AUTONOMOUS RACE CAR UTILIZING NXP KL64 MICRONTROLLER

Carson Clarke-Magrab, Kenji Aikazawa and Varun Malhotra

Rochester Institute of Technology
Department of Computer Engineering

ABSTRACT

The concept of race cars is an exciting prospect in its own right, but coupled with autonomous robotics it also becomes an excellent educational engineering opportunity. The students in the Interface and Digital Electronics course at Rochester Institute of Technology participate in the NXP Cup during the school's innovation festival, Imagine RIT. The process of developing a competitive autonomous car for said competition is rife with unexpected pit-falls. However, even at its worst it maintains an endearing and valuable educational experience. The authors competed in the race as Team ADP.

Index Terms— Autonomous Car, Control Theory, KL64, Line-following, RIT

1. INTRODUCTION

In a modern-day, tech-based society, there is an emphasis on ease-of-use and ease-of-life. The average member of society is quick to accept any new technology that simplifies an everyday task. The basis of these aforementioned technologies is usually autonomy and artificial intelligence. In the case of the technologies developed by students at Rochester Institute of Technology (RIT) in the Interface and Digital Electronics (IDE) course, there is a strong emphasis on autonomy. The students apply their respective engineering skill sets in order to develop an intelligent, autonomous, line-following car.

In developing the line-following algorithm and associated technologies, it was illuminating to research what methods other students explored in their design processes. In particular, an interesting method of signal-conditioning was explored by participants in the RIT's NXP Cup in 2017. The team mentioned using a Kalman filter in combination with an accelerometer and gyroscope to determine optimal velocity [1].

The authors chose to approach the design of the car's source code in a simplistic, methodical manner by avoiding such over-complications as well as over-simplifications. Much of the methods used in the final iteration of the source code stemmed from information received in the lab and lecture section of the IDE course. Section 2 elaborates on the basic technologies and design choices used to construct the autonomous car while Sec. 3 delves much deeper into how the individual components work together. The results of the project and RIT's NXP Cup 2019 are documented in Sec. 4.

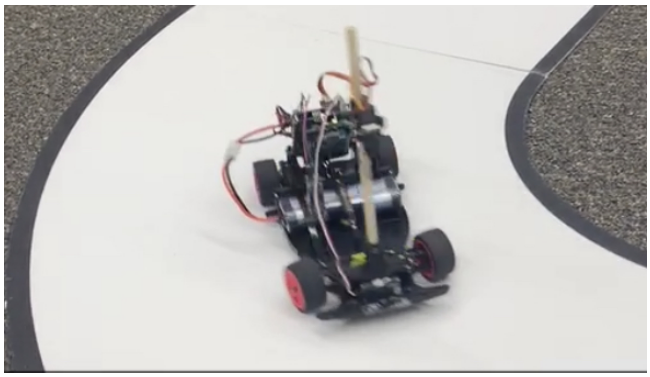

Fig. 1. The NXP Cup Car

2. BACKGROUND

The autonomous capabilities of the car stem from a variety of hardware mechanics and technical concepts. Included in these technologies are the use of variable speed, control theory and the KL64's on-board modules such as the FlexTimer (FTM) [2].

2.1. Bill of Materials

The following is a list of materials used to construct the car:

- KL64 Series Micro-controller (1)
- 3D Printed Micro-controller Mount (1)
- Line-scan Camera (1)
- Motor Shield (1)
- Servo Motor (1)
- DC Motors (2)
- Car Chassis (1)
- Battery (1)

2.2. Variable Speed and Differential Steering

To complete the track at higher velocities the car must be able to dynamically adjust its speed when between straight and

C. Clarke-Magrab, K. Aikazawa and V. Malhotra performed the work as bachelor students in the RIT Department of Computer Engineering

curved parts of the track. This is accomplished by using the corrective value received from the PID controller as detailed in section 3.4. When the value returned from the PID control is close to zero (i.e. within some small tolerance range) the car begins to increase the duty cycle sent to the DC motors. Otherwise, the car assumes it is turning and cuts power to one of the wheels and increases the duty cycle of the other in order to replicate simple differential steering.

2.3. Control Theory

The purpose of control theory is to obtain some form of data from the surrounding environment and perform corrective behavior. To intelligently center itself on the track, the car uses a error-correction feedback mechanism known as Proportional-Integral-Derivative (PID) control discussed in depth in section 3.4. The PID controller attempts to provide a turn such that the car centers itself at the center of the track [3].

2.4. Motors

Two DC motors and a single servo are utilized to provide the car with the capability to turn and propel itself forward and backward. DC motors are current-controlled motors that can be driven in one direction by applying a current from the negative to positive terminals and vice versa for the other direction [4]. Power applied to a DC motor can be controlled by feeding the motor a PWM signal where a duty cycle of 100% induces maximum drive and 0% is effectively off. In order to drive the motor in this fashion, it is often convenient to use some form of switching and voltage-control circuit.

