Justin Aoki
UID: 605084124
William Chong
UID: 205114665

Introduction and Background

The goal of the project was to implement a line-following car that uses infrared phototransistors to detect a path and self correct its direction to successfully complete the course. Further guidelines included meeting the minimum time requirement of 75 seconds and programming the car to turn 180 degrees to traverse up the course and back to the starting line. The built-in hardware included eight infrared LED/phototransistor pairs in a row along the bottom of the car (See **Figure 1** [1]). Being the most crucial element in the design, these phototransistors were used to sense solid black, or the center of the path to follow, and the steady gradient that faded to white on either side of the centerline.

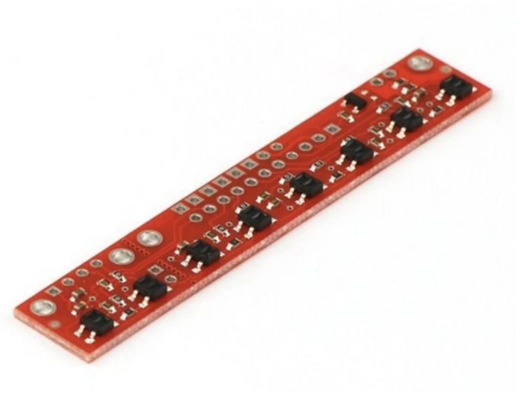
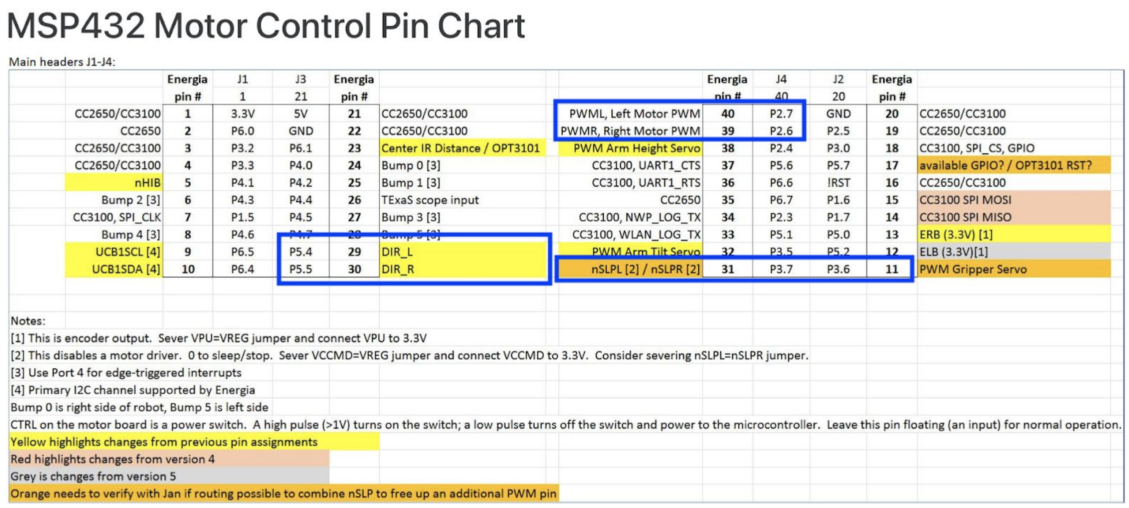


Figure 1. QTR-8RC Reflectance Sensor Array
Source: https://www.pololu.com/product/961

