IEEE Grand PriEEE (Formerly Viacar) & NATCAR 2013-2014

Project Documentation

Ferdinand Soedharto

Foreword from the Author

Because this is my second time participating this year, I decided to improve upon my design from the previous year. I'll outline differences from last year's car and this year's car, and hopefully the next generation people can learn the advantages and fault of this design. I will also show what I think was the mistake that I made for this year and hopefully the next generation team leads will be able to follow and do better than the previous year's design.

Members:

Alexander Chang (Lead Programmer)
James Tang
Shengfei Yu
Yunting Zhao
Borhan Vasli
Albert Xu
Richard Newby
Ihab Salameh

Overview

To design an autonomous robot car that will run on a 100mA RMS current carrying track. The car size must meet the rules specified in the IEEE Grand PriEEE and UC Davis NatCar. Must design the flag to trigger the race timer during the competition.

Car Design Overview

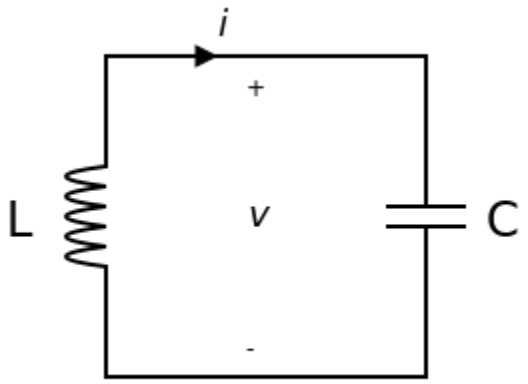
In our design for this year, we opt to use the inductive approach.

We will use a combination of Top-Down and Bottom-Up design to determine the optimal design of our car. First of all, we want the car to work, and then before adding any complicated control system, we want to make sure that the individual system works. Our car system design can be broken down as follows:

1. Sensor Design
2. Motor Control Design
3. Regulation Systems
4. Safety Systems
5. Steering Control Systems

Sensor Design

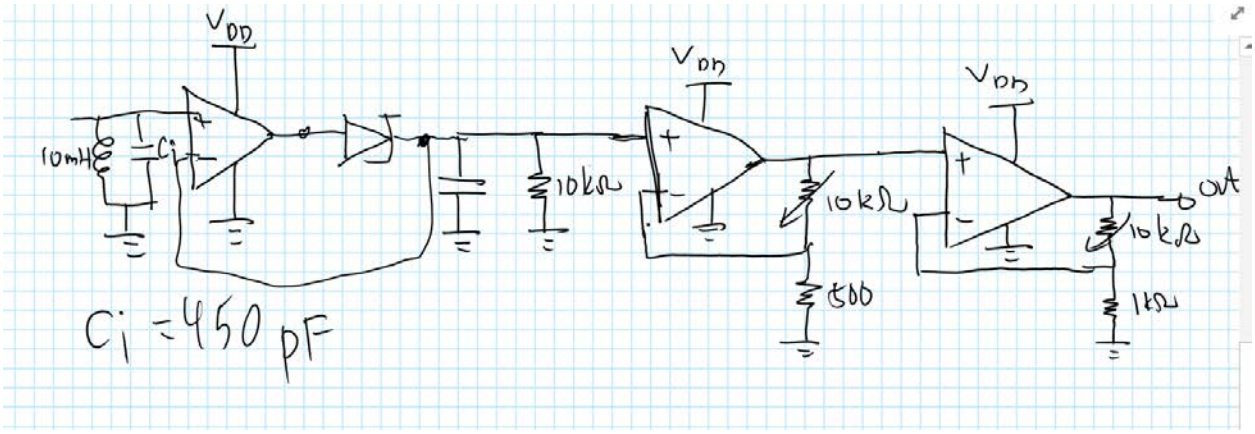
Since we are using inductive sensors, we want to be able to detect the track, and since the track frequency is 75kHz, we will use an LC tank circuit



Knowing the frequency of the track, we can calculate the needed values of inductor and capacitor, given that L = 10 mH, C = 450pF.

3-sensors will be placed at the top of the system. This sensor will need to be very good and needed some amplification in the system.

