Evan Snyder, Denise Wang
EC ENGR 3 Lab 1B

EE3 19S Final Report

1. Introduction

The goal of the final project was to program a car that would be able to drive through a given path with curves under a certain time limit. We were introduced to Arduino IDE, which is C-style programming without project and class setup overhead, and Energia IDE, which is an Arduino variant for Texas Instruments microcontroller platforms.

The board used to drive the car is programed through a UART serial connection and works in a few different ways including digital input and output, analog input and output, analogRead(), analogWrite(), and PWM (pulse-width modulation). The digital input and output consist of digital voltage signals (0 corresponding to low and 1 corresponding to high) and applications responsible for reading (digitalWrite()) “on-off” type sensors and writing (digitalWrite()) sleep and reset pins. Analog inputs and outputs are responsible for reading and writing voltages outside of a discrete digital range, as its signals take in an infinite range of values. [1] The vehicle was able to detect the path due to the QTR-8RC Reflectance Sensor Array. This sensor module consists of sensors that provide a separate digital input output measurable output and used by charging the output node by applying voltage to its OUT pin. [2] The digitalRead() was a useful function for sensing the system because it reads the digital signal by using the digitalRead() method, takes pin numbers as an argument. The following images display the circuit schematic for the sensor module.

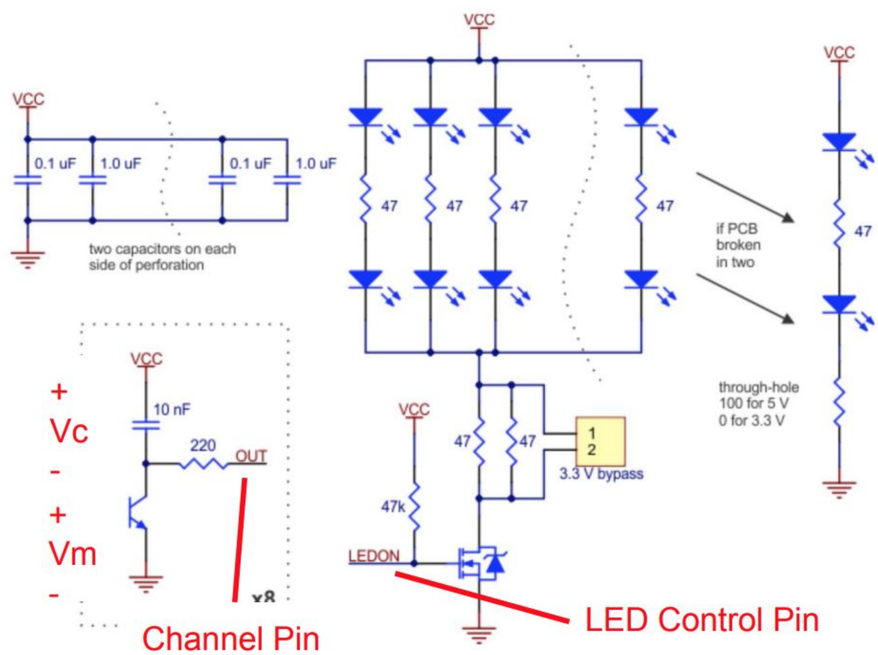

Figure 1- Circuit schematic for the sensor module.
[1] Image credit: https://www.pololu.com/product/961

