ECE3 24F FINAL PROJECT REPORT

Michael Shara

TEAM NAME: RED WINGS

Contents

Planning and Execution

1. Code - Sensor Fusion Construction for PD Control to Follow a Straight Path

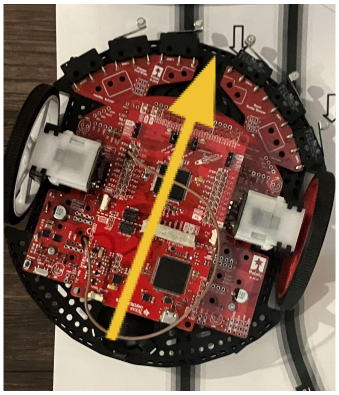
The first step was to plan how we wanted our fusion interface to interact with our car's decisions. There are 8 sensors on the car and each plays a vital role in keeping the car on the track, determined by a black line.

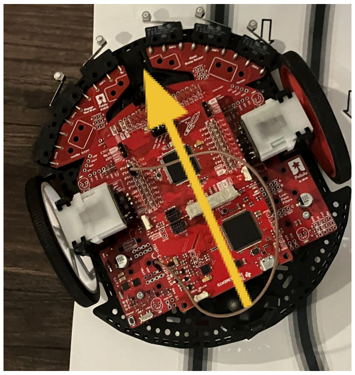
It was determined that sensor values would be stored into a single array of 8 values, one for each sensor. The goal is to determine how the speed of the wheels should change to keep the car directly on top of the black line and complete the track, turn around, and stop at its original position.

From our sensor calibration, we established a baseline for what the sensors should consider as the black line versus the white outer perimeter. Using our fusion code, we transform the batch of 8 elements in the sensor array into an error value that explains where the car is relative to the black line. It can then be determined by a positive or negative error value for which side of the line the car is on and react by adjusting the speed of both wheels to fix its direction to align itself back above the black line. If the error value is positive, our car is to the right of the track and the right wheel should speed up while the left wheel slows down to change its direction toward the track. Equivalently, If the error value is negative, our car is to the left of the track and the left wheel should speed up while the right wheel slows down to change its direction toward the track. The magnitude that our wheels should change in speed is determined by our constants called "Kp" (proportional constant) and "Kd" (derivative constant). In the code, the Kp determines the car's immediate response to the error (deviation from the black line), scaling the correction proportional to the size of the error. The Kd adjusts the car's movement based on the rate of change of the error, smoothing out oscillations and helping the car stabilize its path above the black line.

This was tested by starting our car at different angles with respect to the black line to see which direction the car turns (as shown in Figure 1a. And Figure 1b). This would ensure our error value implementation is effectively working.

[Figure 1a.]

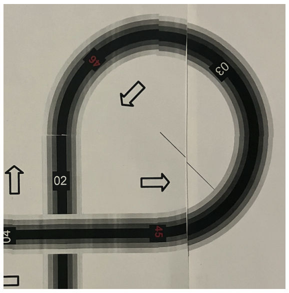




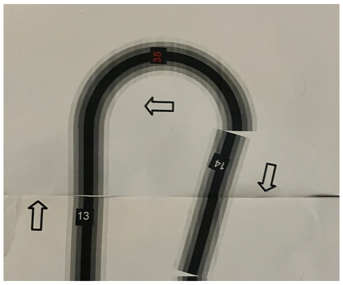
[Figure 1b.]

2. Tight Turns

The next challenge that was tackled were the tight turns. The turns on the track were constructed in a way that the car would sense it is straying off the track and make a tight turn. The issue was the car would then stray off the track, therefore losing positioning of the black line in any one of the sensors. The problem was that the car was not turning tight enough in situations like sudden tight turns or the donut turn to stay above the black line and allow the sensors to maintain contact with the track position. (See Figure 2a. and Figure 2b.)



[Figure 2a.]



[Figure 2b.]

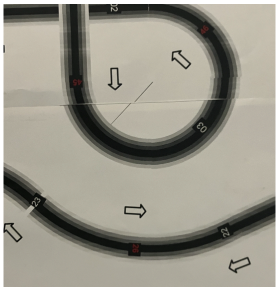
To solve these issues, the wheels were enabled to not only change its speeds in the forwards direction to control the car’s travel direction, but also change its modes from forwards rotation to backwards rotation. The code was written so that if the calculated speed for a wheel becomes negative (indicating the need for reverse motion), the corresponding direction pin is toggled to reverse the wheel’s rotation. This allows the car to make sharper turns or adjust its position dynamically when the error value is significant, ensuring that it can realign itself with the black line effectively even in challenging scenarios such as sudden tight turns.

