ECE 110L Spring 2023

Laboratory 5.0:

Controlled Motion—Line Following

Contents

Reminder on Prompts

Throughout the IDC based lab manuals, there will be two additional prompts to let you know specific IDC work that needs to be completed. The two prompts are as follows, with N representing a number to help make sure you see all the prompts:

- **IDC Checkpoint (N):** Work to be shown to a TA during the IDC Checkpoint Section of the laboratory.
- **IDC Deliverable (N):** Other items to be completed and then included in the post-laboratory assignment.

1 Abstract

As part of the IDC, your team will need to have your 'bot navigate paths marked with dark lines on a light background. Along the paths, points of interest will be indicated by hashmarks that are perpendicular to the line (and significantly wider than the line). Whenever you get to a hashmark, your 'bot will need to use its sensor(s) to take a reading.

Note that this week, the work is divided into four sections - the Pre-Laboratory, the Experiment, the Exploration, and the IDC Checkpoint. IDC Checkpoints will specifically support the IDC Full System Demo. Although the first two IDC Checkpoints are intended to be accomplished by the squads individually, and are based on the success of a particular system or systems, the rest of the checkpoints will require the entire team to accomplish them.

2 Objectives

After performing this laboratory exercise, students should be able to build a 'bot that:

- (Exploration) Uses a set of three or four QTI (q=charge, t=transfer, i=infrared) sensors to navigate a track, and
- (IDC Checkpoint) Uses a set of three or four QTI sensors to navigate a track, stopping and performing tasks at hashmarks.

3 Background

Take a look at https://pundit.pratt.duke.edu/wiki/Following_Lines for a general discussion of how to get a 'bot to follow a line.

3.1 QTI Sensor

Information about the specific QTI Sensor you are using is on the EGRWiki page at https://pundit.pratt.duke.edu/wiki/ECE_110/Equipment/QTI.

4 Pre-Laboratory Exercises

Read through the Pratt Pundit page on Following Lines. Make sure you understand the concepts involved - for the IDC Checkpoints you will be doing at the end of lab, the 'bot needs to (a) follow a path with both straight sections and curves and (b) identify hashmarks along that line. It will do that by gathering information from three or four sensors then using that information to decide how (or if) to move. For this lab and later during the full system demo, every time the 'bot encounters a hashmark it will change the color of the on-board RGB LED. Note that if there are hashmarks, they will be much more than an inch away from each other

Once you are sure you understand the general idea, you will fill out a table that maps each of the eight (for three sensors) or sixteen (for four sensors) states for the QTIs to some kind of action plan for the robot. Below are some blank tables with a few entries filled in for each. Note that the tracks you are using will have both straight and curved sections, but no sharp corners. The hashmarks can show up in the middle of a straight or curved section.

Left	Middle	Right	No.	Description	Action
D	D	D	0	At a hashmark	stop, change light, wait, move straight past
D	D	L	1		
D	L	D	2	Highly unlikely	N/R
D	L	L	3		
L	D	D	4		
L	D	L	5		
L	L	D	6		
L	L	L	7	Not on line	turn left and right until line is found

Left	Mid-left	Mid-right	Right	No.	Description	Action
D	D	D	D	0	At a hashmark	stop, change light, wait, move straight past
D	D	D	L	1		
D	D	L	D	2		
D	D	L	L	3		
D	L	D	D	4		
D	L	D	L	5		
D	L	L	D	6	Highly unlikely	N/R
D	L	L	L	7		
L	D	D	D	8		
L	D	D	L	9		
L	D	L	D	10		
L	D	L	L	11		
L	L	D	D	12		
L	L	D	L	13		
L	L	L	D	14		
L	L	L	L	15	Not on line	turn left and right until line is found

Note: *The last column is based on converting Dark and Light to binary digits 0 and 1, respectively; using Left as the most-significant bit and Right as the least-significant bit.*

During the lab, you will be required to follow different types of paths:

- (Exploration) Line with no hashmarks - your 'bot needs to follow to the end; once it goes past the end of the line, it can stop (i.e. if you have LLL or LLLL, the 'bot should stop).
- (IDC Checkpoint) Navigation–line following with hashmarks–your bot needs to stop at each hashmark for one second before continuing down the path. Also, when it gets to a hashmark, it should change the color of the on-board RGB LED so that at the first hashmark it is red, at the second it is yellow, at the third it is green, at the fourth it is blue, and at the fifth it is purple. Your 'bot should completely stop on the fifth hashmark.