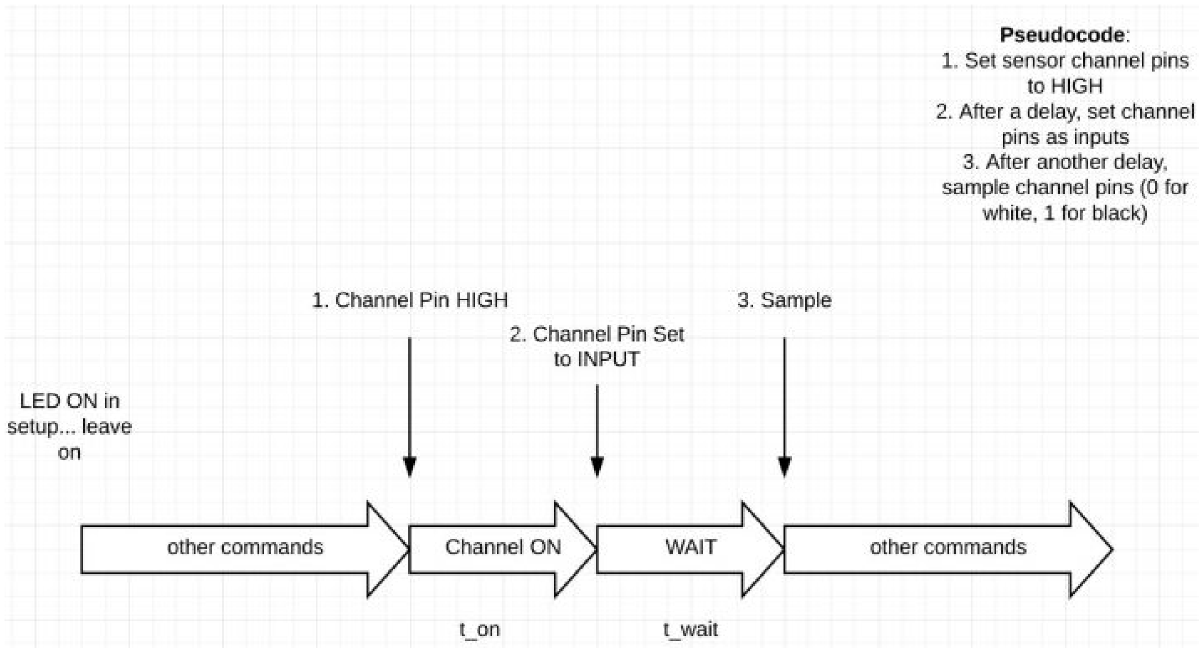

Figure 2- IR Reflectance Sensor- Proposed Sensing Scheme.
[2] Image credit: https://www.pololu.com/product/961

2. Testing Methodology:

Test Setup/Procedures:

The car was powered by 6 AA batteries, which served to power the microcontroller and motor. We tested the car by preloading a control program into the microcontroller, then running the car on one of the test tracks provided.

We initially implemented an integral controller, whereby the car's motion would be adjusted by increasing the speed of one of the wheels by a constant value for every sample that was off the line. To test how effectively the car performed, we ran the programmed car on the test tracks and measured the amount of time it took to complete a round-trip.

Data analysis/Interpretation:

Using the simple integral controller, we were able to program the car such that it would run the route successfully. However, the round-trip time was well over a minute, and any increase in the car's base speed resulted in a wild destabilization of the system, generally causing the car to careen off the track. Additionally, the use of a speed increase to modify direction caused the car to move erratically down the track. Even when it was making progress, it would end up doing so at a much slower rate than the base speed.

To solve this issue, we were forced to modify our approach. Instead of using the finicky integral-based controller we had developed, we reimplemented the car's controlling software, this time using a proportional controller. Under this new controller, the car's direction is adjusted by slowing down the speed of one of the wheels to a specific value, determined by the size of the deviation from the path and a proportionality constant. We found that it was best that the proportionality constant be such that the speed of the wheel at the maximum deviation from the path be 0 in all cases. This ensures that the car never leaves the path. This finished version of the car controller allowed the car to complete the path in around 35 seconds!

3. Race Day Discussion:

On race day, our car performed well. We had the second fastest time of all of the cars in our section, coming in at about 37 seconds. Because the proportional controller would slightly overcompensate in its steering of the car, the movement of the car was still relatively jerky and slower than it could have been. However, the car's movement was significantly faster than the original, integral-controlled version.

If we were to do the project again, we would implement several changes to our methods. First of all, we would try to implement a derivative controller for the car. It is possible that using a derivative-based controller as well as a proportional controller would make the car's movement much smoother than it is currently, leading to a faster race time.

4. Conclusions:

Additionally, it would be interesting to implement a feature whereby the car would slow down going into curves. This would allow the car to be more precise in its turning and stay more aligned on the track. We also hoped to implement some sort of speed monitoring through the built-in speed sensors on the car, but we ran out of time to implement this feature.

5. Illustration Citations

[1,2] Pololu - QTR-8RC Reflectance Sensor Array. (n.d.). Retrieved from https://www.pololu.com/product/961

6. References

[1] Introduction to Arduino and Energia Programming- Rahul Iyer, UCLA EE3 - Spring 2019
https://ccle.ucla.edu/pluginfile.php/2837919/mod_resource/content/0/Rahul%20Iyer%20%28Mentor%29%20EE3%20S19%20Presentation%20-%20Intro%20to%20Arduino%20and%20Energia%20Programming.pdf

[2] Pololu - QTR-8RC Reflectance Sensor Array. (n.d.). Retrieved from https://www.pololu.com/product/961