EE3: Introduction to Electrical Engineering - Path Following Robot -

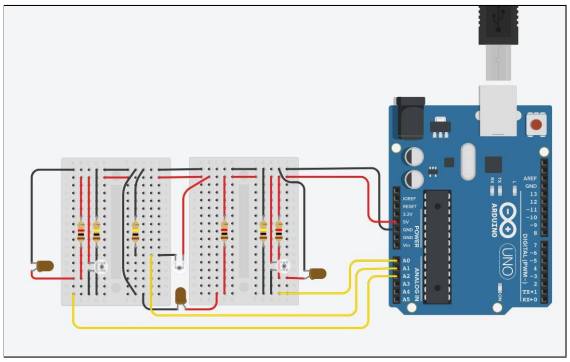
Brian Dionigi Raymond
Kevin Ke-En Sun
Instructor: Prof. Briggs
December 8, 2017

Introduction

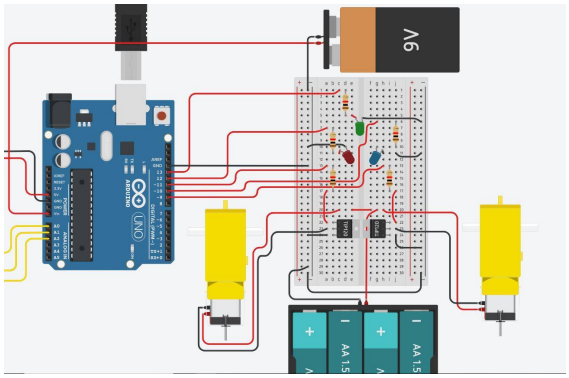
The goal of the project was to design a car able to follow a path represented by a black line on a white background. The car uses three sensors to detect where to go on the path, one each on the front, right, and left. The sensors work by using an IR LED that reflects IR light off the ground and into a phototransistor. They are connected to each other as shown in Figure 1a. A phototransistor works like a transistor which is like a switch. A transistor has three terminals: a base, emitter and collector. When a voltage is applied to the base that is relatively higher than the voltage on the emitter, current flows from the emitter to the collector. On the other hand, if the voltage applied to the base is relatively lower than the voltage on the emitter, then no current flows. A phototransistor is different from a regular transistor, however, in that the base is sensitive to light as opposed to voltage. When light is shown on the base, current flows, but if there is no light, there is no current. An emitter follower setup was used for the sensors (signal response is low in absence of IR light and high in its presence). See Figure 1b for the proper wiring.

The car's motion is controlled by going straight whenever the front sensor triggers low. As soon as either the left or right sensors trigger low, the car turns in the opposite direction (left if right triggers, right if left triggers) until the front sensor trigger again triggers low again. The rate at which the car turns back towards the front sensor was adjusted using PID methods[1] to allow the car to orient itself along the line both faster and more precisely. After the front sensor triggers low, the car would then continue straight again until it finds the stopping condition (black lines perpendicular to the

Circuits for Path Sensing and Movement



(a) IR LED and Phototransistor Circuit



(b) Motor and LED Circuit

Figure 1: Circuit schematics for our path following car. In Figure 1a, the $1k$ resistors attached to the IR LEDs between the V_{in} and anode (+) as well as $10k$ resistors attached to the photo transistors between the emitter and ground. The phototransistors go to Analog pins since their data is transmitted as an analog signal ranging from $0 - 1000(arb.)$. In Figure 1b, the LEDs are wiring the same way as the IR LEDs from before. The TIP120 is connected to V_{in} through a $1k$ resistor on it's base, to the motor through it's collector, and to ground through it's emitter.

