ECE 3 Line Following Car

Jamie Liu 204981426

Spring 2019 Lab 1C

Dr. Briggs

1 Introduction and Background

The goal of this project was to create a line-following car using the TI-RSLK (Robotics System Learning Kit) and TI MSP432 microcontroller. In order to meet the specifications, the car was required to follow a curved track indicated by a black line, do a 180° turn at one end, and follow the line back to the beginning of the track before coming to a stop.

The motors of the car are driven by batteries. The sensor board on the bottom of the car was comprised of 8 input pins and 2 power pins, and had an array of 8 IR LED/phototransistor pairs. The schematic diagram of the module is shown in Figure 1. For each pair, the IR LED shines infrared light down towards the track, and the light is reflected onto the phototransistor. Then, the phototransistor detects the light level and has a voltage reading corresponding to the level of light.

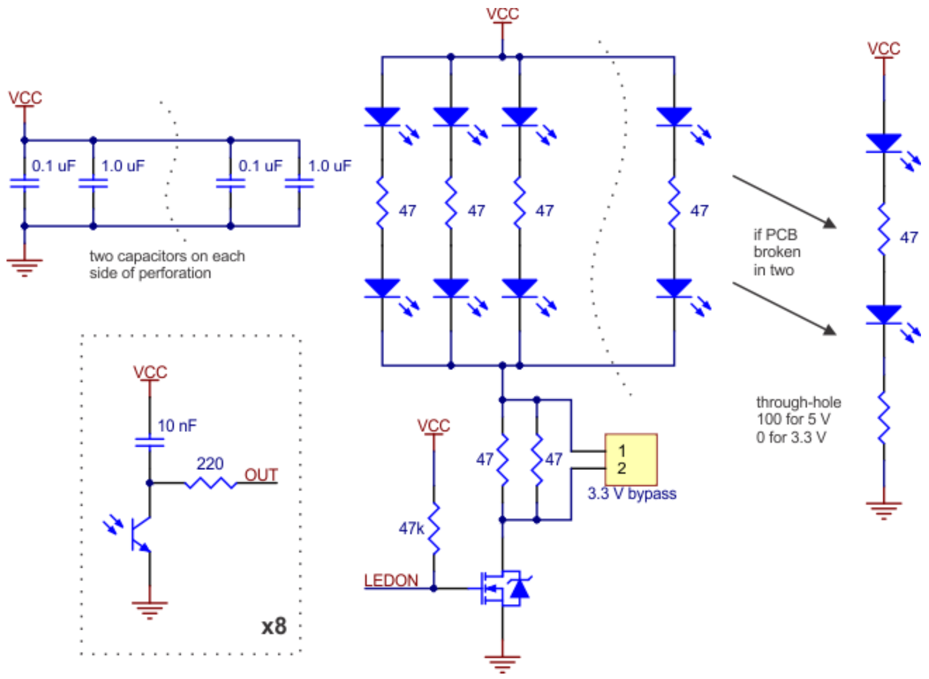


Figure 1: Sensor board schematic (source: https://www.pololu.com/product/961)

Obtaining voltage readings from the sensor array would allow us to determine if the car was above the black line, and if so, what the car's position was relative to the line. For example, if the leftmost sensor detected the black line and the sensors to the right did not, it would indicate that the car was to the right of the line, and would need to adjust its course to the left to stay on the track.

The voltage readings from the sensors were taken as follows. First, the sensor channel pins are set to HIGH, and there is a short delay. Then the channel pins are set as inputs, and after another delay, the channel pins are sampled. See Figure 2 for the schematic. The two delays are set in accordance with the charging and discharging curve for V_m.