To test this, the car was set at the straight aways before the tight turns and observed how the car reacted to the tight turns. It was observed this strategy was crucial in our cars performance. However, the car was still falling off wide on the track and losing contact with the black line. It needed to turn more quickly to stay above the line, so a constant multiplier was included to the speed of the turn rotation to allow the car to quickly change its direction around the tight turns while maintaining the sensors above the black line.

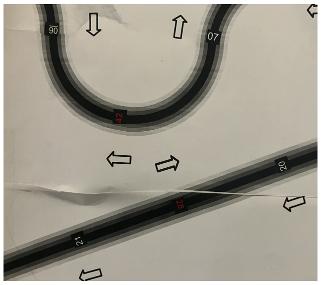
The plan consisted for our car to solely rely on PD control without the use of any encoders. This will confirm that our car can react properly on any track and will perform consistently based on feedback from the 8 Infrared (IR) sensors beneath the car.

3. **Interference**

Another issue the car faced was interference with the outer sensors detecting a separate piece of the track, which disrupted the car's performance to stay aligned with the appropriate black line (See Figure 3a. and Figure 3b). To address this, the code includes a condition that temporarily ignores readings from the outer two sensors depending on the crossPieceCounter value. During specific segments of the track (when the car is on the second or fifth crosspiece), the outer two sensors are disabled by setting their values to zero. This prevents conflicting sensor data from influencing the car's error calculation and makes sure that the car remains focused on the relevant portion of the track in the moment. By dynamically ignoring unnecessary sensor data at specific parts of the track, the car minimizes interference and maintains its intended path effectively.



[Figure 3a.]

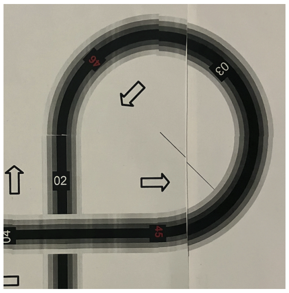


[Figure 3b.]

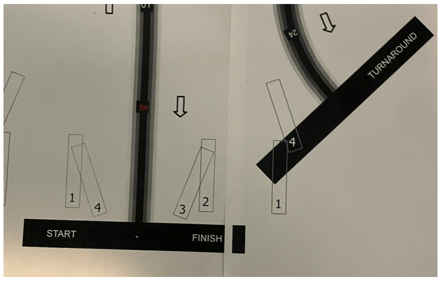
To test this strategy, the car was placed before the areas on the track where there was interference and its performance was observed.

4. Donut/Crosspieces

The challenge with the donut was simple. We needed to detect and properly react to a crosspiece in the track. Crosspieces in the track consisted of the loop intersections as well as the stop markers signaling the car to either turn around and head back to its initial position through the track again, or stop once the track was successfully completed (See Figure 4a. And Figure 4b.). Also, we had to verify that the crosspiece was a valid crosspiece reading, avoiding what is known as a “phantom crosspiece”.



[Figure 4a.]



[Figure 4b.]

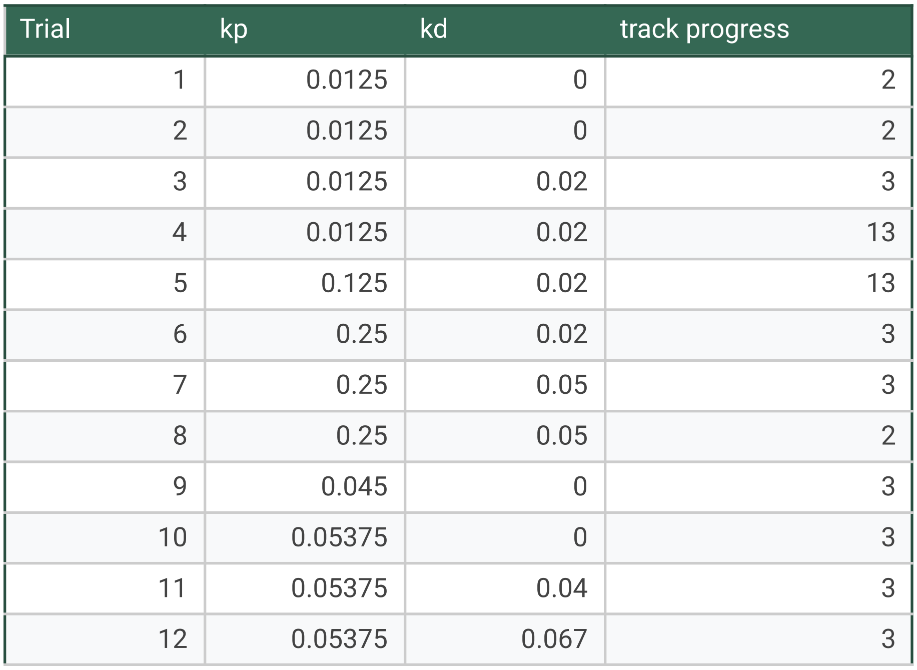
To detect a valid crosspiece, the code continuously monitors the combined values from all 8 sensors. If the sum of the sensor values exceeds a threshold (indicating that the car is completely over a black area), it checks the current state of the crossPieceCounter. Depending on its value, the car either counts the crosspiece as valid, executes specific actions, or prepares for the next maneuver. For example, if the crossPieceCounter is 1, 2, 4, or 5, indicating the crosspieces of the donut turn of the track, the car increments another counter (blackCounter) each time the condition is met. Once the blackCounter exceeds 4, the car acknowledges a valid crosspiece, increments the crossPieceCounter, and briefly moves forward. The code delays the loop for a very short time while allowing the car to keep moving forward. Through this strategy, the car was able to successfully travel through the crosspieces of the donut turns without unintentional reactions in the error values. For the third crosspiece, the car executes a 180-degree turn by reversing the left and right wheels with a delay until 180-degree rotation is complete (tested by trial and error). On the sixth crosspiece, the car stops completely by turning the motor modes to sleep, ensuring that it has reached its final position. These steps enable the car to navigate the track while correctly identifying and responding to crosspieces.

