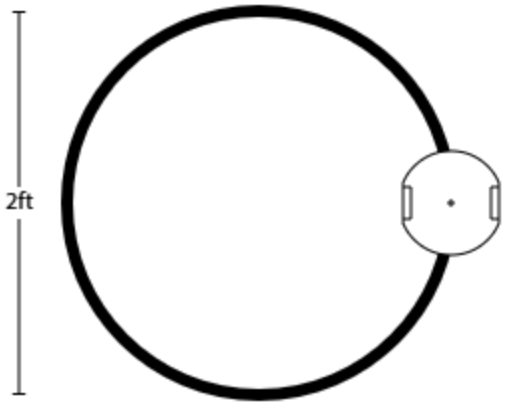
# Lab 0x04 - Line Following

New Attempt

- Due Nov 6 by 12:10pm
- Points 20
- Submitting a file upload
- File Types pdf, mp4, and py

# Overview

The purpose of this assignment is to write a new hardware driver for your line sensors and to get Romi to drive in a 2 ft diameter circle with closed-loop actuation using feedback from line sensors.




# Assignment

It's best to approach this assignment in two steps. First, write and test a driver class to get data from your line sensors in a usable manner. Second, use that driver class with your existing code from Lab 0x03 to get Romi driving in a circle reliably.

## Line Sensor Driver

1. Design a new Python class or set of classes to run on your Nucleo board to facilitate usage of your line sensors. Because each group has the opportunity to select their own sensors, there are no specific requirements for how the driver class is set up or how it will be utilized.

   Your team may want to start by drawing a class diagram so that you can all agree on the set of attributes and methods that will work best for you and your particular sensor selection.

   Consider the following before you begin your class design:

- If your sensor has many channels that are each manipulated individually, it may be convenient to write two classes: one that represents the interface for a single sensor, and a second that represents an array of sensors and internally uses multiple objects of the single-sensor class.

- Alternatively, if your sensor has many channels that share a set of input or control pins, it may be more convenient to write one class that represents the entire array manipulating all the sensors at once.

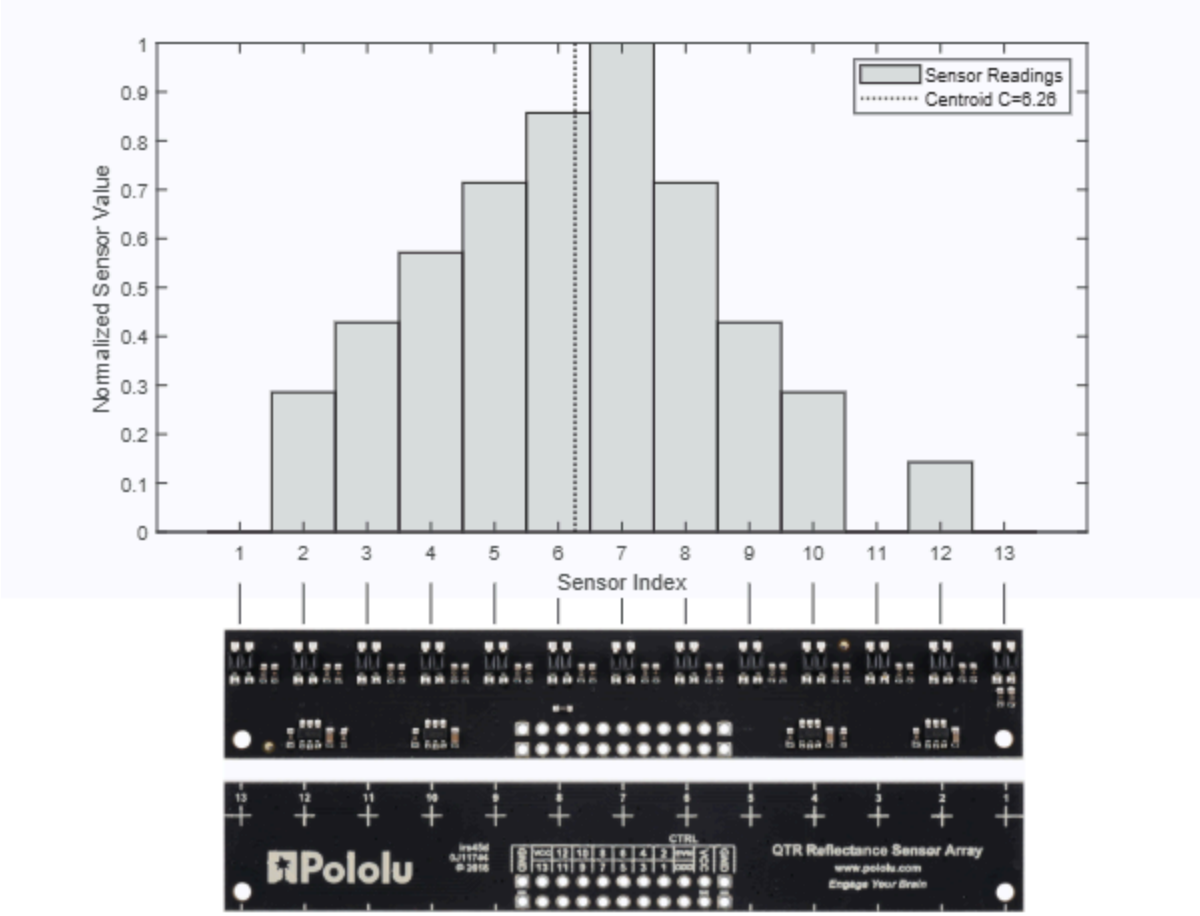
In either case, it will be nice to have all channels of the sensor array handled by one object so that you can keep your tasks tidy; having modular sensor drivers will make it easier to avoid redundant code outside of the drivers.