Below is the sensor design that we used for our PCB:



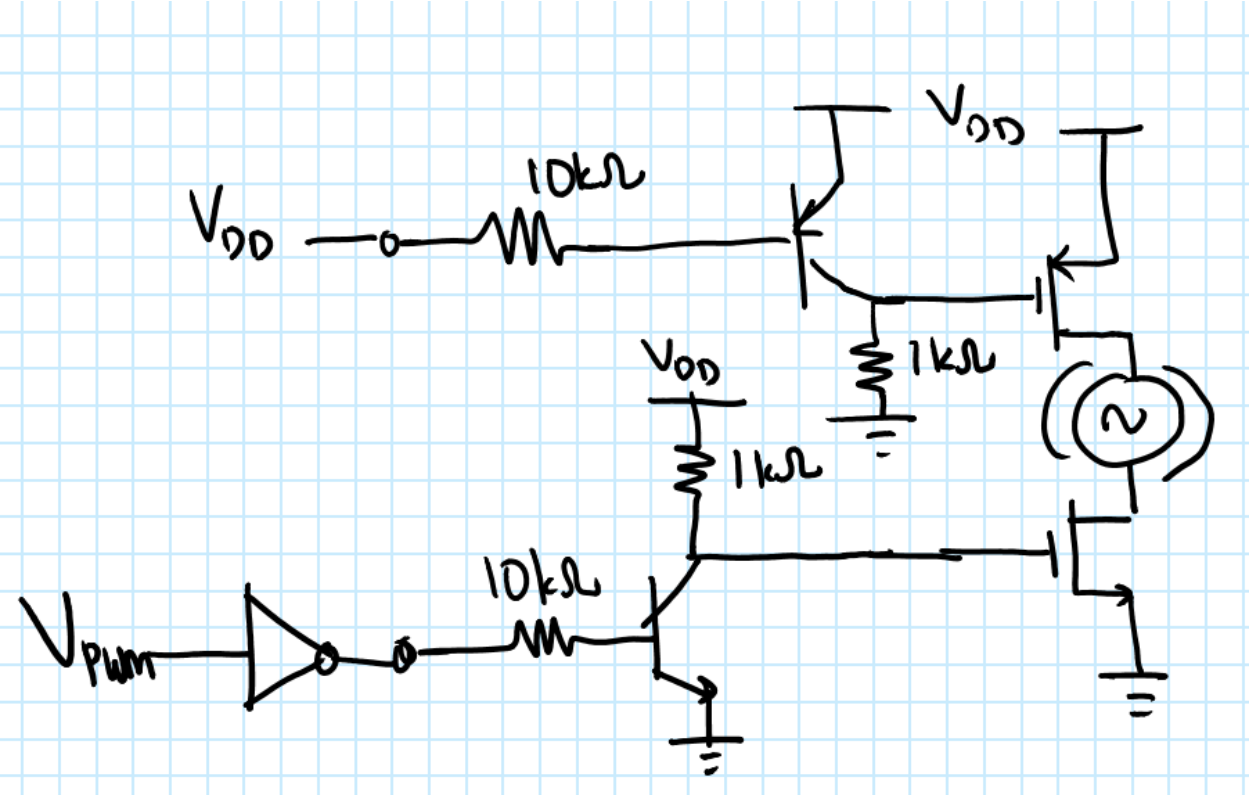
As we can see this is a rectifier followed by two-stage amplification with potentiometers. Rectifier circuit converts AC to DC which will then be amplified. As you can see there is no low-pass filter stage. We designed without the low pass filter stage on the assumption that the signal filtered will be completely at the baseband without aliasing. Thus an improvement to the design could be the use of low-pass filter to prevent aliasing before amplification. The value of the potentiometer does not matter so long as it

can provide enough amplification. Despite all this, the design worked to the specification that we wanted to.

The use of Shottky diode is essential in our design since it gives our system a more dynamic range.

