Thomas Kost & Hoang Tran
Dr. Briggs
EE3, Lab 1F
December 15, 2018

EE3 Final Project Lab Report

Introduction and Background:

The goal of the final project was to create a IR sensing line follower. The small car uses readings from the IR sensors we create to determine the location of a black line and correct the strength of the motors to keep the car on the course marked by a black line. To do this we used a circuit composed of several IR LEDs and an IR phototransistor to determine the location of the black line under the car. This data was read in and used to determine the direction the car should travel. We chose to use three sensors spread out under the base of the car and pulled up towards the car to give us a notable difference in values read in by each sensor to determine the position of the line and the direction the car must travel. The program used to guide the car worked on a guardrail approach—that is, the car would travel in one direction until a sensor says otherwise. For instance, if the car moves too far to the right, the rightmost sensor would trigger and move the car to the left. This keeps the car roughly centered on the track and able to follow the line.

The sensors in the bottom of the car measure the amount of IR light being reflected by the surface—darker colors will radiate more of the IR radiation and thus the sensor will note more IR light. Because our final sensor design consisted of an IR phototransistor, an analog reading point, and then a resistor after it our Arduino read in high values when the amount of IR light increased. This is because the internal resistance of the phototransistor is very large in comparison to the resistor when there is little IR light, thus dropping most of the voltage across the transistor. But when the transistor's resistance drops the Arduino will read most of the voltage and produce a high analog value. This behavior can be explained by the voltage divider equation shown below. Because the presence of the black line produced a high value, we used a threshold value to determine if the movement associated with the sensor should be carried out. That is, if the sensor read high enough, we would turn left, turn right, or keep moving forward as was associated with the sensor. Additionally, our program stopped the opposing wheel when making a turn to ensure our car was able to make any turn—we will later explain the difficulties our motors presented, and thus the need for this measure.

Voltage Divider Equation

$$Vout = Vin(\frac{R}{R + Rtransistor})$$

Testing Methodology:

Test Designs:

Diagram 1 depicts our final circuit schematic for the line tracer. Note that several of aspects of the circuit are marked so that they may be referenced in general later. Note that a schematic detailing the motor setup with the TIP120 is shown in **Diagram 2**.