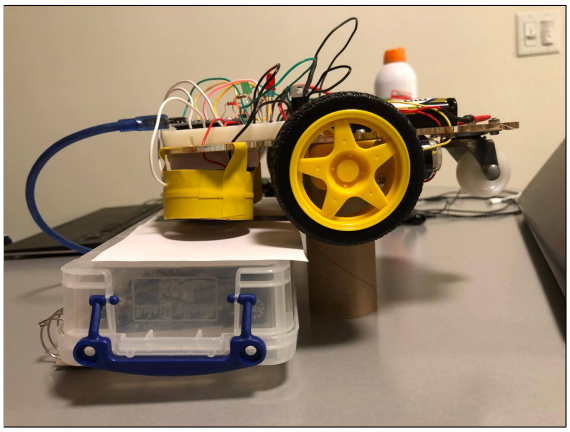
track). Pulse width modulation (PWM) and transistors are used in conjunction to control the speed of the motor. The PWM signal is connected to the base of the transistor and the duty cycle of the PWM is used to control the speed of the motor. Increasing the duty cycle of the PWM increases the speed of the motor by increasing the relative time for which the motor is on to that for which it is turned off. LEDs are used to signal which direction the car is going: a green LED signals forward, a red LED signals left, and a blue LED signals right. In the case of the car's stopping conditions being met, both the blue and red LEDs turn on. LEDs have two sides: a negative ('n') side and a positive ('p') side. The n side is filled with electrons and the p side is filled with holes. Energy is released when electrons annihilate the holes and this energy is most likely released in the form of light.

Testing Methodology

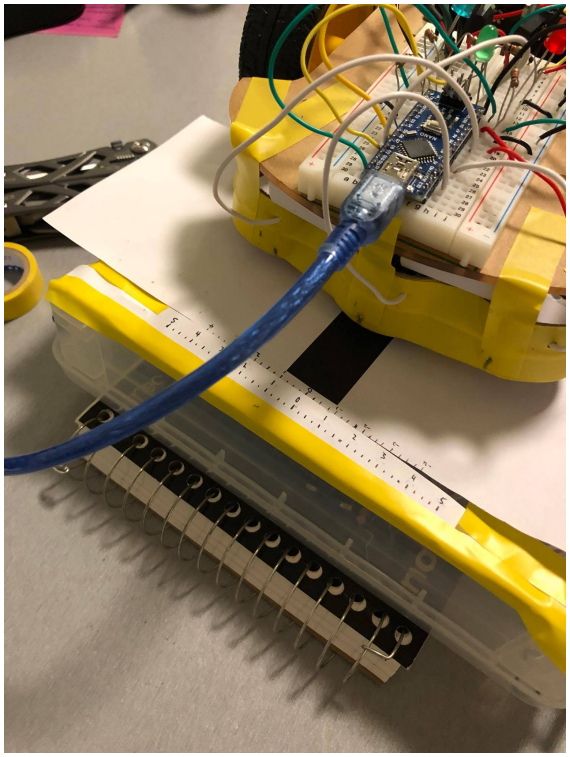
How We Designed the Test

To obtain a comprehensive test of our path sensing subsystem and sensors, we took a multi step approach involving testing the initial, adjusted, and mapped readings from our getDistance() function before testing the entire function. Each of our tests involved a setup where the robot was propped onto a cardboard jack (e.g. paper roll) and the sensors suspended over a stationary piece of paper that was unmarked besides lines acting as a metric ruler. Using another similarly delineated paper with a $2cm$-wide black line approximating that on the course, we aimed to simulate the robot's motion by moving the line from to different distances from each of the sensors.

Circuits for Path Sensing and Movement



(a) Cardboard Jack



(b) Sliding and Stationary Ruler

Figure 2: Figure 2a shows how the car was suspended above the ruler. Figure 2b shows both rulers; the stationary ruler with a range of $\pm 5cm$ and the sliding ruler with a range of $\pm 4cm$.

How We Conducted the Test

The test was conducted in four parts. For the first three parts which consisted of the function component tests, we took measurements for each of the sensors separately, centering them on the black line, and measured $\pm 1cm$ from the edge of the line. Originally, we measured $\pm 2cm$ but noticed that past a centimeter made no difference as the IR phototransistor was unable to read that far away. From these three trials, we were able to tune our processing of the sensor readings to better gauge the distance. Then for the final trial where we tested the overall getDistance() function, we took measurements by starting ($\pm 4cm$) from the front sensor and taking measurements of the estimated distance at $0.25cm$ intervals going back to the center.

