Ana Gu, Noah Himed

Team EndGame

Final Report

Introduction and Background

This report provides a summary of ECE3 final project of coding for a path-following car using IR sensors and Energia. The car will further recognize a “school zone”, in which its speed will be reduced by half when entered and resume previous speed once exiting the school zone.

The line-following car was guided using proportional, integral, and derivative(PID)control closed loop system, in which the car recognized and self-corrected any deviations away from the middle two IR sensors. The user must have an understanding of PID control systems, PWM cycles, IR sensors, and basic if-else statement coding. Using Pololu QTR-8RC, a sensor module which has 8 IR LED/phototransitors mounted on the bottom of the car, this module will output eight digital signals that will output individual HIGH signals when the car is over a black line. Sensitivity of the IR sensors can be further modified by individual teams using delay().

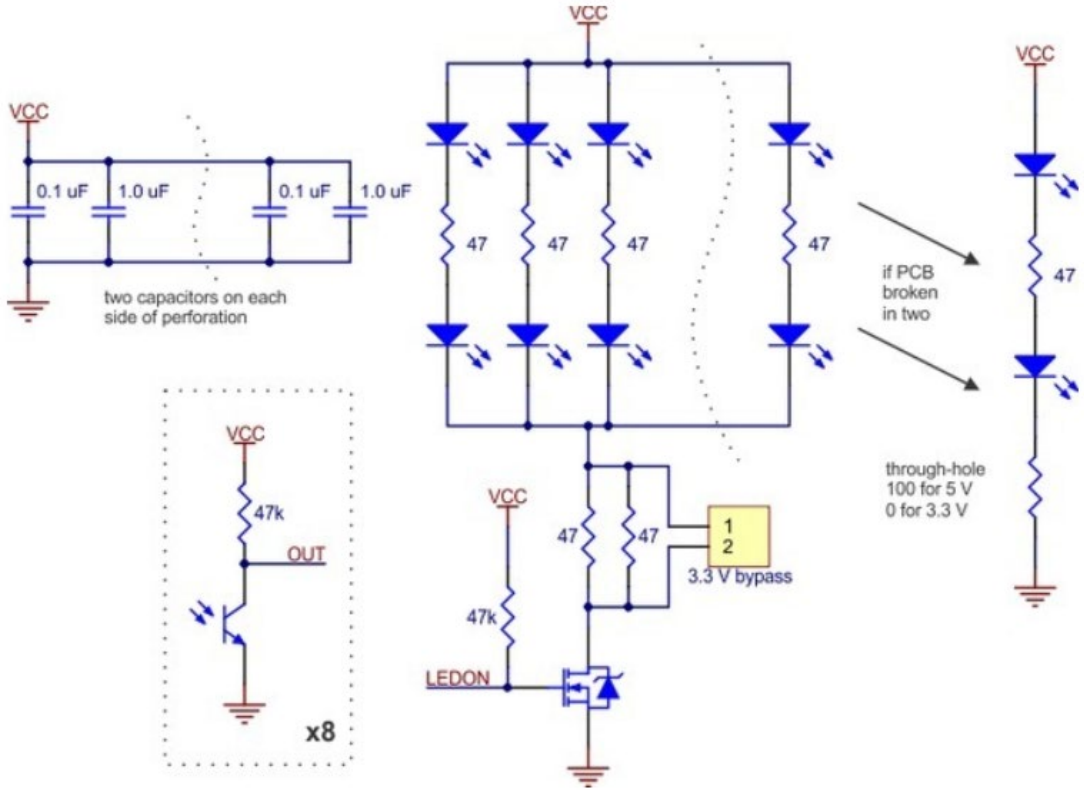


Figure 1 Pololu’s schematic for the QTR-8RC.