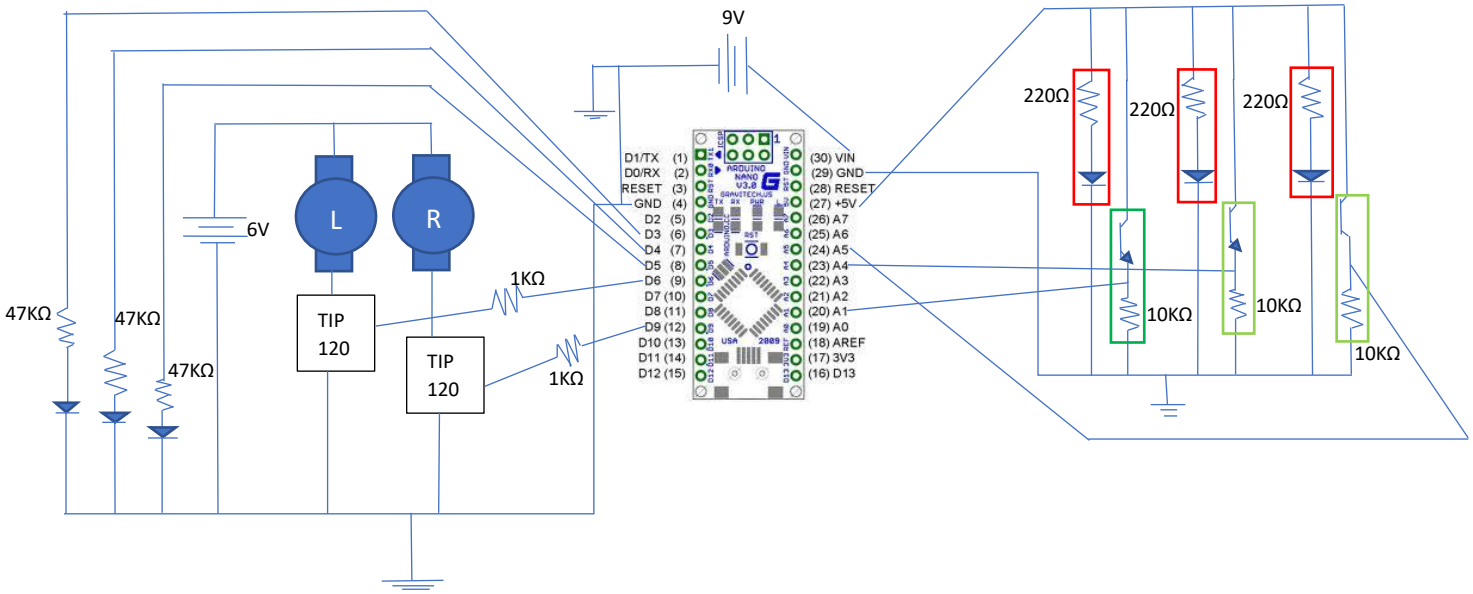


Diagram 1: Line Tracer Circuit Diagram [1] This diagram details the circuit layout for our final design of the line tracer. Note that the red boxed denote the IR LED/ IR light emitters and the green boxes denote the IR Sensors (the phototransistor and analog read pin). In the diagram, any component linked to the ground symbol is assumed to be connected to the same single point ground—they are simply separated for readability.

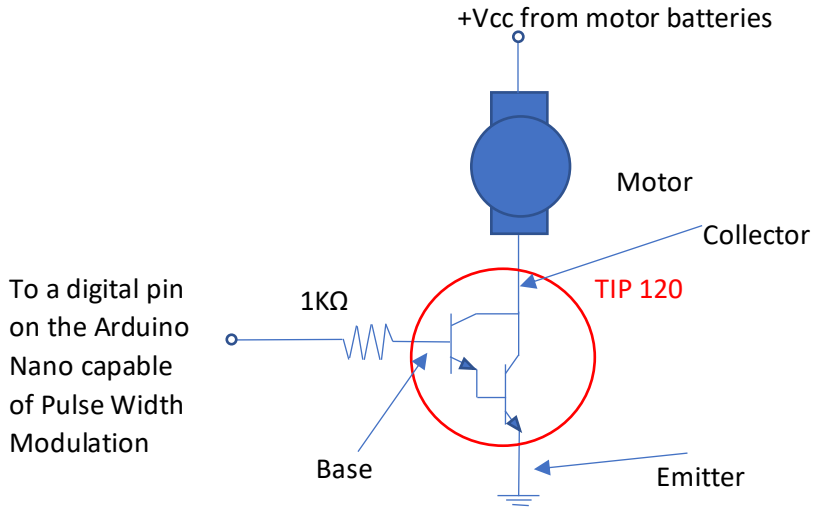


Diagram 2: TIP 120 Motor controller circuit in greater detail [2]

We tested two different circuits for creating IR sensors—denoted by the green boxes in **Diagram 1**, the first one and more difficult to cope with is shown in **Diagram 3**:

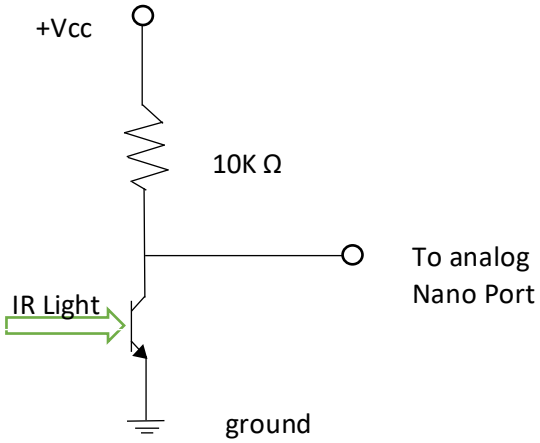


Diagram 3: Read-Low sensor circuit [3]

Because the voltage is read before the phototransistor, the voltage is high when the sensor is not triggered. Therefore, when the phototransistor senses IR light, the resistance significantly drops, and the voltage value read is near zero. This configuration was difficult to create a sensor fusion equation that could allow us the option to implement PID controls.