How We Analyzed the Test Data

After each of our trials, we graphed the estimated distance (function's output printed to Serial) versus the actual distance (as measured). The first trial (raw sensor readings vs. distance as measured) gave us a polynomial graph so we found the polynomial regression and solved in terms y in order to obtain a distance value from the sensor reading. Taking new measurements using our newly found formulas gave us the second trial, where after graphing we noticed our global minima were preserved but the range condensed to not contain the entire range of $\pm 1cm$ that we knew the sensors could read from Trial 1. Having adjusted our code once more before our final trial, we moved onto the fourth and final trial, where we graphed all of the sensors on the same axis with $x = 0$ being the center of the front sensor. From here we began our interpretation of the results.

How We Interpreted the Data

Looking at the trial of the overall performance of the getDistance() function, performance was interpreted according to two metrics: absolute error at different point and overall trend. For the sensor function to be acceptable, the distance should decrease as the black line goes from either side sensor towards the front sensor and reach a minimum once at the direct sensor of the car. In addition to this, we judged success on the difference between the estimated distance and the actual distance. As the sensor an only read a maximum of 1cm away, we aimed for a maximum error of half that or 0.5cm.

Results and Discussion

Test Discussion

Test Discussion plaintext - To be added

Race Day Results

Results Discussion Conduction plaintext - To be added

Conclusions and Future Work

Conclusions plaintext - To be added

Future Work plaintext - To be added

References

References plaintext - To be added

Raw Data from Sensors

	Sensor Values (*arb.*)		
Distance (cm.)	Left	Front	Right
2.00	988	986	982
1.75	988	986	982
1.50	988	985	982
1.25	988	984	981
1.00	987	983	973
0.75	876	980	506
0.50	718	782	443
0.25	680	544	380
0.00	656	499	362
-0.25	634	501	374
-0.50	640	590	381
-0.75	786	743	655
-1.00	985	980	980
-1.25	986	982	982
-1.50	987	986	982
-1.75	987	986	982
-2.00	987	986	982

