Rochester Institute of Technology's NXP Car Cup (May 2021)

Christopher Larson & Jason Au
Computer Engineering Students
Rochester Institute of Technology
Rochester, New York, USA
cel8473@g.rit.edu & jxa2009@g.rit.edu

***Abstract*—Staying in the lab till past midnight the day before the race, some competitors are still there and probably will be for the next few hours, others have resigned themselves to their fate. Waking up, and arriving to see at least one team never left, staying in the lab overnight to get maximal testing time. The energy in the air is that of nervous excitement as more and more teams arrive. The test track gets built before our very eyes, and all the teams are chatting about whether their cars will do well on this previously unknown design. Before you know it, it's time for practice runs. Small tweaks here, expletives as cars go off the track there, it is all part of the experience. Two time blocks of practice runs and it is time for the pre-race car checks. Teams lineup their cars on the table and it looks like one team decided to put duck sauce on their dowel as a pseudo-flag. Everyone passes inspection, and now it is down to just three thirty-second tries. It's time to race!**

I. Introduction

The challenge of designing, building, and racing model cars around a track against other student teams to see who's car is the fastest is a challenge that many Computer Engineering students at Rochester Institute of Technology and many other universities face. In this paper, the authors explore the collaborative, and competitive creation of a smart, autonomous car and the development of software and hardware of motor control to propel and steer the model cars. The concepts are not just limited to the micro-controller, and the sensors, as the basic understanding of motors, drifting, turning, and drag were all key components of success for any team.

The thrill of competition was a very real incentive for the teams that participated in the NXP Cup. Teams, unlike previous years, were all from the Computer Engineering program at Rochester Institute of Technology. The teams came together at the Rochester Institute of Technology's Imagine RIT event to participate in the competition. Of the twelve teams that competed, only eight teams made it around the track, with only five of those teams achieving under the 30 second mark, and the difference between second and fifth was a mere three seconds. This shows just how close the race was for the top spots.

Fig. 1. NXP Cup Cars lined up on a table in Erdle Commons at RIT. The team's car is the closest car on the right.

Fig. 1 shows all of the cars lined up on the table after inspection, ready to race. Once the cars were put onto the table, the teams could no longer edit the code of the car, or have the Bluetooth module attached to change values. The only way to change speeds on the cars was to have preset modes and toggle through the modes via one of the switches on the micro-controller.

The program for driving the autonomous car was very different between teams, even though the end goal was the same. All of the programs at its base used a proportional-integral-derivative (PID) controller. This was because PID was one of the check points that needed to be met along the way, but the differences in PID values, steering logic, acceleration logic, and camera data interpretation ensured that no two cars were the same. For example, there were teams that used quadratic turning logic, while others used a linear approach. Some teams smoothed their camera data using a 5 point average while others used the raw camera data. The

physical car brought many differences too, some teams wiped their wheels down between runs with isopropyl alcohol, some cameras were placed high on the dowel and others closer to the ground, the camera angle was very subjective, and one team decided to add headlights to their car. These differences showed that there was not one singular way to be successful in the competition and it was down to each team's ability to adapt and overcome the environment and limitations that were present.

While testing the car, many problems arose that the team needed to overcome. To ensure that the car stayed on the track but was still competitive, an optimal duty cycle was needed. This was found by repeatedly testing on different practice courses to ensure that the car would not go too fast and slide off the track or miss an intersection. Next, to help with staying on the track, both differential steering, and a quadratic turn speed scaling variable was implemented. The differential steering stopped or slowed the inside wheel during the turn to allow for tighter turning. The turn speed scaling variable slowed the speed of the car down depending on how hard the turn was.

This paper goes on to discuss principles and practices that were learned while building an autonomous car. This includes concepts like proportional-integral-derivative control systems, pulse width modulation for motors, and racing line theory. The paper will also cover lessons learned along the way, discussions about easily avoidable mistakes if the team were to race another car in the future, and observations made throughout the hours that were spent in the Internet of Things (IOT) Security Lab.

