**NATCAR Final Report**

Team: Occam’s Razor

By: Reginald Lai, Andrew Trinh and Thomas Turner

EEC195A-B

Lance Halsted

22 March 2014

**Table of Contents**

|  |  |
| --- | --- |
| Title Page | 1 |
| Table of Contents | 2 |
| Overview | 3 |
| Technical Report - Hardware Design: by Reginald Lai | 4 |
| Technical Report - Steering Control: by Andrew Trinh | 6 |
| Technical Report - Speed Control: by Thomas Turner | 9 |
| Design and Performance Summary | 12 |
| Safety | 13 |
| Appendixes | 15 |

**Overview:**

Team Occam’s Razor’s car was built with the general idea of going fast on straight aways, and making turns as efficiently as possible. This concept was brought from our belief that this car should be ran with a simple and efficient algorithm, rather than a robust and complicated one. Our car utilized 2 linescan cameras and a hall effect speed sensors to move through the track efficiently for the checkpoints and competitions. When focusing on the general idea of completing the track as efficiently as possible, we divided tasks that seemed most reasonable with our members skill sets.

Reginald’s tasks involved building the car and working on putting the PCB together. These tasks included soldering the motor control boards, designing the hardware mounts, and organizing the components connected to the car.

Andrew’s tasks involved coding, designing, and implementing algorithms. This included coming up with ideas and improvements that would increase the overall speed and efficiency of the car. Also, this involved designing motor control equations for steering and speed.

Thomas’s tasks involved the software aspect of the autonomous race car. This included the designing, coding, and debugging of algorithms to increase the overall performance of the car.

By dividing each member with as set of tasks, our team was able to complete each checkpoint at a timely manner. The table following this section analyzes the performance of each team member throughout the process of creating a functional car (See table on next page.)

Team Performance:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Member** | **PCB**  **(25%)** | **Hardware**  **(15%)** | **Debugging**  **(10%)** | **Coding**  **(30%)** | **Testing**  **(15%)** | **Report**  **(5%)** | **Overall Performance** |
| Reginald Lai | 25% | 15% | 5% | 0% | 15% | 5% | 65% / 195% = 33.33% |
| Andrew Trinh | 0% | 5% | 10% | 30% | 15% | 5% | 65% / 195% = 33.33% |
| Thomas Turner | 0% | 5% | 10% | 30% | 15% | 5% | 65% / 195% = 33.33% |

Signatures:

|  |  |  |
| --- | --- | --- |
| Reginald Lai | Andrew Trinh | Thomas Turner |
|  |  |  |

**Technical Report-Hardware Design**: By Reginald Lai

My focus for the NATCAR project was building the mounts for the PCB and most of the hardware that was installed on the car. This included the custom motor control PCB and other parts that were installed to make the car function. My team’s main objective when building this car was to make our car light, well organized, and functional. Before every step was finalized, I consulted with my group mates for suggestions and changes.

The most important part to work with when building our car was where to install the cameras. Since I was given two to work with, I had to find an acceptable height for each camera that would make our car run to its full potential. Mounting the top camera too high or aiming too far ahead caused our car to see the carpet when going up the hills. The bottom camera saw much closer as it was closer to the ground. I installed two red LEDs in front of the bumper to allow the bottom camera see clearer as the cameras given is most sensitive to red and white light. Since we weren’t allowed to use any other external battery, I connected the LEDs to the 7.2V battery to power the car with a resistor large enough to prevent the LEDs from burning out.

The objective that our group came up with when building the component mounts was to make sure that the car is as light as possible. For this case, I chose to use plastic plates (e.g. the house ‘For Sale’ sign). The material is still stiff enough to securely hold our PCBs but is much lighter than using metal. The remaining LEDs and plastic plates were used for aesthetic features like a spoiler and blue LED rail in the back (also connected to the 7.2V battery.)

The most intricate part in physically assembling the car was the Hall Effect sensors and the magnets. I had to make sure the sensors were close enough to the magnets to read voltage values. There were five magnets evenly super-glued on inside of the two wheels connected to the motor. That way, the number of times magnets were to pass the sensor would be consistent over a period. The Hall Effect sensors were then super-glued about two or three millimeters away from the magnets to capture the rpm of the magnets as best as possible.