The theory behind the path sensing system involves converting the sensor readings into a measurable value that can be used to correct the car's path. To accomplish this, we needed to read-in the information from the sensors and then adjust the output speeds of the car's left and right wheels (DC gear motors) according to real-time changes in the sensory input. The sensor readings, which ranged from approximately 400 to 2500, were stored in an array with the first and last values corresponding to the outermost sensor readings and pin numbers were assigned using the information on **Figure 2**. **Figure 3** shows the circuit schematic of the Reflectance Sensor Array. The Reflectance Sensor works by utilizing pairs of IR LEDs and phototransistors; the phototransistors pick up the scattered infrared light from the track or surface. As shown in **Figure 3**, VCC provides the sensor array with power; the LEDs are controlled by the LED control pin. In the bottom left corner of **Figure 3**, the "actual sensor" (i.e. the phototransistor) is shown. The phototransistor is in series with a capacitor, which is charged by VCC; the other end of the phototransistor goes to ground. A "Channel Pin" reads the voltage across the capacitor, and allows us to detect the charging time and settling time of the capacitor. This is because the phototransistor-capacitor has different charging and settling times corresponding with different reflectances. The curves are shown in **Figure 4**; this shows how by mapping the different times to different reflectances, we can use the sensor to map different shades of black to sensor values. With assigned weights, each reading from the phototransistor array was properly scaled and used to calculate an error value. Using pulse width modulation, we applied the error value to control the rotational speed of the motors, aiming for smooth movement with little oscillations.



MSP432 Motor Control Pin Chart

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

Figure 2. MSP432 Motor Control Pin Chart
Source: Professor Briggs located on CCLE Week 3

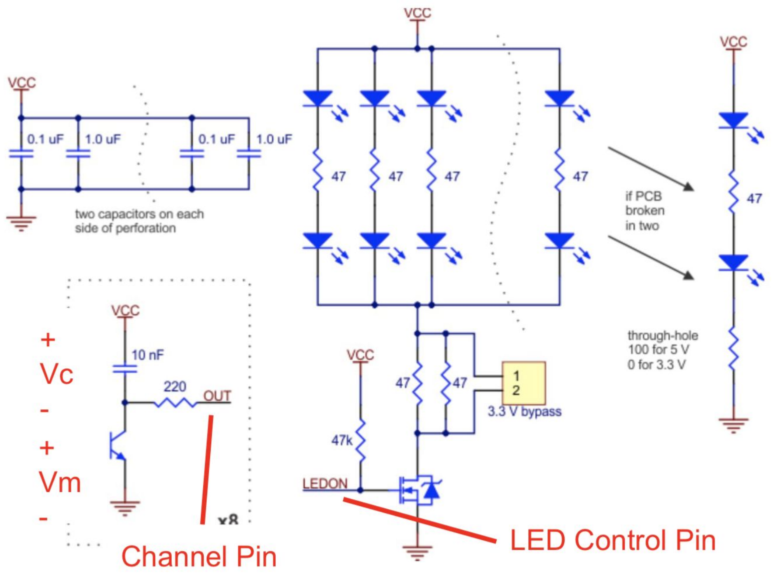


Figure 3. Infrared LED/Phototransistor Reflectance Sensor. The phototransistors pick up the scattered IR light from a surface. The time it takes for the capacitor/phototransistors to charge and discharge correspond with the amount of IR reflected off the surface.
Source: Rahul Iyer, Introduction to Arduino and Energia Programming [PowerPoint]