Conducting Tests

To track the progression of our car, we used the numbered sections along the track, with “0” indicating the start and “24” indicating the end, to monitor the car’s success rate along specific parts of the track and document its performance. During each trial, we recorded these numbers, as well as the Kp, Kd, and battery voltage, in our logbook to identify sections where the car struggled and to track the adjustments made after each trial. Initially, we started every trial from the beginning of the track, but after encountering challenges in certain sections, we shifted our focus. We placed the car at other points along the track to test specific sections, such as straight paths, before the donuts, or the 180-degree turn, to identify recurring problem areas and determine if there were common patterns in the car’s performance.

This segmented testing approach allowed us to analyze how the car performed under different conditions and identify trends. For each trial, we carefully documented the car’s behavior, including any observed difficulties, such as instability or deviations from path. We kept tabs on changes to parameters like Kp and Kd. By systematically isolating variables and testing them across multiple sections of the track, we were able to refine our hypotheses and improve performance. Ultimately, this approach helped us evaluate the car’s behavior on both challenging sections and transitions, ensuring an understanding of its performance.

As discussed in our planning and in our appendix, Kp and Kd data was observed and progress was recorded.

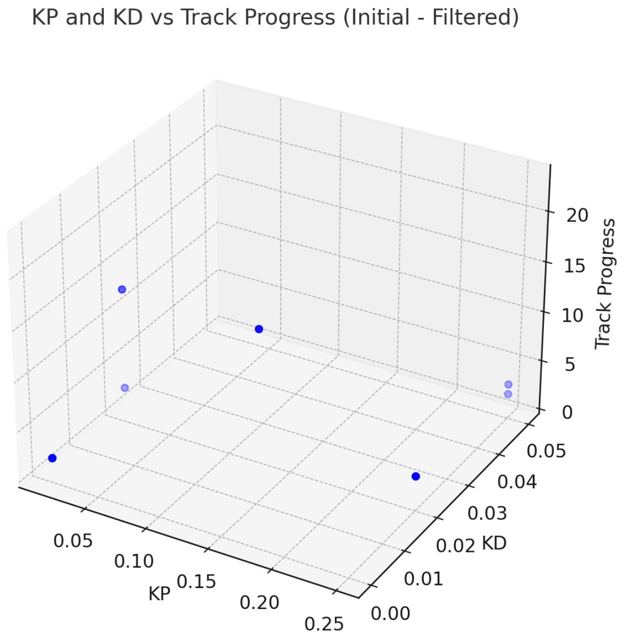


Trial	kp	kd	track progress
1	0.0125	0	2
2	0.0125	0	2
3	0.0125	0.02	3
4	0.0125	0.02	13
5	0.125	0.02	13
6	0.25	0.02	3
7	0.25	0.05	3
8	0.25	0.05	2
9	0.045	0	3
10	0.05375	0	3
11	0.05375	0.04	3
12	0.05375	0.067	3

13	0.05375	0.07	3
14	0.027	0.05	7
15	0.0275	0.05	7
16	0.0375	0.05	7
17	0.025	0.05	13
18	0.025	0.015	13
19	0.025	0.05	13
20	0.0125	0.07	24
21	0.0125	0.25	24
22	0.0125	0.19	24
23	0.0125	0.01	24
24	0.125	0.01	24

