Amy Yu
905138432

ECE3 Spring 2020 Final Report

Introduction and Background

The goal of this project was to apply concepts learned through this course and apply it to a line-following car. The car had to be able to complete both a straight-line and ribbon path, while doing a 180 degree turn at the end to traverse the path again and return to the starting position. For the ribbon path, the car was required to stop after arriving back at the starting point. A Texas Instruments RSLK learning kit was used to achieve this. This includes a sensor array with eight IR/LED phototransistor pairs on the bottom [1]. This is used to detect changes in reflectance on the ground, which is perfect for detecting a black line on a white surface. A RC circuit is the basis for our phototransistor circuit, as we learned in the Week 3 Lab.

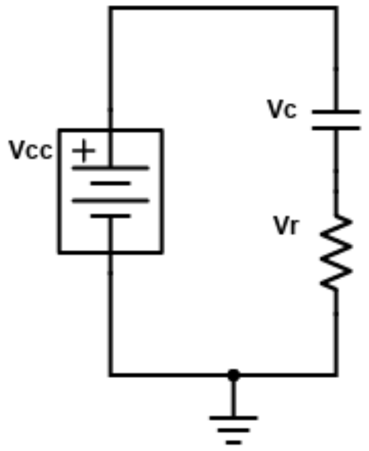


Figure 1. RC circuit.
Created with https://www.digikey.com/schemeit/project/.

A phototransistor can be thought of as a variable resistor, where the phototransistor will have less effective resistance if it is bright. The data is read through the change in the RC measurement time. From the equations $V_c(t) = V_{cc} - V_{cc}e^{-\frac{t}{RC}}$ and $V_r(t) = V_{cc}e^{-\frac{t}{RC}}$, we can see that the smaller the time constant, the faster V_c will converge to its steady state value (V_{cc}) [4].