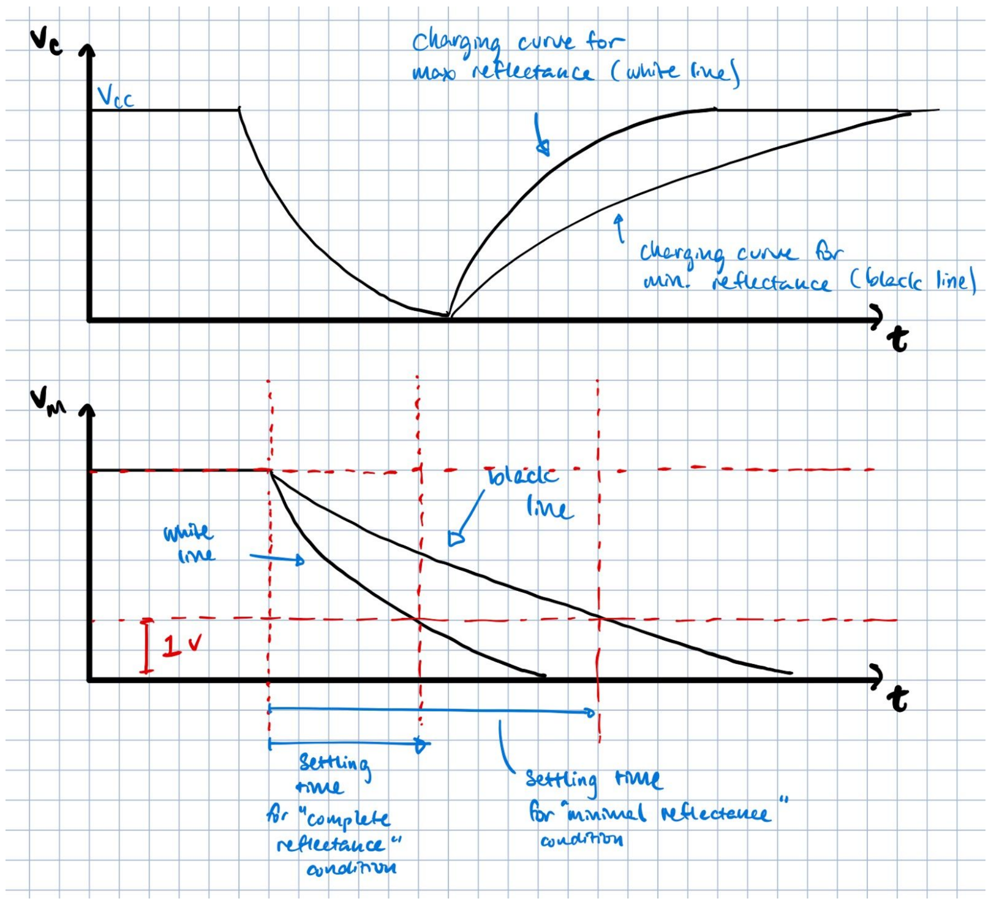


Figure 4. Phototransistor/Capacitor Charging and Discharging Curves. The different settling times and charging times correspond to different degrees of reflectance as shown with the different curves.
Source: Graphs inspired by Rahul Iyer, "Timing Diagram and Corresponding Voltages", Introduction to Arduino and Energia Programming Powerpoint; drawn by William Chong.

Testing Methodology

1) Test Setups (Physical and Electronic Characteristics)

We first verified that the systems including the sensors and the motors were functioning properly. To set up our car for testing, we began by making sure our sensors were functioning properly. We used a small portion of the track and held the car above the line, moving it slowly from left to right to observe sensor readings. Using Energia, we plotted our sensor readings to make sure they were changing as the car moved relative to the center line. Once completed, we implemented our error function to calculate an error value. This value was tested similarly to the sensor readings, by observing the changing output while moving the car side-to-side. Removing print statements, we also tested the motors by using the same method as above, but this time we observed changes to the speed of each wheel to make sure our directions were correct and the output speeds were adjusting correctly. For testing purposes, we included a two second delay that would allow us to place the car on the track before it began moving.

2) **How The Tests Were Conducted:**

With multiple variables including sensor weights, speed, the *proportional controller constant* (Kp) and the *derivative controller constan*t (Kd), we most often chose one variable to change at a time. The testing process primarily involved testing our car on the track, recording observations, changing a variable and repeating. For the sensor weights, we started with arbitrary values and changed them throughout testing whenever we noticed the car's directional adjustments were too strong or weak in relation to the distance away from the line. Refer to **Figure 5** for data collected on various weight values. The scaling often changed drastically through the testing process as we gained advice from instructors and attempted to rethink our values.

Sensor Weights (Left to Right):	Observations:
-6, -4, -2, -1, 1, 2, 4, 6	Car overcorrecting on turns
-3, -2.5, -2, -1, 1, 2, 2.5, 3	Car still overcorrecting
-2.5, -1.75, -1.5, -1, 1, 1.5, 1.75, 2.5	Approaching the turn, car was unable to follow path
-1.875, -1.75, -1.5, -1, 1, 1.5, 1.75, 1.875	Car able to stay on path more consistently on turns than straight path.
-2.65, -2.5, -2, -1, 1, 2, 2.5, 2.65	Car able to successfully complete path
-3, -2.5, -2, -1, 1, 2, 2.5, 3	(Return to previous weights) Small errors of car not registering with inner weights. Larger errors being picked up with outer weights.
-3, -2.5, -2, -1.5, 1.5, 2, 2.5, 3	Small errors of car still not registering with inner weights.
-6, -4, -3, -2, 2, 3, 4, 6	Sharper oscillations with small errors.
-5, -4, -3, -2, 2, 3, 4, 5	Final Weights Consistently successful runs.