In terms of circuitry and the custom PCB, our group decided to use the motor control board that was provided by Lance. Unlike the TFC shield provided by Freescale, the motor control board only had a discrete half-H bridge to control the motors of our car. With two DMOS per motor, the car would only be able to move forward or brake. Wiring the DMOS correctly without burning leads on the custom PCB was the most challenging part.

For the purposes of safety and assurance, our team decided to only use the low side DMOS to power our motor controls. Initially, we wanted to implement braking onto our car, but we kept getting irregular signals going into the high side motors. Instead, we configured the motor control board to only run forwards or freewheeling (See Figure 1 below.) I still installed the high side DMOS onto the board, but I grounded the high side inputs. That way, the high side DMOS were guaranteed to be off as all times while the car was running. I was going to use protection diodes in place of the high side DMOS but I figured that turning them off will have the same effect.

There were a total of three different voltage levels throughout the circuit: 3.3V from the freedom board, 5V from a voltage regulator and 7.2V from the battery. The 3.3V was to power most of the headers (e.g. both cameras and the Hall Effect sensors) and the 5V from the voltage regulator powered the freedom board and the Maxim 620 MOSFET driver. The 7.2V battery was to power up the LEDs, the servo, and act as the input for the voltage regulator.

I added a switch to create a power sequence with turning on the freedom board first, then the Maxim MOSFET driver. The switch was to turn on the driver after the microprocessor board to prevent any PWM signal that was trigger before our code ran to reset the signal values (usually instantaneous.) After the PWM signals are set to the desired values from the microprocessor, I would then be able to turn on the Maxim MOSFET driver. I also added a 100KΩ potentiometer and two switches to replicate our start-up sequence with our TFC shield. The schematics for those can be found on the TFC shield schematics. Pin headers for all the components (e.g. camera, servos, and speed sensors) were created avoid messy wires running throughout our board.

**Technical Report - Steering Control**: By Andrew Trinh

An important part of an autonomous vehicle’s motor control is its steering. Steering control determines where the car is going to go next by taking in data from its surroundings and estimating where the car should be. Our NATCAR vehicle’s steering control revolves around the car’s servo to turn the wheels. The control is based off of 2 main inputs, track location and speed. Our main objective for the car’s steering control is to efficiently steer through the course without going off track. We’ll first discuss about how we use the inputs to derive our control equation, then about the special cases we had to address such as the wavy track.

The first input we use for steering control is the track location. This data is gathered by our 2 mounted TSL1401R−LF linescan cameras in front of the car. One is mounted up high to see up ahead, while the other is mounted low to see close. We use a voltage threshold to discern which values of the linescan camera are the track and which are the white floor. This gives us an idea of how far the track is relative to the center of the car. We generally use the track location given by the higher mounted camera, but with some special cases (where the track is visible to the low camera only) the lower camera’s track location is used. Having a camera that sees farther allows the car to predict a turn before it arrives, while the close camera prevents going off track. The second input used for steering control is speed. We installed a SR13C-A1 Hall Effect Sensor near our car’s wheels to detect the speed of the car. With the addition of the speed sensor, we were able to adjust our servo more efficiently to stay on the track.

There are several different control equations we used based off of the speed and track location. We split the track location into 3 regions, left, middle, and right region. While the track was in the middle region, we used a proportional equation

(1)

Where is the gain of the proportional control, which we use as ½ . is the reference point to the center on the track, we used 62. is the input value of the track location. is the servo value to go straight, we used 48. We calculate the error from the reference point (center of the track) and use it to slightly adjust the servo to keep straight. With a small gain like ½, we avoid overshooting the turn and oscillating left and right on the track.

If the cameras detect a turn up ahead, there were 2 different steering equations used that were dependent of speed. A speed threshold is used to determine whether the car can make a turn at its current speed. If the speed was above the threshold, the car’s servo would hard turn left or right immediately. This was done to compensate for the overturning that would occur when coming into a turn too fast. If the speed of the car was slow enough to make the turn, then we use a proportional control equation such as

(2)

is the value that separates the middle region from the right region. is the gain of the error, we used 0.02 . This equation was designed to turn the servo harder as the distance away from the track increases. By including ) , we can “exaggerate” the turn, which helps prevent overturning. This control equation is used for turning right. Left turns are done similarly, with .

