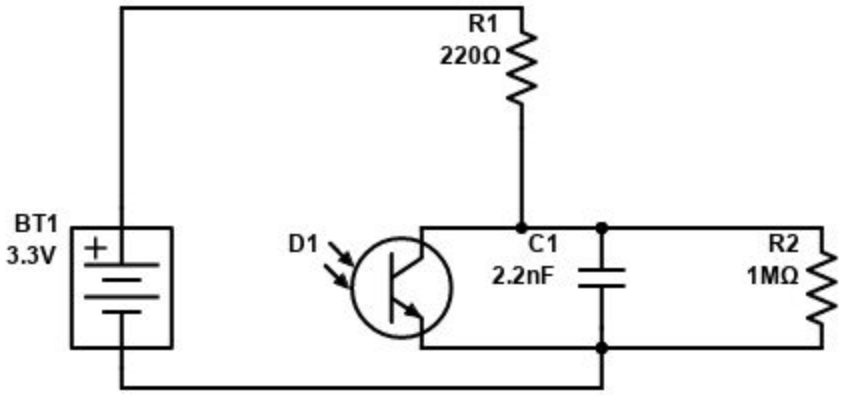
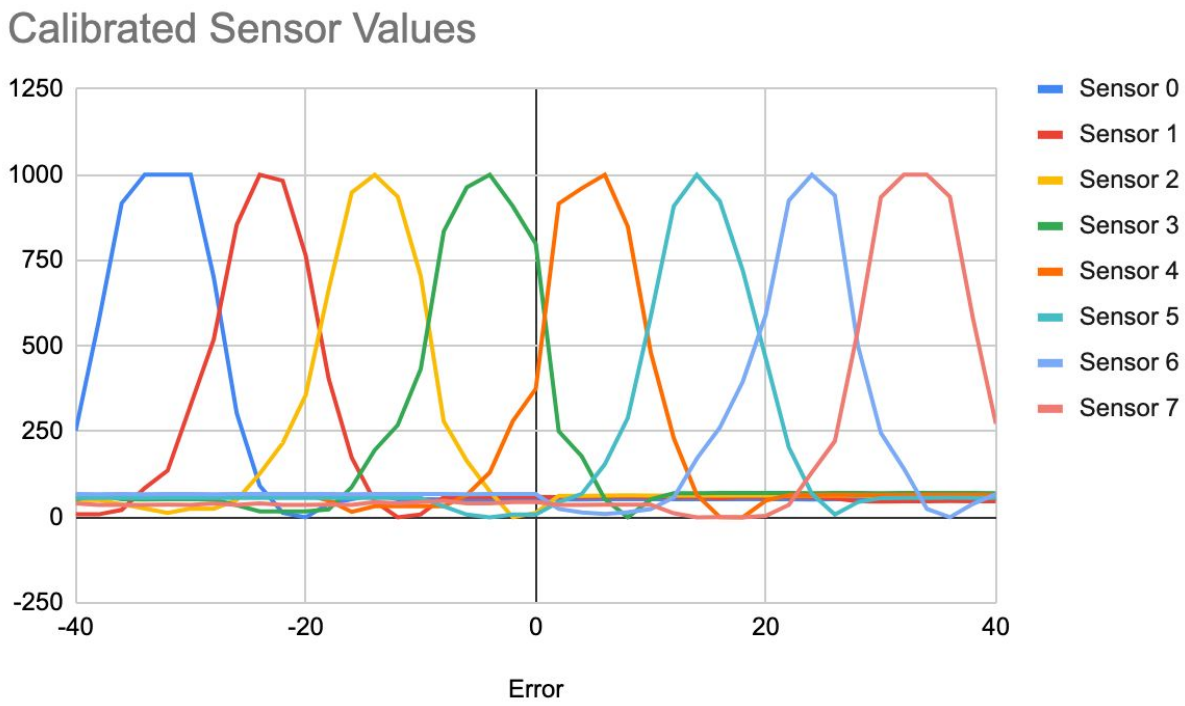
1. Introduction and Background

The goal of this project was to program the TI development robot car to follow a curved track, turn around at the end, follow the track back to the beginning, and stop at the beginning in the shortest time possible using the onboard suite of eight infrared sensors on the bottom of the car to guide its steering. These sensors work by projecting infrared light and measuring the amount of light received. The basis of this receiver circuit is the NPN phototransistor. Just like a normal BJT, the phototransistor has a base, collector, and emitter, but the distinguishing factor is that the base is not open and is tied to a light sensitive material that supplies different amounts of base current depending on the amount of light of a frequency band present. Therefore, we can treat this phototransistor as a variable resistor of sorts dependent on the amount of infrared light present. The receiver circuit takes advantage of this part by putting it in conjunction with a capacitor to form an RC circuit.