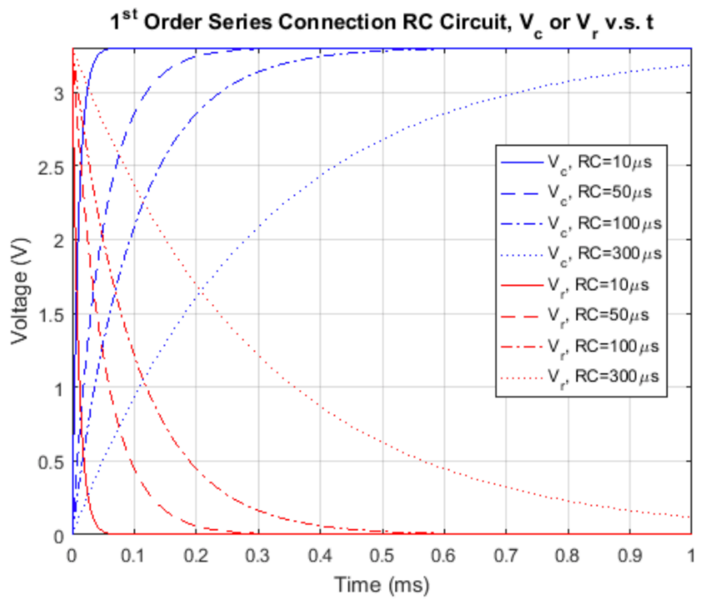


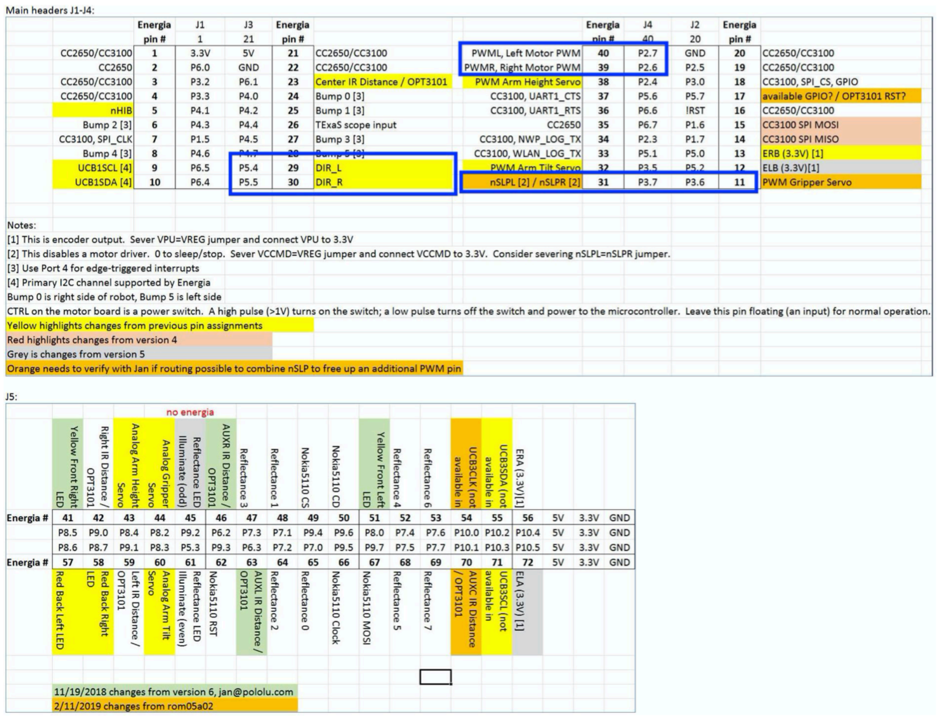
Figure 2. Voltage across resistor and capacitor at different time constants. V_{cc} is set to 3.3V and $V_c(0)$ is 0.
From Stafsudd, O.M., EE3 Introduction to Electrical Engineering Laboratory Manual.

From Figure 2, it can be seen that we reach the steady state after five time constants, which can be used as the settling time. This will change depending on the reflectance, so the time constants can be used to tell us if the phototransistor is detecting a light or dark surface [4].

Testing Methodology

a) Test Setup

The car was physically tested by first inserting batteries and checking for both blue and green LEDs. Then, the Blink code that was provided by Energia, an integrated development environment used to connect Arduino to the LaunchPad, was uploaded to the car [2]. This ensured that code could be successfully uploaded and that the Chassis Board was working. Next, using the pin chart from Figure 3, the appropriate motor pins were identified and code was uploaded to make the car perform a donut, proving that the wheels were working. Lastly, a small portion of the track was printed. The given IR Sensor code was uploaded to the car, and the car was placed on top of the track. This code printed each of the eight sensor values as a number between 0 and 2500, where 0 was maximum reflectance and 2500 was minimum reflectance. The sensor values were taking at two-millimeter increments, and resulted in values ranging from around 400 to 1500.



Main headers J1-J4:

	Energia	J1	J3	Energia			Energia	J4	J2	Energia	
	pin #	1	21	pin #			pin #	40	20	pin #	
CC2650/CC3100	1	3.3V	5V	21	CC2650/CC3100	PWML, Left Motor PWM	40	P2.7	GND	20	CC2650/CC3100
CC2650	2	P6.0	GND	22	CC2650/CC3100	PWMR, Right Motor PWM	39	P2.6	P2.5	19	CC2650/CC3100
CC2650/CC3100	3	P3.2	P6.1	23	Center IR Distance / OPT3101	PWM Arm Height Servo	38	P2.4	P3.0	18	CC3100, SPI_CS, GPIO
CC2650/CC3100	4	P3.3	P4.0	24	Bump 0 [3]	CC3100, UART1_CTS	37	P5.6	P5.7	17	available GPIO? / OPT3101 RST?
nHIB	5	P4.1	P4.2	25	Bump 1 [3]	CC3100, UART1_RTS	36	P6.6	!RST	16	CC2650/CC3100
Bump 2 [3]	6	P4.3	P4.4	26	TExaS scope input	CC2650	35	P6.7	P1.6	15	CC3100 SPI MOSI
CC3100, SPI_CLK	7	P1.5	P4.5	27	Bump 3 [3]	CC3100, NWP_LOG_TX	34	P2.3	P1.7	14	CC3100 SPI MISO
Bump 4 [3]	8	P4.6	P4.7	28	Bump 5 [3]	CC3100, WLAN_LOG_TX	33	P5.1	P5.0	13	ERB (3.3V) [1]
UCB1SCL [4]	9	P6.5	P5.4	29	DIR_L	PWM Arm Tilt Servo	32	P3.5	P5.2	12	ELB (3.3V)[1]
UCB1SDA [4]	10	P6.4	P5.5	30	DIR_R	nSLPL [2] / nSLPR [2]	31	P3.7	P3.6	11	PWM Gripper Servo