Table 1: text

Adjusted and Mapped Distance from Sensors

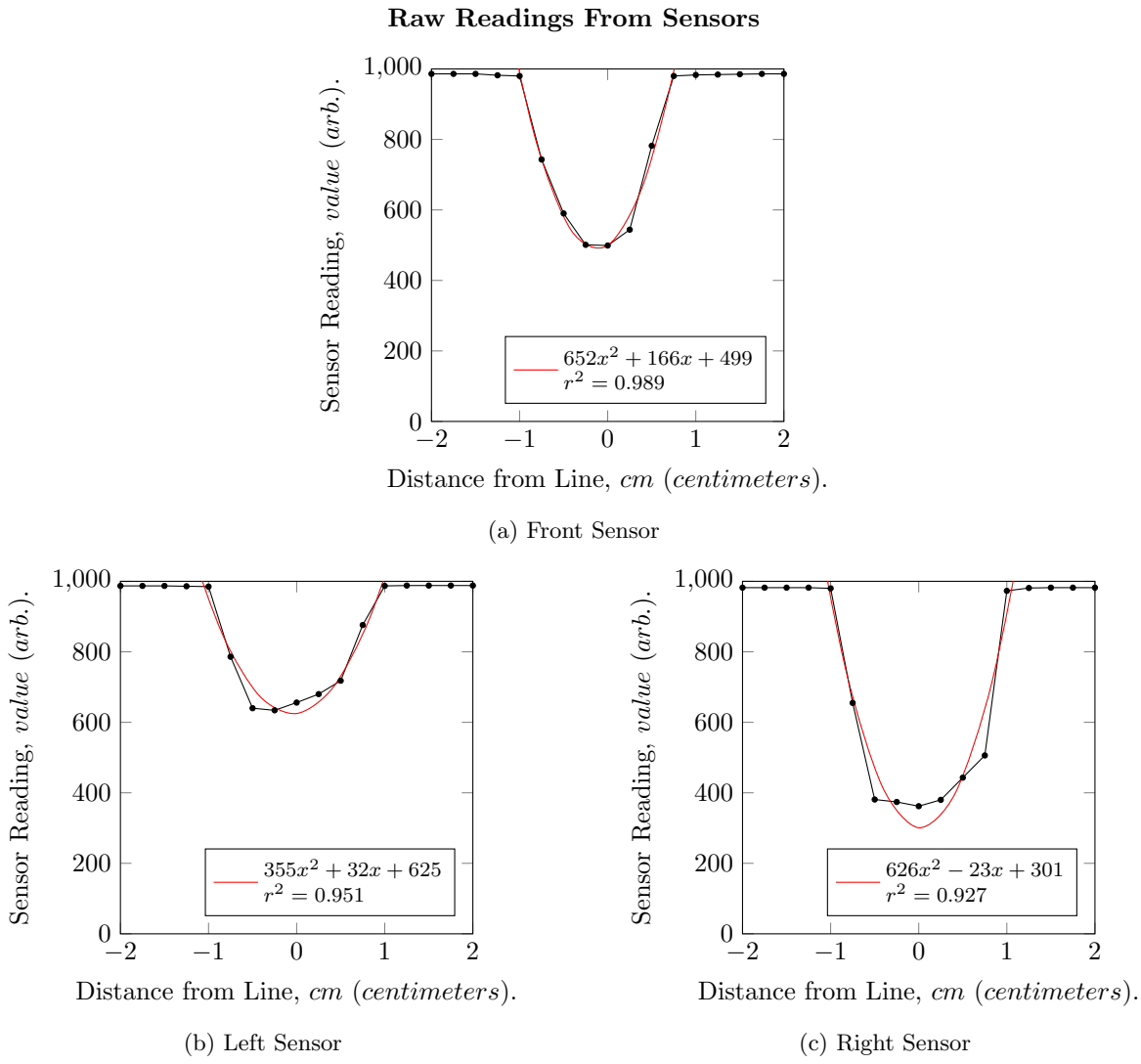
	Distance, Adjusted (*cm.*)			**Distance, Mapped** (*cm.*)		
Distance (cm.)	Right	Front	Left	Front	Left	Right
-1.50	1.06	0.75	0.97	0.99	1.00	1.00
-1.25	1.06	0.74	0.97	0.99	1.00	1.00
-1.00	1.05	0.74	0.97	0.99	1.00	1.00
-0.75	0.62	0.74	0.94	0.96	0.96	0.96
-0.50	0.49	0.47	0.94	0.70	0.91	0.37
-0.25	0.47	0.42	0.94	0.08	0.65	0.19
0.00	0.41	0.30	0.87	0.01	0.02	0.01
0.25	0.42	0.37	0.92	0.09	0.31	0.12
0.50	0.67	0.50	0.94	0.29	0.48	0.37
0.75	0.91	0.70	0.95	0.72	0.70	0.76
1.00	1.05	0.74	0.96	0.96	0.92	0.99
1.25	1.06	0.75	0.97	0.98	0.99	1.00
1.50	1.06	0.75	0.97	0.99	1.00	1.00

Table 2: text

Adjusted and Mapped Distance from Sensors

Distance (cm.)	**Actual** (*cm.*)	**Measured** (*cm.*)
-4.00	4.00	3.88
-3.75	3.75	3.84
-3.50	3.50	3.84
-3.25	3.25	3.70
-3.00	3.00	3.25
-2.75	2.75	2.00
-2.50	2.50	2.00
-2.25	2.25	2.00
-2.00	2.00	2.00
-1.75	1.75	2.00
-1.50	1.50	2.00
-1.25	1.25	2.00
-1.00	1.00	0.96
-0.75	0.75	0.70
-0.50	0.50	0.26
-0.25	0.25	0.12
0.00	0.00	0.04
0.25	0.25	0.43
0.50	0.50	0.97
0.75	0.75	0.98
1.00	1.00	2.00
1.25	1.25	2.00
1.50	1.50	2.00
1.75	1.75	2.00
2.00	2.00	2.00
2.25	2.25	2.00
2.50	2.50	2.00
2.75	2.75	2.00
3.00	3.00	2.00
3.25	3.25	3.11
3.50	3.50	3.24
3.75	3.75	3.43
4.00	4.00	3.62