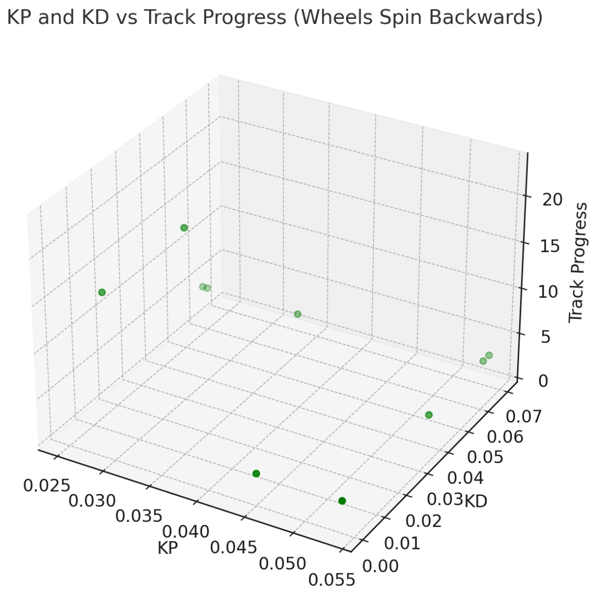
Analyze Data

Below, Figure 5a shows the trials of our Kd and Kp values plotted along the bottom axis while the track progress plotted on the vertical. This set of data was collected in our early stages after creating the fusion code.



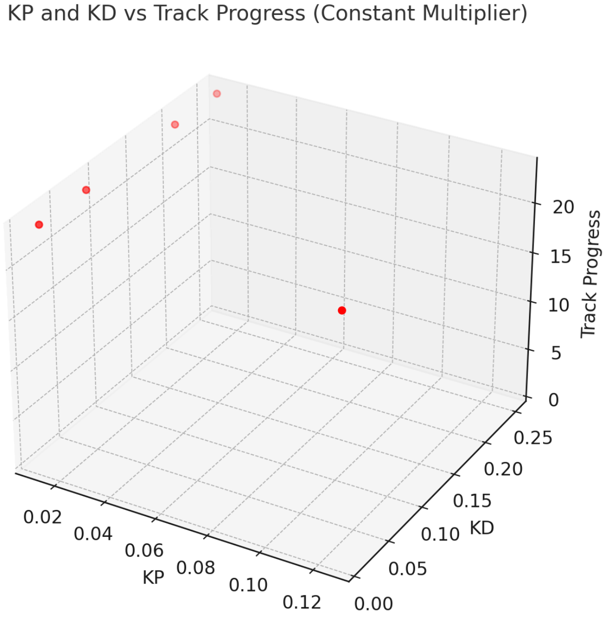
[Figure 5a.] KP and KD vs Track Progress (Initial) 3D Scatter Plot

Below, Figure 5b. shows the trials of our Kd and Kp values plotted along the bottom axis while the track progress plotted on the vertical. This set of data was collected after allowing the direction of wheel rotation to change as discussed in the planning section “Tight Turns” and “Interference”.



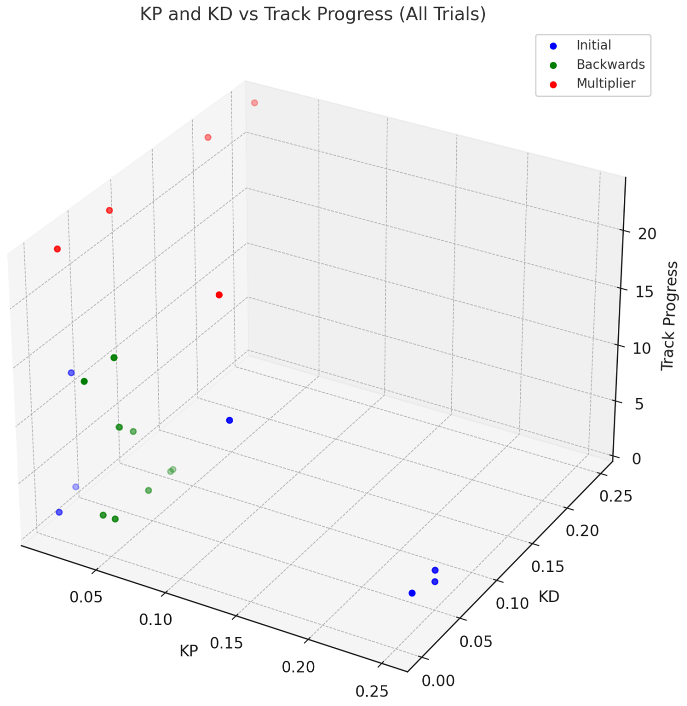
[Figure 5b.] KP and KD vs Track Progress (Allow Wheels to Spin Backwards) 3D Scatter Plot

Below, Figure 5c. shows the trials of our Kd and Kp values plotted along the bottom axis while the track progress plotted on the vertical. This set of data was collected after adding the constant multiplier to our magnitude of wheel rotation as discussed in our planning section above “Tight Turns” and “Donut/Crosspieces”.