Notes:
[1] This is encoder output. Sever VPU=VREG jumper and connect VPU to 3.3V
[2] This disables a motor driver. 0 to sleep/stop. Sever VCCMD=VREG jumper and connect VCCMD to 3.3V. Consider severing nSLPL=nSLPR jumper.
[3] Use Port 4 for edge-triggered interrupts
[4] Primary I2C channel supported by Energia
Bump 0 is right side of robot, Bump 5 is left side
CTRL on the motor board is a power switch. A high pulse (>1V) turns on the switch; a low pulse turns off the switch and power to the microcontroller. Leave this pin floating (an input) for normal operation.
Yellow highlights changes from previous pin assignments
Red highlights changes from version 4
Grey is changes from version 5
Orange needs to verify with Jan if routing possible to combine nSLP to free up an additional PWM pin

J5:

				no energia															
	Yellow Front Right LED	Right IR Distance / OPT3101	Analog Arm Height Servo	Analog Gripper Servo	Reflectance LED Illuminate (odd)	AUXR IR Distance / OPT3101	Reflectance 3	Reflectance 1	Nokia5110 CS	Nokia5110 CD	Yellow Front Left LED	Reflectance 4	Reflectance 6	UCB3CLK (not available in	UCB3SDA (not available in	ERA (3.3V)[1]			
Energia #	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	5V	3.3V	GND
	P8.5	P9.0	P8.4	P8.2	P9.2	P6.2	P7.3	P7.1	P9.4	P9.6	P8.0	P7.4	P7.6	P10.0	P10.2	P10.4	5V	3.3V	GND
	P8.6	P8.7	P9.1	P8.3	P5.3	P9.3	P6.3	P7.2	P7.0	P9.5	P9.7	P7.5	P7.7	P10.1	P10.3	P10.5	5V	3.3V	GND
Energia #	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	5V	3.3V	GND
	Red Back Left LED	Red Back Right LED	Left IR Distance / OPT3101	Analog Arm Tilt Servo	Reflectance LED Illuminate (even)	Nokia5110 RST	AUXL IR Distance / OPT3101	Reflectance 2	Reflectance 0	Nokia5110 Clock	Nokia5110 MOSI	Reflectance 5	Reflectance 7	AUXC IR Distance / OPT3101	UCB3SCL (not available in	ELA (3.3V) [1]			

11/19/2018 changes from version 6, jan@pololu.com
2/11/2019 changes from rom05a02

Figure 3. MSP432 Pinchart.
From Dr. Briggs, CCLE Week 4.

As we learned from class, the phototransistor circuit used to sense the path can be represented as two stages.

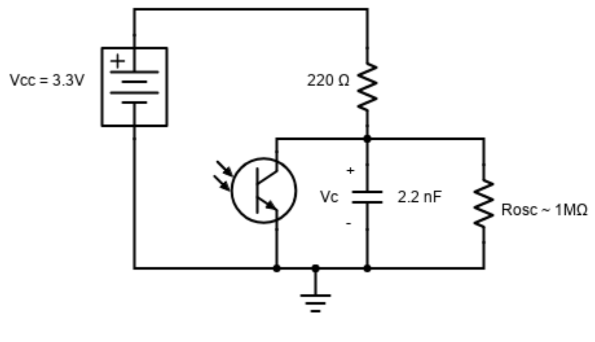


Figure 4. Stage 1 of the phototransistor circuit.
Created with https://www.digikey.com/schemeit/project/.

As can be seen from Figure 4, the phototransistor can be modelled as a variable resistor ranging from around 5kΩ to 1MΩ is used, depending on if the sensor is detecting light or dark.

Using Kirchhoff's Voltage Laws, we eventually get $V_c(t) = V_{cc} - V_{cc}e^{-\frac{t}{C(220\Omega)}}$. This forces the voltage across the capacitor to be around 3.3V and gives us the starting voltage for stage 2 [4].

