F1/100th

Jacob Tamor, Maria Boza, Leo Davies, Jessica Henson

Cornell University, Ithaca, NY

Abstract— **As self-driving cars become more prevalent in society, research and competitions surrounding autonomous racing vehicles are expanding. F1TENTH is a community in autonomous racing through scaled-down versions of F1 racing vehicles. This paper reviews a project aimed at further miniaturizing these autonomous race cars, creating an F1/100th scale model using the frame of a Hot Wheels car, an M0 Trinket microcontroller, and a time-of-flight (ToF) sensor. Our goal was to build a matchbox-sized autonomous racecar with basic AI and sensor implementation for navigation and obstacle detection. The microcontroller handles sensing, navigation, and movement, enabling the car to stop when an obstacle is detected in front of it. This project demonstrates the potential for a new autonomous racing league at the micro-robotics scale, highlighting the advancements in micro-robotic autonomous systems.**

I. Introduction

A. Motivation

F1TENTH (or F1/10th) is a community of engineers that foster the research of autonomous systems through the lens of racecars, resembling scaled-down versions of F1 racing vehicles. F1TENTH provides a collaborative platform for engineers, researchers, and students as well as brings light to new innovations of micro robots. Our objective is to learn more about anonymous racing and micro-robotics by scaling down these F1/10th racecars to an even smaller scale naming them F1/100th. With the F1/10th cars being around 40 centimeters (about 1.31 ft) in length this project will bring it to a much smaller scale of about 7.6 centimeters (about 2.99 in) in length. Our goal was to create a matchbox (around six centimeters in length) autonomous racecar with basic AI (Artificial Intelligence) and sensor implementation through a small MCU that allows for GPIO interaction, which allows for navigation of a possible course like the F1/10th racing competitions.

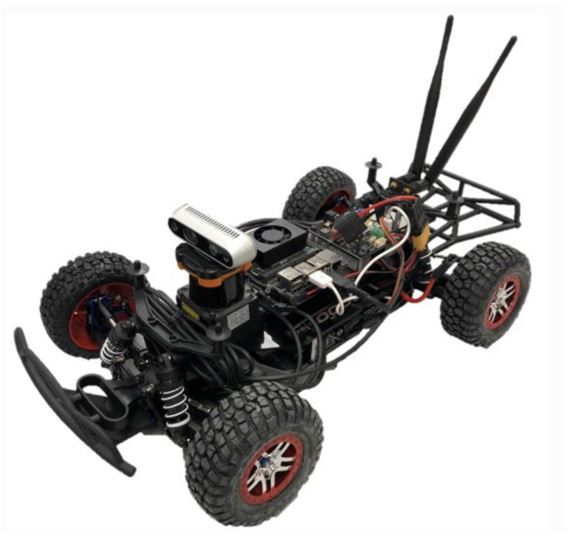

Figure 1: F1TENTH Car

B. Related Work

As described in Johannes Betz et al.[4], autonomous racing should be defined by the features of four wheels, an engine, and a clear connection of software and hardware to racing. As we design an autonomous car the size of a matchbox, we need a frame, wheels and axils, motors, sensors, and a computer small enough but powerful enough to control said motors and sensors. For this reason, we found a remote-controlled Hot Wheels car (Illustrated in Figure 2) to take apart and rebuild, allowing us to encapsulate Betz's definition.

Other literature, such as in Benjamin David Evans et al.[6], also examines programming algorithms for path planning and training, such as deep reinforcement learning (DRL). While this too may be excessive for our project, it is interesting to learn about and begin our discussion of how path planning and the general course will be developed. Our decisions pertaining to path planning had a great effect on how we incorporated our software and sensors. We had to choose a microcontroller that was compatible with sizing and programming needs, and our sensors needed to have similar standards as well.

Next, we needed to find sensors that would be compatible with a car this size. Reviewing the Weiqiang Xie et al.[5], which covers sensing within F1/10th racing, we see most of the cars are equipped with LiDAR (Light Detection and Ranging) and fiber-optic gyroscopes, which for this project would have been too large and unnecessary. Based on the collective experience in our group, we considered using infrared sensors, ultrasonic sensors, and time of flight sensors. After examining the options, we chose to use a time-of-flight sensor. This sensor was readily accessible, easy to use, lightweight and small, and allowed us to quickly identify the distance of objects in front of the car.

