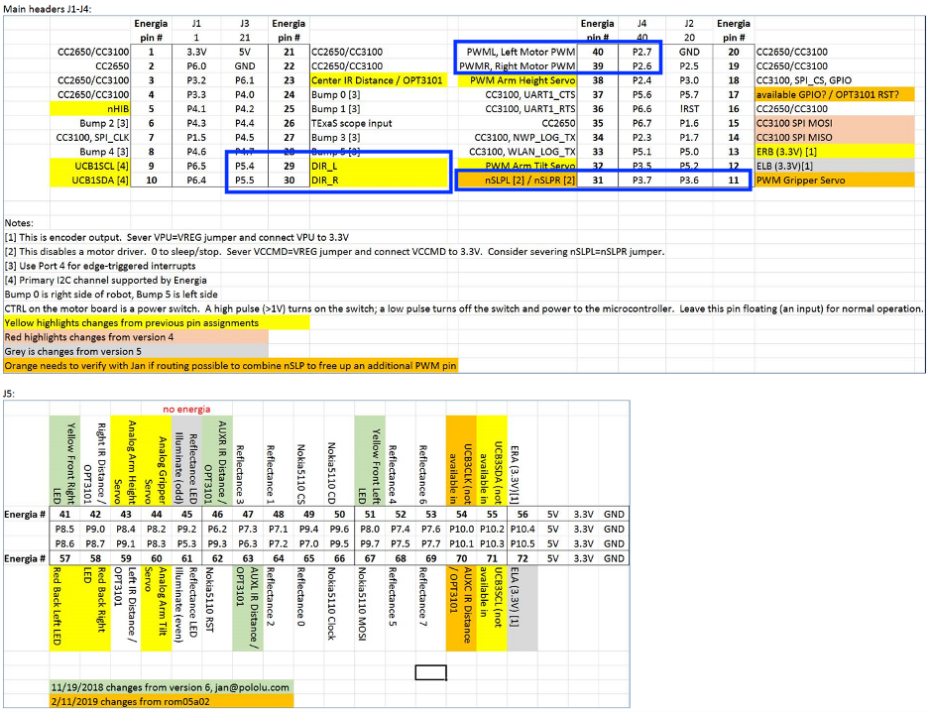
**Electrical and computer Engineering 3 Car Project**

## 1. Develop:

**First step:** understand how to code in Arduino and check the car works properly.

In the First project Day Instructions Lab, we plugged in the battery and checked that both the green and blue LEDs were on. Then, we inserted the USB cable and uploaded the Blink code to RSLK via Arduino. the red LED started blinking, which indicated that LaunchPad was working well. In the next step, we wrote a simple program based on the Basic code downloaded from CCLE and verified that the motor worked properly by setting the PWM correctly using the Pin Chart shown in Figure 1. In the last step, we used IR_Sensor_Example to test if the sensor works properly and read the numbers between 0 and 2500.



Main headers J1-J4:

| | Energia pin # | J1 | J3 | Energia pin # | | | Energia pin # | J4 | J2 | 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 | IRST | 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 Servo | Analog Arm Tilt | 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 1. MSP432 PinChart.
From Dr. Briggs

In this process, we know the car will work properly, which prepares us for the future steps. At the same time, we know how the code will affect the performance of the car. For example, the speed in the setup is not allowed to be less than 0 and the maximum speed is 255.

**Second step**: understand how the sensor works for the car and how PD control affects the car.

We will use the 8-Channel QTRX Sensor Array, known as a phototransistor, to get the position of the car by reading the values from sensors. The schematic diagram of an individual RC sensor channel is shown in Figure 2.

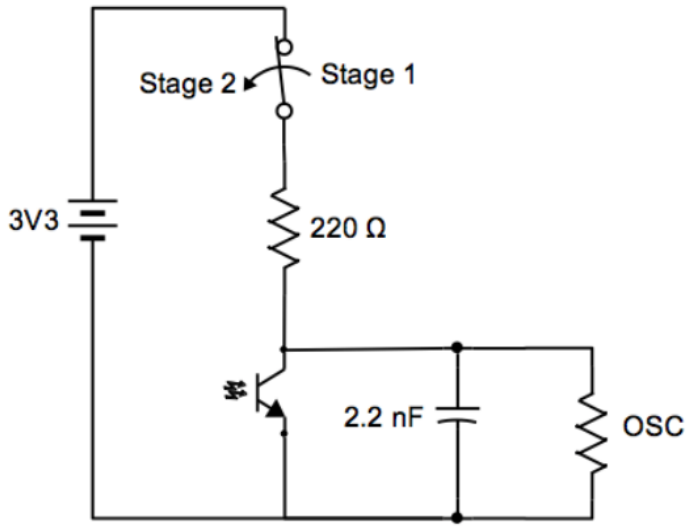



*Figure 2: Simulation of phototransistor circuit*
(Source: EC ENGR 3 week 2 Lab Worksheet)

When the sensor needs to be read by the system, the circuit needs switch stage 1, which is shown in Figure 3.

