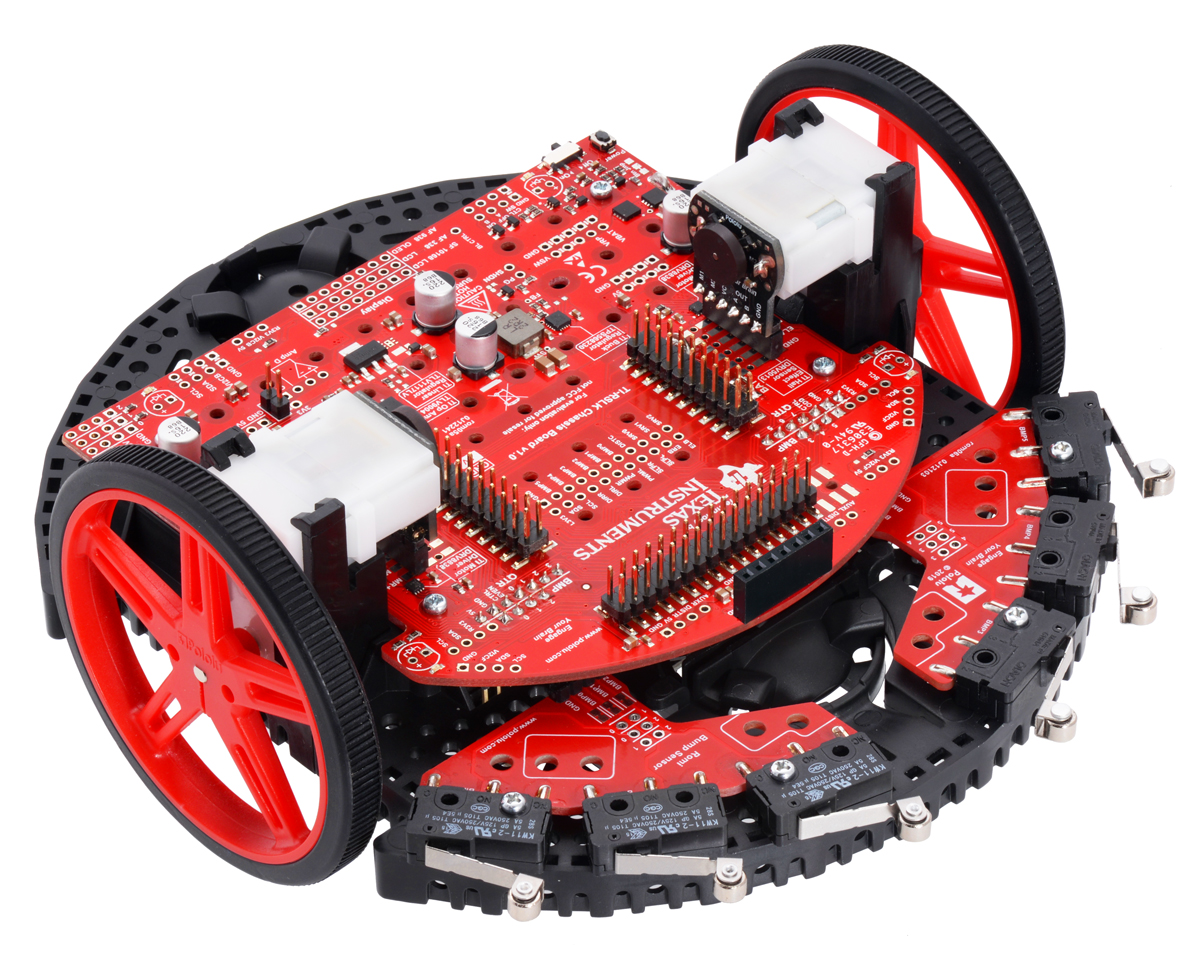
Line-Following Arduino Car Project Report



Team Name: Snail

 Chandaralong “Dylan” Phe (UID: 505 834 475)

Raphael Navarro (UID: 305 837 229)

EC ENGR 3: Introduction to Electrical Engineering, Lab 1E

Dr. Dennis M. Briggs

December 11, 2021

1. ***Objective***

To program the TI-RLSK car so that it can autonomously navigate and complete the provided track on race day.

1. ***Background Information***

In preparation for the project, we individually reviewed prior lab documents to recall knowledge of the components (e.g. TI-RLSK car, Analog Discovery 2, and Arduino IDE) and how to properly utilize them in favor of our project’s requirements.

1. ***Movement Capabilities of the Car***

For movement, the TI-RSLK car uses two DC motors connected as H-bridge circuit integrated with various components such as switches to activate or disable the right and left motor (nSLPR (pin 11) and nSLPL (pin 31)), switches altering the direction of rotation of the two motors (DIR\_R (pin 29), DIR\_L (pin 30)). Additionally, the speed of the motors are controlled using pulse-width modulation (PWMR (pin 39) and PWML (pin 40)).

