Artavazd Torosyan and Angelina Tram
19 December 2020

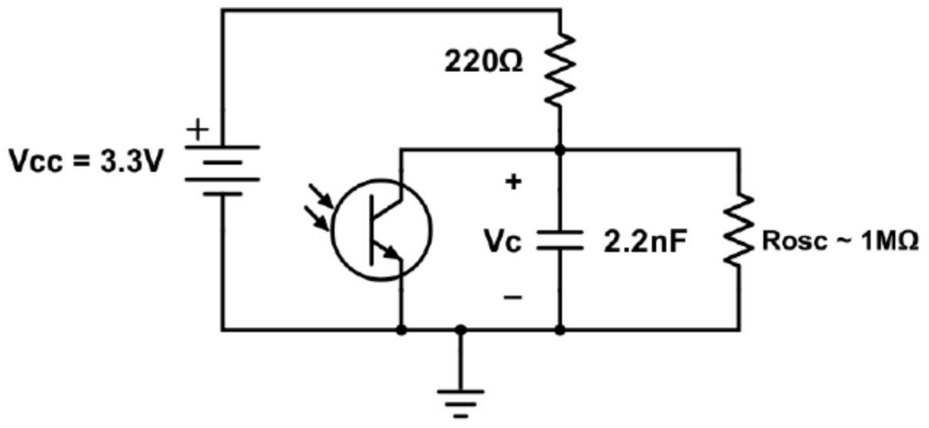
ECE3 20F Final Report

1. *Introduction and Background:*

The goal of the project was to make the car be able to follow a black curved track to the end, turn 180 degrees, come back to the starting position and stop. The car must have been able to start from any four of the starting position options provided and complete the route successfully. Another optional challenge was to complete the route in under eleven seconds. The design that was selected uses Infrared light LED sensors to read the intensity of light reflection from the track and make decisions based on those readings. We used the concept of PD control to control motor speeds in proportion to the car's position. PD control[1] takes into account the difference in position and change of position relative to the black curve. The 'P' stands for Proportion Control and 'D' stands for Derivative Control.

The basic theory required for understanding the path sensing system first includes the path sensing circuitry[2]. Within the car, an RC circuit powers each phototransistor, helping each detect whether they are above a light (white) surface or dark (black) surface. The circuit works by first charging up the capacitor, going from 0 to 3.3 volts. The charging time is relatively short, about 0.4 μs. Here is a schematic for the circuit during charging:

Charging:

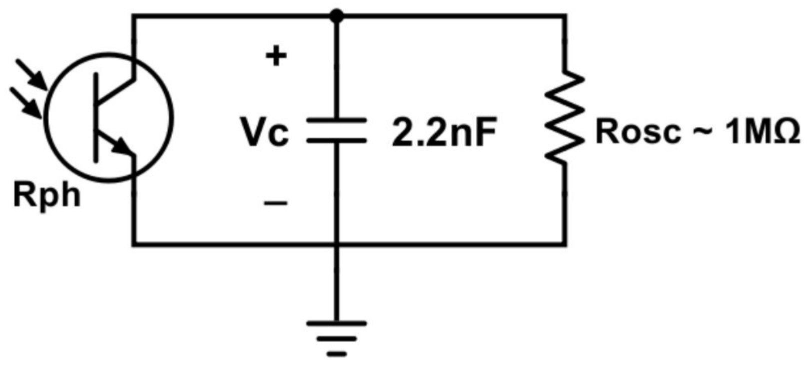


ECE 3 Fall 2020
Lab Section 5
Notes (annotated)

$$\tau_c = C(R_{220} // R_{ph} // R_{1M}) \approx 220\Omega \times C$$

Once the capacitor is charged to 3.3 volts, the capacitor begins to discharge. During the discharge time, the readings for the sensory input data are taken. Here is the schematic for the discharging RC circuit:

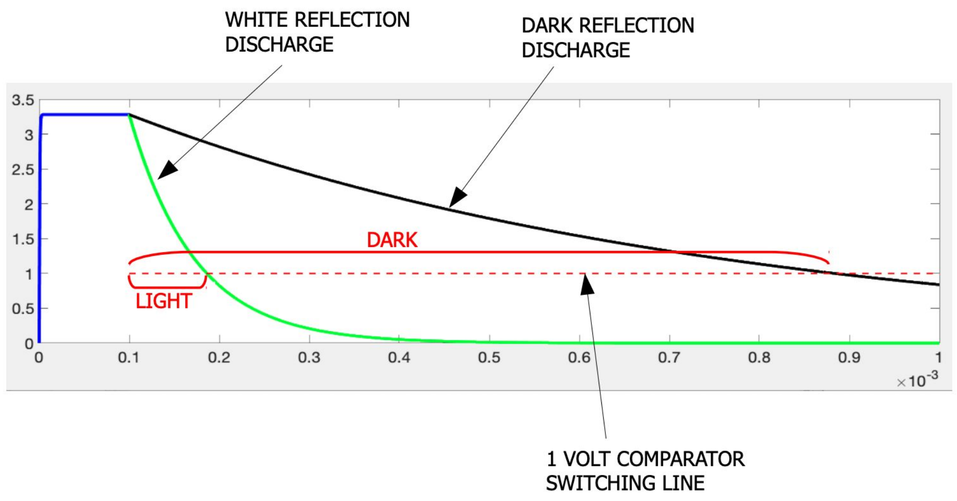
Discharging:



ECE 3 Fall 2020 Lab Section 5 Notes (annotated)

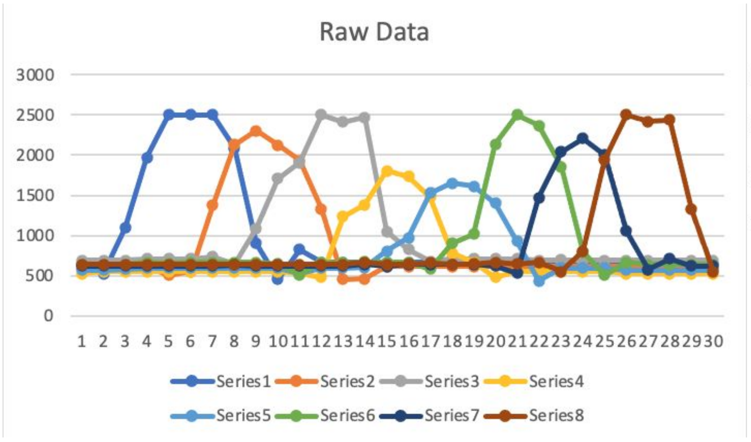
$$\tau_c = C(R_{ph}//R_{1M}) \approx R_{ph} \times C$$

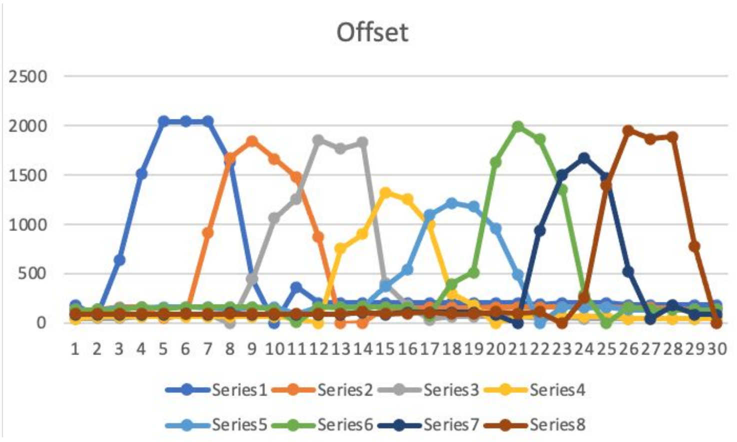
The time taken for the capacitor to discharge from 3.3 volts depends on whether the phototransistor is seeing a dark or light surface, and it gives the sensory input data, which ranges from 0 to 2500. If the phototransistor is above a white surface, the Rph would subsequently be low, making Rph * C be low, causing the time taken for the capacitor to discharge to be small. In the same vein, if the phototransistor is above a dark surface, the Rph would be high, causing Rph * C to be high, making the time taken for the capacitor to discharge to be long. Thus if the sensory input data reads 0, the phototransistor would be reading white, and at 2500, the phototransistor would be reading black. Here is a graph that depicts the charging and discharging time:

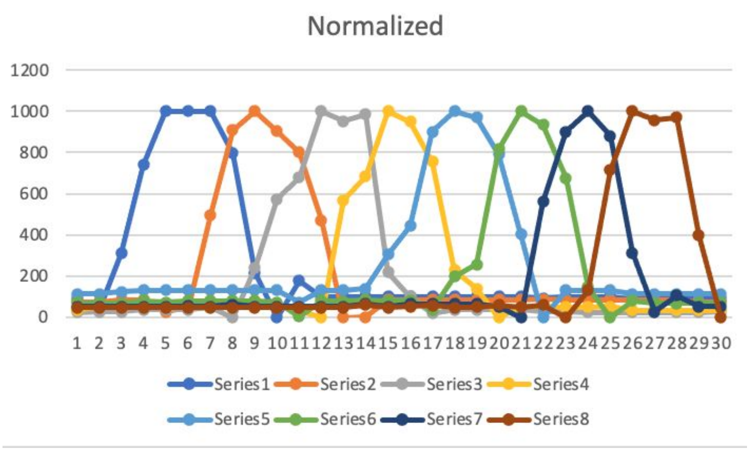


ECE 3 Fall 2020 Lab Section 3 Notes (annotated)

Calibrating the sensory input data includes two components: killing the offsets and normalizing the data through scaling.

