DEPAUL UNIVERSITY
COLLEGE OF COMPUTING AND DIGITAL MEDIA

CSC 375/475: Introduction to Robotics - Spring 2019/20. **Instructor: Isuru Godage, Ph.D.**

Lab 5: Week 6

Due on 5/13/2020 at 5 PM.

This group lab assignment has two questions. Develop the following RobotC programs.

Read the instructions carefully.

- Provide solutions to written questions in a single PDF file. In the title page of this document, identify group members, their affiliations, and percentage-wise contribution.
- Use separate c files for programming questions with filenames in the form Qnm.c, where n is the question number and m, is the part number.
    - E.g. **Q1b.c** is the solution file for the Q1 part b programming question.
- **Zip** your solution files in a file named Gn.zip where n is the group number and upload on D2L. Upload only one solution per group.
    - E.g. G3.zip is the solution by group 3.
- If you have any questions, contact the instructor.

Q1. Line following Robot I (binary state solution).

The idea of the line following programs is to sense whether the light sensor either sees black or white (i.e., dark or light) or adjusts the motion accordingly (Figure 1). The robot should see both the states (light and dark) to maintain the forward movement along the track edge. Otherwise, it would either be venturing into a dark or light area.

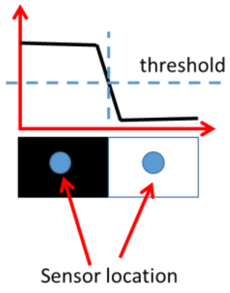

Figure 1: Sensor value change

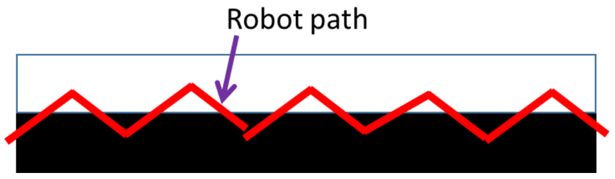

Figure 2: Robot traveling while changing the direction.

As the programmer, you need to decide if you wish to follow the left or the right edge of the line and write the code accordingly. The robot will oscillate back and forth over the line continuously checking for either black or white values (Figure 2); (for the reasons mentioned above) even if the line is straight, the robot continues to turn left and right searching for the track. Depending on the left or edge you are following, the robot will either go to the right or the left.

The key to a good program is to control the motion appropriately. You can use `setMotorSync()` and control the turning with `nTurnRatio` and speed with `nSignedPower` parameters. The more the robot oscillates, the slower it will perform. The main objective is to follow the line as smoothly and as fast as possible. Develop a RobotC program to read the color sensor using `getColorReflected()` and control the motion to follow a line (dark tape on a light surface).

a) Design a finite state machine to solve this line tracking problem. Clearly identify the states the actions you take within those states, and the conditions that dictate the transition between the states. Include the state machine with your submission.
b) You will be tracking a dark line on a white surface. To measure the values sensed by the color sensor, first placed the robot in the starting position D (light surface color) in the Color Sensor Table under the Utility Tables in the Robot Virtual Worlds (Figure 3). Record this value. Similarly, with the robot on the C starting position (dark color) and record the value. Calculate the threshold value by taking the mean of these two values.
c) Develop the RobotC program. Test your program using the Line Tracking Challenge (Figure 4) under the Program Flow. Test your program with different forward motion speeds and comment on the performance of the algorithm.

Hints: Play with the `nTurnRatio` variable to understand the amount of turning per given power to the `setMotorSync()` method. Then define the desired `nTurnRatio` value to change (+/-) the turning direction when you detect a state change. Start with less power.

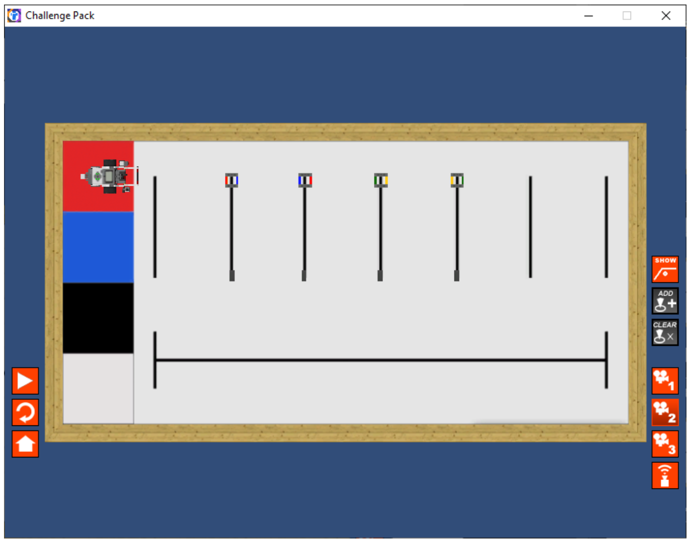

Figure 3: Color Sensor Table

Q2. Line following Robot II (proportional solution).