Figure 2: Remote-controlled Hot Wheels

C. Contributions

Throughout this paper we present the effort to build an autonomous racing vehicle at the micro scale with off the shelf components and implementations of basic path planning. This work attempts to have the following five main contributions:

i. By scaling down the F1/10th autonomous racecar to an F1/100th scale, we demonstrate the feasibility of creating highly compact autonomous vehicles and encourage developments for autonomous microscale systems.
ii. The creation of this race car is a proof of concept for a new micro-robotics racing league. This new category can stimulate interest in the autonomous racing field by drawing in others from the micro robotics field.
iii. Utilizing readily available components such as the frame of a Hot Wheels car, off-the-shelf microcontrollers, and a time-of-flight sensor, we showcase how existing materials can be repurposed for research with cost effective autonomous vehicles.
iv. Our attempts for basic implementation of AI in navigation and obstacle detection show the potential for effective integration of AI into a small-scale autonomous system, enhancing their functionality.
v. By equipping the micro race car with a ToF sensor, we enable it to detect obstacles and respond appropriately by stopping. This capability is crucial for the safe and effective operation of autonomous vehicles, even at a micro scale, and lays groundwork for the integration of more complex sensor and decision-making algorithms.

II. High Level Overview

A. Description

As a result of the limitations of the equipment at hand, the ultimate goal of this F1/100th car was reduced to a simple behavior: with a singular distance sensor, stop driving when a barrier is detected such that the racecar comes to a complete stop, based on a pre-set distance threshold, before hitting the barrier.

The Trinket M0 microcontroller (MCU) breakout is used to interface with the distance sensor and the car motors. The size of the Trinket M0 MCU for this application is advantageous as to not add much weight to the system, but this microcontroller, also, limits the car's capabilities to a behavior where only one sensor is required, and one direction is possible. The singular distance sensor used is a laser-based time-of-flight (ToF) Adafruit VL53L4CD sensor that is mounted on the front "bumper" of the car and only measures the distance in the forward driving direction of the car. Both the ToF sensor and the MCU were small and light enough to fit on the car. The MCU measured 27mm x 15.3mm x 2.75mm (1.07" x 0.6" x 0.1") and the ToF sensor measured 25.5mm x 17.7mm x 4.6mm / 1.0" x 0.7" x 0.2". Together, both boards weighed 3.2 grams, a weight that the wheels of the car can accommodate. Note this measurement does not include the wires used for connections. The ToF sensor used is able to detect up to 1200mm (about 3.94 ft) and can have a ranging speed of up to 100 Hz. The ranging speed limits the stopping distance from a barrier, governed by the equation

$$v/d < 100\text{ Hz},$$

where d is the stopping distance of the car and v is the average velocity of the car. Suppose the car moves any faster, for the same stopping distance (or in other words, the inequality does not hold true). In that case, the car can move too fast and can potentially hit a front-facing obstacle if a distance reading is taken right before the pre-set distance threshold. The car is moving at a fast enough speed such that it hits the barrier before the next distance reading is taken.

As for the mechanical facets of the car, the chassis of the car is from a pre-built remote-control car made by Hot Wheels. Minor modifications were made to ensure that our electronics can interface with the motors inside. Additionally, any electronics that come with the product that are unnecessary for our project are taken out to reduce the weight of the system.