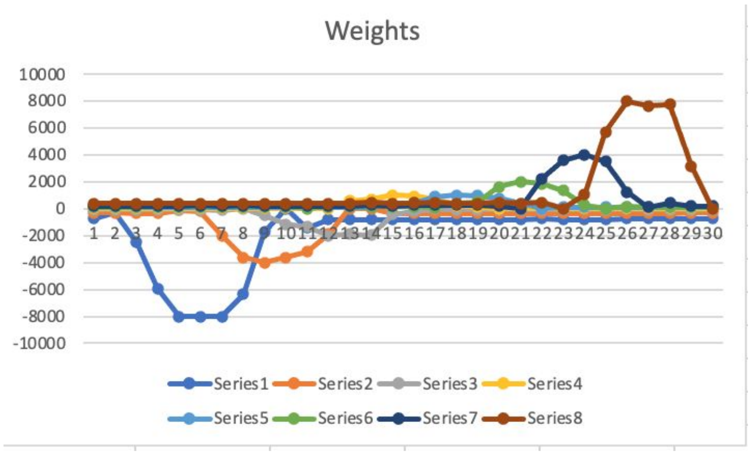






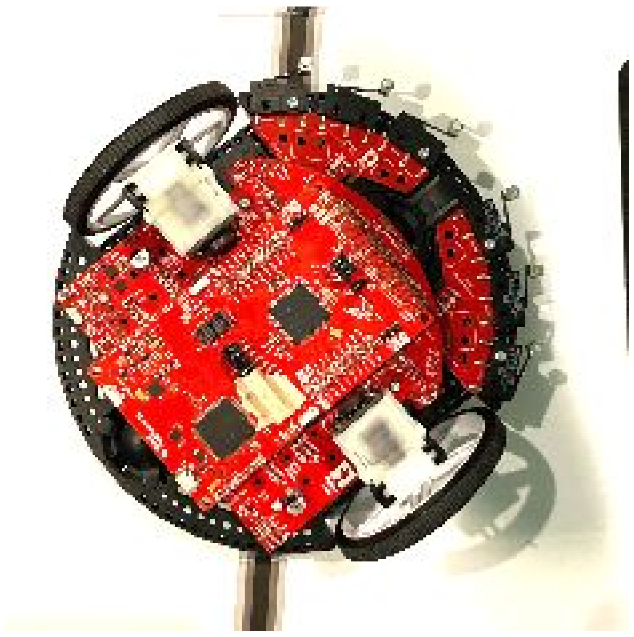
(Data taken from our group's excel sheets from one of many calibrations)

With the raw data values from the phototransistors as pictured on the left, we first took the minimum value from each curve and subtracted each point with the minimum value, bringing the bottom of the curves to meet the x-axis. This process is known as "killing the offsets". Then with the offset values, we take 1000 divided by the maximum value of each curve and multiply all points on the curve by this value, bringing the maximum amplitude of each curve to 1000. This process is known as "normalizing" the curves.



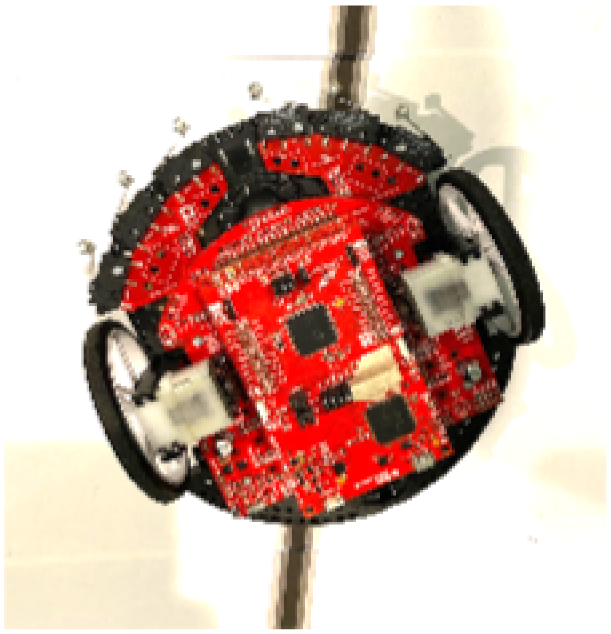
(Data taken from our group's excel sheets from one of many calibrations)

The calibrated sensor data was then fused together. This process is called sensor fusion and it helps us to determine an error from the track. The error is basically the distance away from the track. By finding out the distance away from the track, we are able to send instructions to the power motors to run at different speeds in order to bring the car back on track if it is off track. The way this is done is by assigning different weights to different sensors. The middlemost sensors, Sensor 4 and Sensor 5 were given the smallest weights while the farthest sensors from the middle were given the highest weights. This is because if we want the car to be following the track, then we want the middle sensors to be above the black curve. If the outermost sensors are reading black then the car is deviating away from the track. Which is not what we want. We also want to have polarity in the weights because the polarity will help us determine whether we should be turning left or right. So, sensors 1-8 were assigned the weights of {-8,-4,-2,-1,1,2,4,8} respectively in ascending order. These weights were multiplied to the calibrated readings of the sensors and are added together to get the fused value of error, the distance away from the track. As an example, if our readings for sensors 1-8 are {1000,900,300,100,0,0,0,0}. Then our error = -8*1000+-4*900+-2*300+-1*100+0+0+0+0 = <u>-12,300</u>. The way we defined our system says that if we have a negative value then the car is too much to the right so we should do something to the motors to make it start turning left and getting back on track. We would get a case of readings like this if the car is oriented similar to what is illustrated below.

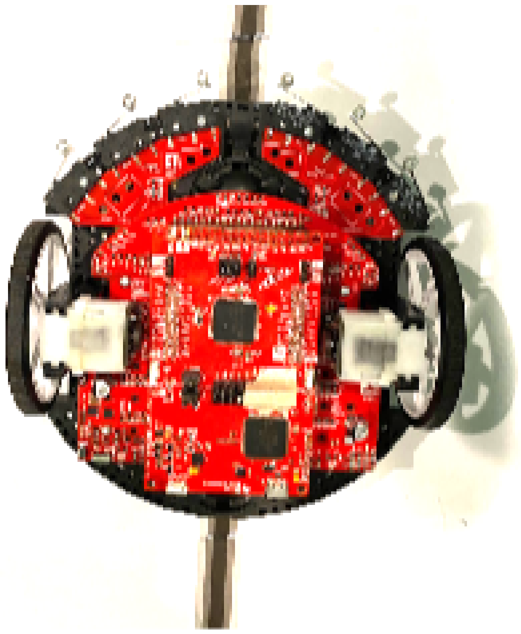


Case similar to {1000,900,300,100,0,0,0,0}

To give another example, let us say that our calibrated readings are {0,0,50,100,200,800,1000,200}. Then using the weights {-8,-4,-2,-1,1,2,4,8}, our fused error would give 0+0+-2*50+-1*100+1*200+2*800+4*1000+8*200 = <u>7200.</u> The positive value to us means that the car is too much to the left and we must do something to make the car turn to the right. See below figure for an illustration of a similar case.


Case similar to {0,0,50,100,200,800,1000,200}

In a perfect world, if the car is exactly centralized on the track, our calibrated sensors would read something like {0,0,300,1000,1000,300,0,0}. See below figure.