Table 3: text

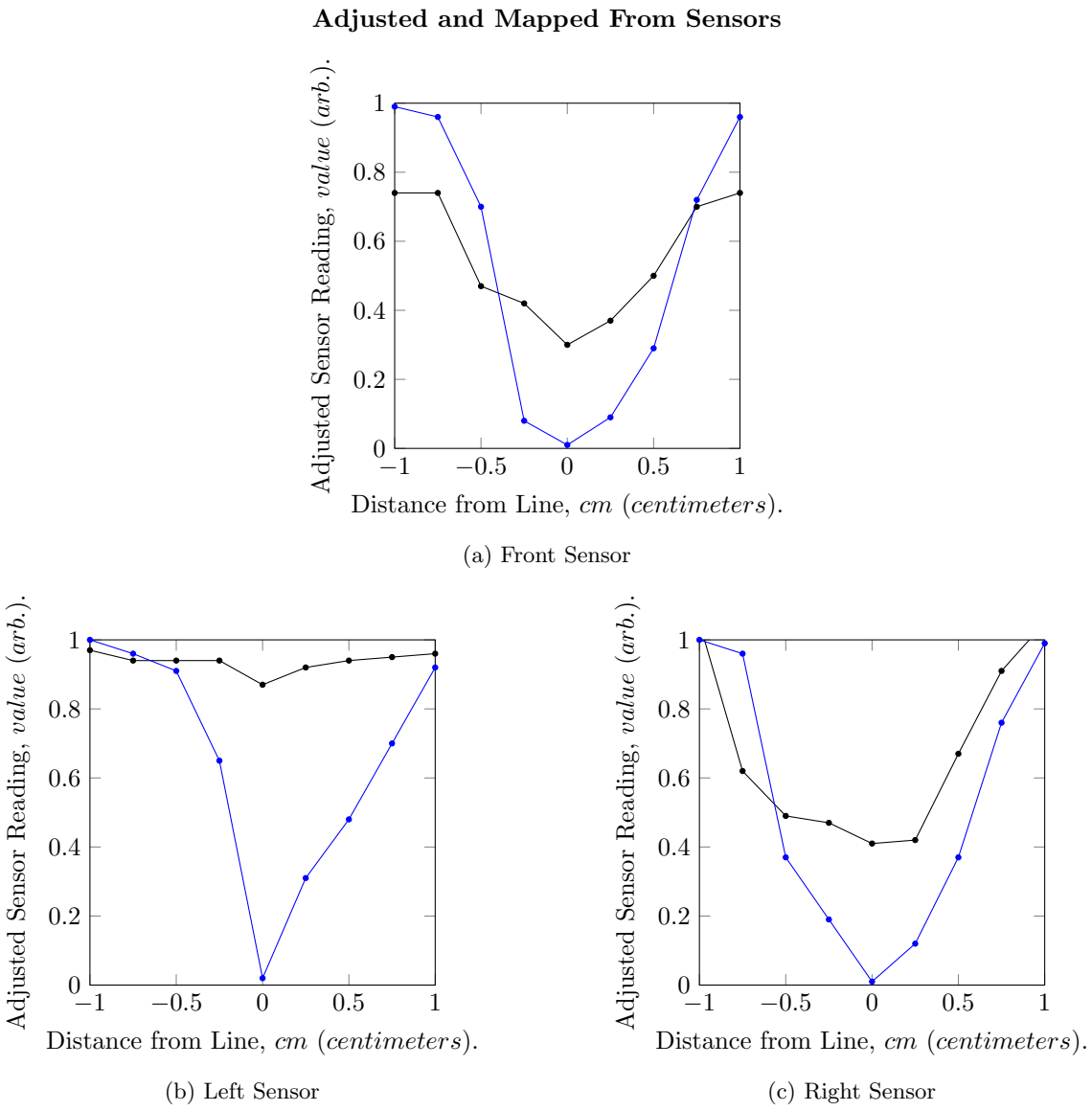


(a) Front Sensor

(b) Left Sensor

(c) Right Sensor

Figure 3: **Raw readings from the IR sensor, IR phototransistor combinations used on our path-following robot.** From the graphs, the value of the maximum readings range from about 400 for Graph 4c to about 600 for Graph 4b. Also worth noting is that the sensors each reach trend down to a global minimum of around $0cm$ when moving from $\pm 1cm$ to the black line.