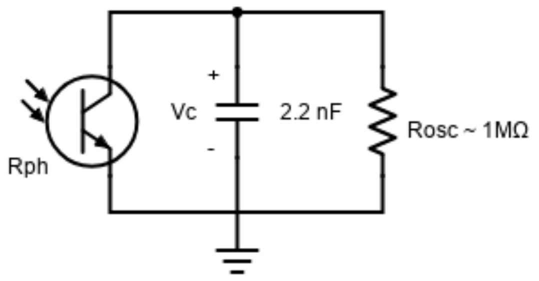


Figure 5. Stage 2 of the phototransistor circuit.
Created with https://www.digikey.com/schemeit/project/.

Similarly, for stage 2, we get that $V_c(t) = e^{-\frac{t}{C*R_{ph}}}$. After five time constants, the charged capacitor will be fully discharged in stage 2. The time constant will be longer than that of stage 1 [4].

b) How the tests were conducted

When testing the car on the track, different sensor weights, proportional controller constants (K_p), derivative controller constants (K_d), and speeds were tested. First, I started by tested different sensor weights as shown in Table 1. Every test was conducted on both the straight and ribbon paths, and the results are described in the table.

Sensor Weights	Results
15-14-12-8	Car overcorrects after doing 180 degree turn and does not return to path for both tracks.
10-6-4-1	Works on straight path, but does not stay on path during ribbon turn.
FINAL: 8-4-2-1	Completes path.

Table 1. Results of testing different sensor weights.

Next, the proportional controller constants and derivative controller constants were tested as seen in Table 2. An extreme value for K_p was first chosen and K_d was kept at zero to be able to see its effect on the car. This resulted in a lot of sharp oscillations on the straight path and the car would not stay on the path during the turn for the ribbon path. After finding a K_p that successfully completed both tracks, K_d values were tested. The same thing was done again, where extremes were first tested. After finding values that seemed to work, the car was placed at different starting positions and adjustments were made as needed. The goal was to have the car smoothly traverse the path and correct the right amount on the turns for the ribbon path.

K_p	K_d	Results
10	0	Lots of oscillations, does not stay on path during turn.
5	0	Lots of oscillations, does not stay on path during turn.
1	0	Stays on path, but does not correct fast enough on turns.
1.5	1	Does not find path when starting on counterclockwise offset.
1.5	10	Does not correct fast enough on turns.
1.5	5	FINAL

Table 2. Results of testing different K_p and K_d values.

Lastly, different speeds were tested. During initial testing, the speed was kept at 55, with the speed during the donut at 100. After the car was able to complete the track at that speed, I increased it in increments of five. I was able to increase the speed to 65 while keeping the donut speed the same. At higher speeds, the car had problems completing the ribbon path.

c) Data Analysis

When testing the car on a small portion of the track, the data was collected and the eight sensor values were plotted with position on Excel. Then, the values were normalized by finding the maximums and scaling them to 1000. Different values were then multiplied with the sensor values and fused, as can be seen in Figure 6. The ideal sensor fusion would be a linear line going through the origin, and the data shows that sensor weights of -8, -4, -2, -1, 1, 2, 4, 8, from left to right, is the most similar to the ideal. This ended up being my final sensor weighting.