Case similar to {0,0,300,1000,1000,300,0,0}

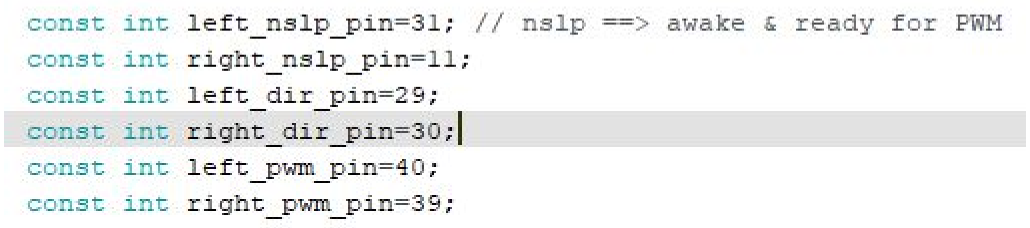
So, if calculate the error, we get 0+0+-2*300+-1*1000+1*1000+2*300+0+0 =-600 -1000 +1000 + 600 =0. NO ERROR. What does that mean? We are exactly on track and since our error is 0. So, nothing is going to change on the motors. A different number weighting system could be implemented but we found that {-8,-4,-2,-1,1,2,4,8} worked beautifully in our personal experience with this project.

2) ***On-Track Development Methodology:***

a. ***Test Setup:***

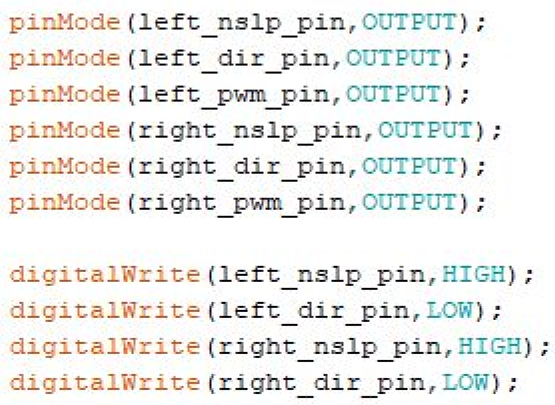
The environmental characteristics of the test setup included ensuring that there was an even distribution of light on the track, as the phototransistors were particularly sensitive to any changes in light and shadows. Additionally, any sunlight needed to be blocked, as sunlight disrupted the infrared light sensors on the phototransistors, which would cause the sensors to fail. Since we used a split copy of the track, it was necessary to tape the paper onto the floor to ensure that the track wouldn't move out from under the car, especially considering that the car turned at high speeds. Another point of concern for the setup was the darkness of the floor. Putting papers underneath the track, especially on the sides where the car made the bend, would help the sensors detect the black track more effectively.

As for the electrical characteristics of the test setup, we used we first had to define the pins to our code so that we can manipulate the parts of our car in order for it to run on the track. Please see the pinchart provided in the appendix. We read the sensors' values in our code with the help of a library provided by Dr. Briggs, *ECE3.h*[3]. This library uses pins: 65, 48, 64, 47, 52, 68, 53, and 69. First, we had to set the output voltages of left and right nSLP pins 31 and 11 to HIGH, so that our power motors can run when we give them a direction and speed. Then we set the voltages of the left (pin 29) and right direction (pin 30) pins to LOW, so that they run in the same direction. If one of them is set to LOW and the other to HIGH, then they would run in different directions and do a donut instead of going straight. Next, we had to also define the power motor output pins (30 & 40), so that we could supply it with the motor speed (an integer between 0-255, 0 is the slowest and 255 is the highest). We tested the speed at different values: 20,40,80,110, and 120. We settled for 110 since it was the fastest speed we could figure out the rest of our parameters. The figures below show the definitions of our C++ code for these values.



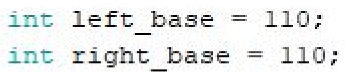
```
const int left_nslp_pin=31; // nslp ==> awake & ready for PWM
const int right_nslp_pin=11;
const int left_dir_pin=29;
const int right_dir_pin=30;
const int left_pwm_pin=40;
const int right_pwm_pin=39;
```

Definitions of pins



```
pinMode(left_nslp_pin,OUTPUT);
pinMode(left_dir_pin,OUTPUT);
pinMode(left_pwm_pin,OUTPUT);
pinMode(right_nslp_pin,OUTPUT);
pinMode(right_dir_pin,OUTPUT);
pinMode(right_pwm_pin,OUTPUT);

digitalWrite(left_nslp_pin,HIGH);
digitalWrite(left_dir_pin,LOW);
digitalWrite(right_nslp_pin,HIGH);
digitalWrite(right_dir_pin,LOW);
```

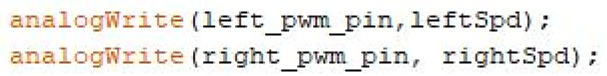
Assigning pins for OUTPUT



```
int left_base = 110;
int right_base = 110;
```

Assignment of base speed

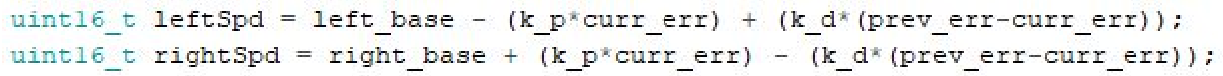
For assigning value to the PWM pins, we didn't use the base speeds. This is because the speeds of each power motor would change based on the error derived from the sensor data.



```
analogWrite(left_pwm_pin,leftSpd);
analogWrite(right_pwm_pin, rightSpd);
```

Output of speed given to PWM

The left speed and right speed were determined by the base speed, current error, previous error, and the constants of our PD control Kp and Kd. The current error was read in an iteration of a loop that always runs during our program and the previous error was recorded from the previous iteration of that same loop.



```
uint16_t leftSpd = left_base - (k_p*curr_err) + (k_d*(prev_err-curr_err));
uint16_t rightSpd = right_base + (k_p*curr_err) - (k_d*(prev_err-curr_err));
```

Assignment of speed

Since the program was figuring out the error values for us, we had to figure out the value of Kp and Kd by trial and error. Since our errors ranged from ~ -12,000 to ~12,000, we thought that Kp and Kd would be small values since our base speed is a double or triple-digit number. So, we tested out numbers that would change the speed in a similar order of magnitudes.

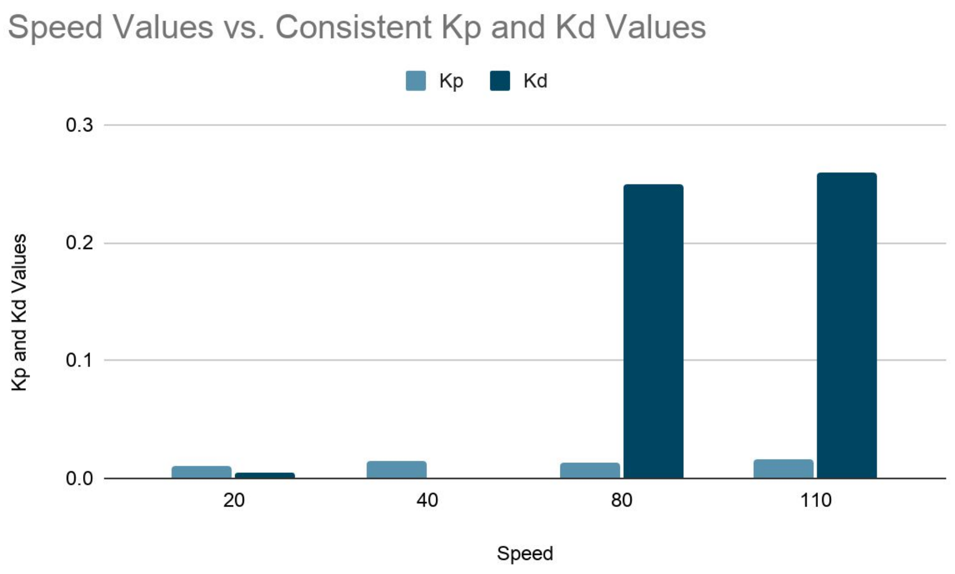
b. ***How the tests were conducted:***

Before testing, we calibrated the car with new offset and scaling values to ensure the sensors were being normalized to the appropriate setting. We first began testing the car at lower speeds. During this process, we did not run the serial monitor simultaneously. The parameters we were looking for were Kp and Kd. First, we tested Kp, starting at a small initial value and gradually increasing the value until the car began to wiggle around the track. We tested the car on each Kp value on each of the four positions. Since the car must be able to complete the track regardless of its starting position, testing to ensure the Kp value worked with each position was absolutely necessary. Once the car wiggled around the track, we meant to start increasing the Kd value to "smooth" out the wiggles of the car. Each time we ran a test, we noted the time that it took for the car to complete the track, whether the car completed the track or not, and the movement behavior of the car around the track (whether it wiggled, whether the turns were jerky, or where the car failed to complete the track). If the car failed on a position, we would diagnose the issue to determine whether we should continue increasing Kp or choose a lower value.