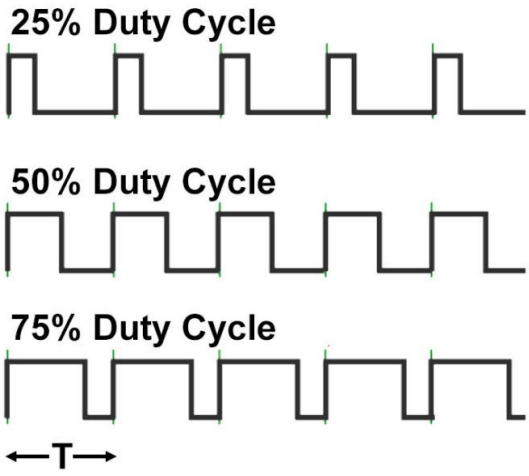


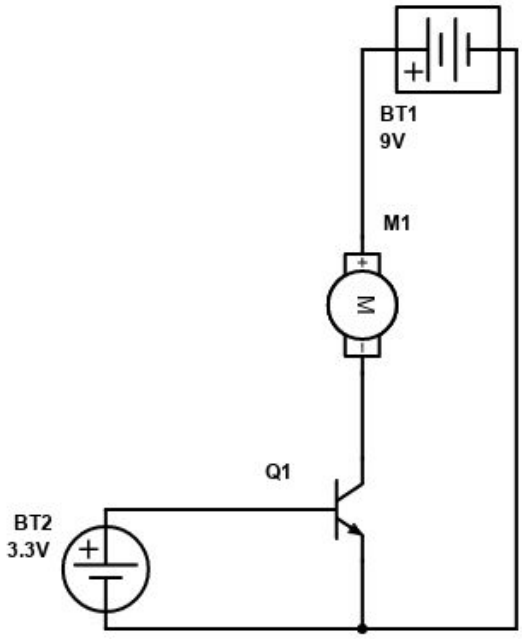
If we measure the time it takes the capacitor (C1) to charge or discharge by monitoring it with an input from the breakout board (modeled as a high impedance resistor R2), we can then figure out the value of the effective resistance of the phototransistor (D1), then the current that must be going through the base, and correlating it with the phototransistor's datasheet, the intensity of the infrared light it measures. For our specific setup, this system took values of 0 for the maximum light measured, and 2500 for no light measured. With the variability of real world parts, however, these 8 measurement circuits were bound to produce values that differ compared to each other, and this is why we chose to calibrate the sensors so we could compare the values they produce without having to worry about variability. To do this, we gathered test data, zeroed their minimum values, and scaled their maximum values down to a range of 0 to 1000. This way, regardless of the variability between components, all sensors reported values of 0 to 1000.



To convert these sensor values into a control algorithm, we had to fuse them into a single value that we could create a motor output from. After consideration, we decided that the best way to combine these values was the weight them with increasing intensity from the inner sensors to the outer sensors and with opposite polarity, then sum them so that we end with a final fused value that is zero if the car is directly in the middle of the track, and exponentially increasing positive or negative intensity as it veers to the left or right.

We chose to convert this fused value into motor control using a PD control system. The proportional controller was selected because it was simple, we had prior experience with it, and it was an effective way to quickly react to and minimize the car's error, and the derivative controller was added as an effective way to stabilize and smooth out the over corrections of the proportional controller, which was especially useful on reacting to sharp corners. After the output intensities were computed from the control algorithm, they were output as a PWM signal (from BT2) with a duty cycle equal to the respective intensities for each motor to the active pin of the transistor (Q1) that switches the double-A batteries (BT1) between floating and the motor (M1).