The second Circuit was a simpler circuit to use and produced more logical values. This circuit is the one we eventually used on the final version, shown in **Diagram 4**:

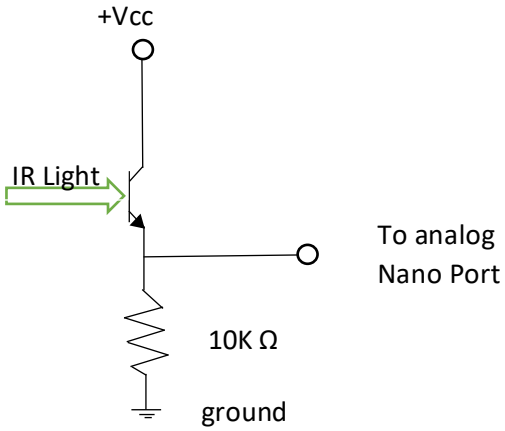


Diagram 4: Read-High sensor circuit [4]

Because the resistor is after the node where the voltage is read the read voltage is near zero when there is no IR light—recall the effects of the resistance of the phototransistor being much greater. As more IR light enters the phototransistor the resistance decreases, which raises the voltage read in by the analog port. This made the processing of data easier because we had peaks for the location of the black line and near zero values for a white space. This gave us the option of more easily creating a PID control system. We also found this setup to be very sensitive—we experimented with resistance values of 10k, 50k ,100k, and 1M ohms and found 10k to give us a gain of nearly the full 5 volts with relatively little noise.

We also tested three physical designs of the sensors. The first design was simply the phototransistors and IR LEDs placed in the bread board located on the bottom of the car. This sensor setup provided very sharp and clear results—the curves created by the sensors were sharp peaks that gave clear values. However, the leads were quite long and thus were subject to change in their movement—making results difficult to repeat. This is depicted in **Diagram 5**:

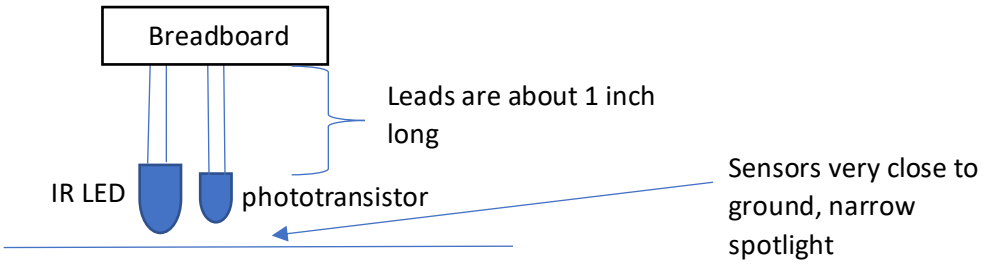
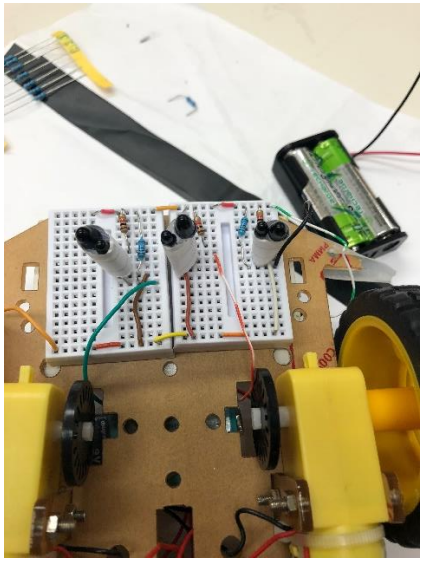
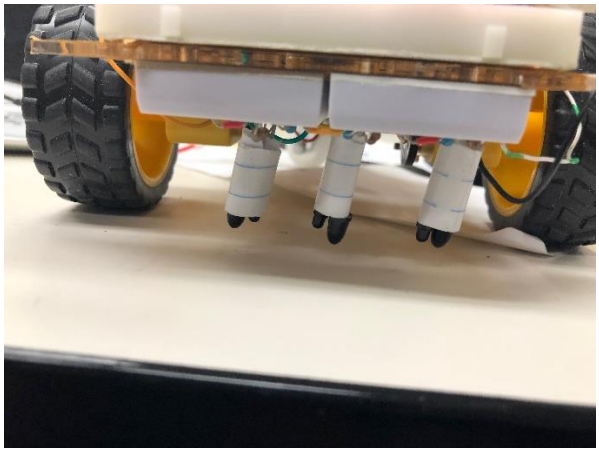


Diagram 5: Long Lead Design

This is a head on view of the sensors—circuit depicted in the green box of diagram 1, the image depicts a single sensor. Note that the car has three of these on the breadboard on the underside of the car.