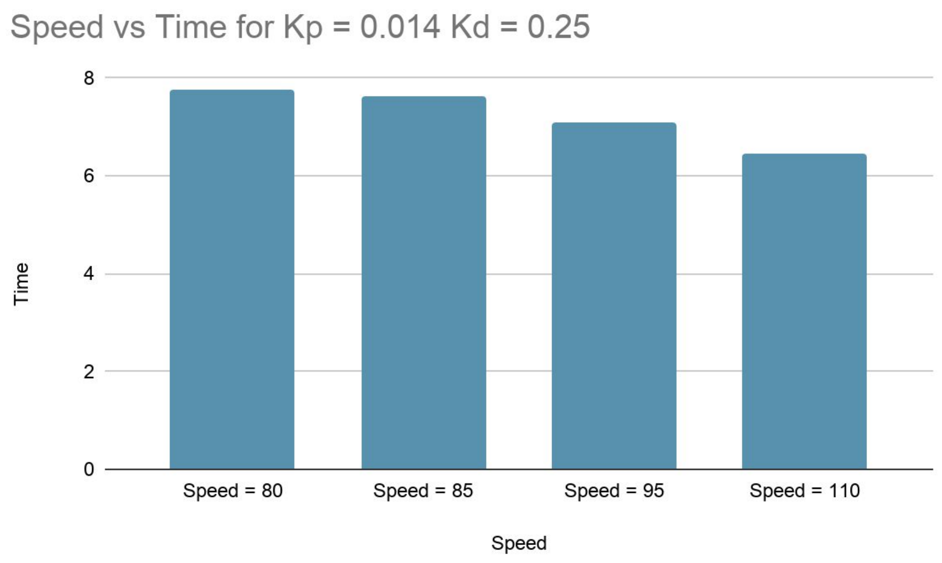
We tested the car at 4 different speeds: 20, 40, 80, 110, and 120. However, we were unaware of the extra credit challenge of completing the track under 11 seconds until after we tested the car at speed 40. We only began vigorously testing the Kd values after, on both speed 80 and 110. However, when testing high speeds of 80 and 110, we focused heavily on adjusting Kp to get the car to complete the route consistently on each position. Once we chose a Kp value to go with for speed 80, we chose a random value for Kd to start with, and it luckily worked consistently on all four positions without wiggling. So we did not do much hunting for Kd.

For testing the cross piece detection, we had to change the phototransistor sensing value depending on the difference in lighting each new session of testing. Should the sensing value be changed, the time that the car spun would have to be adjusted slightly in order to ensure the car turned right back onto the track again. We designed the code in a way that once all of our sensors saw values greater than near black, we made it do a 180 degree turn. And for the second time, to come to a full stop.

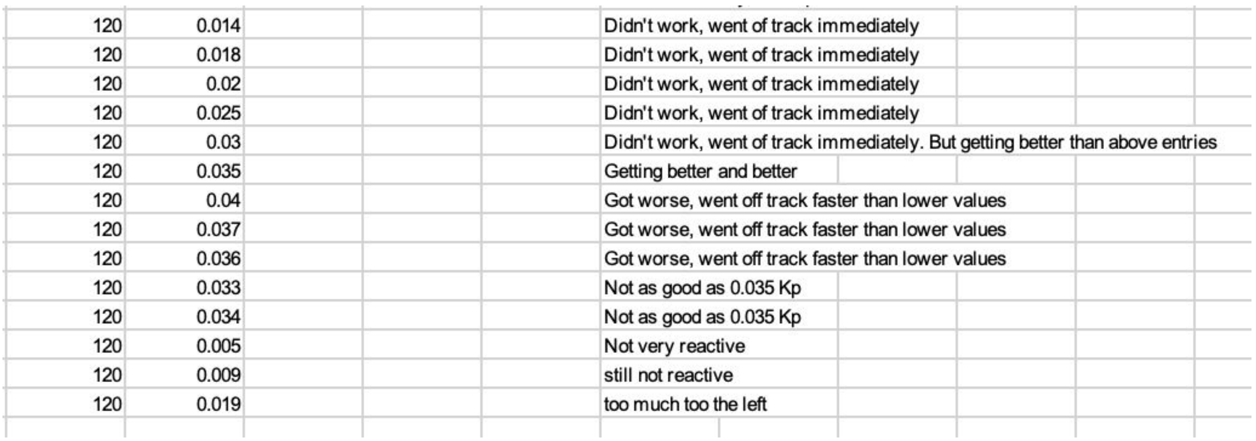
c. ***Data Analysis:***



(Figure A): As speed increases (taking a closer look at speed 80 and 110), Kp and Kd seem to scale up together correlating with the increase in speed (Taken from Appendix Fig 1-5)



(Figure B): Fixing the Kp and Kd values, the time taken to complete the course decreased as the speed increased, and the car completed the track consistently at all four of these speeds. (Taken from Appendix Fig 6)



120	0.014			Didn't work, went of track immediately
120	0.018			Didn't work, went of track immediately
120	0.02			Didn't work, went of track immediately
120	0.025			Didn't work, went of track immediately
120	0.03			Didn't work, went of track immediately. But getting better than above entries
120	0.035			Getting better and better
120	0.04			Got worse, went off track faster than lower values
120	0.037			Got worse, went off track faster than lower values
120	0.036			Got worse, went off track faster than lower values
120	0.033			Not as good as 0.035 Kp
120	0.034			Not as good as 0.035 Kp
120	0.005			Not very reactive
120	0.009			still not reactive
120	0.019			too much too the left

(Figure C): Trying to find Kp for speed 120

d. ***Test Data Interpretation:***

In Figure A, speed and the Kp and Kd values that produced the most consistent results were compared. As for speeds 20 and 40, we lack data because we neglected to focus on testing for Kd, and we were still unsure at the time what exactly was the best behavior for the car to be moving at any certain Kp, and we were unaware that we had the extra credit time challenge goal. Thus, once we were aware of such a challenge, we upped our tests to speeds 80 and then to 110, and we began testing for Kd. Though there is not a wide range of data, from the graph, we can infer that the Kp and Kd values continue to scale higher as the speed increases and that the ratio of Kd to Kp is a rather large number.

In Figure B, after finding a Kp and Kd value that worked consistently at speed 80, we wanted to see how high we could up the speed when fixing the Kp and Kd values until the car does not complete the route consistently on all four positions. We luckily found that the car did work consistently on these fixed values up until 110 (+30 speed from 80), which is what we decided to run for Race Day.

For Figure C, we also tried running Kp = 0.014 and Kd = 0.25 for speed 120, but the car was extremely inconsistent for all positions. After trying to steadily increase Kp, we found that we would have to do much more rigorous testing in order to find the optimal combination of Kp and Kd, but by this point in time of the project, we would not have time to find it.

3) ***Results and Discussion:***

a. ***Test Discussion***

Since our project goal became clearer halfway through our project, the testing that we did for the higher speeds were much more beneficial than the testing at lower speeds (20 and 40). Additionally, it was unnecessary to test for the time at the lower speeds, as those speeds would not have enabled the car to complete the route in under eleven seconds regardless of any Kp or Kd adjustments. Since we were somewhat aimless at the beginning of the project, Figure A shows that our data from our testing for speeds 20 and 40 helped very little in determining any sort of Kp to Kd ratio for higher speeds. Only when we began testing for Kp at speed 80 did we make any progress in enabling the car to reach the goal within the time.

Figure B illustrates our attempt at pushing the speed of the car with the Kp and Kd values that we found for speed 80, and this was arguably our most useful finding in our tests. Taking our time from just under 8 seconds to just above 6 seconds gave our group insurance that the car would run well under the time limit. However, we found that at 120 speed, as pictured in Figure C, the fixed Kp and Kd values did not work anymore. Yet, by examining Figure A again, we could assume that Kp and Kd would continue to scale larger with speed, but we did not have the time to do so. Thus, if we had simply started testing at speed 80, we may have had time to find a proper ratio for Kp to Kd at higher speeds. This would have been more conducive to reaching our project goals at a faster rate.

b. ***Race Day Discussion***

i. ***(5 points) Discuss how well your vehicle performed on Race Day***.