II. Background

The NXP car was the culmination of many technical concepts and ideas that together create a fast race time. A discussion of what other teams in the past, basic strategies for going fast, what a PID control system is and how it works, and the interfacing of a flex timer will provide background to the design of the team's car.

A. Comparing to other methods

The strategy that was implemented by the team was unique, but there can also be a comparison drawn between what the team did and what other teams did. This is an important aspect of engineering, analyzing what other engineers were doing and understanding why their approach worked for them. There were countless differences between methodologies of the teams but a few key ones are as follows. Smoothing of the camera trace, rather than analyzing the raw camera data as employed by a majority of the teams. This was used to decrease the amount of noise. The team did not employ this because we found that our camera output was not noisy enough to warrant the smoothing. Another method that was employed by other teams was to take the derivative of the smooth camera data to find the edges of track, where the left-most side of the track was represented by a peak and the right-most side of the track was represented by a trough. This was not employed by any teams this year but in years prior it was used. The team found it best to use a static threshold and scan the array of camera data until a value that was above the threshold was found and used as the edges, this was done both left-to-right and right-to-left to find the respective edge.

B. Basic strategy for going fast

The general strategy, employed by nearly all teams for going fast was to drive fast on the straightaways and slow down around turns. This was achieved by increasing the duty cycle of the rear motors on the straights, and decreasing, in some cases reversing, the motors before the turns. Once the camera detected a turn, the motor on the inside of the turn was lowered significantly compared to the outside motor which was slowed slightly. A turn is detected when the camera sees a change in the edges of the track, and the middle starts shifting one way or the other. This was the basic strategy behind differential steering.

C. PID control systems

A proportional-integral-derivative (PID) controller is a control loop mechanism employing feedback for systems and applications that require continuous modulated control. A PID controller continuously calculates an error value as the difference between a desired set point and a measured process variable, then applies a correction based on proportional, integral, and derivative terms.

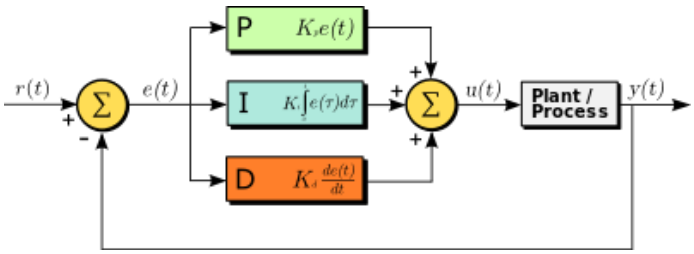

Fig. 2. A block diagram of a PID controller in a feedback loop. r(t) is the desired value, and y(t) is the measured process value.

Fig. 2 shows the block diagram that represents the PID control equation. In this model the **P** term is proportional to the current value of the error, the **I** term accounts for past values of the error and integrates them over time to produce a value, and the **D** term is a best estimate of the future trend of the error, based on its rate of change. [1] The effects of the PID control system allowed for the car to easily find the center of the track, a great advancement from the pseudo bang-bang control that was initially implemented.

D. Flex Timers

The Flex Timer Module (FTM) is a very simple timer built upon the one used in the NXP's 8 bit micro-controllers. It can be used for input capture and output compare as well as generating Pulse Width Modulated (PWM) signals. This timer is ideal for motor control and timing associated with external inputs and outputs. [2] The FTM modules were used

to generate PWM signals for the servo motor that steered the car and the DC motors that controlled the propulsion of the car. This was done by modifying the duty cycle of the module to turn the vehicle left or right, or to speed up/slow down.