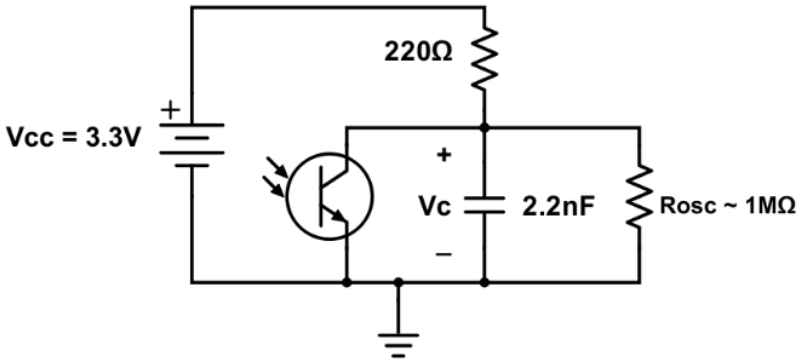



*Figure 3: Simulation of phototransistor circuit, stage 1*
(Source: EC ENGR 3 week 2 Lab Worksheet)

The capacitor will have enough time to charge, by Voltage Divider, KVL and Differential

Equation. We get the equation: $V_c(t) \approx V_{cc} - V_{cc}e^{-\frac{t}{C(220\Omega)}}$ .

After the capacitor is fully charged, the phototransistor will switch to the stage 2, the capacitor will discharge, which is shown in Figure 4.

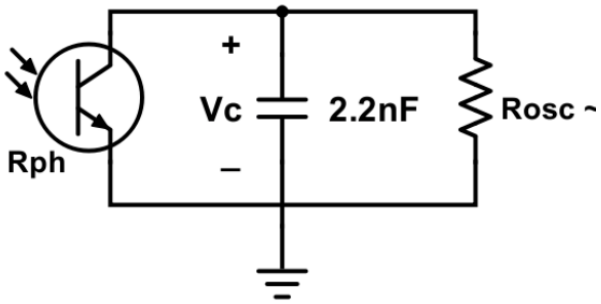



*Figure 4: Simulation of phototransistor circuit, stage 2*
(Source: EC ENGR 3 week 2 Lab Worksheet)
Then we get the equation:

$$V_c(t) = V_{cc}e^{-\frac{t}{C(R_{ph}//R_{OSC})}} \approx V_{cc}e^{-\frac{t}{C \cdot R_{ph}}}$$

According to Lab 2, we know that the phototransistor will be lower resistance when there is a brighter environment (Figure 5).

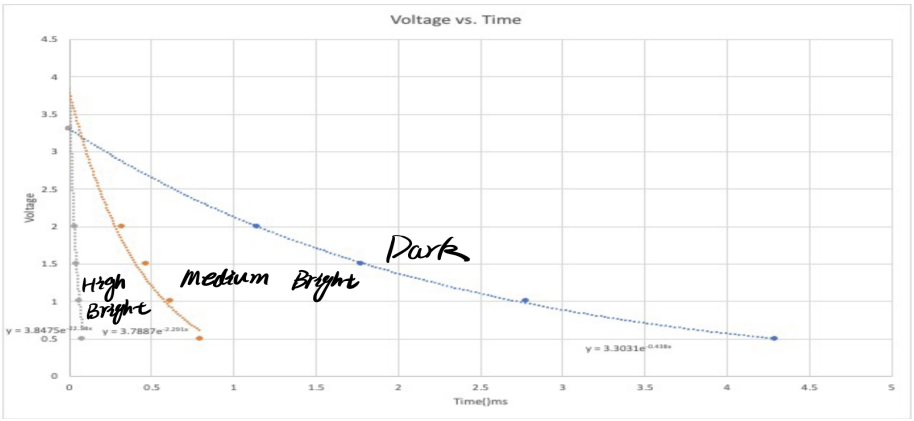



*Figure 5: lighting conditions resulting curves*
By the constant of RC circuit: $\tau$=RC, when the environment is brighter, the sensor resistance will be lower, then the smaller $\tau$ will be, which will reveal and print out as the value between 0 to 2500 to indicate the degree of light change in the RLSK car.

Based on the sensor readings, we can get the error from the 8-Channel QTRX Sensor Array to get the position of the car. The wheel speed is then adjusted using PD control methods. In this project, We chose Proportional control and Derivative control.
**Proportional control:** To respond to the current error term.
**Derivative control:** To Prevent over adjusting and improve stability.
Collecting the error from the 8-Channel QTRX Sensor Array then using the PD control method to adjust the speed of the wheels to promote the car on the right track.