On Race Day, both vehicles completed the track successfully. However, Angelina's car failed the first run but made it on the second run. Artavazd's car completed the route in 6.5 seconds, whereas Angelina's car completed the route in 6.6 seconds in its second run. The reason for the failure of Angelina's car's first run was most likely due to a lighting issue or anomaly, as it failed to follow the bend of the track back around, continuing straight instead, meaning the phototransistor didn't read the track as darker than the shadows surrounding it. Nevertheless, both cars performed as expected from our testing. We were declared to be Champions of Lab Section 1F by Xin Li.

ii. ***(5 points) Discuss the limitations of your code and of the car and tracks***.

As for the limitations of the code, we had to manually change the blacks value (the sensor value that signaled for the car to make a 180-degree turn) every time our lighting condition changed, as well as the time for the car to rotate. There was no way for the car to automatically adjust this in its code (similarly to how the offset and normalizing values must have been manually done as well).

As for limitations for the car and the tracks, the tracks were merely pieces of paper taped onto the ground, which doesn't necessarily provide the most traction for the car, especially during its turns. For example, when testing for the cross piece detection, the car would often spin out and stop. However, had the car been set on a surface that provided more traction, the spin would not have been such an issue. There was also a problem with printing the track, some dark parts were not as dark as we would want to be but we compensated for that by changing some values in our code.

iii. ***(5 points) Talk about how you would conduct the project differently, using what you have learned along the way***.

If we were to conduct the project differently, the first thing we would alter is how we tested the car to find the correct Kp and Kd values for each speed. Starting to test Kp and Kd at higher speeds first would accelerate our progress, especially considering that we wished to make our run time under the optional challenge of eleven seconds. Additionally, we would have kept track of our trials in more detail, as well as log more of the trials. More often than not, we would run our car without taking down every trial and noting every behavior, especially towards the end of the project as Race Day encroached. This becomes an issue as the data isn't organized enough to understand what precisely was occurring during each trial.

4) ***Conclusions and Future Work:***

Our designs met our goal very well. Besides the discussion on beginning to test on higher speeds as mentioned in the previous sections, it would be interesting to explore the differential equation behind the equation for current error. Our group solved the differential but found it tricky to actually implement into the code for the car, and we eventually abandoned this idea. However, implementing the differential equation would make the error calculations running in the car to be much more accurate, which would eventually allow the car to run at a much quicker pace. Additionally, exploring changing the value of the weights was something that piqued our interest, but we found it unnecessary to explore as by the time our car ran well for the challenge presented to us. However, with some extensions, both of these concepts would be something that would be worth delving into.

6) ***References:***

1. Courtesy of Xin Li's explanation in ECE3 Lab Section at UCLA. Also Courtesy of Dr. Brigg's explanation of PID control.
2. Courtesy of Xin Li's explanation in ECE3 Lab Section at UCLA.
3. Courtesy of Dr. Brigg's ECE3 class at UCLA

7) ***Code:***

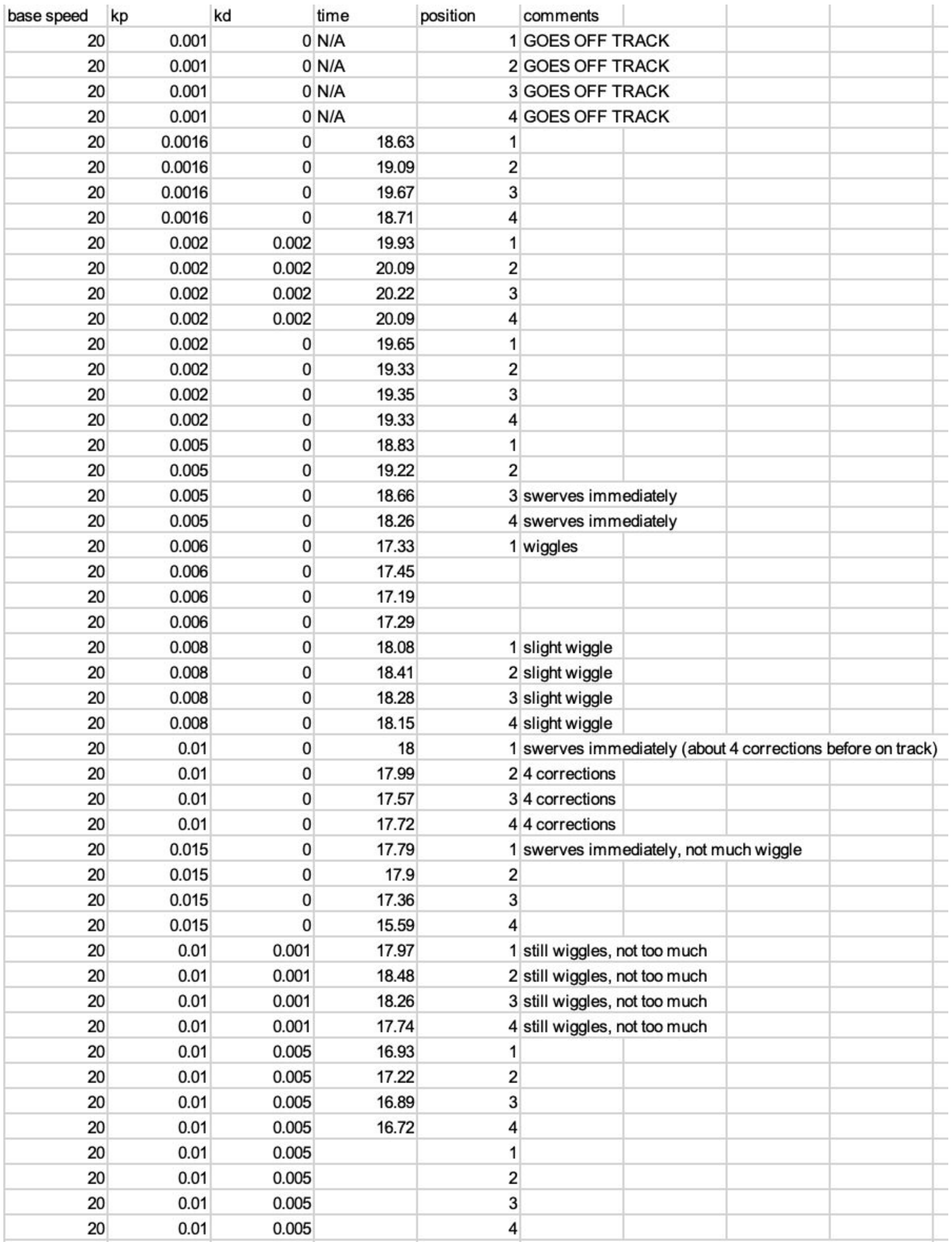
```
#include <ECE3.h>

  uint16_t sensorValues[8];
  uint16_t offset[8] = {505, 412, 551, 573, 573, 505, 575, 458};
  float scale[8] = {1.995, 2.088, 1.949, 1.927, 1.927, 1.995, 1.925, 2.042};
  int16_t sensorValues_2[8];
  int16_t weights[8] = {-8,-4,-2,-1,1,2,4,8};
  float k_p = 0.014; //0.01 for base_speed 20
  float k_d = 0.30; //

  int16_t curr_err = 0;
  int16_t prev_err = 0;
  int left_base = 110;
  int right_base = 110;

  const int left_nslp_pin=31; // nslp ==> awake & ready for PWM
  const int left_dir_pin=29;
  const int left_pwm_pin=40;
  const int right_dir_pin=30;
  const int right_pwm_pin=39;
  const int right_nslp_pin=11;

  int16_t blacks = 0;

  int16_t time_count = 0;

  bool donut = false;

void setup() {

  ECE3_Init();
  //Serial.begin(9600); //if we need serial monitor delete // in the beginning of this line

  pinMode(left_nslp_pin,OUTPUT);
  pinMode(left_dir_pin,OUTPUT);
  pinMode(left_pwm_pin,OUTPUT);
  pinMode(right_nslp_pin,OUTPUT);
  pinMode(right_dir_pin,OUTPUT);
  pinMode(right_pwm_pin,OUTPUT);

  digitalWrite(left_nslp_pin,HIGH);
  digitalWrite(left_dir_pin,LOW);
  digitalWrite(right_nslp_pin,HIGH);
  digitalWrite(right_dir_pin,LOW);




  delay(200);

}

void loop() {
 ECE3_read_IR(sensorValues);


  //offset and scale
  for (uint16_t i = 0; i < 8; i++)
  {
    sensorValues_2[i] = (sensorValues[i]-offset[i])/scale[i];

  }

  //weighs sensors and adds them
  for (unsigned char i = 0; i < 8; i++)
  {
   curr_err += (sensorValues_2[i]*weights[i]);
  }
```

```
uint16_t leftSpd = left_base - (k_p*curr_err) + (k_d*(prev_err-curr_err));
uint16_t rightSpd = right_base + (k_p*curr_err) - (k_d*(prev_err-curr_err));

prev_err = curr_err;
curr_err = 0;

//Serial.println(curr_err); //if we need serial monitor delete

//light = low, dark = high
//LEFT WHEEL S8 S7 S6 S5 S4 S3 S2 S1 RIGHT WHEEL


uint16_t sensorSum = 0;

for (uint16_t i = 0; i < 8; i++)
{
  sensorSum += sensorValues[i];
}

if (sensorSum > 19000)
 {
   blacks++;
 }

if(blacks > 2)
 {

  digitalWrite(left_dir_pin,HIGH);
  leftSpd = 180;
  rightSpd = 180;
  time_count++;
    if(time_count > 41 )
    {
     digitalWrite(left_dir_pin,LOW);
     blacks = 0;
     time_count = 0;
     donut = true;
    }
    else if(donut && time_count > 3)
    {
      digitalWrite(left_nslp_pin,LOW);
      digitalWrite(right_nslp_pin,LOW);
    }
  }
```

```
    analogWrite(left_pwm_pin,leftSpd);
    analogWrite(right_pwm_pin, rightSpd);
      //delay(50);

}
```