There are several special cases that we had to address specifically for this kind of race track. The first one was having our car handle the wavy track. This issue was simple to fix by increase are mid-bands. By increasing the mid-band, then the car would not use the left/right control equation (2), but use the middle steering control (1). We also had a case where if the two mounted cameras had disagreeing values (one read left and the other read right), then we would ignore it.

Another special steering case is when the camera did not see the track, but saw the white floor or off-track rug. Both of these cases typically would occur when the car is veering off track by overturning. We had to figure how to deal with staying on the track once we have already overturned. The solution for steering when only the white floor is seen was solved with an optimistic approach by simply finishing the turn. We would have a variable that remembers if the turn we were making was a left or right turn, and if we came across a white floor, we blindly continue the turn (at a sharper rate) until we reach the track again. When the camera saw the rug, we took a similar approach by finishing the turn at a sharper rate. Differentiating the detection of the rug and a track was done by checking if the car was on a left or right turn.

**Technical Report- Speed Control:** By Thomas Turner

**3. SPEED CONTROL**

The main objective for speed control are different for the TFC shield and custom PCB board. For the custom PCB board we used did not have the ability to perform reverse braking due to unsupported functionality. Because of the objective difference between the TFC shield and PCB board, this speed control section will be divided in two: Explaining the methodologies used for the TFC shield to achieve our objectives and how the TFC shield braking methodology differs when using the PCB board.

**3.1 TFC shield:**

The objectives of speed control for the TFC shield are implementing:

* Braking (reversing the wheels)
* Current Feedback
* and Speed sensing

**3.1.1 Braking:**

There are two reasons why we implemented reverse breaking. The main reason is to prevent the cameras from detecting the 'black' rug because we did not have a method for ignoring the rug while using the TFC shield. If the camera that sees further out detects the black rug during a turn, our autonomous race car will run off course because our code will calculate the middle value of the rug (any black it sees) instead of the black strip and use that calculated value to steer the car. Because we did not implement any off track sensing, this leads us to our second reason for using reverse braking, which is to avoid overshooting turns even while traveling at fast speeds. Also note that while the car is in a turn, after it does reverse breaking, we slow the inner-wheel down to help prevent our car from overshooting turns.

**3.1.2 Method for Reverse Braking:**

To implement reverse braking in our code we decided to use an interrupt handler to determine how long we wanted to brake for. After the duration of the brake we then slowed the inner-wheel more than the outer-wheel. The method is as follows:

* When a turn is detected and reverse braking is necessary, the car will start to reverse its wheels and a braking flag will be set.
* Inside the interrupt we check for the braking flag, if the flag then a counter is initiated.
* Once the counter reaches a certain amount of time, the wheels will then begin to move in the forward direction.

**3.1.3 Current feedback:**

The reason why we implemented current feedback is to speed up the car while going over hills. When going over hills there is current build-up inside the motors. We take an advantage of this and use feedback to detect how large the current build-up is.

**3.1.4 Method for Current Feedback:**

To implement current feedback we set a threshold value so the car would know when to increase its speed. We chose a specific threshold that was large enough to detect if the car has encountered a hill. We knew that as the current build-up increased, the feedback value increased. The process to determine the correct threshold was a heuristic process because there is still some current build-up in the motors while the car is driving on the track. So we had to change the threshold and run the car on the track to see how it performs when going over hills. We did not want the car to unexpectedly speed up because we set the threshold value really low. We also did not want to set the threshold value really large because the car would not go over the hill very smoothly (The car will lose momentum and stop, then speed up rapidly due to the large current build-up).

**3.1.5 Speed sensing (Hall effect):**

The purpose of using speed sensing is to determine when the car actually needs to do reverse braking. If the car is traveling very fast, it needs to do reverse braking in order to prevent overturning and the detection of the rug. While looking at the other case, if the car is traveling slow enough to make sharp turns efficiently than it does not need to do reverse braking.

**3.1.6 Method For Speed sensing:**

The method used to know when reverse braking is needed is similar to the method used for the current feedback; by using a threshold value. In this case the threshold value is compared with the average input capture value of the hall effect sensors. The slower the wheels rotate the larger the average input capture value. So the speed of the wheels is inversely proportional to the average input capture value. How we determined the threshold value was again another heuristic process; setting the threshold and testing how the car performs on the track. We needed to find the middle ground to actually know when reverse braking should be used at certain speed. So we tested the car to find the maximum speed it can make a sharp turn without actually braking. We then set the average input capture value that corresponds to maximum speed, at which no braking is necessary, as the threshold.