Servo motors are motors that utilize some form of position encoding [4]. The hobby servo used in the car is attached to the front wheels such that the turning of the motor turns the wheels. The servo motor receives some PWM signal from the micro-controller that is usually 50Hz and a duty cycle that controls the positioning of the servo.

2.5. Flex Timer

Motors like the DC and servo motors used on the car require a pulse-width modulated (PWM) signal [4]. The KL64 series' on-board modules include Flex Timers (FTM) which can be used to generate PWM signals with various duty cycles [2]. The FTM modules were configured to generate the proper frequencies to drive the car's various motors.

3. PROPOSED METHODOLOGY

3.1. Control Loop

In order for the car to autonomously follow a white track with two defined black lines, it must be able to detect the location of the two lines and adjust its steering and speed accordingly. The algorithm defined in Algorithm 1 in Appendix A contains

Fig. 2. The Rochester Institute of Technology NXP Cup

pseudo-code for the main control loop the car uses to navigate the track. The algorithm can be split into four sequential components: reading the camera signal, conditioning the signal, centering the car and controlling the speed.

3.2. Line-scan Camera

The line-scan camera is mounted on a short pole at the front of the car, aimed downwards to be able to see the track. It generates an analog signal based on the light levels it detects, with higher values at brighter areas and lower values at darker areas in the cameras field of vision. Since the track being used is white with black tape at its edges, when centered on the track, the line-scan camera will produce a signal with a large number of high values near the center and a sharp drop in values on each side where the camera sees the tape.

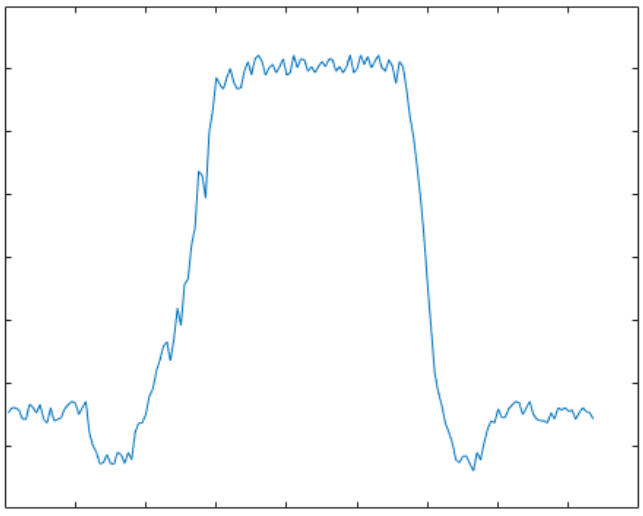

Fig. 3. Example Track Reading

3.3. Signal Conditioning

Once the signal has been read into an array of 128 unsigned 16-bit values, the signal must be conditioned such that the left and right lines can be easily detected. This is accomplished using digital filtering; the camera signal is treated as a discrete function and is convoluted with a kernel (Another discrete function that induces some form of filtering when convoluted with a function) [5]. The first kernel function used is a simple 7-point average to smooth any noise from the camera.

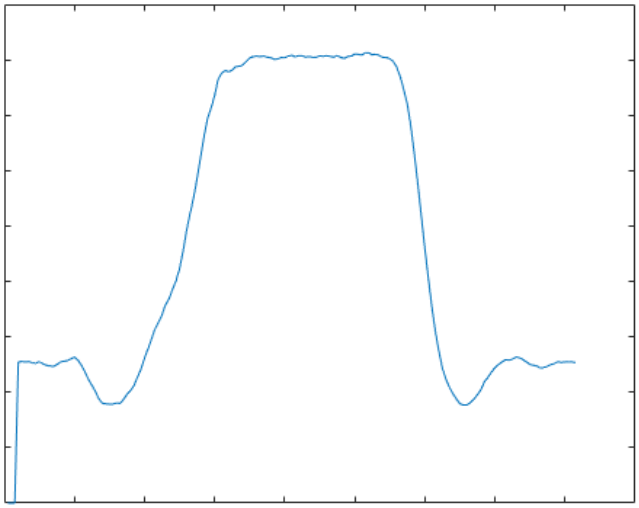

Fig. 4. Results of Smoothing Filter

Once the signal is smoothed out, the resulting function is convoluted with a derivative filter which simply results in a very basic derivative of the function. The derivative signal significantly spikes upwards and downwards where the black lines meet the white track. Finding the minimum and maximum of the derivative signal yields the location in a valid line-scan of the left and right black lines.

