ECE 3 Final Report

Kameron Carr 504988167

Hien Ho 505190709

TA: Xin Li

Section 1A

June 15, 2019

1 Introduction and Background

The goal of this project if to use knowledge from the course to program a small car to follow a path designated by a black line within a given amount of time. As an additional part of the project, you must read data from the shaft encoders to more precisely control the speed of the car.
The car must determine its error from the the center of the path can correct itself. This is done by using two feedback loop controllers. One is a linear controller so it simply takes the difference between where it is and where it wants to be in order to make adjustments. For example, if I am to the right of the line, turn left. The second is the derivative controller. This one looks at how its position is changing to make adjustments. For example, if I am drifting left, turn right. These two controllers can give the car different suggestions on which way to turn. Taking the sum of the two controller values can give the car a good sense of what adjustments need to be made.
The path following system uses IR LEDs to shine light onto the ground, then the IR sensors detect how much light reflects off the ground below it to determine if the ground is black or white.

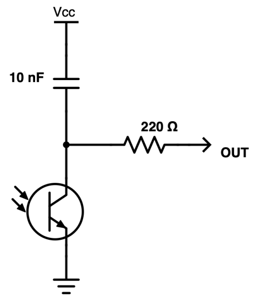

Figure 1: Schematic for IR sensor[2]
Credit: Digi-Key.com/SchemeIt

As you can see in Figure 1, the path sensor uses a resistor, capacitor, and phototransistor to detect IR light. When you put the sensor into output mode and set it to high, the out port becomes an in port and is set to V_{cc}. Therefore the voltage across the capacitor becomes 0V as long as you give it enough time. Upon changing the sensor pin to input mode, charge flows through the phototransistor depending on how much light is detected. This means that the voltage read on the pin will start at V_{cc} and decay to zero over time. The rate of this decay depends on the amount of IR light entering the phototransistor[2].

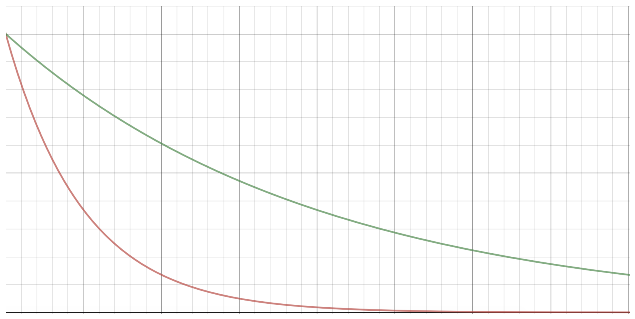

Figure 2: Example decay curves[2]
Credit: Desmos.com

In Figure 2 you can see what the curves may look like for different amounts of IR light going into the phototransistor. The red curve represents if more light is going into the phototransistor and the green curve represents less light. The difference in voltages after a certain amount of time is what allows the sensor to read in either a 1 or a 0 depending on how much IR light ground below it reflects. When you do a digital read, it returns a 1 when the voltage is over a certain value, and zero if the voltage is below that value. To get a good reading, you must take the reading when the high curve is above the threshold and the low curve is below the threshold.

2 Testing Methodology

2.1 Motors

2.1.1 Test Setup

To start, you may need to assemble your car. This process includes assembling the wheels, putting the motors into the clamps on the car, and putting the LaunchPad on the car. You may need a TA's help to solder the header connections. Also it is important to remove the 5V jumper on the LaunchPad. If you are using the same car as the one used in this lab, not removing the jumper **may cause permanent damage to the car when plugging into a computer**. Also make sure to put fresh batteries in your car and turn it on using the switch on the back end.
In order to be able control the motors in Energia, we first looked up the pin numbers on the MSP432 Pin Chart. From this chart we were able to find all the relevant pins to control the

motors.