III. Proposed Method

A. Camera Output and Interpretation

A line-scan camera was used to capture the track in front of the car. A line-scan camera is a sensor that consists of 128 photo-diodes arranged in a linear array. Light energy impinging on a photo-diode generates photo-current, which is integrated by the active integration circuitry associated with that pixel. During the integration period, a sampling capacitor connects to the output of the integrator through an analog switch. The amount of charge accumulated at each pixel is directly proportional to the light intensity and the integration time.[3] The integration time for the camera was set to a value that was low enough to allow for quicker response time but high enough to acquire precise data from the camera, this was very subjective and differed between many of the teams. The pixel values varied from 0 to over 100,000 depending on the reflection of the light on the surface in front of the car. To visualize the camera output, a Matlab script was used to plot the graph of the data as shown in Fig. 3.

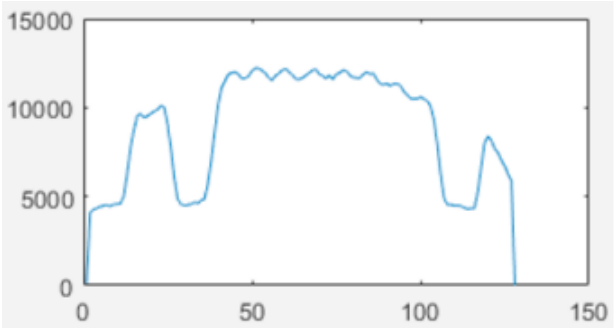

Fig. 3. Plot of the output data of the camera and graphed by a Matlab script[4]

In Fig. 3, the middle portion between the two troughs was the track, and the troughs were the black tape that runs along the edges of the track.

B. Centering the Car

To center the car, the edges were found by iterating over the array of integer values that represented the pixel values. The array was iterated over twice, once starting from the beginning of the array to find the left most edge and a second time from the end of the array to find the right most edge. The value was considered the edge if it was the first value above the static threshold that was set to differentiate the track and the carpet. The average of the two edges was taken to determine the position of the car on the track and that value was then inputted into the PID control system to figure out how much the car had to turn to return to the center or desired point on the track. The desired point on the track was calculated by printing the average of the two values while the car was held in the middle of the track and used as a constant for the PID control system.

C. Turning

When the camera detected a turn, the turn amount was calculated using the PID control system. The team found that the turning amount on sharper turns was not large enough to stop the car from driving off the track so a turn boosting variable was added to increase the turning amount. This allowed for sharper turning and lowered the chance of the car to slide off the track, this variable could not be too high though as overturning caused a lot of problems right before intersections. After the ”boost” was applied, the turn amount was capped between -100 and 100 to ensure that the values were within the necessary bounds. The turn amount was then scaled to be within 0 and 100 because of how the team used the turn amount to set the duty cycle of the servo motors. The duty cycle of the front wheel servos was calculated by finding the duty cycle for the left most turning and right most turning positions that did not grind the servo gears or brush against the chassis. An equation was made so that when a 0 was inputted, the output would be the left most position and 100 was the right most position as shown in (1).

$$\begin{aligned} duty\_cycle = & turn\_amount \\ & * PWM\_MULT\_FACTOR \\ & + PWM\_CONST \end{aligned} \tag{1}$$

PWM_MULT_FACTOR for the team’s car was 0.034, and the PWM_CONST for the team’s car was 5.2, which gave a duty cycle range of 5.2-8.6.

D. Variable Speed

During the straights the team wanted the car to go as fast as possible, while still being in control to take any turns that were on the track. The car was driven at the straight speed when the turn amount was between a duty cycle of 40 to 60, which was when the car was going straight or close enough to straight that it was able to speed up. The duty cycle of the rear motors during the straights was a constant that was found through trial and error. The value was increased until the car could not consistently get around the track. The duty cycle of the rear motors during the turn was calculated using a quadratic equation as shown in (2). This was chosen because the sharper the turn, the slower the team wanted the car to go. The turn_speed variable was a constant that was found to by subtracting a constant from the straight speed constant, similarly found using trial and error.

$$speed = turn\_speed * (1 - 0.04 * (turn\_amount - 50)^2/100) \tag{2}$$

The speed value was then given to both the rear motors, except that the inside motor’s duty cycle was multiplied by a differential factor to greatly reduce the speed of that tire.