3.4. Centering Algorithm

In order to center itself the car must dynamically adjust its turning depending on how close it is to one of the lines. To accomplish this, Proportional Integral Derivative (PID) control theory was applied:

$$r(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \qquad (1)$$

PID control simply produces some correction value based on a given error by using three different numerical methods in tandem [3]. Taking the midpoint of the index of the maximum and minimum determined in section 3.3 and comparing it with the desired value of 64 (derived from being half the length of the camera signal) produces the error used for the PID control. The resulting correction value is added to some

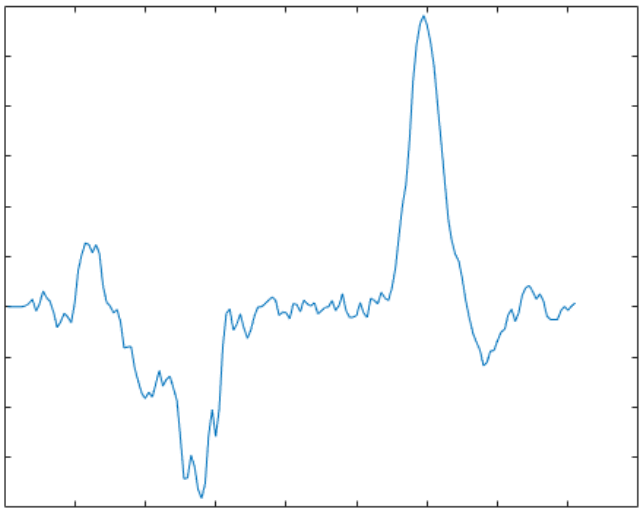

Fig. 5. Results of Derivative Filter

predetermined value for the servo that causes a straight orientation. The final value is the duty cycle to run the servo at to induce the proper corrective turn.

3.4.1. Proportional Control

The proportional control is the core of the PID control system. It creates its correction value by simply multiplying the error by some adjustable constant. This serves to essentially convert the range of error to a range of values that can be used to control the amount the servo wheels turn.

$$p(t) = K_p e(t) \qquad (2)$$

Equation (2) shows the P function where K_P is the constant "P-term" value and $e(t)$ is the error at time t. Too high of a P-term causes the car to oscillate left and right as it tries to center itself, while too low of a P-term causes the car to take too long to correct itself. This steady-state behavior can be minimized by either utilizing an ideal P-term value or coupling the proportional control with integral control.

3.4.2. Integral Control

The purpose of the integral is to limit the steady-state oscillations caused by the proportional control. The integral control term essentially integrates over the past errors to accelerate the movement towards the steady state.

$$i(t) = K_i \int_0^t e(t)dt \qquad (3)$$

3.4.3. Derivative Control

With just proportional control alone, the time it takes to reach the desired value can adversely affect performance, especially

around corners. The solution to this is the derivative control which simply causes the error function to achieve the desired value faster. This is accomplished by simply taking the instantaneous derivative.

$$d(t) = K_d \frac{d}{dt} e(t) \tag{4}$$

3.5. Implementation

Upon detecting the maximum and minimum of the filtered signal, it was found necessary to ensure that the peaks exceeded some dynamic tolerance value. Utilizing the calculated standard deviation and mean of the filtered camera signal, the peaks were compared to tolerance values to ensure detection of legitimate lines. In the case where the car failed to detect any lines, the car assumes it has reached an intersection and drives straight until it reads valid line values.

Due to the fact that the error data is discrete, the integral and derivative term are approximated using previous error values. It was found necessary to cap the corrective term created by the PID control so the wheels did not turn too far left and right and produce friction against the chassis of the car. Additionally, the authors found the integral term inconsequential for their purposes and neglected it in the error calculations.

3.6. Wiring Schematic

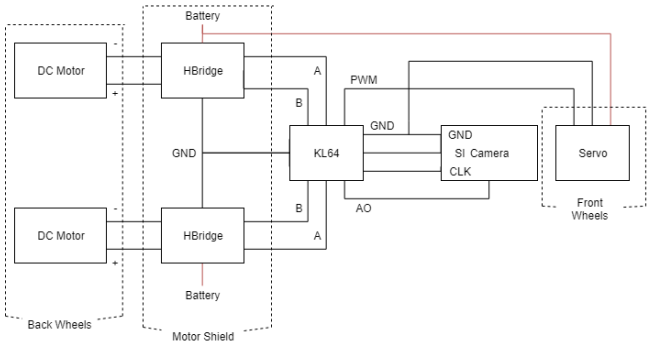

Fig. 6. Wiring Block Diagram

In order to safely interface with the various hardware components, a motor shield was provided to drive the motors and reroute the pin-out of the KL64 [6]. The motor shield attaches to the pin-out of the KL64 and provides power to the KL64 using a battery strapped into the chassis of the car. The motor shield provides a set of H-bridges, an amplifying, transistor-switching circuit, to drive the DC motors either forward or backward. While not shown in Fig. 6, the servo and line-scan camera are routed to the KL64's I/O pins through the motor shield. The servo motor is fed a PWM signal from the micro-controller and connected to a common ground and 5V output from the motor shield. The line-scan camera is fed a clock signal and an integration signal and outputs a voltage for the KL64's (Analog to Digital Converter) ADC to convert.