[Figure 5c.] KP and KD vs Track Progress (Add Constant Multiplier) 3D Scatter Plot

Below, Figure 5d. shows the trials of our Kd and Kp values plotted along the bottom axis while the track progress plotted on the vertical. This set of data shows all pieces of data collected during this project.



[Figure 5d.] KP and KD vs Track Progress (All Trials)

Interpretation of Analysis

The two main components of the car's success were the decision strategy implemented in our code and the fine-tuning of the Kp and Kd values to achieve optimal reaction along the track.

Increasing the Kp value allows the car to respond more aggressively to deviations from the black line, as it directly scales the correction based on the size of the error. This can help the car make quicker adjustments, especially on sharp turns or sudden deviations, improving its ability to stay on track. However, if the Kp value is set too high, the car may overcorrect, leading to oscillations or instability as it struggles to settle above the black line.

On the other hand, the Kd value helps smooth out these oscillations by considering the rate of change of the error. A higher Kd value makes the car less likely to overshoot by damping rapid corrections, making for a smoother and more stable alignment with the track. On the other hand, if the Kd value is too high, it can cause the car to react slowly to sudden changes, making it difficult to navigate tight curves and likely to stray off the track.

By carefully adjusting these values, we balanced responsiveness and stability, allowing the car to react properly to all situations on the track.

Initially, we set arbitrary Kp and Kd values based on recommendations from the course materials and LAs. However, the car struggled as it sometimes interfered with other parts of the track during turns. To avoid interference with other pieces of the track, outer sensors were ignored at specific sections of the track to decrease the risk of false detections from nearby lines during turns. Despite these changes, the primary issue remained the car's inability to follow the loop at point "3" in the track, the donut.

To improve this, we experimented with various Kp and Kd values, aiming to find a balance that could enhance the car's turning rate. Adjusting these values directly influenced the car's responsiveness to deviations from the black line. Higher Kp values increased the immediate correction force, while Kd provided stability by reacting to rapid changes in error. As shown in Figure 5a., the car still struggled to complete sharp turns effectively. Graph Figure 5a. shows that despite the changes in Kp and Kd values, the car was not successfully making it through the first loop/donut of the track (mark "3" on the vertical). The car would react with a sharp turn and then lose contact with the line, straying away from the track. Note that the outlier in this graph is a data point taken testing from a point beyond the loop after mark "3" to test the car's ability to follow a straight line.

Recognizing the limitations of this approach, we implemented a new strategy: we enabled the direction of the wheels to rotate backwards in order to make a sharper turn. This allowed for sharper turns by increasing the car's rotational agility to stay above the black line and not deviate from the track. We also included a crosspiece detection for the car to properly ignore significant directional changes when encountering the crosspieces of the donut, This allowed the car to smoothly complete the donut turns. The results, as shown in Figure 5b., demonstrated significant improvement. Shown on the vertical, the car's track progress increased and made it through the donut turn "3". At this point, it is also shown that the car consistently plateaued at the half-circle section of the track at "13."