To simplify conditioning the car’s behavior during the coding process, we created a table of control signals for different movements:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **nSLPR (11)** | **nSLPL (31)** | **DIR\_R (29)** | **DIR\_L (30)** | **Translated**  **Movement** |
| LOW | LOW | - | - | Brake |
| HIGH | HIGH | LOW | LOW | Forward |
| HIGH | HIGH | HIGH | HIGH | Reverse |
| HIGH | HIGH | HIGH | LOW | Spin CW |
| HIGH | HIGH | LOW | HIGH | Spin CCW |

**Table 1**: Control signals for different car’s movements

\*\*\*Turning is made possible by passing different PWM signals to each motor.

|  |  |  |
| --- | --- | --- |
| **PWMR** | **PWML** | **Direction of turning** |
| < PWML | > PWMR | RIGHT |
| > PWML | < PWMR | LEFT |

**Table 2**: Conditioned for different turning directions.

1. ***Implementation of the Phototransistors***

The eight phototransistors are essential to the line-following capabilities of the car. The phototransistors are aligned perpendicular to the direction of the car, and they must be activated in the Arduino IDE in order to read the light reflected off of the track. Through these readings, we are able to infer the positional error of the car relative to the line and correct its movement.

1. ***Calibration***

Given inconsistent lighting and shadows casting onto the car and track, it is important to note that due to the different placements of each sensor, the range of its readings (offset value to peak value) will also be different. Therefore, a calibration process must be carried out in order to normalize every sensor reading to be within the same range of 0 (denoting the offset value for each sensor) to 1000 (peak raw data reading) ,which can be achieved through the use of arithmetic operations.

Chart, line chart

Description automatically generated

**Graph 1**: Original reading for each sensor prior to calibration.

Chart, line chart

Description automatically generated

**Graph 2**: Normalized reading for each sensor after calibration.

1. ***Sensor Fusion***

In order for the car to autonomously follow the track, the car must always prefer to be centered over the line. To encourage this preference, we assigned higher weightings to the outer edge sensors as opposed to the most center sensors. Specifically, for our project we chose a system of weights as follows {-8, -4, -2, -1, 1, 2, 4, 8} for the sensors from left to right respectively.

Afterward, we fused (sum) those weighted sensor data together into a single quantity representing the error in the current coordination of the car. A larger sum, regardless of its sign, indicates that the car is going further off track to which the sign indicates the direction of the car is offset towards (left or right). On the other hand, a sum of 0 denotes the car in a straight position parallel to the track.

**Chart, line chart

Description automatically generated**

**Graph 3**: The possible range of error relative to position (measured in mm).

1. ***Implementation of PD Controller***

Once we establish the method for the car to autocorrect its position, we figured we also have to develop a method for the car to also autocorrect its motor speeds. For this project, we chose to implement a PD controller with Kp as the proportional controller constant to increase the car’s velocity in correcting its error in position whereas Kd serves as the derivative controller constant to adjust the car’s velocity based on the differences between the current and previous error in the car’s position.

The value of “addSpd” is either added or subtracted to each motor in order to automatically adjust PWM speed depending on positional error.

1. ***Development of Tests:***

To acquire the most useful data while testing, we preferred to conduct the tests in the lab room or under conditions that are similar to the lab room (e.g. a room that is well-lit by white light, no sunlight, etc.); additionally, we recalibrated the sensors for each testing location.

After we found suitable areas to test our car and code, our plan for the project was as follows:

1. Complete the turnaround
   * Find the best settings that will allow the car to successfully complete the 180o turn.
2. Implement PD control
   * Find the best values for both Kp and Kd that will allow the car to smoothly follow the line.
3. ***Conducting Tests***

***Controlled Parameters:***

While testing the car, we adjusted the values of base motor speed, Kp, Kd, turning constant (affects the sharpness of turns), turnaround speed, and turnaround time.

***Measured Variables:***

In order to measure the effectiveness of each controlled parameter, we measured the number of completed turns and voltage; additionally, we noted qualitative observations regarding turnaround reaction, reactiveness, oscillation, and stuttering.

***Test 1: Finding Turnaround Parameters***

We found that an effective method of finding the turnaround parameters was to place the car on a straight track or at the end of a track and see if the car successfully completes the turnaround.

***Test 2: Finding Values of Kp and Kd***

We tested the car at various speeds on the full track and narrowed down the values for Kp and Kd by trial and error.