E. Hardware of the Car

The FRDM-K64F micro-controller was used to control the standard NXP cup car kit that was used. This included a Line Scan Camera, the chassis with motor kit, and the MC33886 Motor Driver Board. The motor kit was comprised of two DC motors, and one servo motor. The motor shield allowed for the motors to be connected to the micro-controller via wires. The motor shield was equipped with an H-bridge to control the direction and speeds of the DC motors. The wiring connections for the car is shown in Table I.

TABLE I
WIRING CONNECTIONS FOR THE MICRO-CONTROLLER, CAMERA, AND MOTOR-SHIELD.

Component	K64F Pin	Motor Shield Pin
Camera Signal	PTC8	SIG
Camera Clock	PTB9	CLK
Camera Serial Input	PTB23	SI
Camera Analog Output	ADC_DP0	AO
Left DC Motor Forward	PTC2	M1
Left DC Motor Backward	PTC3	M2
Right DC Motor Forward	PTC1	M4
Right DC Motor Backward	PTC0	M3
Servo Motor	PTC8	J10-2
Battery Power	3.3V	VCC
Battery Ground	GND	GND

Table I shows all of the components that needed to be connected to the motor shield by wires using the screw terminals or by a pin socket connector. The micro-controller and motor shield were mounted on a 3D printed mount above the motors in the back. The camera was mounted on a dowel in the front of the car above the servo, on the dowel a hinge was used to hold the camera and allowed for angle changes.

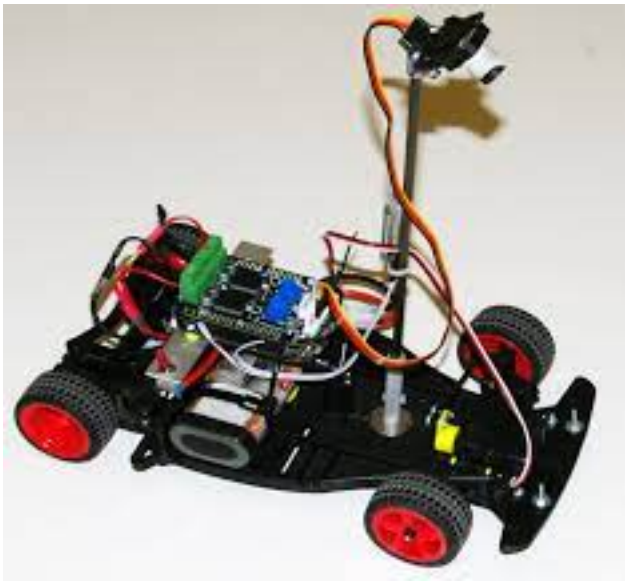

Fig. 4. Example of an NXP Car to show where the parts are on the car

The car shown in Fig. 4 is not the exact same as the team's car that was driven in the race, but the two are very similar.

F. Car Parts

The hardware bill of materials included the microcontroller, the motor shield, the line-scan camera, the chassis, the motor kit, and the batteries.

TABLE II
HARDWARE BILL OF MATERIALS

Component	Price	Description and HyperLink
Micro-Controller	$53.85	FRDM-K64F
Motor-Shield	$28.99	MC33886
Camera	$55.73	CJMCU-1401 TSL1401CL Module
Chassis	$98.75	ROB0170
Motor Kit	$43.75	KIT0166
Two Batteries	$53.99	7.2V Battery Pack 3800 mAh

Table II lists all of the hardware components for car, the current price of the components according the links, and the part number or name of the component with a hyperlink for where to purchase the components.