B. Components

Below is a complete list of the off-the-shelf components we incorporated into our design of the F1/100th race car.

i. Hot Wheels 1:64 scale RC Remote Control Porche 911 Shell & Wheels
ii. Hot Wheels 1:64 scale RC Remote Control Porche 911 Motors
iii. Adafruit Trinket M0
iv. VL53L4CD Time of Flight sensor
v. Protoboard
vi. Solid Core Wire
vii. Miro-USB Cord

III. Construction and Design

Figure 3 shows the connections between the ToF and the Trinket M0 MCU. The crucial connections between these two devices are for the inter-integrated circuit (I^2C) communication interface, which allows the MCU to retrieve information from the ToF sensor. In this interface, the SDA wire transmits data back and forth between the two devices and is synchronously clocked according to the SCL wire. This communication interface on the ToF sensor is one of the sources of distance sensor limitation on the entire system. The I^2C interface requires any peripheral device that the MCU communicates with to be on the same bus (separate peripheral devices are still connected to the same SCL and SDA wires).

Each of these peripheral devices have an address (specified as a number) so that specific devices can be chosen as the “target” to communicate with. This addressing design forces each peripheral device to have unique addresses so that there is no conflict when trying to access or receive data. However, these ToF sensors have a built-in and unchangeable address that is the same for each unit. As a result, only one of these sensors can be used on the same microcontroller for data transfer to occur properly without addressing conflicts. Originally, the design was to incorporate two to three sensors on the car for full terrain navigation. With this hindrance of connection capabilities, the design was simplified to only include one ToF sensor.

The XSHUT pin on the ToF sensor is connected to a general-purpose input/output (GPIO) pin on the MCU, where the MCU pin acts as an output. The XSHUT pin is an active-low shutdown pin for the ToF sensor, so the sensor will turn off if XSHUT receives a digital low signal and will otherwise stay on throughout a digital high signal from the MCU. The handling of the XSHUT pin is abstracted inside of the ToF sensor library that we use to program the MCU and receive information from the sensor.

The GPIO pin on the ToF sensor is present to be used as an output for programming schemes that utilize data retrieval based on an interrupts service routine (ISR). In this programming mechanism, the output from the ToF sensor will be seen by an MCU GPIO pin being used as input. When it is time for the MCU to read a distance from the sensor, the sensor will output a specific signal, and the MCU will trigger the interrupt and run the ISR. This ISR contains all the code that is needed for reading the distances from the ToF sensor. The alternative method for communicating with the sensor that we use here (and that does not require the pin-labeled GPIO on the VL53L4CD) is polling mode. In polling mode, the communication with the ToF sensor is done in the normal control loop of the code. In this scenario, the MCU controls the timing of which information is retrieved from the sensor, as opposed to the interrupts version where the sensor controls the timing. Polling mode is chosen for the applications here because it does not require the extra MCU pin; as seen from Figure 3, all the accessible GPIO pins on the Trinket M0 are used in polling mode, so there is no space to use the sensor in an interrupt mode. A discussion of an alternative design that operates according to an ISR will be in the Results and Conclusions section.

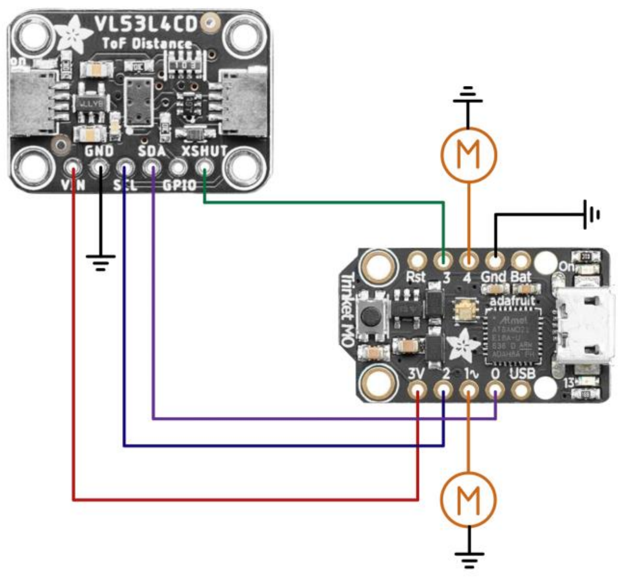

Figure 3: Electronics connections with VL53L4CD ToF sensor, Trinket M0 MCU, and car motors. The ToF sensor is powered by outputs from the MCU, and the MCU is powered through the micro USB port connected to a laptop. This micro USB port is also the progamming port for the MCU.