Pin Functionality	**Pin Number for Left Motor**	**Pin Number for Right Motor**
No Sleep (nSLP)	31	11
Direction (DIR)	29	30
Duty Cycle (PWM)	40	39

Some of these pins also needed to be set to the correct mode and values. In Energia, the modes are set with the `pinMode` function, and the pin value is set through the `digitalWrite` function[1].

Pin Functionality	**Pin Mode**	**Pin Value**
No Sleep (nSLP)	OUTPUT	HIGH
Direction (DIR)	OUTPUT	HIGH

2.1.2 How the Tests were Conducted

Knowing that the power to the motors ranges from 0-255, we wanted to get a feel for how fast the car went at different power levels. These are only rough guidelines of what we found to be true for our car. Note that these speeds can fluctuate depending on your car and the charge of your batteries. If you want to control the speed of the car with more precision, you must use the motor shaft encoders to measure the rotations.

Car Speed	**Motor Power**
Not Moving	0 - 20
Slow	20 - 40
Medium	40 - 90
Fast	90 - 255

Next we tested the evenness / straightness of the motors. We wanted to ensure that the car drove as straight as possible at the base power we would use in our path following program. Knowing the length of the course and the time limit, we were able to estimate that for our car to go fast enough, the base power of the motors would have to be around 60 on each motor. This is the target power we used when testing straightness. We noticed that going from zero power immediately to the target power would cause the car to jerk very quickly. For this reason, we used a for loop that changed the power of the motor incrementally with a small delay. This allowed the car to accelerate slowly and evenly without jerking around. We found this gave the most accurate results when trying to determine what power on each motor was needed to get the car to go straight.

Left Motor Power	Right Motor Power	Result
62	60	Pulled Right
60	60	Straight
60	62	Pulled Left

2.1.3 Data Analysis

From these tests we found that the motors must have power of at least 20 to move consistently. A base power above 90 was found to be quite fast. We suspect a speed above this would make it significantly more difficult to respond to input from the path sensors and make appropriate adjustments.

We also found that our car goes straight when the power of each motor is set to 60. This suggests that the car will go straight unless the power of the motors is different.

2.1.4 Test Data Interpretation

From these tests, we generally deduced that the motors were functioning as expected. By observing the general speeds of the motors at different power levels, we determined that a base power of 60 or greater would be a good starting goal for our car to finish the track in time. Also when using the shaft encoders to determine the speed of the vehicle, it is important to not go below 20 power on the motors, otherwise the feedback loop breaks because the motors are not turning. If the motors are not turning, the shaft encoders will not return information that can be used to calculated the speed.

The tests on motor straightness showed that our motors went at the same speed when the powers were equal. Therefore we did not include a bias value in our final code.

2.2 Path Sensors

2.2.1 Test Setup

As with the motors, testing the path sensors requires the car to have the batteries put in and the switch turned on.

In order to see the values from the sensors, we will need to be able to print to the serial monitor. The serial monitor must be initialized using `Serial.begin` and a data rate of 9600 bps.

To get readings from the sensors, we need to initialize the pins for the sensors as well as the infrared LEDs. We look at the pin chart to find the pin values we can use in Energia to refer to the LEDs and sensors.

Device	Pin Number
Sensor 0 (Right Most)	65
Sensor 1	48
Sensor 2	64
Sensor 3	47
Sensor 4	52
Sensor 5	68
Sensor 6	53
Sensor 7 (Left Most)	69
IR LEDs (Odd)	45
IR LEDs (Even)	61

All of the LEDs are set to `OUTPUT` using the `pinMode` function and to `HIGH` using the `digitalWrite` function.

2.2.2 How the Tests were Conducted

To get readings from the sensors, we created a for loop that goes through the 8 sensors. For each sensor, you first have to set it to `OUTPUT` and `HIGH` then delay for 10 ms to charge the capacitor in the sensor. Then you set the sensor to `INPUT` and delay for a set amount of time before getting a reading using the `digitalRead` function. We tested a few values for this second delay. We did this by running the program and printing the sensor values to the serial monitor with the `Serial.print` function. We then moved the car side to side over a paper with a black line printed on it.