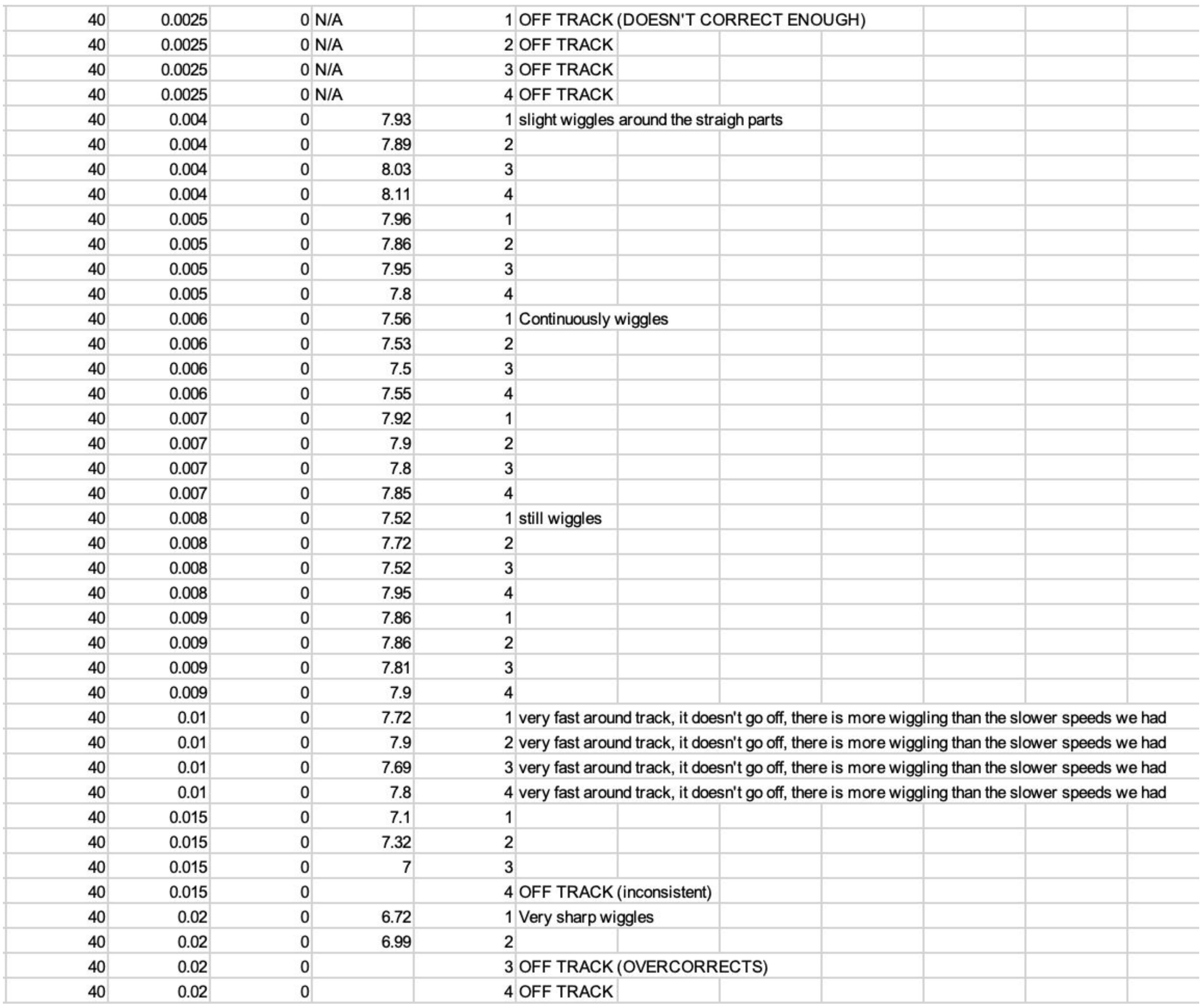
8) ***Appendix:***

Figure [1] Testing Kp and Kd at speed 20



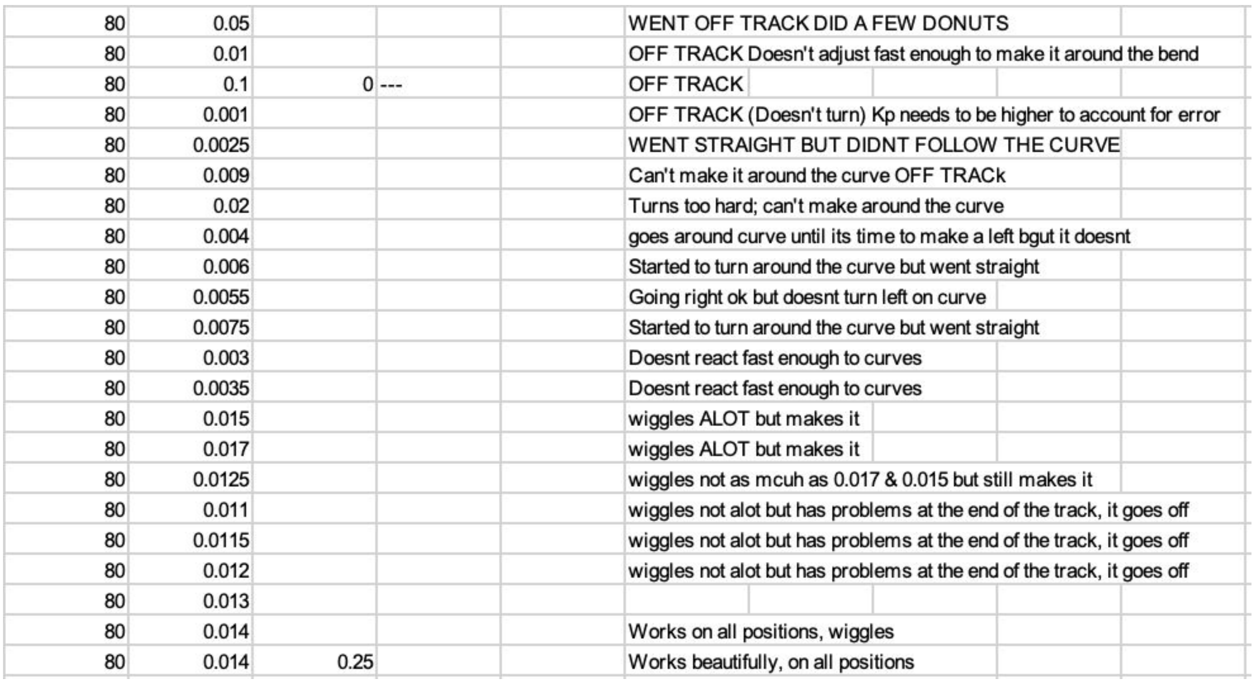
base speed	kp	kd	time	position	comments
20	0.001	0	N/A	1	GOES OFF TRACK
20	0.001	0	N/A	2	GOES OFF TRACK
20	0.001	0	N/A	3	GOES OFF TRACK
20	0.001	0	N/A	4	GOES OFF TRACK
20	0.0016	0	18.63	1	
20	0.0016	0	19.09	2	
20	0.0016	0	19.67	3	
20	0.0016	0	18.71	4	
20	0.002	0.002	19.93	1	
20	0.002	0.002	20.09	2	
20	0.002	0.002	20.22	3	
20	0.002	0.002	20.09	4	
20	0.002	0	19.65	1	
20	0.002	0	19.33	2	
20	0.002	0	19.35	3	
20	0.002	0	19.33	4	
20	0.005	0	18.83	1	
20	0.005	0	19.22	2	
20	0.005	0	18.66	3	swerves immediately
20	0.005	0	18.26	4	swerves immediately
20	0.006	0	17.33	1	wiggles
20	0.006	0	17.45		
20	0.006	0	17.19		
20	0.006	0	17.29		
20	0.008	0	18.08	1	slight wiggle
20	0.008	0	18.41	2	slight wiggle
20	0.008	0	18.28	3	slight wiggle
20	0.008	0	18.15	4	slight wiggle
20	0.01	0	18	1	swerves immediately (about 4 corrections before on track)
20	0.01	0	17.99	2	4 corrections
20	0.01	0	17.57	3	4 corrections
20	0.01	0	17.72	4	4 corrections
20	0.015	0	17.79	1	swerves immediately, not much wiggle
20	0.015	0	17.9	2	
20	0.015	0	17.36	3	
20	0.015	0	15.59	4	
20	0.01	0.001	17.97	1	still wiggles, not too much
20	0.01	0.001	18.48	2	still wiggles, not too much
20	0.01	0.001	18.26	3	still wiggles, not too much
20	0.01	0.001	17.74	4	still wiggles, not too much
20	0.01	0.005	16.93	1	
20	0.01	0.005	17.22	2	
20	0.01	0.005	16.89	3	
20	0.01	0.005	16.72	4	
20	0.01	0.005		1	
20	0.01	0.005		2	
20	0.01	0.005		3	
20	0.01	0.005		4	