Table 1. Macro Settings

Macro	Safe	Danger
ACC	2	2
VEL	47	53
VEL_CAP	53	65
SLOW_DOWN	40	75

4. RESULTS

4.1. Tuning Parameters

Many aspects of the car's autonomous capabilities rely on constant values that often change based on power draw from the battery or physical damage to the car itself. As such, all constant parameters were implemented as macros in the source code. To increase the flexibility in the programming of the car's performance, the following values were implemented as macros:

- Maximum left- and right-turn
- PID control constants (K_P, K_I, K_D)
- Tolerance for turning with differential steering
- Sensitivity for track detection (K_L, K_R)
- Acceleration (ACC)
- Base velocity (VEL)
- Maximum velocity (VEL_CAP)
- Wheel speed differential steering (SLOW_DOWN)

In order to increase the probability of a successful completion of the track, two separate sets of the macros were implemented: a "safe mode" and a "danger mode" (See Table 1). Despite boasting the moniker "danger mode", the settings for this mode were conservative in nature to prevent underestimating the power supplied by the batteries.

4.2. NXP Cup

During the NXP Cup competition, RIT students were given the opportunity to compete in a race to complete a track. Teams were given three chances to complete the track and upon first completion their time was recorded. Team ADP found success in being able to completely navigate the track on the first attempt and placing 5th overall.

The car was set to "danger mode" and maintained a low acceleration and decent forward velocity. It managed to cross intersections and take turns without any erroneous behavior. The success of the unstable "danger mode" is attributed to conservative re-evaluations of the macros prior to the race.

In order to improve performance, it would be beneficial to dynamically produce values for differential steering rather than using some static turn value. This would allow the car to handle turns at higher speeds and optimize the car's maximum handled velocity. Additionally, there may exist some benefit to having specific behavior when only one of the two lines are detected.

5. CONCLUSION

RIT's Interface and Digital Electronics course and the NXP Cup provide students with a valuable learning experience in the field of embedded systems and robotics. The construction of an autonomous car touches upon many important aspects of engineering and brings insight into more advanced electro-mechanical systems. The participation in both class and the competition provides a valuable learning experience.

6. APPENDIX A - PSEUDO-CODE

$signal$:= short[128];
$last$:= 0;
err_1 := 0;
err_2 := 0;
$speed$:= BASE_SPEED;
while *true* **do**
    $signal$ = get_line_scan();
    $signal$ = condition($signal$);
    (l, r) := get_extrema($signal$);
    $mean$:= get_mean($signal$);
    std_dev := get_std_dev($signal$);
    $left\_cutoff$:= abs($mean$ + (K_L * std_dev));
    $right\_cutoff$:= abs($mean$ + (K_R * std_dev));
    if *edge_detected(l, r, left_cutoff, right_cutoff)* **then**
        (err, cor) := PID(l, r, $last$, err_1, err_2);
        SetServo(STRAIGHT + cor);
        err_2 = err_1;
        err_1 = err;
        $last$ = cor;
    else
        SetServo(STRAIGHT);
    end
    if $abs(last) \leq TOLERANCE$ **then**
        $speed$ = Accelerate($speed$, ACC, VEL_CAP);
        SetMotorSpeed($speed$);
    else
        $speed$ = BASE_SPEED;
        SetDiffSteer(SLOW_SPEED, $last$);
    end
end

Algorithm 1: Main Control Loop

[4] [6]

7. REFERENCES

[1] J. Judge, S. Prasathong, and D. Iqbal, "Learning through racing: Rochester institute of technologys imagine rit nxp car cup (may 2017)," Tech. Rep., May 2017.

[2] "Kl64 sub-family reference manual," Freescale Semiconductor, Inc., January 2014.

[3] R. Ptucha, "Topic: Control systems," in *Interface and Digital Electronics*. RIT, 2019.

[4] R. Ptucha, "Topic: Locomotion," in *Interface and Digital Electronics*. RIT, 2019.

[5] R. Ptucha, "Topic: Digital filter design," in *Interface and Digital Electronics*. RIT, 2019.

[6] "Frdm-k64f freedom module users guide," NXP Semiconductors, August 2016.