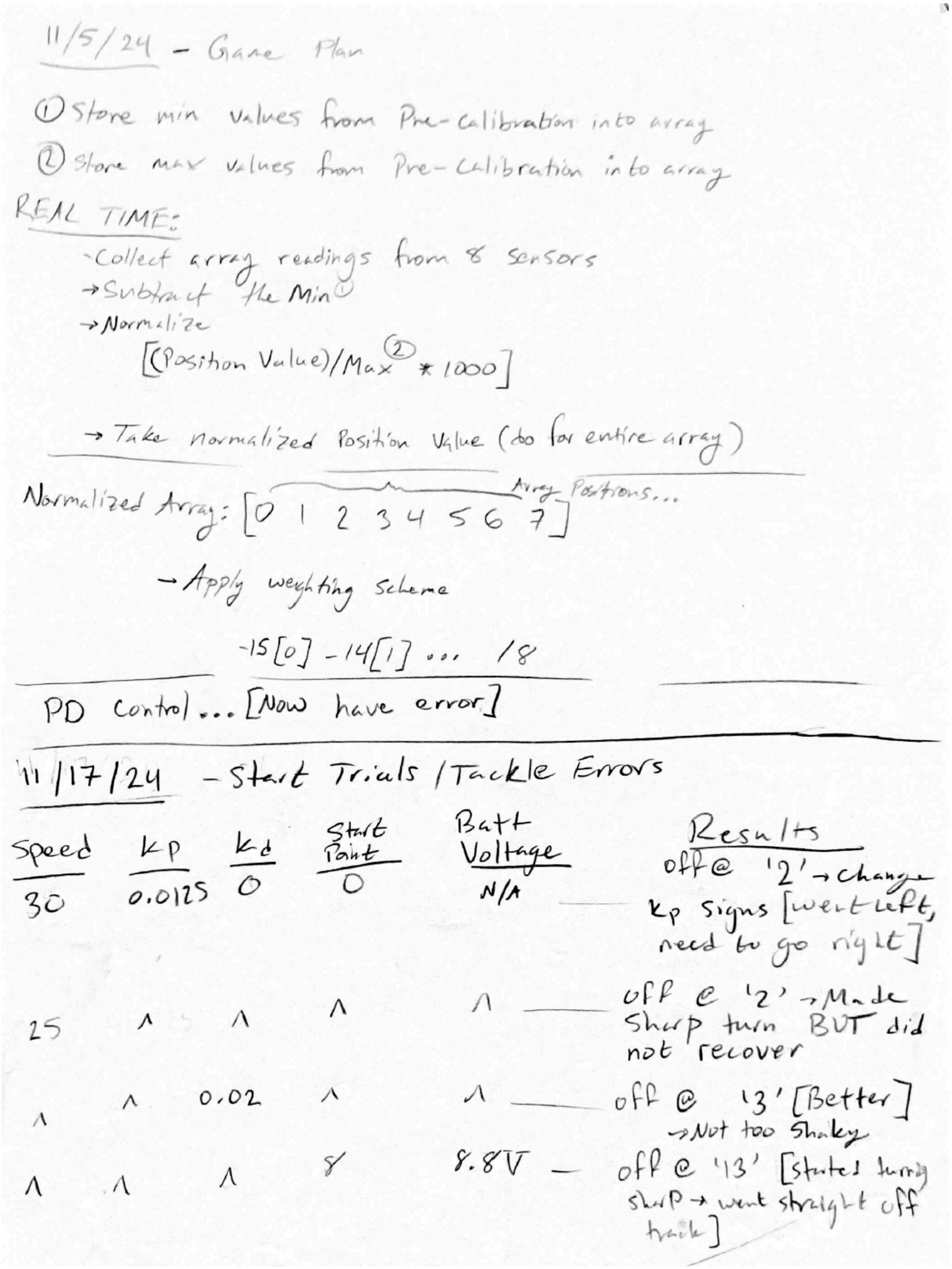
To overcome this, we introduced a constant multiplier to the added Kp and Kd values that change the speed and direction of the wheels to dynamically increase the wheel speed in areas requiring tighter turns, such as the loop at "3" and the half-circle at "13." This adjustment ensured the car maintained alignment with the black line throughout sharp turns by increasing the magnitude of rotation when the car needed to consistently and smoothly make tighter turns. As shown in Figure 5c., this strategy was the key to solving the turning challenges, ultimately allowing the car to consistently complete the track successfully. It is shown in the graph that the car was able to make it to the final mark (mark “24”) after making these corrections and fine tuning the Kp and Kd values. Once the car was able to make it to the end of the track, all we had to do was encode a 180-degree donut and the car was able to successfully return to its initial position through the track.

Discussion/Conclusion

In conclusion, engineering our car to follow the black line using sensor feedback and PD controls involved a lot of systematic problem-solving and iterative testing to accomplish. Specifically, through calibrating sensor data and dynamically adjusting wheel speeds, the car demonstrated a robust ability to follow the black line while responding to deviations. However, the car struggled with immediate corrections and interference when taking sharp turns on the track. To address these challenges, we implemented reverse wheel rotation, allowing for tighter turns and better alignment with the track. We also disabled the outer 2 sensors at specific points in the track. While this improved performance in certain areas, limitations persisted, particularly at complex turns such as the donut loop at "3" and the half-circle at "13." The introduction of a constant multiplier to dynamically enhance wheel speed in these areas proved to be the final key, enabling precise maneuvering and consistent track alignment.

Appendix

1. Log



11/5/24 - Game Plan

① Store min values from Pre-Calibration into array

② Store max values from Pre-Calibration into array

REAL TIME:

- Collect array readings from 8 sensors
→ Subtract the Min①
→ Normalize

$[(\text{Position Value})/\text{Max}^{②} * 1000]$

→ Take normalized Position Value (do for entire array)

Normalized Array: [0 1 2 3 4 5 6 7] Array Positions...

