Michael Mitsch and Clare Barnes
September 28, 2021

Lab 2 Report: Line Sensor

Video: https://youtu.be/rVqncuPK7XQ (link also in the readme)

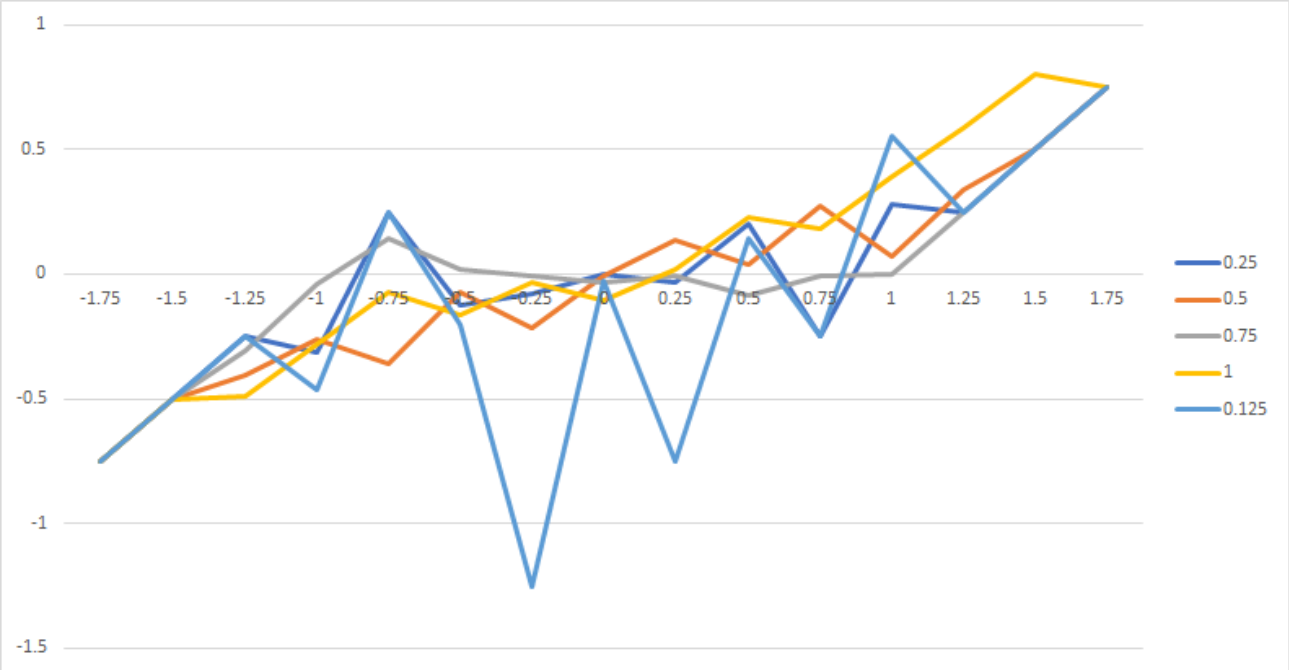
1. **Line Sensor Module Code**

The line sensor class in our code was somewhat redundant, as the TRSensor class could have probably been used directly, but the use of the botPosition class cleaned up the main function. The initialization initializes both the TRSensor class and the AlphaBot2 class, and then it calibrates the sensors with a method borrowed from the LineFollow code provided. Lastly, it instantiates botPosition.sensors and botPosition.position. botPosition.sensorValues() returns just the sensor array from the TRSensor function readCalibrated(). This function did not end up being used in the main function, as the function botPosition.getPosition() set the values for both position and the sensor array. It returns position and then one can just call botPosition.sensors to get the corresponding array. This way the position and sensor values were from the same read, and the display would be more accurate, as they tend to fluctuate slightly even without movement.

2. **Design of Curses App**

Our curses application uses the bottom few lines to print out a description of the robot's position of the line, the sampling frequency, and the calibrated read value each frame. The rest of the screen is filled with a bar chart that shows the value read by each sensor. The tallest bar in the bar chart will always go up to the top of the screen. The rest of the bars in the chart will have a height equal to their ratio to the largest value read, times the number of lines available. The values read by the sensors are printed on each bar and the sensors are labeled on the horizontal axis of the chart. The sampling frequency is set to 3 Hz right now, but we varied that frequency while testing the curses application.

3. **Discuss Line Sensor Accuracy**



1, 0.75, and 0.5 yielded the most accurate measurements. 0.25 and 0.125 seemed to be too small, because the sensors would often lose the line and give one of the edge values of -2000 or 2000. That is why one can see, in 0.125 especially, that the line would spike between an edge, and then a more accurate measurement.

4. **Track Width Recommendation**

The y-axis of the graph in number 3 represents the error in inches from the measured value. To calculate that error we mapped the output position that was between -2000 and 2000 to a value in inches from the center, by using the distance from the far right sensor to the far left sensor. The x-axis represents the position of the robot relative to the line. The graph leads us to believe that 0.75 inches is the optimal line width because the error is small while the robot has the line within 1 inch of the center.