The second physical change we made was the addition of paper holders to keep the led and phototransistors in place. The paper was wrapped into a circle with a divider in between the phototransistor and the IR LED—this ensured the leads did not short, that the LED and phototransistor were close together and directionally aligned and were not as susceptible to movement. This configuration produced our first successful test result. However, we continued to explore other options later. The following photos shows the paper wrappings used for the forward, left, and right sensors [5]:





The third and final design we used was pulling the sensors—each IR LED and phototransistor pair up towards the car. This was done for two reasons. The first was that it would make the car less susceptible to markings found on the track, ensuring that the car would not deviate from the path. The second reason was to create a larger overlap of the sensors. By pulling the led and phototransistor up, we gave each phototransistor a larger section of the track to look at. This created a slight overlap in the areas each sensor was responsible for. This ensured that as soon as one sensor was no longer triggered, the adjacent sensor would be triggered. This was important to ensuring our car (which ultimately used a guardrail approach to guide the car) would not fishtail down the track. To ensure that we still got adequate readings on the analog pins, we reduced the resistance in series with the LED from 1K to 220 ohms. This change reduced our original time of 37 seconds to complete the track to 31 seconds. The final setup is shown in **Diagram 6**:

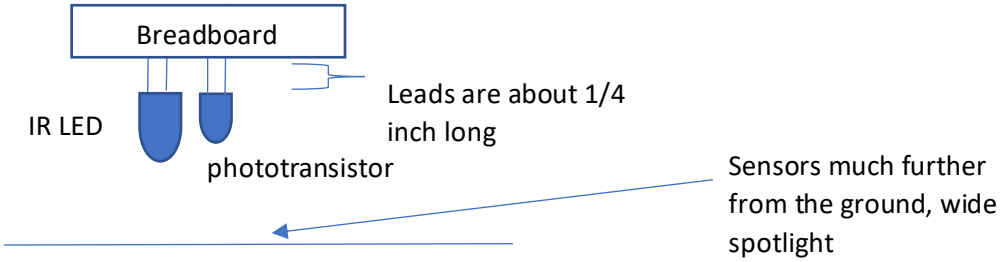


Diagram 6: Short Lead Design [6]

This is a head on view of the sensors—circuit depicted in the green box of diagram 1, the image depicts a single sensor. Note that the car has three of these on the breadboard on the underside of the car.

Test Conducting:

Analytical Tests:

PID Calibration:

The data that was collected through swiping a black line on a piece of paper underneath the car so that each sensor can see the line at one point or another [6]. The table containing the data is shown in **Table 1** below (there are nearly 600 values, so for readability the table will be attached as a link—graphs of the data are depicted in the analysis section):

Table 1: Data generated from swiping a black line under the car

https://docs.google.com/spreadsheets/d/1pO0GBg-SQO9LESK4Ygl3mp7E4Oi0AG822jdtGygPVS4/edit?usp=sharing

Guardrail Calibration:

The same data collected in the PID calibration (seen in **Table 1**) was used to the guardrail calibration. However, this data was taken more frequently and used in the Overlap Analysis described below.

Experimental Tests:

The bulk of our experimental testing was done in simply trying to get the line follower to complete the course. Therefore, the following tables will mostly describe the test results in terms of success or failure. Each row describes a test performed, each test was conducted multiple times. We then considered the behavior the car was exhibiting and tried a change to the code, the control method, or the physical setup. This data is displayed in **Table 2**:

Method of Control	Threshold Value (L,R,F)	Left turn PWM (L,R)	Right Turn PWM (L,R)	Forward PWM (L,R)	Physical Setup	Success	Time (s)
PID	N/A	115±Error	135±Error	Error =0	Low Sensors	No	N/A
Guardrail	500,500,500	130,135	115,165	115, 135	Low Sensors	No	N/A
Guardrail	500,500,500	130, 145	115, 175	115, 145	Low Sensors	No	N/A
Guardrail	500,500,500	130,155	115, 185	115, 155	Low Sensors	No	N/A
Guardrail	500,500,550	130,155	115,185	115,155	Low Sensors	No	N/A
Guardrail	400,400,550	130,155	115, 185	115,155	Low Sensors	No	N/A
Guardrail	400,400,550	130, 155	115, 185	115, 155	Protected Sensors	No, but better	N/A
Guardrail	400,400,450	130,155	115,185	115,155	Protected Sensors	No	N/A
Guardrail	400,400,450	115,0	0,155	115,155	Protected Sensors	Yes	37
Guardrail	400,400,450	130,0	0,170	115,155	Protected Sensors	no	N/A
Guardrail	250,250,300	115,0	0,155	115,155	Pulled Up Sensors	Yes	31