**Third step**: Write the code and test it on Straight Line Stability Check Light AllRows shown in Figure 3. Adjust the PD values based on its performance.

By the sensor reading values, write the function to weight each sensor to promote the car to keep on the center of track.

**Last step**: Test the car on the Race Day Path 21Su Light AllRows and adjust the PD value and speed.

## 2. Conduct Tests:

**Idea and Process:**
In order to successfully run the car on the race day path, we should successfully run the car in a straight line, as it is easier to make the correct adjustments in a straight line than on the race day path.

First, we chose 15-14-12-8 as our sensor weighting scheme, which is more sensitive than the 8-4-2-1 based on the calibration part shown in Figure 5. We want our car to go faster, therefore we want our sensors to become more sensitive.

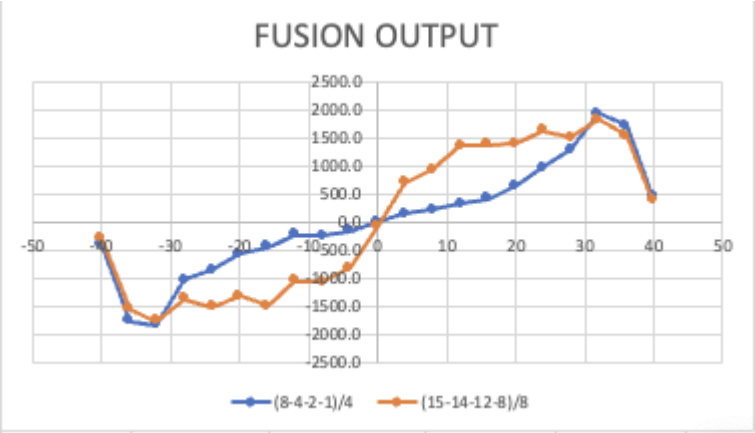


Figure 5. Weighting Scheme

Created on Excel

After correctly adjusting all settings, including the sensor weighting scheme, speed, Kp and Kd values, we can add more features to make the car turn 180 degrees at the end of the path and stop at the start position.

We can then test the car on the race day path and increase the speed by adjusting the values of Kp and Kd to make the car run faster.

In the following tests, they all use regular paths and are tested during the daytime.

**Test 1:**
We set the initial velocity to 50, the value of Kp to 0.01, Kd to 0, and the weighting scheme of the sensor to 15-14-12-8. In this process, we want to approach the appropriate value of Kp at a velocity of 50. This will help us to accomplish the goal of the third step in the development plan. We can find out the appropriate Kp value for the base speed, which is 50, and can make the car run successfully.

The car can run successfully in positions 1 to 3 but fails in position 4. For rounds 1 to 3, the car starts with frequent oscillations. Then, in the second half of the path, it becomes more stable. However, it took a lot of time to finish the first stroke. Therefore, we decided to increase the value of Kp to reduce the frequency of oscillations while keeping the other elements.

**Test 2:**
We keep the weighting scheme of the sensor, the initial speed, and the value of Kd. Then, we increase the value of Kp to 0.02 to find the appropriate Kp for a speed of 50. once we find the appropriate Kp. We can test the value of Kd in order to improve the development plan of the PD control in step 3.

The car has a better performance in this path and succeeds in all starting positions. The car oscillates at a lower frequency at the beginning and gains stability faster than in Trail 1. However, the amplitude of the car's oscillations remained large. So, we decided to adjust the Kd to reduce the amplitude of oscillations.

**Test 3:**
We retain the values of all the elements except Kd. Note that the value of Kd should be greater than Kp, so we set Kd to 0.03. If the car performs well in this test, this indicates

that we have found the right values for Kp and Kd for a speed of 50. We can then try to increase the speed and adjust the PID control values to help the car go faster.

For rounds 1 to 3, the car performs better than before. It has a lower oscillations frequency at the beginning and the swing is smaller than before, which helps the car become more stable on the track. Although round 4 was still successful, the performance was not as good as the first 3 rounds, which is acceptable. The next step is to increase the speed to reduce the time.

**Test 4:**
We preserve the weighting scheme of the sensors, and the values of Kp, Kd. Then, we increase the velocity to 100. In this test, we need to increase the velocity first and then adjust Kp, Kd to reduce the time of one trip on the path.

It succeeds only in the first round and fails in all the others. In the first round, it oscillated at a high frequency and could not remain stable on the track. In this case, we decided to increase the value of Kp to keep the car on the right track.

**Test 5:**
In this test, we keep the speed at 100, kd at 0.03, and the same weighting scheme for the sensors. We increased the value of kp to 0.029 to find a suitable kp at a speed of 100. We are adjusting the PID control element used to match the faster speed to reduce the time the car takes to run.