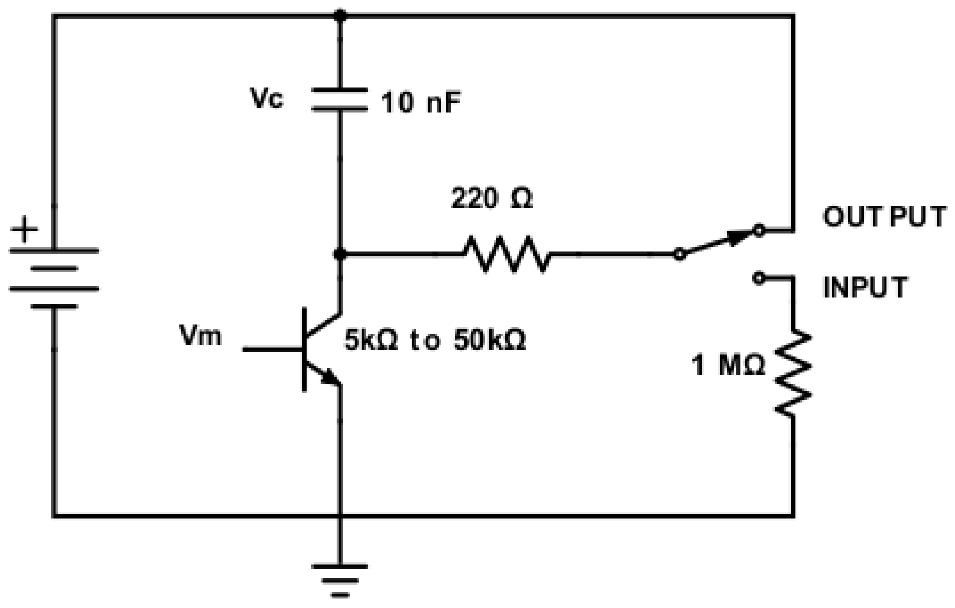


Figure 2: IR Reflectance sensor schematic, created using DigiKey's SchemeIt tool, based off of the circuit shown in lab [1].

The MSP432 microcontroller is mounted on top of the TI-RSLK, and sits on the top of the car. The microcontroller pin map is shown in Figure 3, and the pin chart

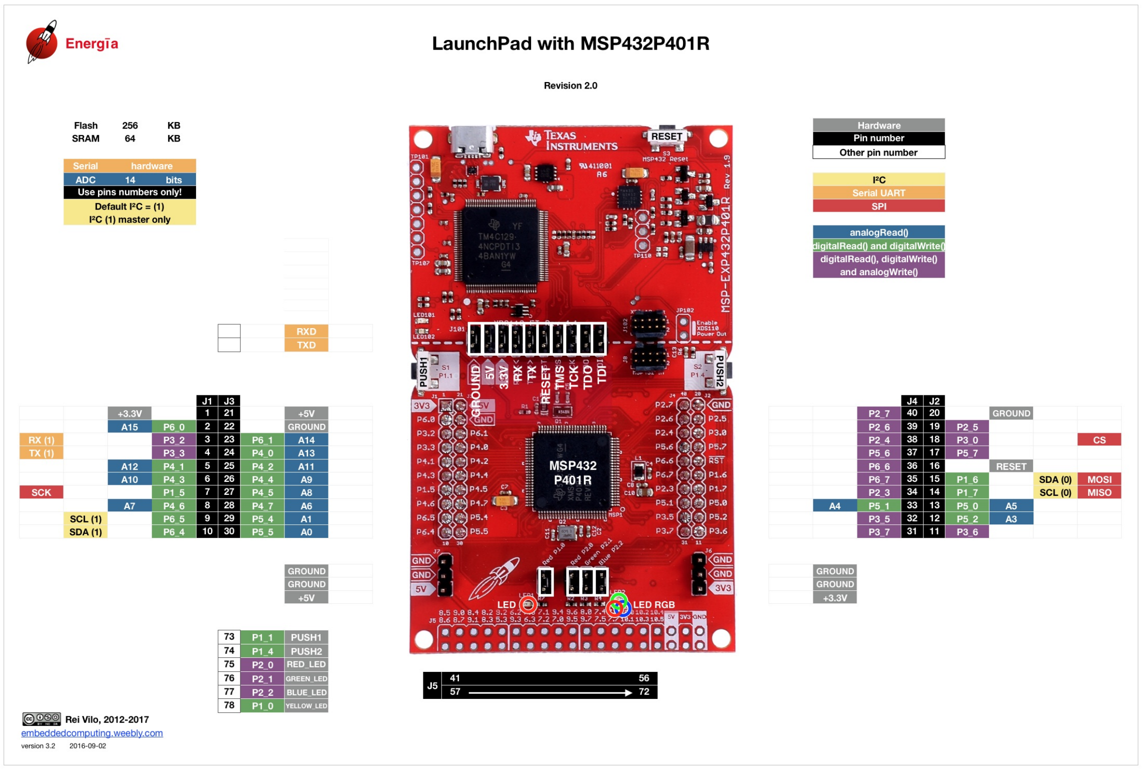
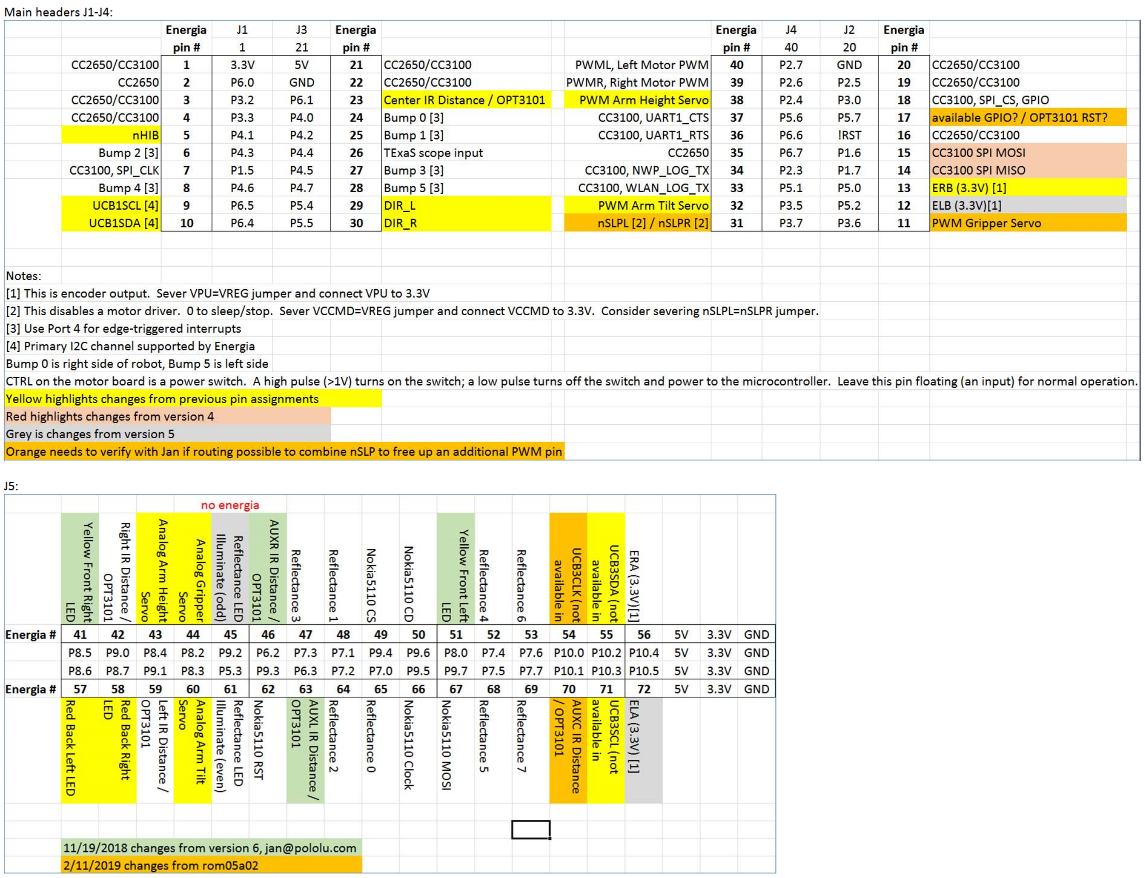