Figure [2] Testing Kp and Kd at speed 40



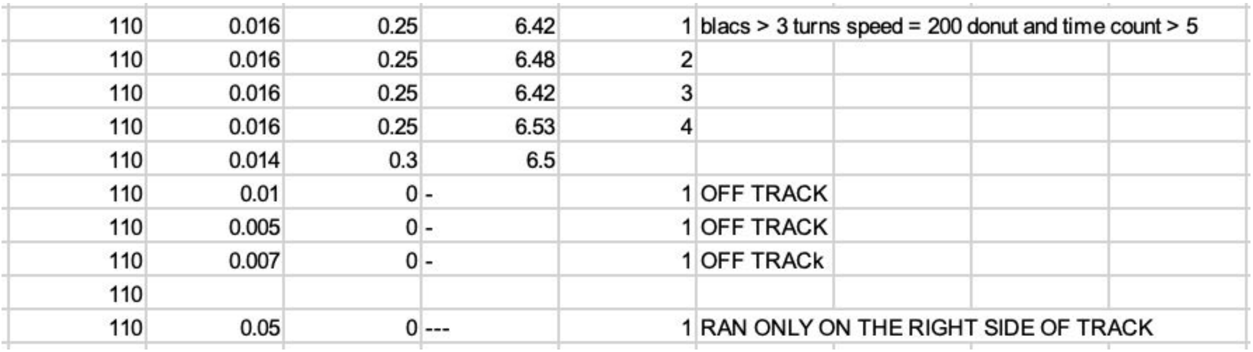
40	0.0025	0	N/A	1	OFF TRACK (DOESN'T CORRECT ENOUGH)
40	0.0025	0	N/A	2	OFF TRACK
40	0.0025	0	N/A	3	OFF TRACK
40	0.0025	0	N/A	4	OFF TRACK
40	0.004	0	7.93	1	slight wiggles around the straigh parts
40	0.004	0	7.89	2	
40	0.004	0	8.03	3	
40	0.004	0	8.11	4	
40	0.005	0	7.96	1	
40	0.005	0	7.86	2	
40	0.005	0	7.95	3	
40	0.005	0	7.8	4	
40	0.006	0	7.56	1	Continuously wiggles
40	0.006	0	7.53	2	
40	0.006	0	7.5	3	
40	0.006	0	7.55	4	
40	0.007	0	7.92	1	
40	0.007	0	7.9	2	
40	0.007	0	7.8	3	
40	0.007	0	7.85	4	
40	0.008	0	7.52	1	still wiggles
40	0.008	0	7.72	2	
40	0.008	0	7.52	3	
40	0.008	0	7.95	4	
40	0.009	0	7.86	1	
40	0.009	0	7.86	2	
40	0.009	0	7.81	3	
40	0.009	0	7.9	4	
40	0.01	0	7.72	1	very fast around track, it doesn't go off, there is more wiggling than the slower speeds we had
40	0.01	0	7.9	2	very fast around track, it doesn't go off, there is more wiggling than the slower speeds we had
40	0.01	0	7.69	3	very fast around track, it doesn't go off, there is more wiggling than the slower speeds we had
40	0.01	0	7.8	4	very fast around track, it doesn't go off, there is more wiggling than the slower speeds we had
40	0.015	0	7.1	1	
40	0.015	0	7.32	2	
40	0.015	0	7	3	
40	0.015	0		4	OFF TRACK (inconsistent)
40	0.02	0	6.72	1	Very sharp wiggles
40	0.02	0	6.99	2	
40	0.02	0		3	OFF TRACK (OVERCORRECTS)
40	0.02	0		4	OFF TRACK

Figure [3] Testing Kp and Kd at speed 80



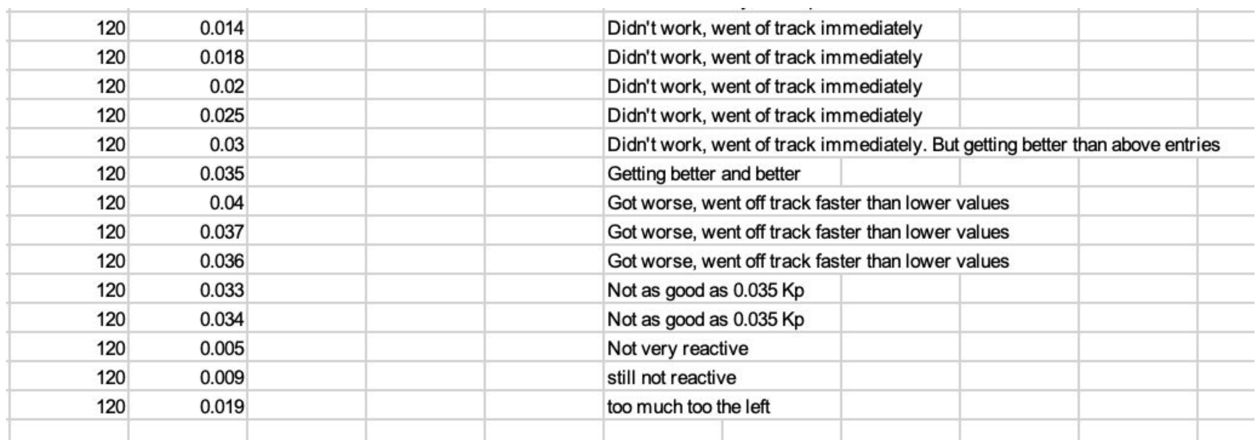
80	0.05				WENT OFF TRACK DID A FEW DONUTS
80	0.01				OFF TRACK Doesn't adjust fast enough to make it around the bend
80	0.1	0	---		OFF TRACK
80	0.001				OFF TRACK (Doesn't turn) Kp needs to be higher to account for error
80	0.0025				WENT STRAIGHT BUT DIDNT FOLLOW THE CURVE
80	0.009				Can't make it around the curve OFF TRACk
80	0.02				Turns too hard; can't make around the curve
80	0.004				goes around curve until its time to make a left bgut it doesnt
80	0.006				Started to turn around the curve but went straight
80	0.0055				Going right ok but doesnt turn left on curve
80	0.0075				Started to turn around the curve but went straight
80	0.003				Doesnt react fast enough to curves
80	0.0035				Doesnt react fast enough to curves
80	0.015				wiggles ALOT but makes it
80	0.017				wiggles ALOT but makes it
80	0.0125				wiggles not as mcuh as 0.017 & 0.015 but still makes it
80	0.011				wiggles not alot but has problems at the end of the track, it goes off
80	0.0115				wiggles not alot but has problems at the end of the track, it goes off
80	0.012				wiggles not alot but has problems at the end of the track, it goes off
80	0.013				
80	0.014				Works on all positions, wiggles
80	0.014	0.25			Works beautifully, on all positions