Motor Control Design

We opt to use a half-H bridge for our motor control circuit since we will not be going in reverse. Below is the overview of our motor circuit:



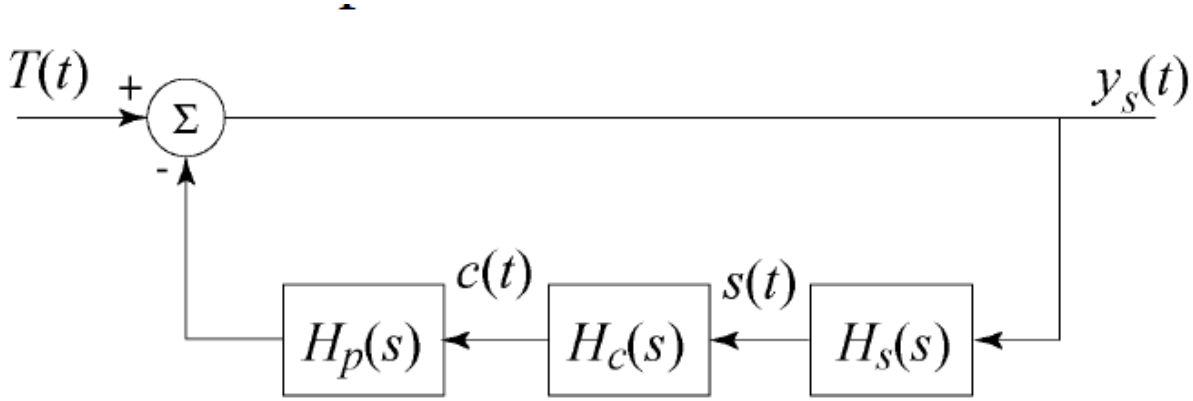
As you can see here, the circuit consists of MOSFETs and BJTs. The MOSFETs are primarily used for the control of the motor, while the BJTs act as gate drivers, which will supply the needed current to switch the gate on. Through experimentation, we determined that the ratio of gate to collector resistance to be 10. A higher ratio will cause the motor to not turn on and lower ratio will drive the motor towards one state. As you can also see the motor control pull up gate drivers is inverted, and therefore it is necessary to control its pull down state by inverting its output. If you also see the pull up network, you can see that

it is directly connected to the VDD since we do not need to switch both state and therefore we can always keep one of the MOSFET on.

Steering and Speed Control Systems

Design of the steering loop will be dependent on the hardware and software execution time. In the aspect of a hardware, servo delay plays an important role and therefore to design the best car, it is necessary to get a high performance servo, which is a servo with a high torque and high speed. The fastest 1/10 servo could go as far as 0.05 sec/ 60 degrees. Therefore, it is necessary to spend a considerable amount of budget in buying a high grade servo since it will increase the loop stability of the car as delay is a primary problem in the steering systems. Also, execution time will decrease the loop stability since it will add a delay to the system.

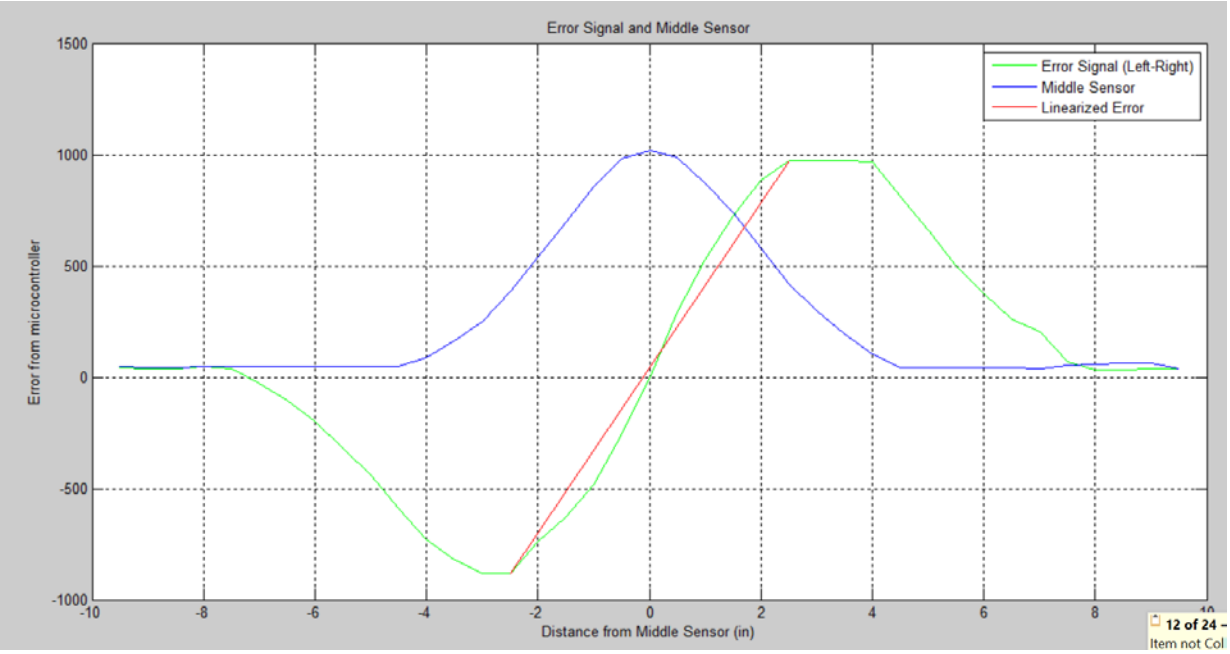
I'll use a paper from Berkeley and class slides from UC Davis NatCar to model our system.