Code implementation of this polling mode was written in Arduino. The setup function was comprised of the setup functions of the devices. In order to check if the ToF data is ready, a do-while loop was created. Following is the do-while loop to ensure there is enough data to gather.

```
do {
status =
sensor_vl53l4cd_sat.VL53L4CD_CheckForDataReady(&NewDat
aReady);
} while (!NewDataReady);
```

Also in the main loop function, an if-else statement was created to turn on and off the motors placed on analog pins 1 and 4 (driving and stopping). The threshold that was set for the distance was around 180 mm (about 7.09 in). Following is the if-else statement implementation:

```
if (results.distance_mm < 180) {
analogWrite(1, 0);
analogWrite(4, 0);
}
else{
analogWrite(1, 250);
analogWrite(4, 250);
}
```

Another facet of this system’s electrical design is the power. Removing the tethered power supply from the MCU would allow the car to behave as an autonomous stand-alone system. However, the batteries that we had access to did not seem to be powerful enough to fully turn on the MCU, the sensor, or the motors, thus forcing us to use the micro-USB port on the MCU to receive power from a laptop. A further discussion about onboard power can be seen in the Testing and Debugging section IV.

Shown in Figure 2 is the body of the remote-controlled car used for this project. Inside this remote-controlled car was a small circuit board used to receive the provided controller data using an antenna and thus control the motors. However, for this paper's purposes, this circuit board was not used. In addition to the board, the toy car came equipped with a battery and two motors to maneuver the rear wheels only. Although the battery and the circuit were not used the motors and wheels were used to implement the project.

The initial step for this project was the dismantling of the remote-controlled car. Once the circuit board was retrieved, both motors were carefully desoldered and detached from the provided board. Once the body of the toy car was empty, the next phase was to take the original circuit board and replace it with the MCU and ToF sensor. Originally the MCU board came without header pins soldered on. After these pins were placed it was then soldered onto a protoboard. A protoboard was used to facilitate connections of the board to sensors and motors. Originally, both the MCU and ToF sensors were placed on a breadboard. This allowed us to protoype and validate both the circuit design and programming, and ensured the integration of the MCU, ToF sensor, and motors to the protoboard and chassis was seamless.

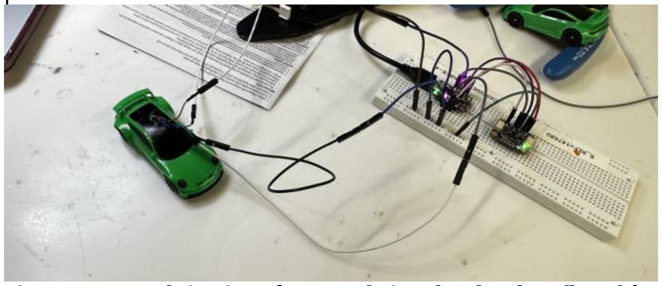
Figure 4: External circuitry of our car design placed on breadboard for experimentation and development. The Trinket M0 and ToF sensor are both placed on the breadboard and connected to cars motors through male to female wires.

When optimizing the placement of the boards on the chassis of the car, changes to the design were made to only have the MCU on a protoboard and simplify the ToF sensor with just the header pins. The ToF sensor was placed in the front of the car facing forward with the pins facing backward. To keep it in place, a piece of Velcro was placed on both the car and the sensor. As for the MCU once it was soldered onto the protoboard and cut to the right size it was held together by two zip ties to the spoiler of the car. The MCU was placed with the micro-USB connector facing the back of the car. Power and the uploading of code was funneled through the micro-USB. Placing these boards in this specific configuration helped minimize wire length and weight. Both motors were placed inside their designated place and two holes were made in the top of the car body to allow the motor wires to feed through and connect to the MCU.