Once the channel pin is turned into an output and set HIGH, the circuit effectively flips the switch shown from position B to position A, causing the capacitor to discharge its stored charge and begin to act like a short, allowing all of the voltage Vcc to appear across Vm. Then, the channel pin is

turned into an input, causing the circuit to flip its switch from position A, back to position B. This causes the capacitor to begin charging, and therefore taking on voltage across Vc. During this process, the voltage of Vm is measured from the channel pin as the circuit acts like a basic voltage divider during this stage. The rate at which voltage builds across Vc--and consequently the rate at which voltage drops from Vm--is determined by the current through the photodiode, which depends on the resistance of Vm . This resistance level is higher when the photodiode is exposed to less light, causing Vm to drop faster when the sensor is above a black line. A resistor is attached to the terminal which the depicted switch attaches to in order to impose a 1 MΩ input impedance on the current flowing from the capacitor to allow the voltage measurement at the channel pin to not affect the current flowing through the photodiode. After waiting for a delay for this process to occur once the switch is moved back into position B, the voltage Vm is sampled from the channel pin. Because Vm decreases faster when the sensor is above a white or light colored surface and slower when the sensor is on top of a black line, the sampled voltage will be lower and read as either above or below a predetermined threshold as a 1 or a 0 respectively; this allows the system to determine if a sensor is on top of a black line [2].

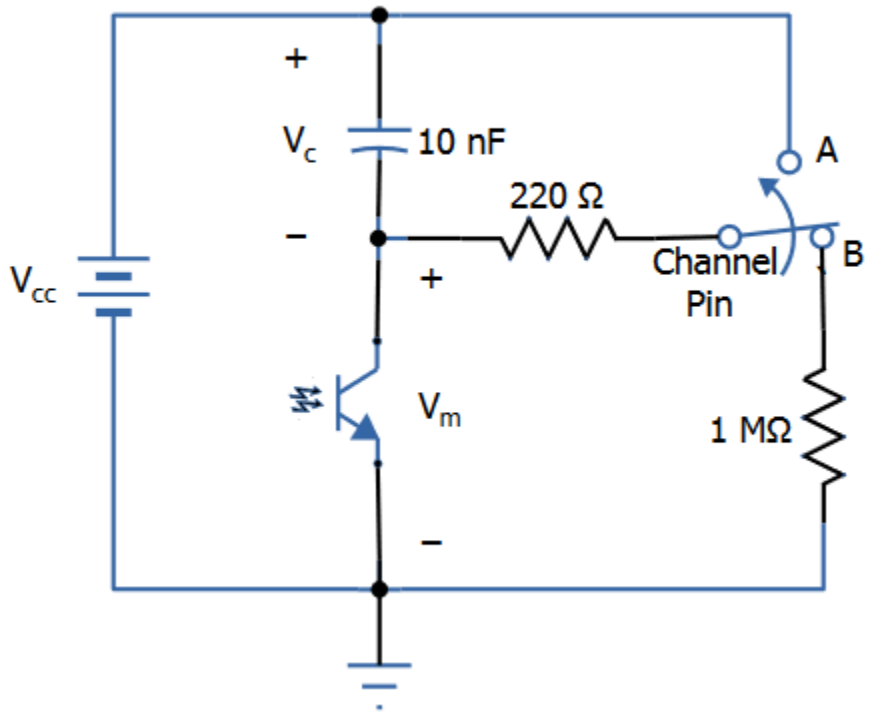
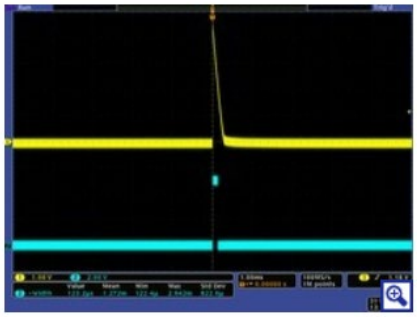
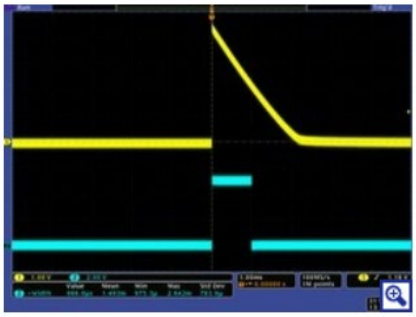


Figure 2(Taken from Practice Problem Set 5) This circuit is the same as one of the eight IR sensors on the RSLK.