Figure 5. Data from Testing Sensor Weights
Source: Justin Aoki and William Chong

To find the correct Kp and Kd values, we first set Kd equal to 0 and just tested Kp. Our process involved testing extreme values, both high and low, to better observe the effects of Kp. Then we worked our way inward to find a Kp that would result in the car oscillating but still following the line. This was a very tedious process and involved restarting when we changed our weight values. Once we were happy with Kp, we worked on Kd in a similar way. We looked for smooth path following and oscillations that dampened over time. Testing the car by starting it slightly off from the centerline on the straight path was a good way to observe the effects of Kd.

Only after the car was able to successfully complete the course did we go back and increase the speed. Our initial speed for the majority of our testing was 60. However, to complete the course in under 45 seconds, we increased our speed slightly to 65. The increase in speed also required some adjustments of other variables like increasing our Kp value.

3) **Data Analysis:**

As described earlier, the process of finding suitable weight values for the car involved a lot of trial and error and required frequent reevaluation of our implementation. See **Figure 6** for our final weight values.

Sensor (Left to Right)	0	1	2	3	4	5	6	7
Weight	-5	-4	-3	-2	2	3	4	5

Figure 6. Weighted Values for the 8 Sensors
Source: Justin Aoki and William Chong

The following table, **Figure 7** shows the data and analysis collected while testing for the Kp and Kd in the experimental process described above. The final values for Kp and Kd were 1.85 and 3.25, respectively.

Kp	**Kd**	**Other Variables:**	**Observations:**
15	0	Using weights: -6, -4, -2, -1, 1, 2, 4, 6	(Testing extreme value) Sharp corrections, went off the track at the turn
1	0		(Testing extreme value) Barely corrected, when approaching turn it went off track
5	0		Able to follow first turns but over corrected on later turns
3	0		Follows most of track, goes off track on turn with strong oscillation
3	-6		(Testing extreme value) Smooth on straight track, on turn car left the track
3	-3		Oscillations still strong
5	0	Changed Weights to: -1.875, -1.75, -1.5, -1, 1, 1.5, 1.75, 1.875	Reevaluating Kp; Oscillations increase
4	0		Less oscillations, able to follow track
4	1.5		Oscillations still too large
4	2	Changed Weights to: -2.65, -2.5, -2, -1, 1, 2, 2.5, 2.65	Successful run, frequent oscillations

4	-1		Reevaluating sign of Kd; still oscillating frequently
4	-1	Changed Weights to: -3, -2.5, -2, -1, 1, 2, 2.5, 3	Oscillations did not dampen on straight path
3	0		Reevaluating Kp; Smaller oscillations, still inconsistent
0.1	0		Reevaluating Kp; No corrections made by car
1	0		Car not correcting enough
2	0		Corrections too strong
1.5	0	Changed weights to: -3, -2.5, -2, -1.5, 1.5, 2, 2.5, 3	Oscillations but able to follow the straight path
1.5	0.8		Oscillations less intense than before
2	3	Changed weights to final values (**Figure 5**)	Corrections slightly too strong
1.75	3		Successful run completed in 48 seconds
1.85	3.25	Speed increased to 65	Successful run, completed in 43 seconds

Figure 7. Table of Test Data for Kp and Kd
Source: Justin Aoki and William Chong

4) Test Data Interpretation

Throughout the testing process, the weights were manipulated several times. We realized that, while the values used as weights are important, the relationship between each adjacent weight was also important. Note our final values (**Figure 6.**) have a nearly linear relationship but our inside weights are -2 and 2 instead of -1 and 1. This was decided so the car would react strongly to any small error from the inner sensors and promptly return itself to the center of the track. The Kp value of 1.85 was the sweet spot between larger values we tested, which caused the car to overcorrect, and smaller values which prevented the car from reacting to the turns in the course. The Kd value of 3.25, when paired with the Kp value, proved the best to damp oscillations of the car. Any lower value and the car showed stronger oscillations and was less consistent in being able to follow the line.