**3.2 How Does The TFC and the Custom PCB Braking Methodology Differ?**

The custom PCB board used only contained a half h-bridge, so we are only able to implement free-wheel braking instead of reverse braking. Without reverse braking the car does not turn as efficiently when traveling at faster speeds and thus will over overshoot turns. The prevention of overshooting now entirely depends on steering control.

The code implementation of braking while using the PCB board is still similar to the code while using the TFC shield; Instead of the wheels reversing, it is now free-wheeling.

**Design and Performance Summary:**

Overall, we were successful in completing our objective. Our first race time was 29.92 and our second race without the TFC shield was 30.9 . Our car was able to completely the track smoothly and executed the way we liked; Go fast on straight aways and turn efficiently.

The car during our first race ran exactly as we wanted to. Despite it having a few hiccups during the timed run, the design choices we made fell inline with our vision of the car. Reverse braking with the full H-bridge helped tremendously. We were able to run through straight aways with great speed and could break strong enough to make the turn while keeping the momentum of the car. There were a several tweaks here and there that we did not get to make prior to the race that we have solved now. We also changed a few design choices since the first race that greatly improve the turns.

Our car performed above our expectations during the second race. With the half H-bridge, we were left to use free-wheeling to slowdown the car rather than braking. This made the car go slower on the straight tracks, but forced us to improve on our turning. From the time of our first race to the second, we implement a few turning design choices that helped us make smoother turns. One fixed issue was when the car would confuse the track with the rug. We also came up with better controller equations and eliminated almost all oscillation.

One thing that we would do differently is to use a custom PCB that we designed ourselves. We wanted to build our own PCB motor control board, but we did not have enough time to plan nor submit our design. By using the full H-Bridge that was given to us from the TFC shield, we were able to make better turns by using reverse braking. On the custom motor control board that Lance gave us, we were limited with only forward and freewheeling. Our braking system was not as effective on the custom PCB than on the TFC shield. As a result, our performance in terms of speed was slightly slower in Competition II compared with Competition I.

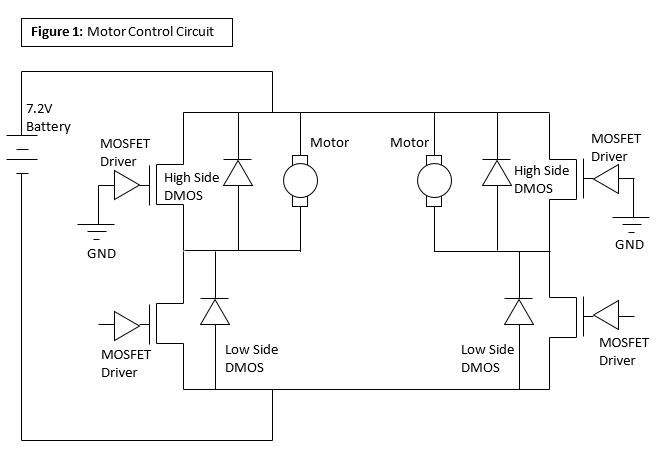
Another thing we would do differently is to adjust the height of the camera mounts. We were satisfied with the current height, but as we improved our car more and more, we found that a higher camera mount would be more efficient. This would let us see farther ahead of the track and cut turns much faster. Our current height limits us from cutting turns, which leads to a lot of overturning when our car is going fast.