Figure 3: The MSP432 pin map (source: http://energia.nu/pinmaps/msp-exp432p401r/)

is shown in Figure 4. In order to program the microcontroller, we use the Energia IDE, a fork of Arduino specifically for Launchpads from Texas Instruments [2]. The microcontroller's digital pins can be used as input or output pins, and the analog pins can be used to read or write analog voltage values. In order to make the car move, all that needs to be done is to set up the motor's sleep (SLP), pulse-width modulation (PWM), and direction (DIR) pins and use analog writes to the PWM pins.

As the car navigates across the track, the sensor array values are continuously updated, allowing the car to change its direction responsively in order to stay on the track. Since the array consists of 8 sensors, we are able to achieve a finer resolution reading of the car's position compared to having only 2 sensors, for example. In addition, we can use sensor fusion as we learned in class in order to represent the car’s horizontal position relative to the track in a single number. I used the weighted values provided in the fifth set of practice problems. We obtain the number by summing up the weights for the sensors that detected the black line. Then, the single sensor value can be used to determine the car’s movement.



Main headers J1-J4:

	Energia pin #	J1 1	J3 21	Energia pin #			Energia pin #	J4 40	J2 20	Energia 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 4: The pin chart for the MSP432 board (source: project-based files on CCLE)

2 Testing Methodology

2a. Test Setup

First, I wrote a simple program to make the car drive in a straight line in order to verify that the motors were working properly. Then, I began to setup the tests for the sensor delay values. In order to test the sensors, I wrote a simple program to set the sensor pins to OUTPUT mode and HIGH. Then, each pin had a short delay before the pinmode was set to INPUT. After another brief delay, the pin was digitally read in order to obtain a white (0) or black (1) reading. In order to see the data collected, these values were printed out to the monitor as the car was moved to different positions over the line. Initially, I tested the sensors with the first delay being 10 microseconds, and the second delay being 600 microseconds.