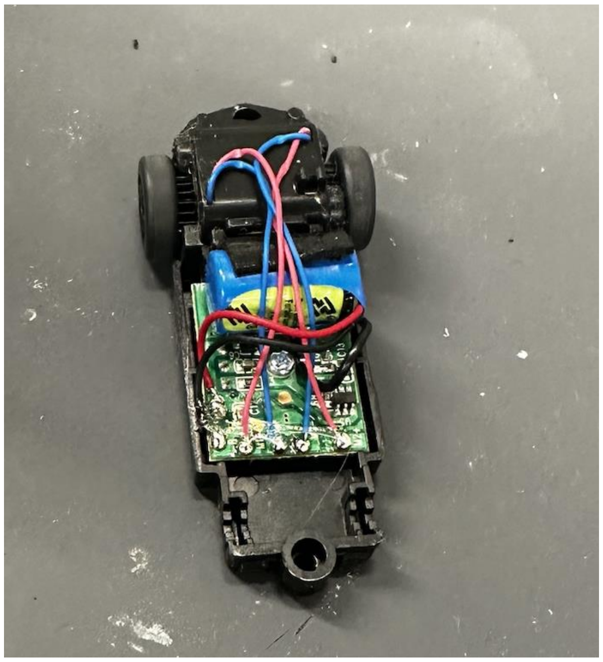
Figure 4: Internal circuitry of the original car. The large cylindrical component is the battery, powering the main chip. Wires coming from black box are connected to the DC motors, which are connected to the wheels. The signal coming off the chip to the motors is PWM. The chip also contained a receiver to communicate with the remote control.

IV. Testing and Debugging

After taking apart the car and exposing the motor and its wires, we decided to discard the control board that the car comes with because there was no documentation on that circuit and how we might use our electronics to interface with it. In doing so, we sacrifice the ability to move the car wheels in different directions since we do not have the space to implement an H-bridge into the system. Because the electronics that came with the Hot Wheels car do not have datasheets for their specifications, we needed to directly test the motors to determine the DC voltage that allows the motor to move. We obtained this voltage by connecting the motor terminals to a power supply and increasing the voltage from 0V until the motor started spinning, which was around 0.11V. This indicated to us that the 3.3V output from the MCU's GPIO pins is at a high enough voltage to turn on the motors.

Next, we wanted to see the minimum duty cycle (with a 0-3.3V signal at 1kHz) the motor needs in order to move. This step was important for determining how slow the motors can spin with respect to the pulse width modulation (PWM) output that the MCU will eventually control the motors with. Though the scope of this project had changed to simpler locomotion, we wanted to keep the possibilities open and record specifications for controlling the car's direction by using the speed of the wheels. This minimum duty cycle was found by connecting the motor's terminals to a function generator, and outputting a square wave with variable duty cycle, and increasing that duty cycle from 0% until the motor started spinning. These tests showed the minimum duty cycle for motor operation to be ~40%. In addition to a minimum duty cycle for motor operation, we wanted to see if there was a maximum duty cycle that would destroy the motor. So, we continued to increase this duty cycle above 40%. Getting to a

90% duty cycle showed that the motor would be faster under a higher duty cycle, and we observed that there were no significant thermal effects on the motor that could be felt by touch. Thus, we determined that with the 0-3.3V PWM signal, there was no maximum duty cycle that we needed to consider when programming the MCU to control the car.

As mentioned before, these tests discussed in the rest of this section were all done using a breadboard for prototyping so that permanent connections were not made before the system was validated. After testing the PWM output from a signal generator, the motor control was tested with MCU PWM output. The test from the MCU was simple: output a signal with a 50% duty cycle for one second, output a signal with a 0% duty cycle, then repeat. Because any of the pins on the MCU could be used as a PWM output pin, the motor could be connected to an arbitrary pin. We could see visually that the MCU was providing adequate control of the motor as evident by one second pulses of the motor spinning.

Testing of the motor with respect to MCU control was done for only one unit. However, once we attempted to spin the motors together while the motors were connected to the car wheels, we observed that they would spin when the car was picked up and floating in the air, but as soon as the wheels touched the ground, they would stop spinning. Upon closer observation of the spinning of the wheels, we found that the motors were spinning in opposite directions with respect to the front of the car. This had made sense because we had wired the motors with the same polarity, and thus both motors were moving in the same angular direction with respect to the motor orientation, which ends up being opposite directions when actually attempting to drive the car forwards. As a result, one of the motors needed wiring such that the red wire was connected to ground, and the blue wire was connected to the MCU's GPIO pin.