Hp(s) will be the car transfer function
Hc(s) will be the controller (PID)
Hs(s) will be the sensor transfer function
Ys(t) is the distance deviation from the center track (error)
T(t) is the track signal.

Based on this control scheme, we will need to tune the controller Hc(s) since we are using PD control that will respond better to the track. Simulations will be a good idea, but trial and error, which we used, are much better since we get faster results. In designing the controller, we decided to empirically figure out the error signal deviation from the center of the track. Because the sensor gain might not be equal, calibration is necessary at first, so we need to calibrate the sensors so that at the center position the

center sensor is maximum at that point and make sure that the two sensors left and right are equal. We also need to increase the sensor so that each sensor is right at maximum when they are placed on top of the track. After the calibration, we need to look at the voltages of the error of Left-Right and the middle average value as we position the sensors along different points away from the track and putting them into Matlab, we get the following values:



From this graph, we already know how to construct a controller scheme. As we can see here the graph of the sensor error of Left-Right is very linear around the center point. Negative represents the distance to the left of the track. However, as you can see the error peaks at some value and the error signal decreases, and thus this signal is non-monotonic. However, looking at the middle values, we can create an algorithm that will lock the wheels to the maximum and ignores the PID control when the middle average reaches a certain value and the error signal is on a certain value. The disadvantages of this, however, is if the distance of the sensor goes by more than 7 inches to either the left or right of the center, then the locking algorithm will not lock at the correct angle. However, this is assuming that the locking algorithm starts at a point far away from the center of the track. However, if the deviations from

the center are small enough, this algorithm will be able to correct for the error and it will return to the track.

For our speed control system, we will go with an open loop speed control. Our speed control algorithm will depend primarily on the servo position. If the wheels are pointing straight then it senses are straightaway and it speeds up. If it turns, it will slow down. If it senses a sharp turn, it will slow its speed even more. Therefore, our speed mode are divided into three components, high speed for straight, mid speed for mild turns, and low speed for sharp turns (i.e., steps).

Regulation Systems

The purpose of regulation system is to create an acceptable voltage to other systems. The battery supplies 7.4 V and will render a component useless if the component has a lower maximum rating, which most of our subsystems have. Below are subsystems which need lower voltage from the 7.4 V Gens Ace LiPo:

- Sensor Regulation (5 V)
- Servo Regulation (6 V)
- Remote Switch Regulation (6 V)
- MCU Regulation (5 V)
- Additional 3.3 V Regulation Systems (3.3 V) (Added later)

We will use a TO-220 regulator packages since they are cheap and abundant and easy to use.

Safety

Remote Switch and Switches

The remote control only stops the car from moving, so if the remote is triggered, the microcontroller Arduino Nano will count the time of pulse. If the time of pulse matches, then the Arduino Nano will send a signal to the ChipKit Max32 microcontroller to stop the car for a set period of time.

Differences from Last Year's Car

Some differences can be found in this car's design compared to last year.