A



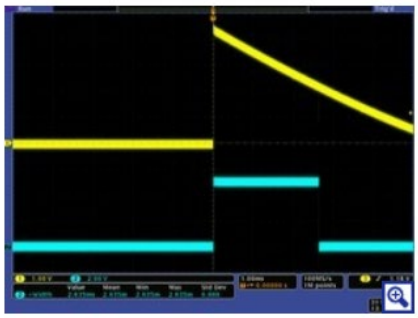
QTR-1RC output (yellow) when 1/8" above a white surface and microcontroller timing of that output (blue).



B

QTR-1RC output (yellow) when 1/8" above a white/black interface and microcontroller timing of that output (blue).

C



QTR-1RC output (yellow) when 1/8" above a black line and microcontroller timing of that output (blue).

Figure 3 (From https://www.pololu.com/docs/0J13/all) Three oscilloscope screen captures that shows the voltage output of the IR sensors. The blue oscilloscope channel represents the output of the digital signal of the QTR8RC. A was taken over the white portion of the surface, B was taken at the edge of the tape, and C was taken while the car was fully over the black tape. [4]

Testing Methodology

To test if the sensors are reading in values correctly, papers with black bars down the middle of the sheet were placed underneath the car and outputted the sensor data to Energia serial monitor. Since the IR sensors only read 0's and 1's, the user can track how the car would react when programmed to recognize straight lines. If the car were to veer off to the right, the pins left of the middle two would read 1's and vice versa.

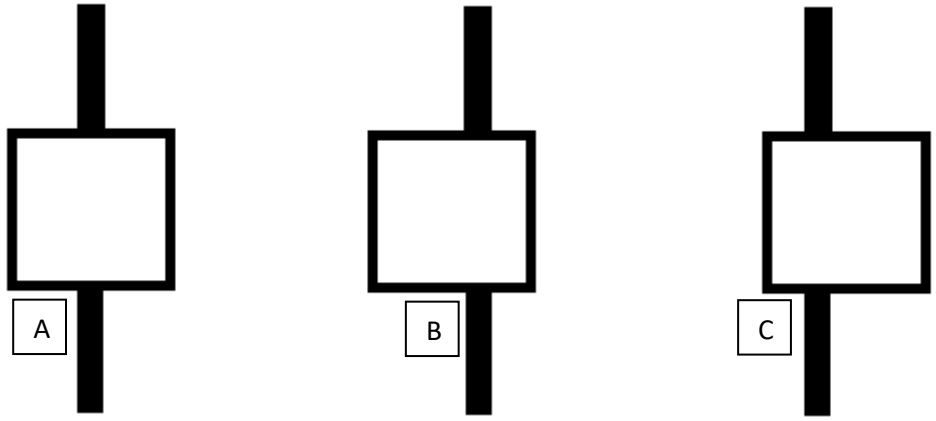


Figure 1 (made using Paint 3D) A shows a car following the path correctly. Serial.print() would produce 00011000, which means the IR sensors detect only the middle two lanes. B shows a car that is veering towards the left, Serial.print() would produce 00000110. C shows a car that is veering towards the right, Serial.print() would produce 01100000 .

Initially, the 1's were flipped from what was expected from the output from Energia. The pins were set incorrectly as our group initially placed the car backwards with the sensors facing the back instead of the front. Adjustments were made to ensure proper detection.

After proper detection was ensured, the team noticed that that the serial plotter had some frequent 00111000 or 00011100 while dead center on the middle of the track. This is due to the set delay() being too sensitive. Since delay() takes only integer number, the team had to use delayMicroseconds() in order to adjust the sensitivity.

Different weights for each individual sensors were given. Initially, Dr. Briggs provided a helpful tip on setting the different weights of the sensors[1]. Our group found the weight to be incompatible with the PWM cycle set and increased the weight of the outer sensors on both ends.

