print("OSError")
            #If an error occurs you may want to
            #add a stop command
```

Figure 1: Example of how to create and read the ultrasonic sensor.

1. Ultrasonic Sensor Follow with Reactive Control
    a. With the ultrasonic sensor attached.
    b. Program the robot to keep a set distance from an object in front of it with reactive control.
        i. If the robot is very close to an object and the distance is low, it should move backwards

            1. As the robot moves backwards and the distance grows, its speed should decrease to close to zero
        ii. If the robot is far from an object and the distance is large, it should move forwards
            1. As the robot moves forwards and the distance shrinks, its speed should decrease to close to zero
    c. Develop an equation that relates the speed of the robot to the distance measurement
        i. This should be a single equation where speed is a function of distance
    d. Take a video.
    e. Naïve Observers
        i. Show at least two people, not affiliated with the class, your working version of Task 2. Don't tell them how its programmed.
        ii. Ask them – 'What do you think the robot is doing?' and record their response/answer. Summarize in a table.
2. Color Sensor Line Following - FSM
    a. Attach the color sensor as shown in Color Sensor Down instructions.
    b. Draw a squiggly circuit on the floor with tape. About 3-4 feet in total length.
        i. Create a read sensors method for the color sensor and figure out what color the sensor things the tape is.
        ii. Note, there is 'yellow' tape in the Lab but that doesn't work great for hardwood floors. You can also try blue painter's tape. **<u>But fundamentally, verify that you can pick out the tape with the sensor.</u>**
    c. Program the robot to follow the line with only a **<u>single</u>** color sensor.
        i. Create a finite state machine diagram with at least two top level states. Make sure you include this diagram, and an explanation of the states in your Methods.
    d. Take a video.
3. Wall Following - FSM
    a. Attach the two touch sensors and small lego axles to the end to give them a little more reach.
        i. <u>Do not further modify the ends of the touch sensors.</u>
    b. Program the robot to wall follow.
        i. Create a finite state machine diagram with at least two top level states. Make sure you include this diagram, and an explanation of the states in your Methods.
    c. Take a video.

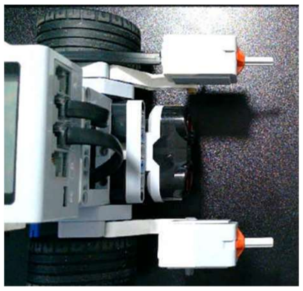


Figure 2: Image of the robot with touch sensors and small lego axles to extend the reach. The axles are the smallest ones at just short of 2.5 cm

Questions

1. What did the naïve observers say about Task 1? What did they think the robot was doing?
2. How did you decide how far to breakdown the states in the FSM for Task 2 and 3?
3. Comparing the tasks 1-3 to the maze navigation in Lab 1, what are some of the pros and cons of closed loop solution over open loop solutions?

4. What were the pros and cons of reactive control vs the FSM?
5. (Optional) What was the most challenging part of this Lab? Explain why.

Write up notes

1. Methods – **This Lab is mainly focused on solutions to the tasks which means the Methods and your algorithm explanations carry extra weight. In your methods, you should have at least**
    a. Speed equation from Task 1. Very brief explanation of reactive control to give context to the equation.
    b. State machine diagrams for Tasks 2-3. Very brief explanation of FSM to give context to the section.
        i. Explain what each module is doing. You can use basic text, pseudocode, equations, flow charts, etc to describe your algorithms.
        ii. This is one of the places I will be checking your logic. Your metric is that a reasonable programmer should be able to follow your outline and recreate your code and results.
2. Results
    a. Pictures of the robot in action and explanation and descriptions of the behavior. You can use screenshots from your video. The behavior and actions are the results so you need to document that.
    b. If there were errors or things the robot couldn't do correctly, document that as well.
    c. I should be able to get a sense of how your Tasks went from your still images and explanations.
    d. Note, the videos are tied into the Results score.
3. Supplementary Material – Each of these points should be in its own sub-section.
    a. Video files will be submitted via Canvas. In the Supplementary Material, provide a list of the videos and a brief description of each.
    b. List of the python files you will be submitting and a 1-2 line description of each.
        i. Each experiment or task should at least be in a separate python file that is imported or method
    c. Pdfs of your code

Rubric – Lab 3

General formatting (15)

- Title, date, name, section headings
- Miscellaneous style and formatting

Abstract (15)

- Gives a brief intro or overview of the Lab and/or topics covered.
- Gives a brief summary of the Results and main points from the Discussion.
- Well written and concise without typos or grammar issues.

Methods and Code (20)

- Possess all relevant equations and algorithm explanations

- Looks nice and is easy to follow.

Results (30)

- Figures and text for all experiments.
- Figures are properly formatted (labels, legends, captions, etc.).

Discussion (20)

- All questions are answered.
- Well written and concise without typos or grammar issues.