2. On-track development methodology
    a. Test setup

Our test setup primarily consisted of the straight line track to help tune kp, the official track for more complex testing, a spare piece of completely dark and completely white paper for examining sensor values when confused, and computers with the car's software loaded for easy tuning.

b. How the tests were conducted

The variables we had to work with in our system were kp, kd, the maximum speed, and the minimum threshold to start correcting. The first variables that we chose to examine were the threshold because it was mostly independent of all the other variables and thus would stay the same between the testing with all of the other variables. To figure out a good value for this variable, we set the car on the calibration strip and constantly printed out and monitored the fusion value to figure out what the threshold between fusion values that could be considered noise and actual error was, then implemented it in the code and ran it a few times to make sure it made sense. After we settled on the threshold value, we decided to hold the speed constant and try to find kp and kd. We knew that kp and kd would vary as we increased the speed, so it made sense to us to hold the speed constant to try to find reasonable values for a slower speed which would then give us a solid starting point for tuning for higher speeds. For kp and kd, we first started trying to find kp and ran the car first on the straight track, trying to find a value that would make the system stable with small oscillations. After it could complete the straight track, we took it to the official track and kept tuning kp so it could complete the track repeatedly. We then concluded finding a starting value for kp and started tuning kd. We only tuned kd on the official track because it had the highest rates of change of error and thus the easiest to discern effect. Our methodology for tuning kd was to use it to smooth out the oscillations and jitters of the kp only system, so we studied the car as it went on the track after each run and raised kd if the car either went off the track by overcorrection or appeared unstable and oscillated continuously around the line. It is also worth noting that kp was altered at points during the process of finding kd as we realized that the addition of the derivative term to the proportional controller changed how robust the system was. After we achieved a stable system for the speed that we had picked at the beginning of this process, we raised the speed to make the car run faster and started the process again, but starting with the previous kp and kd values instead of starting from scratch.

c. Data Analysis

Threshold	**Speed**	**Kp**	**Kd**	**Observation**
500	0	0	0	Seemed to be too low, could trigger without moving much
4000	0	0	0	Too big, had to move car too far to start correcting
1500	0	0	0	Tiny amount too long to kick in when diagonal
1000	0	0	0	Seems to be a good amount
1000	100	0.01	0	Way too fast
1000	50	0.01	0	Off track, still seemed fast for starting

1000	30	0.01	0	Corrected in wrong direction, switched in code
1000	30	0.01	0	Corrected one wheel at a time, kp seems high
1000	30	0.005	0	Realized one wheel at a time was a coding error related to unsigned integers, fixed it
1000	30	0.005	0	Found another coding error related to how output signal was actually being output
1000	30	0.005	0	Missed the corner , decided to change weighting to 1,2,4,8,8,4,2,1 instead of 8,4,2,1,1,2,4,8
1000	30	0.005	0	Missed the corner
1000	30	0.01	0	Overcorrection caused missed turn
1000	30	0.007	0	Overcorrection caused a missed turn
1000	30	0.002	0	Completed to the end, but not gracefully
1000	30	0.0002	0	Still completed to the end and looked much better
1000	30	0.0002	0.01	Very unstable on turns
1000	30	0.0002	0.001	Proportional seemed to overpower kd
1000	30	0.0002	0.005	Unstable on turns again
1000	30	0.0002	0.002	Seems to be nice combination of proportional correcting and smoothness
1000	50	0.0002	0.002	Understeering missed turn
1000	50	0.001	0.002	Oversteer pretty badly and missed turn
1000	50	0.0005	0.002	Completed track, looked unstable though
1000	50	0.0004	0.002	Completed track and looked better, but still jerky
1000	50	0.0004	0.008	Understeer caused off track on turns
1000	50	0.0004	0.007	Understeer caused off track on turns, closer
1000	50	0.0004	0.006	Completed track and less jerky than before increasing kd
1000	50	0.0004	0.0055	Settled on this combination for speed 50
1000	90	0.0004	0.0055	Bad understeer, missed first corner

1000	90	0.001	0.0055	Oversteered past second corner but not terrible
1000	90	0.0005	0.0055	Understeer again missed the first corner
1000	90	0.0008	0.0055	Completed track, jerky and lots of oscillations though
1000	90	0.0008	0.01	Not stable, went straight line then tried correcting too late in the turn
1000	90	0.0008	0.008	Worked so well! Completed course smoothly
1000	120	0.0008	0.008	Understeer and went off track
1000	120	0.001	0.008	Oversteered off the track
1000	120	0.0009	0.008	Didn't complete, understeer
1000	120	0.00095	0.008	Off track, understeer, revert to lower speed
1000	100	0.0008	0.008	Completed track and worked well
1000	100	0.0008	0.008	Off track (think was due to new batteries after previous trial)
1000	100	0.00081	0.008	Completed track, but not every time
1000	100	0.00083	0.008	Completed track more than before but still not robustly
1000	90	0.0008	0.008	Slightly understeer but still completed track
1000	90	0.00081	0.008	Settled on this for compromise on speed and robustness. Final time ~10.3s with new batteries

d. Test Data interpretation

Looking through the observations in the data table above, it seems that the primary reasons for the