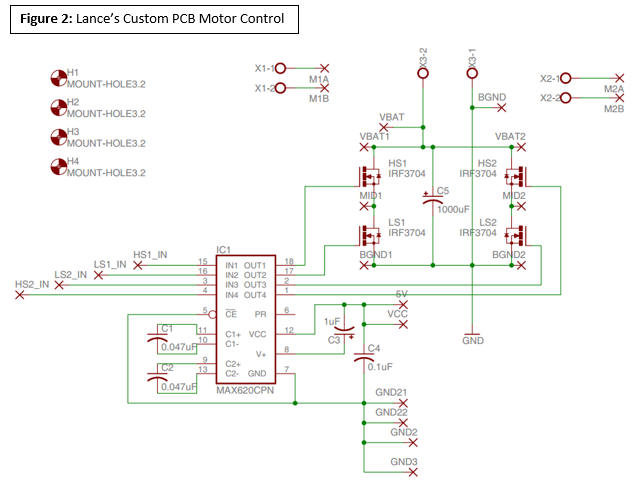
**Safety:**

The first step we took to ensure the safety of our car and the people around us was to have a steering control that prevents the car from losing the track. The method used to add this steering control feature is as follows: Keep track of the last turn made, if both cameras detect white floor (about to drive off the track) then check the last turn that was made and set the servo to turn in that direction. This concept works with the idea that we don’t expect the car to run off track unless its on a turn. By turning once its goes off road, the car avoids dashing across the room and attempts to get back on the track instead. A safety feature that we could have also added was to immediately stop the car once the cameras detect the rug, but we opted to allow it to find its way back on the track. Another safety concern was protecting the hardware of the car.