Also, earlier versions of our code included a more complicated implementation of the sign of the error value and our speed control value. For this reason, we tested positive and negative values for our Kd. After correcting the math and simplifying our code, the Kd value had to be positive. Though this Kd value is larger than the Kp value, our data proved this combination of constants to be the most effective when implementing our PID controller.

Results and Discussion

1) **Test Discussion**

   For the goals of our project, the final variables found through testing were appropriate. Using the results after all the testing, the car was able to consistently complete the track. Because our process involved occasional reevaluating and taking a few steps back, we tested a range of values, some of which were very different from the values we concluded our experimentation with. While helpful in the experimental process, some of the data collected for specific weights and/or specific values for Kp and Kd don't provide much useful information when analyzing our implementation. Still, further work could be done to tweak our variables and create a smoother traversing car.

2) **Race Day Discussion**

   a) **How vehicle performed on race day**

      On Race Day, our vehicle successfully completed the course. During our allotted time on the track, we first ran our car at a slower speed because we were most confident it would be able to complete the track. We accomplished our goal and the car finished the course in 47 seconds. Being close to the 45 second cutoff we also prepared to run the car at a faster speed with some variables slightly changed through testing. At this speed, the car successfully completed the course in 43 seconds. The values of our weights, Kp, Kd, and speed are mentioned above as well as in the code provided alongside the report. The car remained fairly smooth throughout the run and avoided noticeably jagged turns. This may be attributed to the darker track used.

   b) **Limitations of code/track**

      A noticeable limitation of our car was its speed. Though completing the course in under 45 seconds, the car's speed could have been much faster with further testing and adjusting of variables. Also, the car did experience noticeable oscillations on the test track and, with more testing, those oscillations could have been decreased and allow our car to be much closer to the line. Also, the track used on Race Day was printed darker than the track used during our testing so the cars performance was better on the actual run compared to our tests. The car oscillated less on the final track than it did on the track used during our testing process.

   c) **What we would have done differently**

      If we were to conduct this project again, we would have spent less time
      worrying about getting the theoretical concepts correct at first, and instead put more emphasis on experimental trial and error to learn what worked and what did not. This would potentially allow us to get a working trial version of our code such that we could actively start testing our car's performance earlier--giving us more time to fine tune and speed up our car. With our knowledge from this project, we would be able to probably increase the speed of the car at least two-fold, and even explore some of the other extra credit tasks offered.

Conclusions and Future work

Our design successfully met our goal and completed the track in a reasonable amount of time. However, the car was not perfect and could be worked on further to improve speed and strengthen its ability to follow the path. Future work can be done on making the car run much smoother. It would be interesting to reach a point at which the viewer can't tell the car is adjusting to follow the line, and instead it looks like the car is connected to a track. At this point, it would also be worth testing on different tracks, playing around with longer paths and sharper turns. Through the project we learned technical skills, such as implementing the PID controller, that serve as a foundation for projects in the future. Alongside the technical work, we also learned the importance of data collection during the experimental process. Often, we stumbled across difficulty when trying to improve our implementation; however, with each trial we recorded our data. This practice not only helped us keep track of what variable was being changed and what differences we were noticing in the car, but it also allowed us to continuously refer to collected data when working towards our goal. Instead of speculating possible solutions to our problem and forgetting the progress we had made, we were able to look back and what we had already tested to strategize our next steps.

In the future, if we were to revisit this project, some extensions we would aim to perform would be to increase the speed of the car, such that it would be able to complete the track in about 20 seconds. Another aspect we could work on could be calculating variance and displaying additional information on an LCD display. Other extension projects could include modifying the car with additional sensors to navigate around a room--similar to how a like a robotic vacuum cleaner

References

[1] Pololu Robotics & Electronics, https://www.pololu.com/product/961