In the binary state example, the program only had two conditions to deal with: a sensed value either greater or less than the threshold. The problem with the binary approach is that the robot will tend to overcompensate for changes in the detected value with hard turns (only two turn values, +/-, used). For instance, imagine a car traveling down the road, and it starts to go off the roadway. Turning the car's steering wheel drastically to the left will bring the vehicle back onto the road (like we did in the first question with just one `nTurnRatio` value) but could quite possibly cause the car to lose control. Instead, the driver will gradually turn the vehicle back toward the road, and while maintaining the control of the vehicle, the correction is proportional to the error.

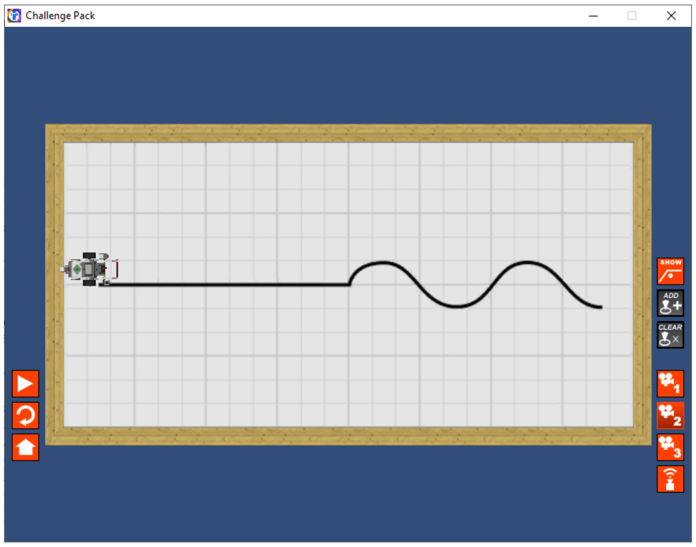

Figure 4: Line Tracking Challenge

If you want to follow lines, especially of curved lines, even smoother, you need a proportional controller. A proportional controller calculates the amount of correction that is required to get the robot back on the track that it is following. Instead of using a set value to correct the direction of the robot, we calculate the direction change based on the value read by the color sensor. If the error value is small, the robot corrects very slightly, whereas a more substantial value results in a stronger correction.

Now you can relate this example to the proportional control that you have been using so far. Now we can extend the same concept for the line following scenario. Think about a value read from the color sensor; it will be between `minValue` (track) and `maxValue` (surface). If the value is close to `minValue`, we are going to want to make an aggressive change of direction compared to a value around closer to the `threshold`, where we would only need a slight correction in direction. Define the error to be

$$error\ (e) = currentSensorValue - thresholdValue$$

Then define the desired direction change as a function of error as

$$directionChange\ (nTurnRatio) = k \times e$$

where k is an appropriate gain, which determines how steep the turn when the robot goes off the track. You can experiment with different k values and select the one that works best for the speed you are using.

Note that the sign of the error guides the turning direction. But in this case, you need not implement a slew rate scheme because we start closer to the desired value (threshold) and try to maintain that, instead of starting far away from the desired value. Also, the k value is important, because, for higher speeds, you only need smaller k values. Develop a RobotC program to read the color sensor using `getColorReflected()` and control the motion to follow a line (dark tape on a light surface) using a proportional control scheme. Experiment how your algorithm performs with different `motorPower` levels and k values. Similar to Q1, You can use `setMotorSync()` and control the turning with `nTurnRatio` and speed with `nSignedPower` parameters.

a)-c): complete the parts a) to c) of Q1 related to Q2.

Hint: Start with slow forward motion speed and the proportional gain (k). Increase the forward motion and change the proportional gain until you gain sufficient line tracking capability.

d) Comment on the relationship between the forward motion speed and the proportional gain.