Delay	Behavior
4 ms	All Zeros
3 ms	Some Incorrect Zeros
2 ms	Correct
1 ms	Correct

2.2.3 Data Analysis

The car properly identified the difference between the white paper and the black printed paper when the delay after turning a sensor to input was 1 or 2 ms. A delay greater than 2ms resulted in unexpected / incorrect results. The errors were when the value should have been 1 but was instead 0. This is expected as if the delay is too large, then the capacitor will have too much time to discharge.

2.2.4 Test Data Interpretation

Originally we took these results and decided that 2ms would be a good delay to have for our path following program. However, later when we tried sensing the horizontal black lines for the purpose of turning around or stopping, we found that a delay of 1 ms resulted in more accurate results.

3 Results and Discussion

3.1 Test Results

When testing our car we mainly modified the scaling values of our linear and derivative controllers. We also modified the base speed and the number of cycles over which the derivative controller was calculating its difference, which we will call the look-back amount.

Value	**Behavior When Too Low**	**Behavior When Too High**
Linear Controller	Doesn't follow curve in path	Swerves out of control
Derivative Controller	Swerves out of control	Jerks off path
Base Speed	Doesn't make it in the required time	Doesn't follow curve in path
Look-back	Swerves out of control or is very jerky	Responds too late to curve

Using this understanding of the behavior, we were able to find values that worked and test them to find how fast the car finished the track.

Base Speed	Linear Controller	Derivative controller	Look-Back	Track Time
50	4	3	4	60 sec
75	5	3.5	3	40 sec
60	4	3	4	47.5 sec

In this chart the values for the two controllers is the multiple of the error that was added to the base speed. For the linear controller, the error is the current position minus the center location. In the derivative controller the error is the current position minus the position that was look-back value cycles in the past.

Since the time requirement for this project was 70 seconds, we determined that the observed behavior of the car was acceptable to meet the requirements. On the 40 second run, it was less reliable as the car went so quickly that it would often lag behind the curves in the path and then turn very hard to catch up to it. On race day we decided to use the 47.5 second run as it had a good balance of reliability and speed.

3.2 Race Day Results

On race day, the car completed the track in 47.5 seconds. However, it did take three runs. One reason that the car may have run less reliably on race day than when we were testing is because the track was modified that morning. During testing, there were wrinkles in the track, while on the morning of race day, the entire track was flattened and retaped to the ground.

Our car also successfully completed the extra credit portion. It took data from the shaft encoders to get the amount of time per tenth of a rotation. It then took this information and converted it into a single car speed. It then compared the current speed to the target speed and adjusted by either speeding the car up or slowing it down.

4 Conclusions and Future Work

For the purposes of the assignment, our car fulfilled all of the requirements. We were able to follow the path and met the given time limit. However, for any real-world application, the reliability of the car would likely be an issue. On race day, our car did not succeed on the first run. The speed of the car in the end was not an issue because of the generous 70 second requirement.

Future work would focus on the reliability of the car at higher speeds. The impact of changing the look-back value was not discovered until later in the project. More time would allow for that value to be more finely tuned. Observations of some of the top performing cars looked that they used higher values for the linear controller. More experimentation may discover that a higher linear controller on our own car could boost performance. We would test this by raising the linear controller and see if it results in uncontrollable swerving. If swerving does occur, we could also trying simultaneously raising the derivative controller.

One advantage of our design over some of the other top designs observed is the smoothness of our car. The heavy reliance on the derivative controller resulted in a smooth, continuous motion of the car. This could be beneficial in any autonomous vehicle with passengers or fragile loads.

References

[1] *First Sketch*. Energia. https://energia.nu/guide/foundations/basics/tutorial_sketch/.

[2] Rahul Iyer. *Introduction to Arduino and Energia Programming*. 2019.