Table 2: Testing Record

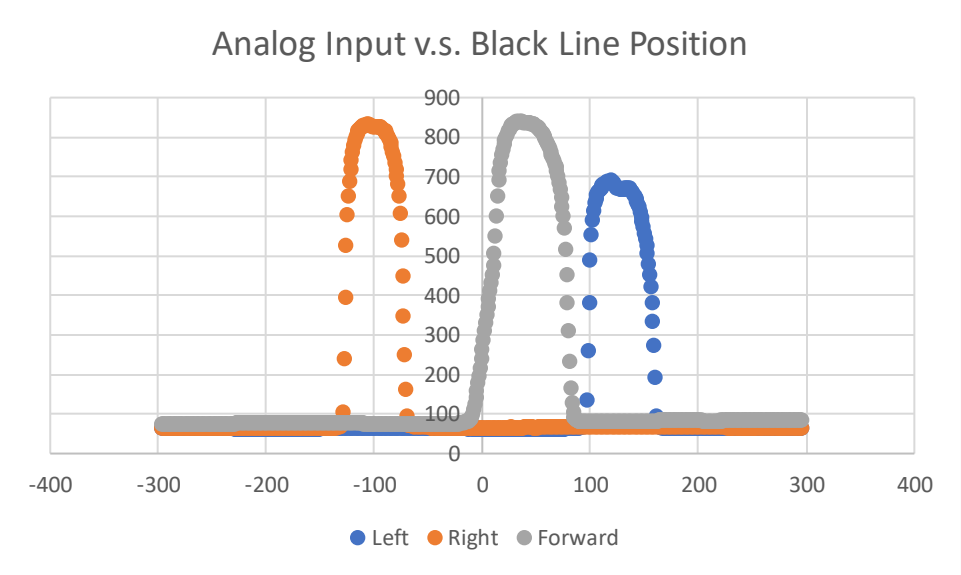
The table displays a rough summary of our trial and error. The threshold value column is organized by the analog input values used as a trigger for the left, right, and forward sensors respectively. For the PWM columns, the values shown are the values written by the Arduino to control the left and right motors respectively. The table focuses more on the changes we made that resulted in failure, because these resulted in the most drastic changes. Most successes and the small changes made after produced near identical times to complete the course. The last row was our final design.

Data Analysis:

PID Analysis[7]:

One of the main data analyses we performed was an attempt to be able to establish PID controls on the car. Note that this mode of control was not our final method of controlling the cars path. We ran into issues with our motors being too uneven—one of which was essentially an all or nothing motor—making slight adjustments impractical.

The data we gathered for the creation of PID controls was a curve tracing of the analog values read from each sensor against distance—note that the distance is not scaled as the actual distance was not needed, but simply a relative numbered position. The distance is approximated, but accurate enough. We obtained this data by copying the data output into the serial monitor while we slowly swiped a black line on a white sheet of paper horizontally underneath the car. The data produced the following plot shown in **Graph 1**:



Graph 1: Analog Input plotted against approximate position and by sensor

The graph displays the value read by each sensor (on a scale from 0 to 1023 shown on the y-axis) as a function of distance. Because we do not care about the actual distance but only the relative distance, we do not need to scale it to an accurate distance. We only care about the order of the points, so the numerical order in which they were recorded suffices and is used as the x axis of the plot. Note that the left sensor was adjusted with a constant *c* to give it a peak equal to the other two sensors.

We attempted to linearize the graphs using the transformation described in class. The transformation is shown in the two functions below (**Equation 1 & Equation 2**):

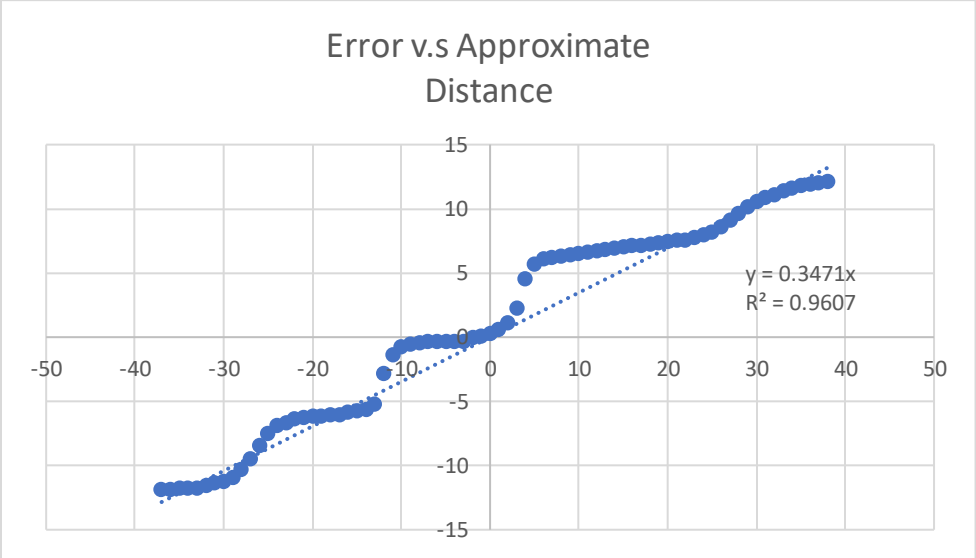
Equation 1:

$$LEFT\ Error = \frac{500}{Fs} - \frac{500}{c * Ls} - 6 : Fs\ is\ the\ forward\ sensor\ and\ Ls\ is\ the\ left\ sensor$$

Equation 2:

$$RIGHT\ Error = \frac{500}{Fs} - \frac{500}{Rs} + 6 : Fs\ is\ the\ forward\ sensor\ and\ Rs\ is\ the\ Right\ sensor$$

The transformation—along with the removal of a few redundant points caused by inconsistencies in the moving of the paper—is shown in the following plot seen in **Graph 2**:



Graph 2 Error as a function of approximate distance

The error—plotted on the y-axis is calculated using **Equation 1** (for left of the origin) and **Equation 2** (for right of the origin). As before, the approximate distance does not necessarily have a correlation to the exact distance between sensors. Rather it is an indication of where in the order of the recorded data the Error value falls. So more negative values indicate the car is veering to the left, and more positive values indicate that the car is increasingly veering to the left.

Through taking a linear regression of **Graph 2** (done above) and setting the intercept to zero, we found that the distance from the center of the track to the center of the car could be modeled by **Equation 3**:

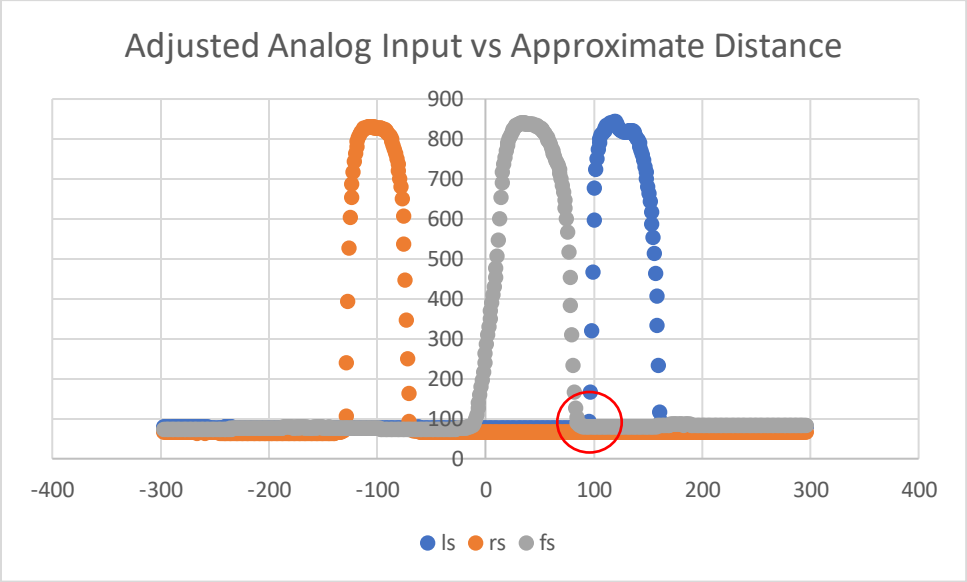
$$Error = 0.3471 * distance$$

$$\therefore distance = \frac{Error}{0.3471}$$

Overlap Analysis[8]:

Another analysis we performed was more key to the guardrail approach we ended up choosing. This was not necessarily a calculation heavy analysis, but it took a look at the curves traced out by moving the paper horizontally across the sensors. When we do so

slowly, we can see the points of intersections of the lines (illustrated below)—that is at what analog value the sensors read the same value. We ran this analysis multiple times, each time adjusting the sensor to put the overlap values at about 200-250. This value we found to be the ideal value for the sensors to be as close as possible to one another, but not have conflicting readings. This value also worked well as an intersection point was far below the threshold value calculated for our sensors (in the final trial we had peak values of around 500). Ultimately, this allowed us to make the changes in steering to be less dramatic and occur quicker, as the sensors more readily noted difference in the position of the line underneath.