→ Apply weighting scheme

-15[0] -14[1] ... 18

PD control... [Now have error]

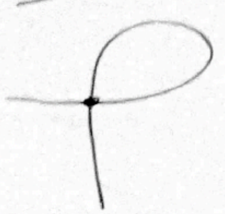
11/17/24 - Start Trials / Tackle Errors

Speed	kp	kd	Start Point	Batt Voltage	Results
30	0.0125	0	0	N/A	off@ '2' → change kp signs [went left, need to go right]
25	^	^	^	^	off @ '2' → Made Sharp turn BUT did not recover
^	^	0.02	^	^	off @ '3' [Better] → Not too Shaky
^	^	^	8	8.8V	off @ '13' [started turning sharp → went straight off track]

11/23/24 - Tackle the Donut

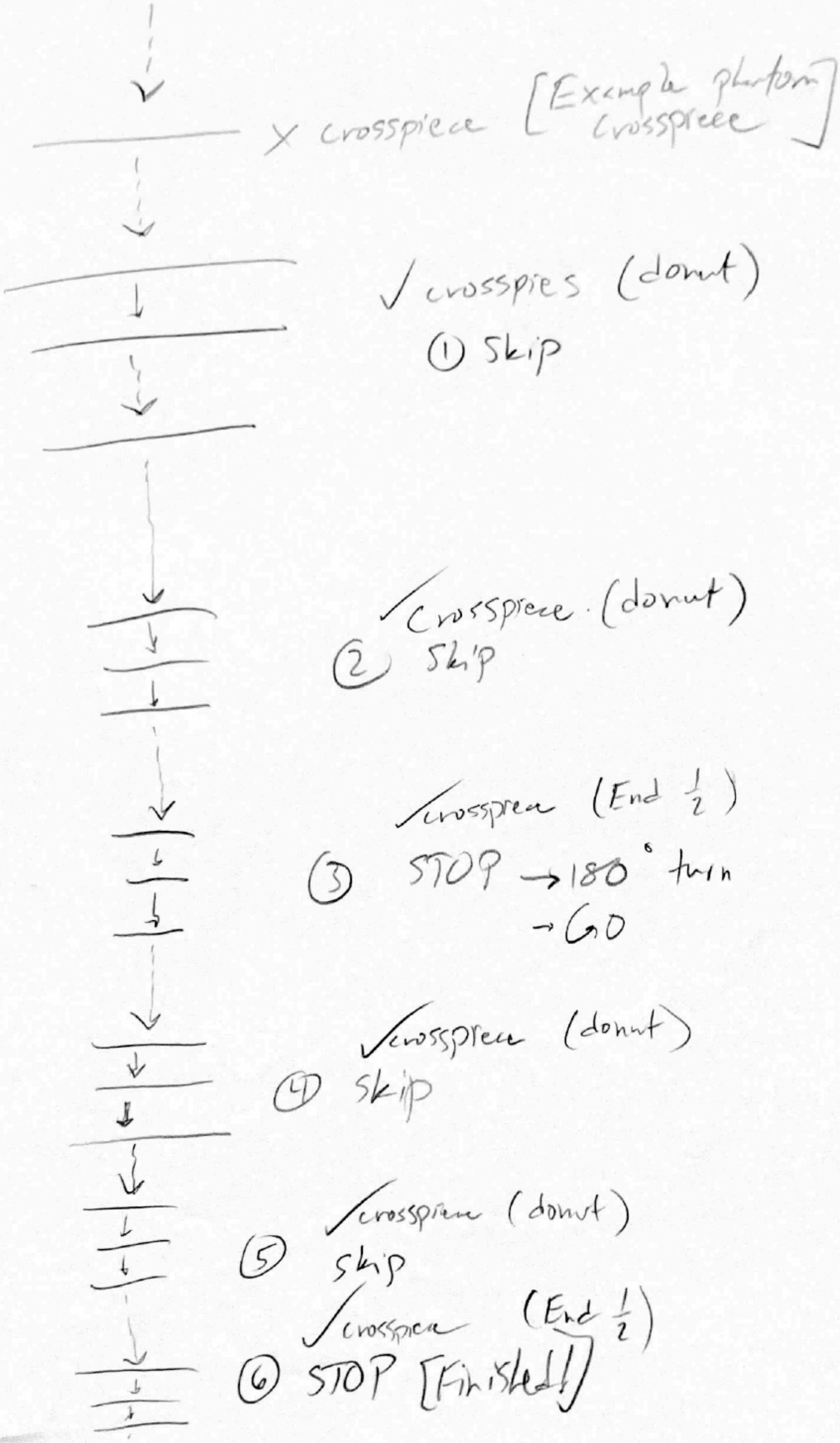
Speed	Kp	Kd	Start Point	Batt Voltage	Results
25	0.0125	^	0	8.9V	- "off @ '3'
15	2*0.0125	^	0	^	— "off @ '3'
25	2*0.0125	0.05	8	8.7V	— "off @ 24[End] [Need it to turn faster @ sharp turns: 2- & 13-]
25	^	^	0	^	— "off @ '2' * Getting [Confused by cross-piece → ADD phantom crosspiece!]
^	^	^	^	^	— "off @ '3' [Included phantom cross-piece]

Sketch:



- loop counter
  ↳ if ≠ All black × 3 loops
    → ignore
  ↳ If == All black × 3 loops
  → = REAL crosspiece
    ↳ Add to crosspiece counter...
    → keep going... (go straight for a delay? → to get over the crosspiece smoothly...)