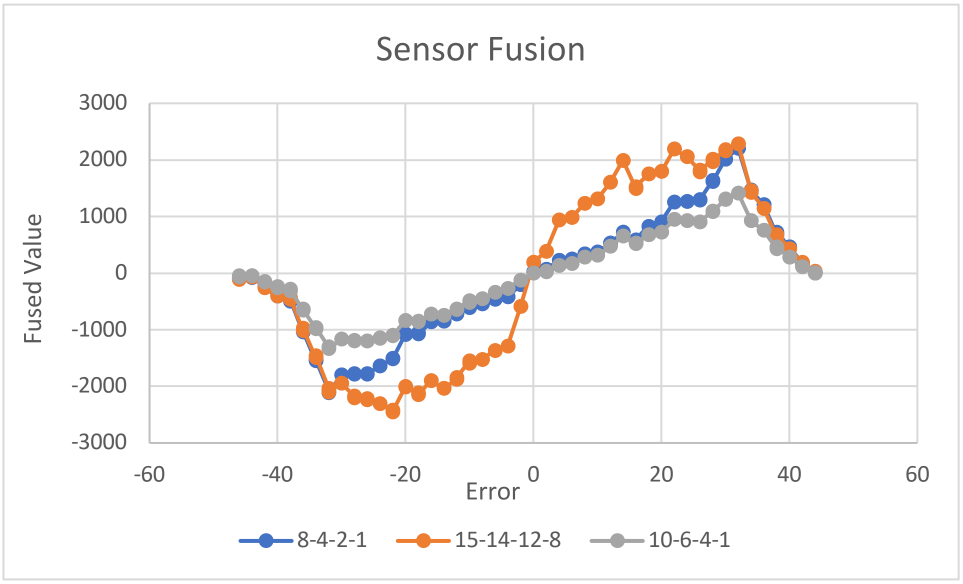


Figure 6. Graph of different sensor weights.
Created on Excel.

d) Test Data Interpretation

As described in Tables 1 and 2, many tests were required to find the values of K_p and K_d. These values were especially important for the ribbon path, as the car had to recognize the blank line and correct to return to the center. The K_p values had to be increased to ensure that the car would follow the line from different starting positions and not only when it is placed directly on top of the line. The K_d values had to be adjusted to decrease the oscillations, but still have enough that it would follow the line. With both K_p and K_d, extreme values such as 10 and 1 were tested, and the correct middle point was found. The final value of K_p was 1.5, and the final value of K_d was 5.

When testing the sensor values, it was found that only one sensor would have a value greater than 1000 at a time, showing that this is the value that determined a black line. Therefore, 1000 was used in the code to detect a black line.

Results and Discussion

a) Test Discussion

The testing that was done was very appropriate to the project goals, as the car was able to successfully complete the track while meeting all the requirements. Sometimes outside factors would contribute to the effectiveness of the tests, such as the lighting in the room or battery power, so many tests had to be conducted multiple times to ensure that the results were accurate. While the testing for sensor weights was sensible since previous analysis had been done through Excel, finding K_p and K_d required a lot of tedious testing. However, they all ended up being useful allowed for the car to perform well on Race Day. Improvements such as increasing speed could be made, but the project goals were still met.

b) Race Day Discussion

i) The vehicle performed very well on Race Day, as it was able to complete both tracks from all the given starting positions on the first try. The car completed the ribbon track, both clockwise and counterclockwise, in around 21 seconds. The final code used is provided and shows the final values used on Race Day.

ii) The track itself had some limitations, as my printer did not print a dark enough line for the car to detect. I solved this problem by covering the track with black electrical tape. One original limitation I had in the code was that the car would think that the intersection in the ribbon path was the end bar and perform a donut. I was able to solve this by adding a counter variable to count the number of times that the car crossed a black bar, perform a donut on the third count, and stop on the sixth count. A delay had to be added for this to work, as the car would otherwise count the same bar multiple times. All of these problems were fixed by Race Day.

iii) Throughout this project, I learned the importance of testing and consistency. I would find that the car would work sometimes and not work at other times, so if I were to conduct the project again, I would make sure to test the car multiple times from all the starting positions, instead of just directly on top of the line.

Conclusions and Future Work

The design met the goal by successfully completing both the straight and ribbon paths from the given starting positions. The car was able to do a 180 degree turn at the end and traverse the path again in a reasonable amount of time. Throughout this project, I learned how to apply many important electrical engineering concepts such as RC circuits and phototransistors. I also learned more about coding using the Energia IDE and how to implement a PID controller. Observing the other cars on Race Day also helped me think about improvements I could make to my own design. If I had more time, I would increase the speed, and this would likely require more testing for better K_p and K_d values to use at the faster speed. Another extension could be to implement the bumper switch, so that the car would be able to determine when it has encountered an obstacle. This would require the bumper switch kit for the RSLK and mounting it to the Chassis Board [3].

References

[1] Pololu Robotics & Electronics, TI Robotics System Learning Kit MAX (TI-RSLK MAX) Parts and Accessories, https://www.pololu.com/category/240/ti-robotics-system-learning-kit-max-ti-rslk-max-parts-and-accessories

[2] Energia IDE, https://energia.nu/

[3] Pololu Robotics & Electronics, Right Bumper Switch Assembly for Romi/TI-RSLK MAX, https://www.pololu.com/product/3674

[4] Stafsudd, O.M., EE3 Introduction to Electrical Engineering Laboratory Manual