However, it failed at all starting positions. We suspect that the speed is too fast or the value of Kp is too large because the first turn is too large and takes the car off the correct track. So, we decided to decrease the value of Kp.

**Test 6:**
We retain the speed at 100, Kd at 0.03 and the same sensor weighting scheme. We lowered the value of Kp to 0.027. To keep finding the appropriate Kp with velocity at 100 which can accomplish the adjustment in the PID control part for Step 3.

Although there are still high frequency oscillations at the beginning and the amplitude of the car's oscillations is still large, it succeeds at all starting positions and for a shorter time than before. Therefore, the next step is to reduce the amplitude of the oscillations in order to improve the stability of the car while driving on the track.

**Test 7:**
In this test, we keep the speed at 100, Kp at 0.027 with the same sensor weighting scheme. We increase the value of Kd to 0.038. In this process, we can reduce the amplitude of the oscillations by finding the appropriate Kd. If the car performs good in this test, it will be the preparation for step 4, as we will adjust the setting of the car on Race Day Path based on this data.

Since the adjustment, the oscillation frequency of the car is reduced, and the amplitude of oscillation becomes smaller. It is easier for the car to obtain a stable performance. As a result, the car has been successful in all starting positions. The next step is to use this data for testing on the road on race day.

**Test 8:**
We will use the same data obtained on test 7. Then, change the Straight Line to Race Day Path to test the performance on the car. This is the starting point of Step 4. We might add the 180 degrees turn function and stop at the beginning position function in the code if it performs well. Otherwise, we will need to adjust the Kp and Kd first.

The car finishes its run with low oscillation frequency and small amplitude oscillation. This shows the stability that the car exhibits on different tracks. This is based on the results of previous tests, where the speed was 100, the kp was 0.027 and the kd was 0.038.

**Test 9:**
We keep the same sensor weighting scheme with Kp of 0.027, Kd of 0.038 and speed of 100. In this test, we will test the car's time after inserting the turnaround function and stop function. The goal is to have the car complete the test in 8.5 seconds to accomplish the final step of the development plan.

The car performed well in the test. It can turn 180 degrees at the end of the path and follow the track back to the starting position in a complete stop with an estimated time of about 10 seconds. Note that in the setup part, the car had to be delayed by 2 seconds before running. This shows that we have basically completed the development plan and met all the requirements for this project.

3. **Analyze:**

| | Track | Starting Position | Speed | Kp | Kd | Weighting Scheme | Success/Fail | Time (s) |
|---|---|---|---|---|---|---|---|---|
| Test 1 | Straight Line | 1 | 50 | 0.01 | 0 | 15-14-12-8 | Success | 8.32 |
| | | 2 | | | | | Success | 8.36 |
| | | 3 | | | | | Success | 7.9 |
| | | 4 | | | | | Fail | |
| Test 2 | Straight Line | 1 | 50 | 0.02 | 0 | 15-14-12-8 | Success | 7.91 |
| | | 2 | | | | | Success | 7.87 |
| | | 3 | | | | | Success | 7.58 |
| | | 4 | | | | | Success | 7.63 |
| Test 3 | Straight Line | 1 | 50 | 0.02 | 0.03 | 15-14-12-8 | Success | 7.77 |
| | | 2 | | | | | Success | 7.64 |
| | | 3 | | | | | Success | 7.89 |
| | | 4 | | | | | Success | 7.83 |
| Test 4 | Straight Line | 1 | 100 | 0.02 | 0.03 | 15-14-12-8 | Success | 6.45 |
| | | 2 | | | | | Fail | |
| | | 3 | | | | | Fail | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 4 | | | | | Fail | |
| Test 5 | Straight Line | 1 | 100 | 0.029 | 0.03 | 15-14-12-8 | Success | 5.35 |
| | | 2 | | | | | Fail | |
| | | 3 | | | | | Fail | |
| | | 4 | | | | | Fail | |
| Test 6 | Straight Line | 1 | 100 | 0.027 | 0.03 | 15-14-12-8 | Success | 5.29 |
| | | 2 | | | | | Success | 5.34 |
| | | 3 | | | | | Success | 5.33 |
| | | 4 | | | | | Success | 5.28 |
| Test 7 | Straight Line | 1 | 100 | 0.027 | 0.038 | 15-14-12-8 | Success | 5.26 |
| | | 2 | | | | | Success | 5.31 |
| | | 3 | | | | | Success | 5.27 |
| | | 4 | | | | | Success | 5.35 |
| Test 8 | Race Day Path | 1 | 100 | 0.027 | 0.038 | 15-14-12-8 | Success | 6.34 |
| Test 9 | Race Day Path | 1 | 100 | 0.027 | 0.038 | 15-14-12-8 | Success with all the requirement | 10.25 |