1. ***Data and Analysis***

***URL to the full handwritten table of on-track test logs:***

<https://drive.google.com/file/d/1OhPZzjO4cDGpDBXe6G6GCka76V05GftN/view?usp=sharing>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Trial** | **Speed** | **kP** | **kD** | **Turn Speed** | **Turn Time (ms)** | **Comments** |
| 1 | 120 | 0.12 | 0.36 | 85 | 700 | Testing at high speed: Fails turn 3 |
| 2 | 120 | 0.12 | 0.5 | 85 | 700 | Fails turn 2, try lower kD |
| 3 | 120 | 0.12 | 0.4 | 85 | 700 | Fails turn 2, not very wobbly; try higher kP |
| 4 | 120 | 0.2 | 0.4 | 85 | 700 | Fails turn 3, small stuttering; try higher kP |
| 5 | 120 | 0.3 | 0.4 | 85 | 700 | Fails turn 1, kP too high |
| 6 | 120 | 0.1 | 0.4 | 85 | 700 | Fails immediately after 2nd turn; try new kD |
| 7 | 120 | 0.1 | 0.7 | 85 | 700 | Fails after 2nd turn, try higher kD |
| 8 | 120 | 0.1 | 2 | 85 | 700 | Fails turn 2, very small, high-frequency stuttering |
| 9 | 120 | 0.1 | 1.5 | 85 | 700 | Fails turn 7, not consistent at all; try new kP |
| 10 | 120 | 0.2 | 1.5 | 85 | 700 | Fails after 2nd turn |
| 11 | 120 | 0.2 | 2.5 | 85 | 700 | Fails after 2nd turn; kD too high |
| 12 | 120 | 0.13 | 1.5 | 85 | 700 | Fails turn 3; try low kP |
| 13 | 120 | 0.05 | 1.5 | 85 | 700 | Fails turn 4; try lower kP |
| 14 | 120 | 0.1 | 1.2 | 85 | 700 | Fails turn 2 |
| 15 | 120 | 0.1 | 0.8 | 85 | 700 | Fails turn 2 |
| 16 | 120 | 0.105 | 0.72 | 85 | 700 | Fails turn 2, try again |
| 17 | 120 | 0.105 | 0.72 | 85 | 700 | Fails turn 2 |
| 18 | 120 | 0.105 | 0.72 | 85 | 700 | Trying new turn const, turns too sharp on first turn |
| 19 | 100 | 0.08 | 0.72 | 85 | 700 | Removing turn const; fails after 2nd turn |
| 20 | 100 | 0.07 | 0.75 | 85 | 700 | Smooth until turn 4 where it went off course |
| 21 | 100 | 0.07 | 0.65 | 85 | 700 | Completes starting position 1 and 2, but fails position 3 and 4 |
| 22 | 100 | 0.07 | 0.9 | 85 | 700 | Fails at turnaround, met turnaround at an angle; adjust kP |
| 23 | 100 | 0.1 | 0.9 | 85 | 700 | Fails position 3 and 4, try higher kP |
| 24 | 100 | 0.2 | 1.5 | 85 | 700 | Fails turn 3; lower kP |
| 25 | 100 | 0.13 | 0.9 | 85 | 700 | Completes turnaround, but travels off course immediately after |
| 26 | 100 | 0.13 | 0.9 | 85 | 700 | Add back turn const; fails position 4, completes other positions |
| 27 | 100 | 0.13 | 0.9 | 85 | 700 | Attempting same settings again, not consistent, stuttering present |
| 28 | 100 | 0.1 | 0.7 | 85 | 700 | Completed track smoothly on all positions |
| 29 | 130 | 0.1 | 0.7 | 150 | 350 | STABLE: Same settings, higher speed & turning. Completed track smoothly on all positions |
| 30 | 150 | 0.13 | 0.8 | 150 | 350 | Aiming for higher speed; very wobbly, fails turnaround after coming in at an angle |
| 31 | 150 | 0.105 | 0.72 | 150 | 350 | Wobbly, but completes all starting positions |
| 32 | 150 | 0.1 | 0.65 | 150 | 350 | Same as above despite different kP and kD |
| 33 | 200 | 0.1 | 0.65 | 150 | 350 | Goes too fast when encountering turnaround |
| 34 | 200 | 0.1 | 0.65 | 150 | 350 | Adjusting code slightly, fails after turn 1 |
| 35 | 180 | 0.1 | 0.65 | 150 | 350 | Reverting code and lowering speed. Runs smoothly, but doesn't stop at turnaround properly |
| 36 | 180 | 0.1 | 0.65 | 150 | 350 | Same as above, stops after turning due to seeing turnaround line twice |
| 37 | 170 | 0.1 | 0.7 | 150 | 350 | Lower speed, fails turn 1 |
| 38 | 150 | 0.1 | 0.7 | 150 | 350 | Lower speed, not smooth. Fails at turnaround (delay too short) |
| 39 | 130 | 0.1 | 0.7 | 150 | 400 | 400ms turnaround delay, completed all positions smoothly |

**Table 3**: Logs gathered from testing on the complete track.

***Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated***

**Graph 4 and 5**: Tuning of Kp and Kd values across 39 tests.

1. ***Interpretation of Data***

* Interestingly, changing the base speed did not result in a large change for the best values of Kp and Kd; despite testing new values for Kp and Kd for every change in speed, the values still trend towards 0.1 for Kp and 0.7 for Kd.
* When the base motor speed is too high, the car has difficulty reacting to error, and it also prevents the car from properly stopping at the turnaround line.

1. ***Conclusion***

On race day, we ran the official track using the stable settings (trial #39) and completed the track in approximately 20 seconds. Afterward, we attempted higher speeds in hopes of completing the track under 15 seconds, but we were unable to get below 17 seconds on the unofficial track due to flaws in our code. Despite an unsuccessful final endeavor, we are glad to have completed this project successfully, and we enjoyed exercising our engineering mindset during the process.