2b. How Tests Were Conducted

The first test that was conducted was testing the motor systems. This was done by setting the appropriate motor pins, and letting the car drive across the ground to ensure the path was straight. I noticed that the car tended to skew a little bit to the right, and to adjust for this, I set the right motor's PWM value to be slightly higher than the left motor's. Then, the sensor tests were conducted. I used a sheet of paper with a black line across it, and varied the position of the car while outputting the sensor values. While testing, I noticed that some of the sensors seemed to not have stable values. Rather, they would fluctuate between 0 and 1 even when they were not over the line.

2c. Data Analysis

As mentioned in section 2b, the sensor readings were not stable. A sample output of these readings is presented below:

Similar data was gathered as the car was moved horizontally. When I positioned

Expected Output	Actual Output
11000000	11100000
11000000	11000000
11000000	11110000
11000000	11000000
11000000	11100000
11000000	11010000

the car to have one or two sensors above the black line, some other sensors would occasionally output a black value rather than white as expected. Since the values were always mistakenly read as black rather than white, I realized that the timing of the delays must have been incorrect, leading to incorrect voltage readings.

In addition, once the sensor issue was fixed, I ran test runs on the curved track and found that sometimes the car would be unable to turn fast enough to stay on the track.

2d. Test Data Interpretation

After conducting these analyses, I realized that the second delay in the sensor reading code was too short. This is due to the nature of the sensing scheme. First, when the channel pins are set to high, the value of V_m begins building up according to the charging curve. So the value of V_m builds up during the first delay. Then, the pinmode is set to input, and the voltage begins discharging. The V_m discharging curves for white and black readings have different time constants. Since the voltages being read were incorrect, leading to the 1 values when 0s were expected, the data indicated that the delays were not correct.

The test data shows that the sensors are not getting correct readings. In order to get an accurate reading, the delay during which the voltage is discharging must be short enough that a black reading does not discharge too much, yet long enough for a white reading to discharge enough to be read as such. In my case, the sensors were mistakenly outputting black instead of white, which meant that the delay was too short for the voltage to be discharged enough to get an accurate reading. Thus, I increased the time for the second delay to 700 microseconds, and found that it was sufficient to obtain accurate, consistent readings.

For the issue with the track, I realized that the issue was with the proportional control, in that the error term was not weighted heavily enough to affect the motion of the car. For example, when the car was trying to turn right, the right wheel would move too fast and the left wheel too slow to do a full turn. Rather, it would stray off of the track and lose it completely. So, I decided to fiddle with the constants to make the error have a greater impact on the wheel speeds, which also made the difference in the wheel speeds more dramatic.

3 Results and Discussion

3a. Test Discussion

On Race Day, the car was able to successfully complete the track as expected based on the testing. I did not have to make any adjustments for better performance, since the car performed about as well (if not a little bit better) compared to the average performance during my tests. Thus, my interpretations from the development testing survived running on the track on Race Day.

3b. Race Day Discussion

Unfortunately, I did not take a video of the vehicle performing on Race Day, but it did pretty well! Although it did not go nearly as fast as the fastest car, it was substantially faster at a 37 second time than the required time of 45 seconds. One limitation of my code was that I only used proportional control, as opposed to derivative control, so my car tended to swerve and oscillate around the track a lot rather than being more smooth and stable. If I were to conduct the project again, I would have tried to implement derivative control as well.

4 Conclusions and Future Work

Overall, my design met my goals in that it was able to complete the track consistently and in the required time limit. The project was a culmination of a lot of the concepts I learned during the first few weeks of the course, and I was able to apply the knowledge I gained about circuits and feedback control in order to create a working system. Furthermore, I was able to solidify my understanding of feedback control by using an error term. I found the integration of software and hardware really interesting, since it was my first experience with Arduino and programming a microcontroller. For extensions to this project, I think it would be cool to make the car navigate through a maze, since that could require some sort of depth-first search and backtracking. I think that would be really fun and interesting to watch if I could implement it. Another possible extension would be to have the car navigate some sort of obstacle course by using different sensors or bumper feedback.

5 Illustration Credits

Figure 1: https://www.pololu.com/product/961
Figure 3: http://energia.nu/pinmaps/msp-exp432p401r/
Figure 4: MSP432 pin chart on CCLE

6 References

[1] https://www.digikey.com/schemeit/project/
[2] https://energia.nu/