IV. RESULTS

A. Tuning

The PID had multiple variables that required tuning to maximize the efficiency of turning. Specifically, the Kp, Ki, and Kd variables were responsible for controlling the micro-controller using closed-loop feedback. The motors were driven at a baseline of a 50% duty cycle. At the given speed, the value of Kp was increased gradually until the car would not excessively oscillate when going straight. Ki served the purpose of minimizing steady-state error that came along with increasing the value of Kp. Changing the value of Kd controlled the process quicker. The final values of Kp, Ki, and Kd were 12.75, 0.0, and -1.0, respectively. Typically, Kd would usually be a non-negative value. However, the implementation for PID was slightly incorrect. The equation used to calculate the turn amount using a PID control system is shown below in (3).

$$\begin{aligned} turnAmt = {} & turnOldAmt \\ & - Kp * (error - errorOld1) \\ & - Ki * (error + errorOld1)/2 \\ & - Kd * (error - 2 * errorOld1 + errorOld2) \end{aligned} \tag{3}$$

Usually, the products of Kp, Ki, and Kd would be added to turnOldAmt. However, the small mistake, as shown in (3), was found too close to the race, and altering our PID implementation resulted in increased oscillation. As a result, our implementation of PID was not changed as it produced satisfactory results.

The car drove at different speeds while it was traveling straight and turning. The straight speeds were manually tuned. It was increased to a speed that the car would not slide off of the track when transitioning from its top speed to a turn. At too high of a speed, the car would have too much forward momentum when transitioning to a turn, causing the car to have more than two wheels off the track. The turn speeds were

tuned to a point where the car would not overturn. When the turn speed was too high, the car overturned and at intersections turned to the left or right instead of continuing straight.

Differential steering was implemented to complement high motor speeds on straightaways. Due to the car having much forward momentum, it was compensated for by driving the inner motor at a much lower speed compared to the outer motor. Doing so decreased the amount of forward momentum and prevented the car from breaking the rules of the race.

While differential steering proved to be very helpful in dealing with high straight speeds, further measures were taken to avoid the car from having more than two wheels off the track. If the result of the PID algorithm produce an output to set the servo to a duty cycle of less than 35% or more than 65%, the turn amount was scaled to turn more by a constant factor of 1.5. This "boosted" the turn amount so that the sharp turns were taken even sharper.

B. Final Results

The team's car came in 4_{th} place, tied with another team to the hundredth of a second. There were a total of twelve teams competing in the race. The car finished the track in a respectable 27.33 seconds. The slowest mode had to be used to complete the track because higher speeds caused overturning when entering intersections just after a turn was made. Overall, our results were successful by producing a car that drove faster than most of the other cars at the competition while also being able to complete the track successfully.

C. Aspects That Worked Best

Much of the initial implementation of the car algorithm worked well because of upfront time invested into implementing proper software engineering practices. Each module and aspect of the car was separated into its own file to very clearly detail what the contents of the file were. Additionally, functionality of the car was modularized in order for changes to be made with ease. Altering algorithms were clear because functions did not take on multiple roles. Define directives were used when possible to illustrate the purpose of values within calculations and functions. Tuning values became simple because they were congregated at the top of the file instead of spread throughout.

Bluetooth worked very well when it came to tuning the car. Much time was saved by not having to reflash code with minute changes. Instead of reflashing, the Bluetooth module was used to change variable values, allowing for more time to tune the car.

D. Improvements

The biggest improvement that could've been made was allocating more time tuning the PID. As noted earlier, our implementation was slightly incorrect. Therefore, there were occasionally oscillations on straightaways, causing the car to not accelerate. More time tuning the PID would result in a smoother car that would drive around the track more efficiently.

Due to the rules of the race allowing for a maximum of two tires to be off at a given time, our turns could've been optimized further by "abusing" this rule. The turn of the car was originally implemented to drive cleanly around these turns at high speeds. However, the team could've taken advantage of this rule by designing our car to cut these turns and turn sooner. Resulting in rounding the turn sooner while still not breaking any rules of the race that would result in disqualification.