After verification of motor control, our next task was to integrate the ToF sensor. We began with a simple script that would read data from the sensor and print out the distances to the serial monitor at a steady rate. Using our hand or a block, we could move closer and farther from the sensor, getting live measurements and values to use later. With the goal of object detection, we created another script to integrate the motor control and the ToF sensor. Upon detecting an object within a certain distance, the motors would shut off. If no object was detected, the motors would run at full speed. We tested this by using our hand or a block, with the car safely off the ground, to check that the motors would turn on and off correctly. Once we knew the car would stop properly, we tested it on the ground with a wall ahead. After tweaking the stop distance a few times, we reached an optimal value for the car which was around 180 mm (about 7.09 in).

Another facet of the system that we had tried to debug was the integrating on-board power. We first started with a singular 3V coin cell battery and a 1.5V coin cell battery in series to produce a 4.5V source that the MCU can run on. The positive end of voltage from these batteries would connect to the "Bat" pin on the MCU and the negative end would connect to ground. When doing this, the MCU showed different results than it did when plugging it into a laptop. By powering the system with the laptop, the multi-colored LED on the MCU's breakout board would glow a bright pink, and the ON signaling LED on both the MCU breakout and ToF sensor breakout would glow green. However, when using this 4.5V source with batteries, we the multi-colored LED was instead glowing red, and the ON signaling LEDs were very dim and not glowing throughout the entire bulb. In addition to these signaling LEDs, the MCU was not running the code that we had uploaded, as indicated by the lack of motor movement. Although the MCU's datasheet says that 4.5V would be a high enough voltage for the system, our set up did not seem to be working. As a result, we decided to increase the input voltage to 6V, using instead two of the 3V batteries in series. The results that the MCU exhibited with the LEDs did not change, and the MCU still did not execute the uploaded code. Because the batteries were not powering the system correctly, we decided to use a bench power supply to check that the voltage connections to the MCU was done properly. To do this, the power supply was set to 4.5V, and its outputs connected in the same way that the battery was connected to the system. Connecting the power rails in this way showed correct operation of the MCU as indicated by both the LEDs and the code execution. Because the bench power supply worked as a valid source and the batteries didn't this indicated to us that the batteries could not output the amount of current that the system had required, and that the load was too high to keep the battery voltages at their nominal values. These tests for the batteries showed the lack of resources for removing the tethered power, and we had to work with using the micro-USB cable for power.

V. RESULTS AND CONCLUSIONS

Figure 5 illustrates the final design of the project. The front part holds the ToF sensor connected to the MCU which is placed on the back end of the car. As for the motors, they were placed in between the top and bottom parts of the car's body and wires were fed through the top. A video of the car's performance and implementation can be found in Appendix 1 of the paper.

One main drawback of the system was the overall weight of the system. We observed a speed reduction of about 80% when driving the car with our custom electrical system compared to the remote-controlled car when operating as is. This speed reduction indicated that mounting components onto the car made the system too heavy for the small motors to spin as fast as they did.

Another aspect of the system that affected the results was the tethering to the car to use as a power supply. In addition to taking away autonomy from the car, this USB tether also made it difficult to obtain consistent results when testing the car's operation. The cord used to connect a laptop to the MCU was a thick micro-USB to USB cable (around 5mm in diameter). Although the cord was malleable, the thickness for the cable's insulation made it sturdy, especially at the ends of the cable where the car and the laptop were connected. Paired with the weakness of the small motors, the thickness of the cable made it impossible for the car to move without any of us intervening and holding the cable in a specific way. However, by holding the cable, the car is susceptible to any small movements based on how the cable is being held, especially because of how thick and sturdy the cable is. When considering the sturdiness of the cable at the micro-USB end that is connected to the car, it can be observed that the cable thickness, along with how well the cable is being held in the air, greatly affects how the car would move. In one way, the car could be completely held back from driving if the cable isn't consistently moving along with the car, since the cable wants to keep its shape and resist the car's movement. In other ways, the car could be pushed forward by the cable if it was moving faster than the speed of the car, since the sturdiness of the cable at the car end would allow the car to move instead of letting the cable bend. Additionally, we often observed that having the cable being held slightly to the left or right would make the car swerve quickly in that direction and away from our barrier. All these effects of the micro-USB cable as a tether made the simple test of the car stopping in front of a barrier inconsistent across trials.