(a) Front Sensor

(b) Left Sensor

(c) Right Sensor

Figure 4: **Adjusted sensor readings using a crude polynomial fit of the data from Graphs 3a, 3b, and 3c with a degree $n = 2$ and then mapping to expand the range across the sensors line of sight (0 to $1cm$).** By using a polynomial fit of the sensor reading (y) versus the distance from the line (x) and solving in terms y, we are able to both find the distance from our readings and preserve each sensors local minimum at $0cm$. From there, we need to adjust the values to fit our known distribution and therefore simply map it as shown. By doing this, we see that our data better fits the distribution and is able to give us more valuable information to use with our PID function as the same change in the sensor reading returns a more significant change (and accurate) change in calculated distance from the line.

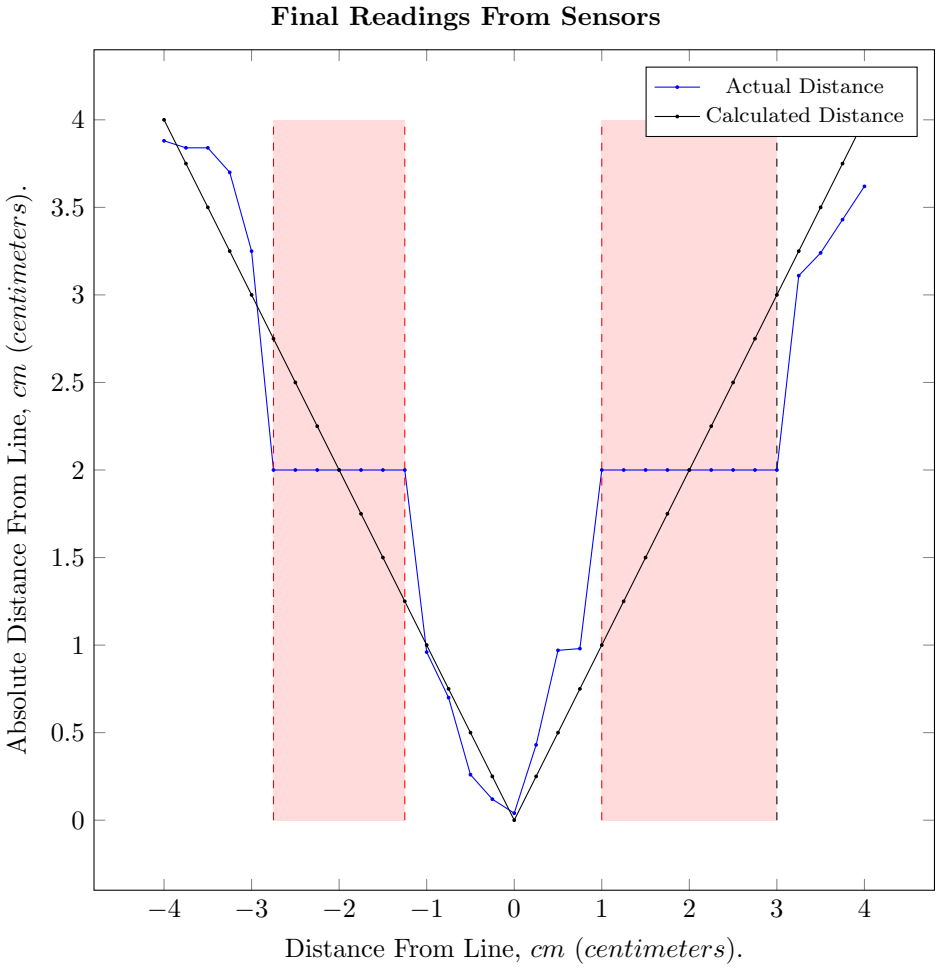


Figure 5: Comparison between actual and measured distance (from our getLocation() function) between center of $2cm$ thin black line and center of robot's front sensor. The red zone in the figure represent the area for which the black line is not within the range of the left, front, or right sensors. The calculated distance for these areas is just the average value of the actual distance. Looking at this graph, the calculated distance is generally accurate to within less than half a centimeter. Most importantly, the graph slopes to a minimum from both sides, an important feature if a function is to be used for a PID controller.