What features will make your sensor class the most useful? Brainstorm with your group, but also consider some of the following suggestions:

- Will oversampling of your sensor data improve the quality of your sensor readings? If so, are the penalties in terms of processing time and memory usage acceptable?

- Will you want to calibrate your sensors? If so, what information is needed? Consider adding features that let you read a white datum and a black datum so that you can normalize the sensor output between these two data.

- Will you want any information from the sensor output that is more processed? For example, finding the centroid of the sensor array readings may help you determine where black lines are with respect to the sensor. Refer to the figure below for an example of how this centroid might be determined.




2. Now that your team has thoroughly considered the design of the driver class it should be much easier to write the code. Make sure that when your team is done writing that you test the code thoroughly and make sure it works as expected.

# Line Following

1. With your sensor class written and tested you can begin work on driving in a circle. Consider how you can achieve this using closed-loop feedback from the sensors. A simple approach would be to choose a constant forward velocity for Romi and to use the line sensor feedback to adjust a steering command for Romi using something like a PID controller.

   In lecture, you will learn about various ways to set up a sophisticated control loop that uses feedback from both encoders, the IMU, and also the line sensors, but for now it is best to keep the design simple.

2. Aim to robustly drive Romi in a circle many times consecutively. That is, aim to design and implement a controller that can stably "lock on" to the circle and do lap after lap without getting off track.

   Eventually you will need to keep track of the distance Romi has traveled so that you can, for

instance, drive exactly one loop around the circle, but that will not be strictly necessary for this assignment's objectives.

# Deliverables

The final deliverables for this assignment will include a memorandum and several attachments.

**Memo:** Write a detailed but concise memo presenting your design choices for this assignment including some discussion of your task structure and selection of shared variables. Why did you choose the task periods and priorities that you did? Why did you pick the specific shares and queues that you did?

**Diagrams:** Include in the body of your memo the task diagram your team has constructed along with state transition diagrams for each task implemented as a FSM.

Also include a block diagram depicting the control structure that you've implemented.

If you feel that the images fit poorly within the body of the memo also include larger high-resolution images as attachments at the end of your memo that each fill up an entire landscape oriented page.

**Shared Variables:** Include a table in your memo that lists out each shared variable with columns describing the data stored in that variable along with the motivation or purpose of that variable in your task structure.

**Program Functionality** Include a video recording of Romi driving on the circle for multiple consecutive laps. Ideally Romi should be able to do this indefinitely, but you can limit the number of laps recorded to something reasonable.

**Source Code** Include at the end of your memo a transcript of the source files you've written for this assignment. Do not include a transcript of the driver classes from previous labs unless you modified them significantly. Also include the source files themselves as attachments to the memo or as additional submissions on Canvas.