Pre-lab Deliverable (1): Create a list or table for the 3-sensor states and actions. This does not need to be in LaTeX but does need to clearly show the status of the three sensors, a description of what that status means, and a statement of the action your 'bot should take. If a state is "Highly unlikely," you can put that as a description with no action plan.

Pre-lab Deliverable (2): Create a list table for the 4-sensor states and actions. This does not need to be in LaTeX but does need to clearly show the status of the three sensors, a description of what that status means, and a statement of the action your 'bot should take. If a state is "Highly unlikely," you can put that as a description with no action plan.

5 Pre-Laboratory Assignment

The documentation for the pre-lab involves submitting a single PDF file. Your document must include your name, NetID, and the Duke Honor Code statement: "I have adhered to the Duke Community Standard in completing this assignment" at the top. **EACH INDIVIDUAL** should submit their own assignment. Your document should also include the tables you created above for Pre-Lab Deliverable 1 and 2.

6 Equipment

- (1) CX-Bot
- Up to (4) QTI sensors with mounts
- Up to (4) Black-Red-White cables
- (1) Parallax screwdriver

7 Exploration

For this exploration, you will use the QTI sensors to aid your 'bot in navigating a track.

1. First, you should decide if you are going to use three or four QTI sensors for line following. There are advantages and disadvantages to each approach.
2. After you have decided, mount that number of QTI sensors and connect them with the black-red-white cables to the dedicated BRW headers on the front of the CX-Shield. Carefully note which pin numbers are on each red pin; the black pins are all connected to ground and the white pins are all connected to 5 V. All three or four sensors should be right next to each other, and they should be collectively centered on the front of the 'bot.
3. Once those are mounted, copy the sample code to a sketch and upload it to your 'bot. The sample code will look at one QTI and assumes it is connected to the BRW where red is 47. This should be your far-left QTI. See what happens if you place the 'bot so that the QTI is over a light surface or a dark surface on a test mat.

   Checkpoint (1): Show your TA that you are correctly reading and displaying the information for one QTI.
4. Copy the sketch to a new sketch and change the code so it reads and displays information from all of your QTIs. See what happens if you place the 'bot so that the QTIs are over different combinations of light surfaces or a dark surfaces on the mat.

   Checkpoint (2): Show your TA that you are correctly reading and displaying the information for multiple QTIs.

   Discussion (1): Discuss the numerical values for the QTI when it is over a light surface versus a dark surface. Come up with what you think is a good threshhold value between "dark" and "light."
5. Write code that converts your QTI states into a single number. Note that Arduino can do math with digital logic - anything true is considered a 1 if you do math with it, and anything false is considered a 0. Assume each QTI's dark/light response can be converted to a binary 0 or 1 (if the decay time is less than or above the threshhold you established above). Next assume the far left QTI's state is the most significant bit and the far right QTI's state is the least significant bit. With that, you can convert the three or four binary "digits" into a single decimal value as follows:

- For three QTIs:

```
int state = 4*sLeft + 2*sMiddle + sRight;
```

  where `sLeft`, `sMiddle`, `sRight` are logical expressions about whether the left, middle, or right sensor is reading light or not. For example, you could replace `sLeft` with `qti1 > 2000`.[1]

- For four QTIs:

```
int state = 8*sFLeft + 4*sMLeft + 2*sMRight + sFRight;
```

  where `sFLeft`, `sMLeft`, `sMRight`, `sMRight` are logical expressions about whether the far left, middle left, middle right, or far right sensor is reading light or not.

Checkpoint (3): Show your TA that your code is correctly reading the QTIs and converting them to single overall state value for several combinations of dark and light.

6. Now that your 'bot can figure out what state it is in, it is time for it to do something about it! There is a structure in Arduino code known as a `switch...case` statement that will be very useful here. When you have a variable that contains an integer that can be mapped to a particular action item, a `switch...case` structure is a much more efficient way to go than a large if tree. Take a look at https://www.arduino.cc/reference/tr/language/structure/control-structure/switchcase/ for the syntax and examples.

   One quick thing to note is that if you have multiple cases that are all meant to do the same thing, you can stack several case statements before the code each is meant to run:

```
switch (var) {
    case 0:
    case 3:
        // stuff for either case 0 or 3
        break;
    case 1:
    case 2:
    case 6:
        // stuff for case 1, 2, or 6
        break;
    default:
        // stuff for any other value of var
        break;
    }
```

[1] That would be a particularly poor choice of threshhold, but you get the idea.