* Went through turn 6-7 & Outer sensors read other part of track → set 'if' condition for if outer && inner sensors read black → ignore outer sensors / → set to 0... [outer 2]


× crosspiece [Example platform crosspiece]
✓ crosspies (donut)
① Skip
✓ Crosspiece (donut)
② Skip
✓ crosspiece (End ½)
③ STOP → 180° turn
→ GO
✓ crosspiece (donut)
④ skip
✓ crosspiece (donut)
⑤ skip
✓ crosspiece (End ½)
⑥ STOP [Finished!]

11/23/24

Note: 'If' [3 & 4 see black / (&) 0&8 see black (&) (1-2 (&) 5-7) don't see black]

→ Ignore kp / kd action

11/25/24 – Last Day Before Race Day → (1) Tackle donut & Tight turn, (2) 180 turn around delay, (3) Stop @ last crosspiece for finish

Speed	kp	kd	Start Point	Batt Voltage	Results
35	0.045	0	0	9.1V	'off @'3'
^	0.05375	^	^	^	off @'3'
^	^	0.04	^	^	off@'13'
^	^	0.067	^	^	off @ '3'
^	^	0.07	^	^	off @ '3'
^	0.027	0.055	^	^	off @ '6'-'7'
50	0.0275	^	^	^	off @ '6'-'7'
45	0.0375	^	^	^	off @ '6'-'7'
35	0.025	^	^	^	off@ '13'
^	^	0.015	^	^	off@ '13'
^	^	0.05	^	^	off@'13'

11/25/24

Speed	Kp	Kd	Start Point	Batt Voltage	Results
45	0.04	0.07	0	9.1V	off @ 13'
^	^	0.25	^	^	Made through end [super shaky] [Inconsistent
45	^	0.19	^	^	Better! [but Shaky]
^	^	0.01	^	^	Good
60	^	^	^	^	Faster! [A tiny bit shaky still but completes track!]

35 seconds Finish!

Notes:

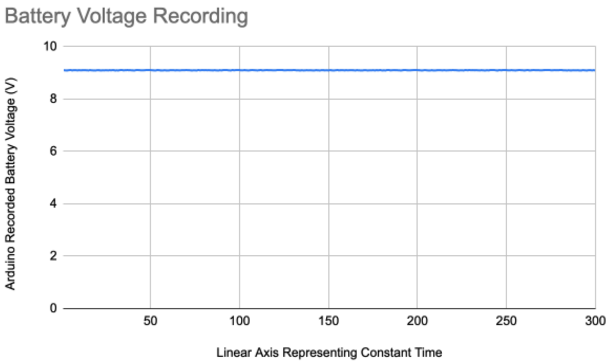
① Allowed the wheels to turn backwards
→ Allows for greater rotation of the car
→ Should stay on track [If (kd + kp) Value gets too (-) ...
"wants to turn more"

② Made a constant multiplier to increase the speed of the wheels when wants to turn quicker..

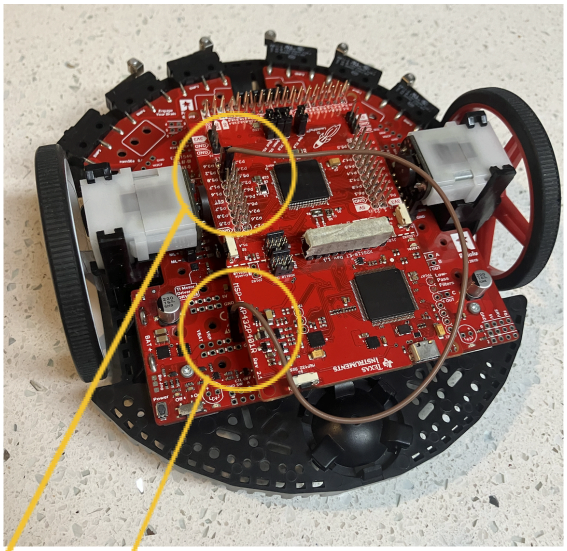
- This will allow the car to stay on track during cycle readings & stay above the black line during sharp turns...

2. Measuring Battery Voltage

A variable that was valuable to keep track of throughout the trials was the voltage of the car. A wire was soldered from the voltage output of the battery to an analog input pin of the Arduino, and the voltage was measured to ensure consistent levels throughout the trials and on race day. A low voltage battery can significantly impact the car's performance by reducing the motors' speed and torque, causing slower or less responsive movement and potentially leading to difficulty staying aligned with the track. Figure 6a. shows the constant voltage being measured as a function of time. Figure 6b. shows the analog input and battery voltage output being soldered together on the car.



[Figure 6a.]



[Figure 6b.]

Analog Input

“V5W” Battery
Voltage Output