3. Results and Discussion
    a. Test Discussion

While we were happy to some extent with the performance of the car in testing, we were not completely satisfied. The end system completed the course robustly and achieved the base goal of getting through the track with the given criteria, more work would definitely have been required to get the car to complete the track faster. As is seen towards the end of the data table, we were not able to tune the PD system to work repeatedly with speeds faster than 100 as we

had hoped, which severely limited the desired speed with which we could complete the track. We think that if we had continued to tweak kp and kd and potentially starting finding kp and kd at the higher speeds from scratch, we might have had more success in making the car go faster, but we also found that operating at those higher speeds was immensely more difficult to tune because factors that are hard to control like battery voltage played a much bigger role in the control system, and any tweaks we made had a much more profound impact than at lower speeds which made it harder to find the optimal values.

b. Race Day Discussion

i. Race Day

Our vehicles performed fairly well on race day. Our expected times were under 11s, around 10.3s, and one of our cars achieved a best time of 10.7s while the other achieved a time of 11.3 seconds which we believe was due to not replacing the batteries again before race day. We are a little disappointed that both cars didn't get under 11 seconds, but are happy that they were both robust and didn't have any issues completing the track.

ii. Limitations

As discussed in section 3a, we would've liked to be able to go faster, but could not seem to find the right combination of constants to enable this. Besides that, for reason on race day, the car seemed to be less smooth in operation than it was during testing, which we would classify as a limitation because it had to slow part of its track run to correct for these larger than usual oscillations. We're not sure why this was, but we would've liked to know this before it happened so we could fix it.

iii. How do differently if did again

If we had the chance to do this project again, there are three main things we would've changed. First, we wouldn't have started tuning the PD system with a lower speed because while it was helpful for gauging a general bound of the constants, there was so much variability between speeds that the time spent tuning for each speed was mostly wasted and could've been used to put more effort into tuning a faster car. Second, we would've changed batteries more regularly. As already discussed, at higher speeds, having new batteries created a noticeably large difference between the same constants having slightly used batteries, and ideally we could've had more batteries to eliminate this variable. Another idea we had that would've even better eliminated this variable if we had more time was building a voltage regulator (most likely a buck converter) to step down the voltage to some constant level and keep it there so regardless of the voltage of the batteries, the motors would always be supplied the same voltage and thus power. Lastly, if we realized that we got more than one chance to have a successful run, we would've pushed the car harder and sacrificed some robustness for a higher speed. We thought that we only got one or two tries to get the car to run so we made sure that the car completed the task almost every time, but we went faster in trials and made the decision that complete robustness was more important than speed.

4. Conclusions and Future Work

Our design met the base goal of completing the track in the specified way, but there was definitely still work to do to make it something we would consider fast after seeing everyone else's times. For starters, we were not able to increase the speed of our cars without sacrificing robustness, and this would be the first place to start improvement. We believe that we built a system that can be used to go faster, but we would need to spend more time to tune it correctly. Another point of future work would be to build a voltage regulation system for the batteries to ensure constant voltage supplied to the motors as discussed at the end of section 3, which would aid in our task of being able to tune the system for higher speeds. Another idea we had was to implement another row of lateral sensors offset behind the already existing ones that we could use to create a better control system. Having only one row of sensors means that there was no way to distinguish between the car being solely offset and offset and turned in one direction, but installing another row of sensors would mean that calculating the angle at which the car is turned from the track would be possible, and we could create a better control system that would be able to correct differently between these two states. Lastly, unfortunately we were not able to collaborate together in person which created difficulties for communicating, problem solving, and creating this system, so further work would also necessitate solving this virtual collaboration problem in some better way.

5. Illustration Credits

PWM: https://www.nightsea.com/articles/pwm-led-dimming/

6. References

Lectures

7. Code