Various physical variables had an impact on the performance of the car that should've been reduced in order to minimize the factors affecting the car's performance. For example, the cleanliness of the car tires had a large impact on the speed and performance of the car. Tires that were not cleaned enough would slide more on turns, occasionally causing the car to go off the track. Keeping the tires consistently clean when testing would've produced test results that were more accurate to different phases of testing. Additionally, having a predetermined angle for the camera would've reduced the amount of factors affecting the car's performance by one. A lower camera angle caused the car to turn later, and a greater angle turned the car sooner. Having the same camera angle would produce more consistent results between different trial runs of a track.

Another improvement that could've been made would be making a much larger track sooner. The track for the official race was much larger than the tracks that multiple phases of testing were performed on. The final track had multiple instances of a turn directly into an intersection. This negatively impacted our car because it would overturn and drive to the left or right rather than straight. To successfully complete the track, our car had to drive on the slowest turn to avoid failing on this edge case. Testing on a larger track from the beginning would have discovered this oversight sooner and allowed for our team to spend more time tuning turn angles and speeds to overcome this obstacle.

V. Conclusion

The purpose of the project was to program a car that could autonomously drive along a track as fast as possible. A successful car would be one that could place within the top 7 fastest cars out of 12 teams. It entailed multiple aspects, such as, motor control, servo control, camera data interpretation, and efficient steering methods. The project developed various engineering skills. Some of which include, effective teamwork, modularizing work, and the implementation of each team member's skills and knowledge to the final product. Common embedded skills were strengthened, such as, reading and interpreting data sheets and how to initialize modules within an embedded environment. Once combined with interfacing with peripherals and racing knowledge to efficiently race along a track, it created a very in-depth project that served to only strengthen one's skills as an engineer. A large skill learned at the end of the project was that even though one's testing may seem very thorough, edge cases may always exist that would lead to unexpected results. The implementation of fail safes such as, slower driving modes, served as a very

useful backup in the case the unexpected occurs. Finishing 4th out of 12 teams, the team composed of Chris Larson and Jason Au can be considered as successful. With more time and a larger reservoir of racing knowledge, an even faster car could've been engineered that would win the race.

REFERENCES

[1] Wikipedia contributors. "PID controller." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 1 May. 2021. Web 2021.

[2] L. Beato, "CMPE-460 Interface & Digital Electronics Laboratory Manual." Rochester Institute Of Technology, Rochester, 2021.

[3] "TSL1401CL," Jul-2011. [Online]. Available: https://www.mouser.com/catalog/specsheets/TSL1401CL.pdf. [Accessed: 2021].

[4] J. Judge, S. Prasathong, and D. Iqbal, "Learning Through Racing: Rochester Institute of Technology's Imagine RIT NXP Car Cup," 2017. [Online]. Available: http://suhailprasathong.com/docs/ieeenxpcup.pdf. [Accessed: 2021].

[5] Waveshare Wiki contributors, "RPi Motor Driver Board Schematic," https://www.waveshare.com/wiki/RPi_Motor_Driver_Board, 2017. [Online]. Available: https://www.robotshop.com/media/files/pdf2/rpi-motor-driver-board-schematic.pdf. [Accessed: 2021].

VI. INFORMATION ABOUT THE AUTHORS

Christopher Larson is currently in his fourth year of study at the Kate Gleason College of Engineering at Rochester Institute of Technology in Rochester, NY. He plans on graduating in May of 2022 with a Bachelors of Science in Computer Engineering, and with a minor in Environmental Studies. He will be working for D3 Engineering during this upcoming summer of 2021.

Jason Au is currently in his fourth year of study at the Kate Gleason College of Engineering at Rochester Institute of Technology in Rochester, NY. He plans on graduating in May of 2022 with a Bachelors of Science in Computer Engineering, and with a minor in Mathematics and Computer Science. He will be working for Anduril Industries this upcoming summer of 2021.