Figure[4] Testing Kp and Kd at Speed 110



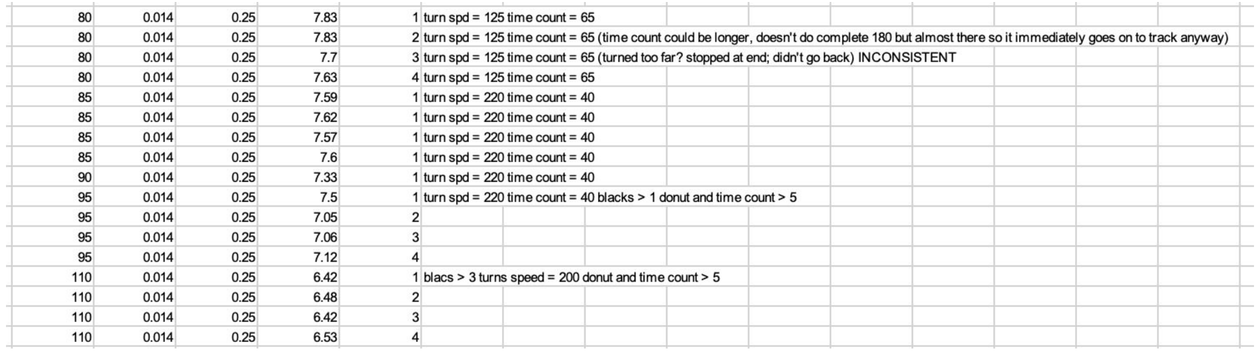
110	0.016	0.25	6.42	1	blacs > 3 turns speed = 200 donut and time count > 5
110	0.016	0.25	6.48	2	
110	0.016	0.25	6.42	3	
110	0.016	0.25	6.53	4	
110	0.014	0.3	6.5		
110	0.01	0	-	1	OFF TRACK
110	0.005	0	-	1	OFF TRACK
110	0.007	0	-	1	OFF TRACk
110					
110	0.05	0	---	1	RAN ONLY ON THE RIGHT SIDE OF TRACK

Figure [5] Testing Kp and Kd at speed 120



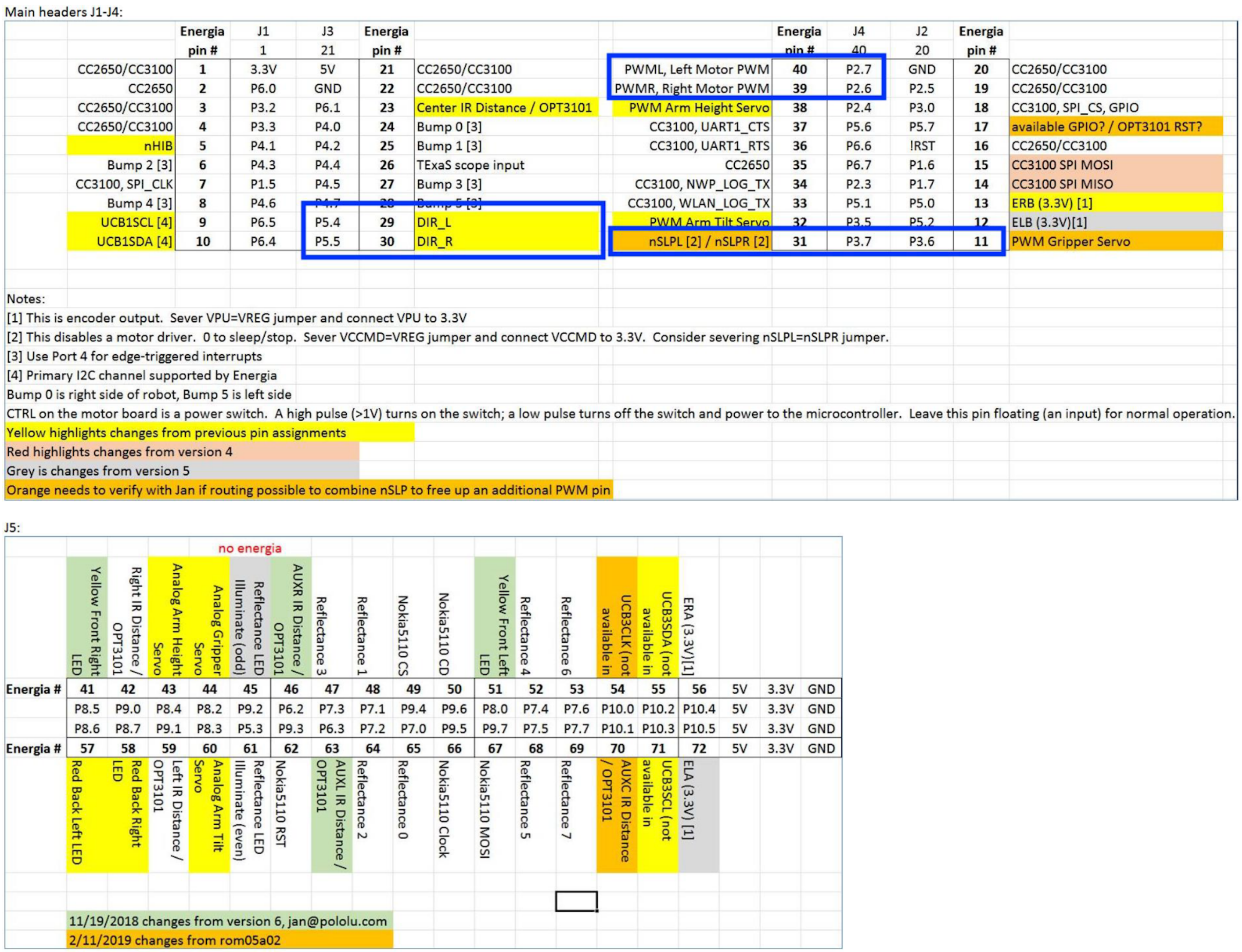
120	0.014			Didn't work, went of track immediately
120	0.018			Didn't work, went of track immediately
120	0.02			Didn't work, went of track immediately
120	0.025			Didn't work, went of track immediately
120	0.03			Didn't work, went of track immediately. But getting better than above entries
120	0.035			Getting better and better
120	0.04			Got worse, went off track faster than lower values
120	0.037			Got worse, went off track faster than lower values
120	0.036			Got worse, went off track faster than lower values
120	0.033			Not as good as 0.035 Kp
120	0.034			Not as good as 0.035 Kp
120	0.005			Not very reactive
120	0.009			still not reactive
120	0.019			too much too the left

Figure [6] Fixing Kp and Kd



80	0.014	0.25	7.83	1	turn spd = 125 time count = 65
80	0.014	0.25	7.83	2	turn spd = 125 time count = 65 (time count could be longer, doesn't do complete 180 but almost there so it immediately goes on to track anyway)
80	0.014	0.25	7.7	3	turn spd = 125 time count = 65 (turned too far? stopped at end; didn't go back) INCONSISTENT
80	0.014	0.25	7.63	4	turn spd = 125 time count = 65
85	0.014	0.25	7.59	1	turn spd = 220 time count = 40
85	0.014	0.25	7.62	1	turn spd = 220 time count = 40
85	0.014	0.25	7.57	1	turn spd = 220 time count = 40
85	0.014	0.25	7.6	1	turn spd = 220 time count = 40
90	0.014	0.25	7.33	1	turn spd = 220 time count = 40
95	0.014	0.25	7.5	1	turn spd = 220 time count = 40 blacks > 1 donut and time count > 5
95	0.014	0.25	7.05	2	
95	0.014	0.25	7.06	3	
95	0.014	0.25	7.12	4	
110	0.014	0.25	6.42	1	blacs > 3 turns speed = 200 donut and time count > 5
110	0.014	0.25	6.48	2	
110	0.014	0.25	6.42	3	
110	0.014	0.25	6.53	4	

Figure [7] Pinchart, courtesy of Polulu (provided by ECE3 instructors at UCLA)



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