Graph 3: Adjusted Analog Input plotted against approximate position and by sensor

Note that the adjusted Analog input is used here to make the location of the intersection between sensors more visible. The graph shows the intersection occurring just above 100 on the analog scale (it is marked by a red circle). Through moving the sensors, we were able ton bring the peaks closer to one another and elevate the point of intersection to around 250 on the analog scale.

Test Data Interpretation:

PID Interpretation:

The most important product of our PID analysis was **Equation 3**, which gave us a relation of the error values calculated from sensor input. Equation 3 gave us a scalar value to calculate distance based from the error value. This allows us to convert the sensor inputs into a position value—this value can be input into the PID controls to determine the position and rate of change of position of the car. Ultimately, this will allow us to control the car with greater precision.

The final equation suggests that every unit change in the error function we would have expected a 0.3471-unit change in distance. This can be used to calibrate the motor values and control the path of the car. Unfortunately, one of or motors was almost entirely an all or nothing motor. This prevented us from making smaller changes in the power the motor output and lead us to abandon the PID method of control.

Overlap Interpretation:

The overlap analysis was important in ensuring the sensors were positioned properly and in determining the appropriate intersection and threshold values used by our Arduino code. The threshold value is the point at which we consider a sensor to be triggered and will follow the direction associated with it. For instance, if the left sensor reads above the threshold the car will turn left. By analyzing the graph, we noticed that our peak adjusted values were at 766 on the analog scale (note this value was determined on the second to last physical design and would change later). This meant that the maximum value we could read in would be 766. So, we determined that 400 would be a good cutoff value, as it was guaranteed to be triggered when the sensor was close to the black line. Additionally, this allowed us to determine a value to allow our sensors to intersect at (where the sensors could be close but never accidentally trigger). We chose 250 as the value, because it got the sensors close to one another but did not risk any accidental triggering of the sensors.

When we later pulled the sensors up towards the car, our peak value dropped to around 550. Therefore, we set the thresholds at 250 and the intersection point at around 200—as the sensors were much more stable this time and less subject to change.

Results and Discussion:

Test Discussion:

The PID analysis culminated in **Equation 3**. This gave us a linear relationship between the values read in by the sensors and the distance the center of the car is off of the black line. This allowed us to create a PID control system for the car. However, as **Table 2** indicates we ran into some difficulties with using the PID control system. The main issue was with our left motor—the motor had difficulties responding to small changed in the PWM value written to the motor. Furthermore, once it turned on, the motor was essentially running at full power and was much stronger than the right motor. Because we could not adjust to motor, PID controls essentially went out the window.

After that, as **Table 2** shows, we moved to a guardrail approach to control the car. This had much better success but did not initially work either. Our initial approach was to set the car's motors to a roughly equal power and increase the power of the motors to either turn left or right. When we did this, we noticed that the car was having difficulties making a left turn. The car would turn left, but not enough and would lose the track. In an attempt to remedy this, we continually increased the power of the right motor to push the car more left. As we continued to increase the value, we noticed very little change, so we moved on to changing the sensors to correct the behavior. However, we did notice that our initial characterization of the motor's

equal power values was incorrect. We retained what appeared to be the new equal point and modified the sensors. We moved the sensors closer together and gave them a paper casing as described in the Test Designs section. This made our sensors more stable and caused the car to correct its path faster, keeping it more in line with the track. We used the Overlap analysis described above to determine how close we could place the sensors to one another without interference in signal. This helped us get towards the end of the track, but again we noticed the same difficulty in turning left. Errors began to pile up by the end of the track, and with the car not able to turn left enough it would lose the track. At this point, we came up with another idea to ensure the car would stay on the track. We chose to make the program for turning only turn one motor—guaranteeing that the car would turn enough to stay on the track. When we implemented this, the car made it all the way to the end of the track and stopped at the horizontal lines. Note that our sensors were quite sensitive, and thus we had little problem with recognizing the stopping condition. However, even though this design was functional, it was quite slow. We decided to continue making changes in the hopes of speeding the car up. We initially tried to just increase the speeds of the motors, however this occasionally caused the car to turn too much and meet the stop condition by facing perpendicular to the track. Because this was not reliable, we decided to change the sensor configuration in an attempt to reduce the fishtailing we had observed. We cut the leads of the sensors and IR LEDs short, and decreased the resistance in series with the LED's to make them brighter. Doing this gave us a wider spotlight from the LEDs and therefore, a wider range of locations each sensor could perceive. This had the intended effect of spreading out the curves shown in **Graph 3**. As a result, there was greater overlap, and the sensors responded to smaller changes in the position of the line). Of course, we had to change the threshold values as well—our peaks were at 500 and our intersections around 200, so 250 was our chosen value (see Overlap Analysis Interpretation for a more detailed explanation). Ultimately, this became our final design and it saved us 6 seconds from our previously functioning model.