Positions	**Initial**	**Final**
10000000(leftmost)	-1.75	-4
01000000	-1.25	-2
00100000	-.75	-1
00010000	-.25	-.25
00001000	.25	.25
00000100	.75	1
00000010	1.25	2
00000001(rightmost)	1.75	4

KP	KD	Success?	Time	Comments
10	2	no		spinned in circles
5	5	no		spinned in circles
1	2	no		veered off without correctly
1	5	no		followed the beginning straight part but did not make the turn
1	10	no		failed at sharp turn
3	40	no		failed at sharp turn coming back
2	38	yes	43.2 s	followed path correctly under initial pwm as 70
2	38	yes	54.3 s	followed path correctly under initial pwm as 60
1	20	yes	45.2 s	followed path correctly but did not work over pwm >60, very jolty
1	20	yes	45.2 s	followed path correctly but did not work over pwm >60, less jolty after averaging error
1	15	no		did not work >70 pwm and could not make the turn back, did average error
2	15	no		did not work when pwm = 60, failed to turn properly at sharp curve, did average error

Figure 3 Trial and Error data from determining the proper KP and KD values. Implemented using Excel.

The ratio of KD/KP seems to be about 19-20 for the car to follow the track successfully. Initially, the car was very jolty but still followed the line correctly. The problem was that the car was making PID decisions based on every preceding position. The "joltyness" was reduced once the group took an average of the last five readings of position. Correction at a low speed took very little time but was a lot harder to fine tune once our group bumped the PWM cycle from 50 to 70. In row 13 and 14, the group tried to test out different ratios at higher PWM but would not make the sharp turn towards the beginning of the track. In the course of testing, we took the average of 3,5, and 7 previous positions and found that the average of last 5 positions worked to be the smoothest.

Results and Discussion

Monday's discussion was allotted 70 seconds due to two disastrous lab section meetings as 14 cars were accidentally shorted. Team End Game completed the track in 40.3 seconds, tying for third out of 8 teams. The car was able to follow the line correctly with relatively smooth control. There was no need to change anything in the code on test day. The car travelled faster than developmental testing by an average of 4 seconds.

There are several limitations of the code used. Firstly, the car cannot surpass above 75 PWM cycles as then the KD and KP values would need to be adjusted accordingly. At the end of the track, the car is supposed to turn 180 degrees and follow the track once again. Limitations to how the turning was implemented is how the car recognizes the black line once it turns. Right now, the code uses delay() in order to circumvent the car from detecting the black bar more than once. In a setting in which there are multiple black lines along the track, our code would not work because of its failure to count the number of the times it's seen the black line.

If we were to redo this project, the turning around would be implemented without using delay(), as that is just an easy way without actually letting the car recognize the turning. Instead of starting the car off slow, therefore having more leeway for the PID function, we should start the PWM cycle at 80 and then fine tune using different weighted sensor systems to tailor the car better for higher speeds.

Conclusion and Future Works

Overall, the car worked exactly how it was supposed to in the time allotted. In the future, increasing the PWM cycle to 100 and ensuring the car turns properly is the next goal. Fine-tuning the KD and KP values along with implementing Ki would foolproof the car. The car would have a black line counter implemented using millis(). An if condition would be implemented to keep track of sensitive readings as the IR sensors would read in multiple 11111111 signals while passing the track. The code will use the millis() function that keeps track of the time and if (millis() – previousTime > 1000), then the counter would increase by one.

In the future, our group can implement a Bluetooth module since keeping the car connected to a USB in order to read the serial.print() data hinders the car from performing while also being an inconvenience to the user. In addition to being a line following car, the TI-RSLK also can be implemented with bumper switches such that it can sense blockage in the front [3].

References

[1] Adapted from Dr.Briggs HW 5

[2] from https://www.pololu.com/docs/0J13/all

[3] from https://university.ti.com/en/faculty/ti-robotics-system-learning-kit/ti-robotics-system-learning-kit

[4] from https://www.pololu.com/docs/0J13/all