Here is where the work on the pre-lab comes into play! Figure out what you want your robot to do for those states you think you might achieve. For the default case, you should have your robot stop moving and you should print a message to the serial monitor essentially indicating that the 'bot does not know what to do there. Every other case should be specifically addressed in the `switch` structure.

You will likely[2] want to write some functions you can call that will have the robot move forward, turn or veer left, turn or veer right, and stop. You performed all these tasks in a previous lab, so time to dust off those routines and bring them back into action!

For this exploration, if your 'bot identifies a hashmark, it should keep going straight. For IDC Checkpoint 1 that will no longer be the case.

7. Once your code is written, test it on the three test tracks:

   - Straight line
   - Squiggle
   - Circle

   For the straight line, there are hashmarks that the 'bot should roll right over. For the straight line and the squiggle, the LLL or LLLL case can be coded to cause the robot to just stop (i.e. you do not need to write code to try to find a line for the LLL or LLLL cases). For the circle, your 'bot should just go in circles forever.

   There are two main items you will need to tweak in your code once you get the `switch` sorted out:

   - How much time does a 'bot spend moving in a particular way before you check the state again?
   - How fast should each wheel be going for each maneuver?

   Hint: Full speed for long periods of time between checks is definitely not a good idea...

   Checkpoint (4): Show your TA that your 'bot can navigate the above paths. To get full credit, it should follow the straight line two separate times, the squiggle two separate times (once in each direction), and go around the circle twice in one direction without stopping and then twice in the other direction (the TA will manually reverse the 'bot's direction).

   Deliverable (1): Include this code in your PDF for this week and a description of additional features that might help improve the code. You are *not* required to come up with code for the improvement, just describe your suggestion.

   NOTE: The IDC Checkpoint code will very much be based on this code; you will want to document this simpler version first.

[2] Where "likely" here means "definitely."

8 IDC Checkpoint

As noted in the Pre-Laboratory exercise and in the IDC documentation, IDC Checkpoint 1 is all about your 'bot's navigation (AKA Line Following). For the IDC Checkpoint, whenever your 'bot gets to a hashmark, it should stop and change the color of the on-board RGB LED. The colors at each hashmark should change between red, yellow, green, blue, and purple. The 'bot should remain at hashmarks for at least one second before moving on.

You already have the code to follow a line while ignoring hashmarks, now consider which overall QTI state value or values mean you have found a hashmark and should thus stop the 'bot and change the color. Also consider how you are going to keep track of *how many* hashtags the 'bot has seen, what that means for the RGB LED, and what it means for the 'bot going forward.[3]

IDC Checkpoint (1): Show your TA that your 'bot can start moving at the beginning of your group's IDC path, stop at each hashmark for at least one second, turn on the on-board RGB LED appropriate color for that minimum 1 second, then turn off the on-board RGB LED before moving forward again, and then repeat this process until the fifth hashmark causes the on-board RGB LED to turn on with the appropriate color and turn off again.

IDC Deliverable (1): Include your code for having the CX-Bot follow the line, change the color of the on-board RGB LED accordingly.

IDC Deliverable (2): Write a brief paragraph reflecting on your experience with this lab. Comment on any struggles your team faced and how you overcame them. What can you improve for next time? Comment on the clarity of the lab manual (Pre-Laboratory, Exploration, and IDC Checkpoint) and supporting material for this week. Let us know of any additions, deletions, or edits you would make to any part of it.

[3] As in, should it go forward anymore?

9 Assignment

The assignment for this laboratory involves submitting a single PDF file. Your document must include your name, NetID, and the Duke Honor Code statement: "I have adhered to the Duke Community Standard in completing this assignment" at the top. **EACH INDIVIDUAL** should submit their own assignment. The documentation for this laboratory includes the both codes and a reflection:

1. **Code for Simple Line Following** 7
    - The QTI code for simple line following from the exploration (following a line, regardless of hashmarks).
2. **Code for Line Following with Hashmarks** 8
    - The code for following lines, stopping at hashmarks, turning on the appropriate RGB LED colors, and turning them off before continuing. Code should show that the 'bot does not move on past the final hash mark. Make sure your TA has marked your group as finishing IDC Checkpoint 1.
3. **Reflection Paragraph** 8
    - Brief paragraph reflecting on your experience with this lab.

This file should be uploaded to the ECE 110L Laboratory **Gradescope** site by the assignment deadline. Each student must submit their own **INDIVIDUAL** assignment.