Race Day Discussion:

While we did have some initial difficulties on race day, our car did perform to all the specifications provided for the project. Our initial difficulty can be attributed to the phototransistor of the forward sensor being drastically moved—we had not initially noticed the change. But this was quickly fixed by running a few of the Overlap analyses by viewing the serial plotter of the Arduino. This allowed us to adjust the sensor to the previous position giving us correct readings once again. After doing this our car ran the entire course and stopped at the end in 31 seconds.

Our car met all of the specifications of the project, but there were a few things of the performance that could be improved. One of the main aspects we could improve upon would be the time taken to traverse the track. One possible solution we would like to explore to improve the function of the car would be to try running the motors at a higher voltage. We hope that at a higher voltage the motor that operated as an all or nothing motor might have more intermediate values we could use. This would allow us to implement the PID controls we had prepared—which would have the end goal of making the cars journey across the track smoother, and thus faster. PID controls, implemented properly, would help to eliminate the fishtail pattern that our car exhibited—this would give a more direct path likely decreasing on

the time of the run. Another option would be to help eliminate the fishtail would be to experiment with further changes to physical designs. One option could be using dividers between the sensors, so they can be placed quite close together. This would cause the time elapsed between the moment one sensor is triggered to the moment another sensor is triggered to be minimal. Therefore, the car would likely not deviate far from the line, making smoother transitions across the track.

Conclusions and Future Work:

Our design final design for the line tracer met all the specifications for race day. It followed the black line drawn on the paper, turned on the appropriate signal lights for the direction it chose to move, and stopped on command (the horizontal stripe at the end of the track). We were also able to implement the extra credit—a wheel sensor and kickstart program to get the car moving on a hill. One extension I would be interested in pursuing with more time would be making the car pilotable. That is, that you would not need to simply rely on IR sensors in the car to drive it. This could be done by making the car Bluetooth or Wi-Fi enabled (possibly with an ESP chip). This would allow for the driver to send override commands to the car and control the car's path manually.

Over the course of the project, one of the most important things we learned was problem solving and being able to work around difficulties encountered in the project. One of the biggest issues that we faced was that one of our motors was fairly unresponsive to PWM. The motor would not move until a given value—115 in our case—and shortly after the motor was operating at near full power. This created a host of problems—firstly, it derailed the option for using PID controls. Secondly, it prevented the car from being able to fully turn right. Even with the right motor written to full power the left motor was too strong. While the car would turn, it would not turn enough and consistently lost the track in the process. This took some figuring out that we could compromise time and smoothness of travel by completely stopping one motor while the car is turning. This while slower and much more prone to fishtailing, allowed our car to successfully complete the track. We learned that the solution that we wanted to work is not necessarily the most effective solution. It was important to learn that it is fine to abandon our personally favored designs for what is more effective. In doing the project we became more comfortable with scrapping current designs in favor of something that should work better. We got more comfortable learning from failure and making changes that address the issues displayed.

Illustration Credits:

1. Arduino, June 29, 2015, "Is there any reason the Tx/Rx Pins are marked so strangely?", Arduino Forum, https://forum.arduino.cc/index.php?topic=332898.0

References:

1. M. Briggs, 2018, "The Project", EE3, Week 5
2. M. Briggs, 2018, "The Project", EE3, Week 5
3. M. Briggs, 2018, "The Project", EE3, Week 5
4. M. Briggs, 2018, "The Project", EE3, Week 5
5. M. Briggs & TA's, In-class suggestion for sensor stability, EE3
6. M. Briggs, 2018, "PID Concepts Natcar", EE3, Week 6

7. M. Briggs, 2018, “PID Concepts Natcar”, EE3, Week 6
8. M. Briggs, 2018, “PID Concepts Natcar” (idea of looking at intersection), EE3, Week 6