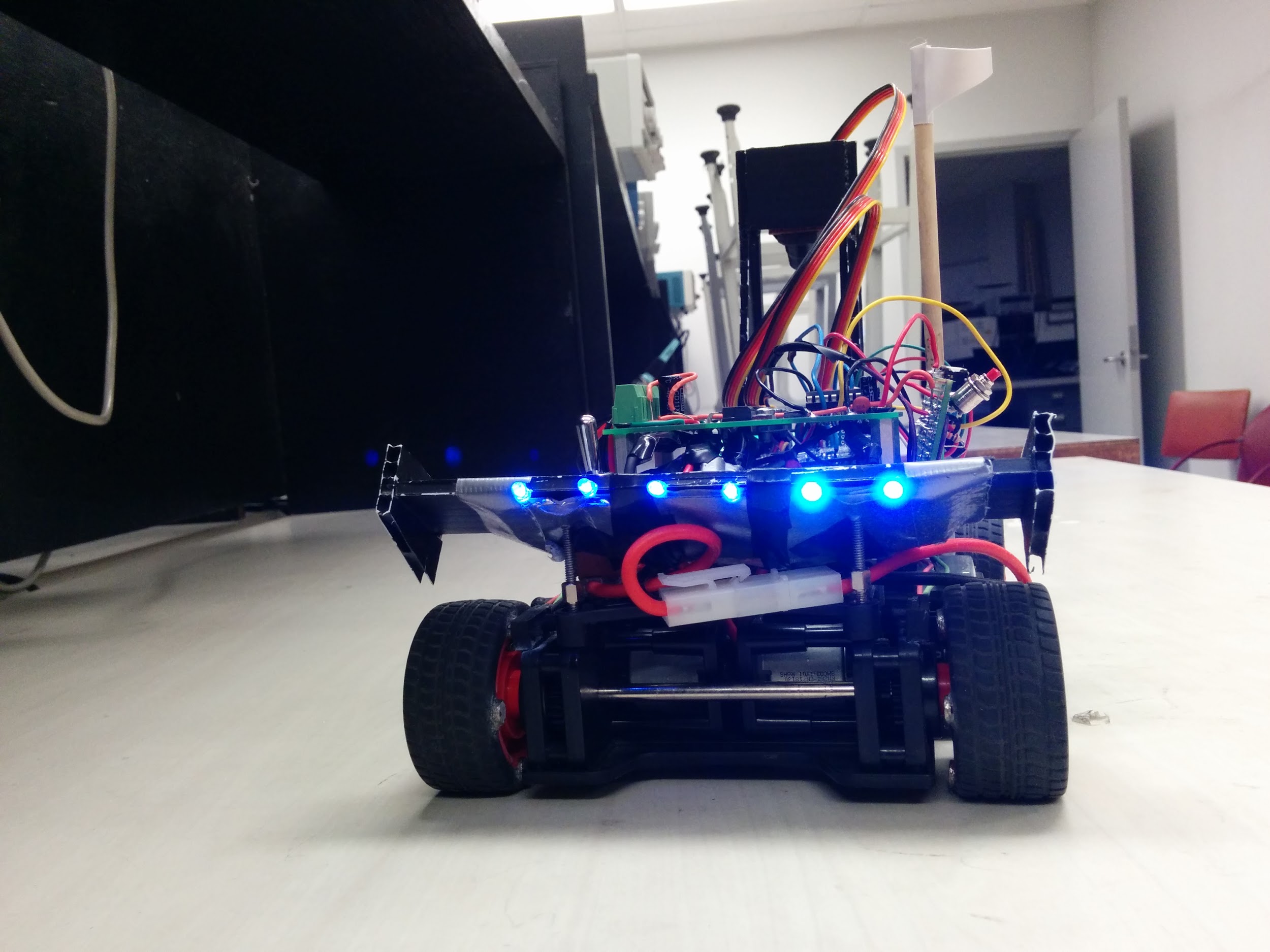
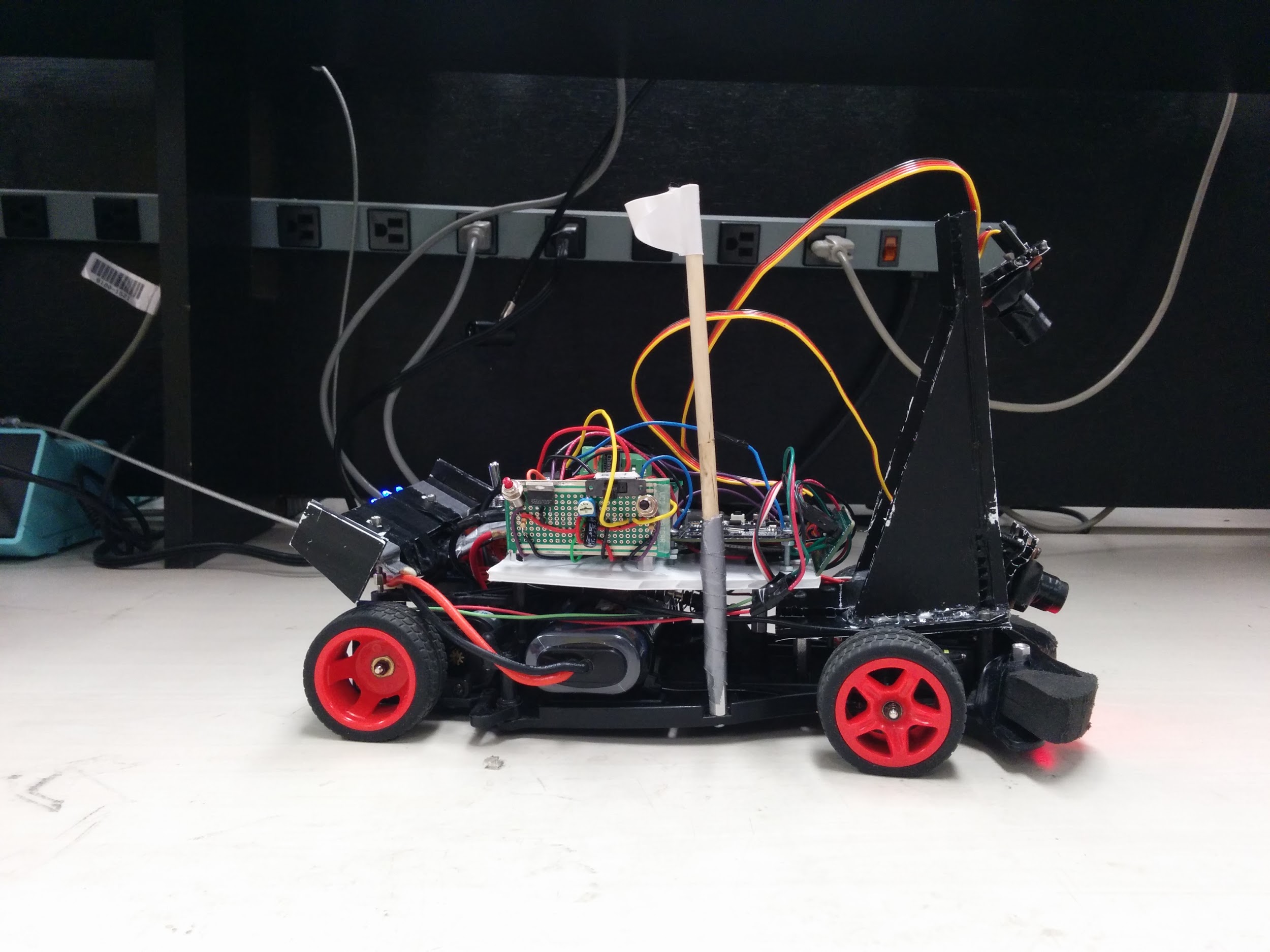
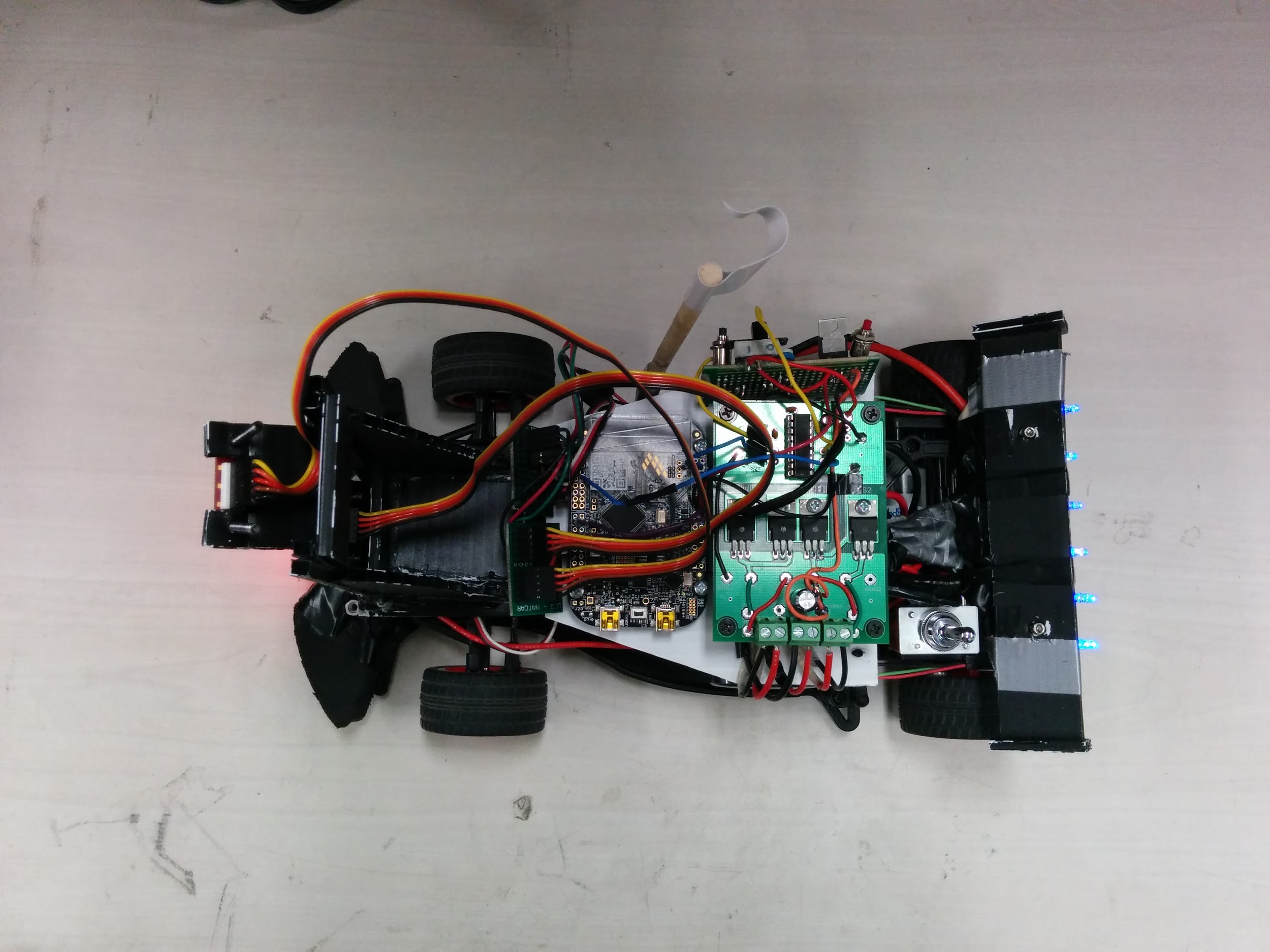
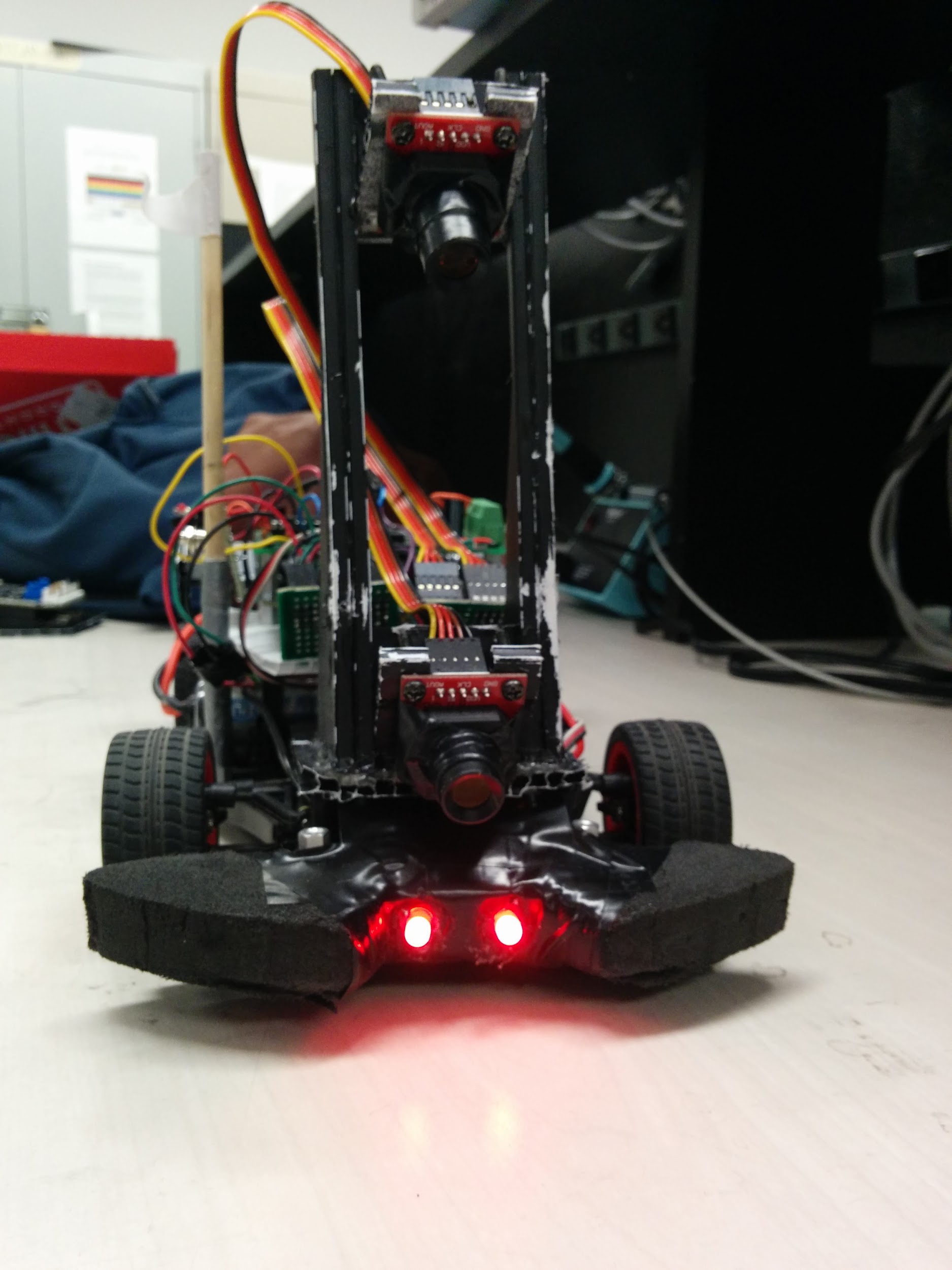
We added a bumper to the front of our car to prevent our cameras from being damaged. The bumper is made of a stiff, foam-like material to absorb impact while maintaining its original shape. The bumper was designed to allow the near camera to still see the road while still providing enough distance to prevent the cameras from smashing into the walls. We also used plastic boards to mount all of our equipment to prevent accidental shocks from any circuits that was attached to the car. We needed to add a voltage regulator to prevent certain components from frying. The battery supplied 7.2V while the Maxim MOSFET DRIVER only needed 5V. Any more voltage would fry the connections between the DMOS components. As mentioned earlier in Reginald’s Technical Report, we also installed a switch to create a safe power sequence.

**Appendixes**





**Pictures of Our Car:**



(The code will be attached separately due large quantity of paper.)

-main.c - 17 pgs.

-tpm.c - 2 pgs.