Our design was tailored towards efficient and cheap production, with an emphasis on achieving feasibility. This leaves room for plenty of further work and improvements to be made. One of which is untethering the power supply to the system. To do this, a battery powerful but small enough is necessary to attach to the system. Finding room on the car with our current design was a challenge, so optimizing space to create room will be important.

Even further work on this F1/100th car would involve building a custom PCB so that the microprocessor on the Trinket M0 breakout board could have the unnecessary components removed and more GPIO pins can be exposed for controlling the system. This custom PCB may benefit from being a flex PCB to reduce system weight, allowing the car to move faster. Having better access to the MCU could also help us implement closed-loop control of the system with feedback so that driving is better controlled with respect to sensing.

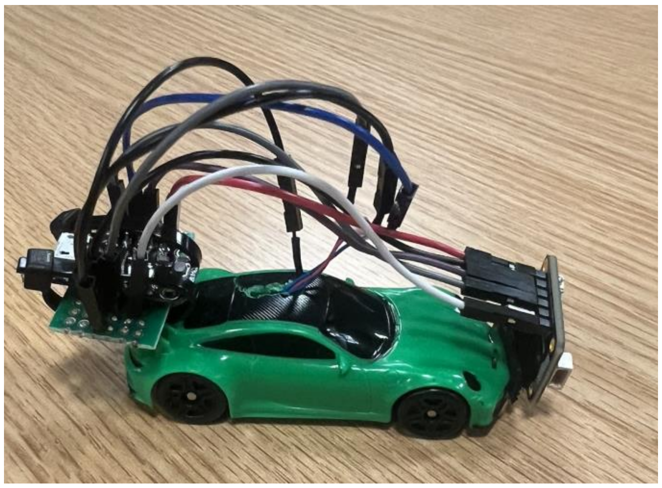
Figure 5: The final design of the car.

APPENDIX 1

The link below is a video demonstrating the implementation and performance of the car:

Video Link

APPENDIX 2

Engineering is never perfect and brings many imperfections but that is the beauty of it. The following video showcases the challenges encountered during the making of this project:

Video Link

ACKNOWLEDGMENT

This project was worked on as a final deliverable for ECE 6960, Special Topics in Electrical and Computer Engineering. ECE 6960 focuses on micro and nanorobotics topics with emphasis on new and upcoming research in the field. Teaching at Cornell University, Professor E. Farrell Helbling fostered learning and passion throughout this project.

REFERENCES

[1] G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.
[2] J. P. Wilkinson, "Nonlinear resonant circuit devices (Patent style)," U.S. Patent 3 624 12, July 16, 1990.
[3] F1/10 Autonomous Racing, "Build your car," [Online]. Available: https://f1tenth.org/build.html.
[4] J. Betz et al., "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing," arXiv:2202.07008 [Online]. Available: https://arxiv.org/pdf/2202.07008.
[5] W. Xie et al., "Heterogeneous silicon photonics sensing for autonomous cars [Invited]," Opt. Express, vol. 27, no. 3, pp. 3642, [Online]. Available: https://opg.optica.org/oe/fulltext.cfm?uri=oe-27-3-3642&id=404589.
[6] B. D. Evans et al., "Bypassing the Simulation-to-reality Gap: Online Reinforcement Learning using a Supervisor," [Online]. Available: https://f1tenth.org/publications/Simulation-to-Reality_Gap.pdf.
[7] Hot Wheels, "RC Porsche 911," [Online]. Available: https://www.amazon.com/Hot-Wheels-RC-Porsche-911/.
[8] Adafruit Industries, "Product Title," [Online]. Available: https://www.adafruit.com/product/3500.