Category	Last Year's Car	This Year's Car
Chasis	Redcat Monster Truggy	1/10 TT-01E
Sensor	About 2'' x 3''	Smaller 1'' x 1''
PCB Regulators	No	Yes
PCB Motor Control	No	Yes
Switches	No	Yes
Wireless Switch	No	Yes
Breakoutboard	No	Yes
Mode Switches	No	Yes
$100 Servo	No	Yes
Battery	NiMH 50 5000mAH	LiPo 50C 5000mAH

PCB Designs (For IEEE Grand PriEEE)

Sensor Systems **Breakout Board for Chipkit Max32**

Sensor Systems (Designed by James Tang)

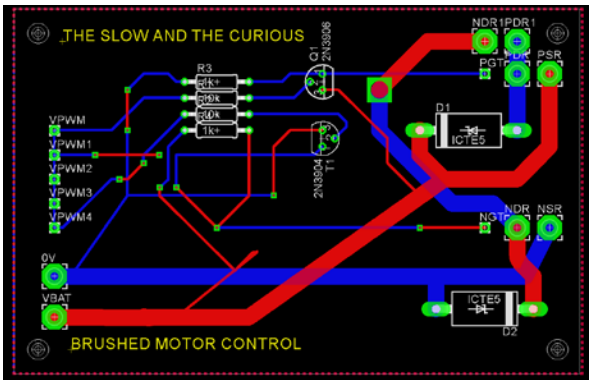
The sensor system is small in comparison to what we did for the previous year. There is a cut in the middle of the center board where the inductors and capacitors will be placed across. In retrospect, it would be much better to have the capacitor not across the inductor but somewhere else in the board area. Also the placement of the sensor output, VDD, and GND wires are very awkward. In retrospect, it would be better to place them all on the back of the sensor where they can be easily replaced. Also, the size of the chip is too small to insert a socket which also makes the design very susceptible to vibrations caused by crashes since it nudges the potetiometer turns. Days before the competition, we solved this problem by using a hot glue to keep the potentiometer and sensor stay in place. However, in retrospect, it would have been better to use a trim potentiometers since they are very sturdy and not susceptible to nudges due to vibrations.

Breakoutboard

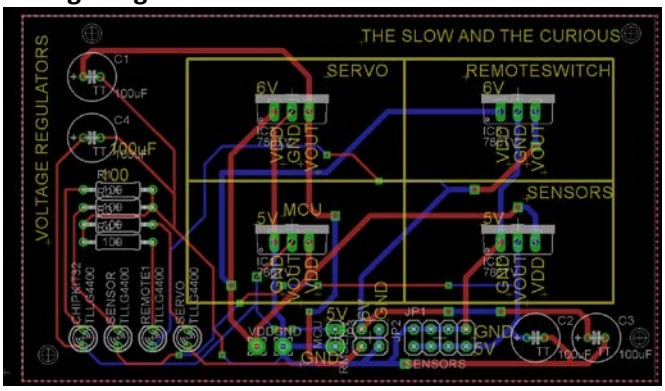
The breakout board is made by taking the schematic from Digilent Chipkit Max32 and modifying its pinouts. The breakout board is made for the purpose of improving the reliability of the system. The idea is to create a solid connection between the pins by soldering the wires from the other systems on top of the board. However, this design also has its flaws since it requires that the wires soldered on top of the

breakout board be far apart from each other to prevent shorting between the pins. The better design would have been to create a similar pinouts extending from both sides of the pinouts to serve as a place to solder and connect the trace to each other. Also the breakout board does not get flushed in due to the DC jack so we had to cut part of the front to flush it in the controller.

Brushed Motor Control



Voltage Regulators



Brushed Motor Control (Designed by Ferdinand Soedharto)

The brushed motor control is integrated with the gate drivers, and features a thicker trace than the sensor design. This is necessary since the motor control will have a large amount of current running through the trace. It also has a power and ground planes. The only thing that this design lacks is a place to put the capacitor across the motor to minimize the noise. However, based on our testing it would seem that it was no problem at all since we placed the sensors far from the car. Also the placement of the motor connectors at the top right corner makes the placement of the MOSFET very awkward.

Voltage Regulators (Designed by Ferdinand Soedharto)

The design of the voltage regulators are very good, there were little problems when soldering the components into the board. The only problem being the distance of the VDD and GND, which makes it impossible to insert a clamper (at least for the ones that we bought) to put larger wires to the regulator. Also the LEDs serve as an indicator as to whether the circuit is powered up or not. They do waste a lot of power but this makes the car look very shiny.

Other Problems

-Lack of switches in the PCB design.

-Lack of 3.3 V regulator and motor inverter. We had to perfboard the switch and regulators.

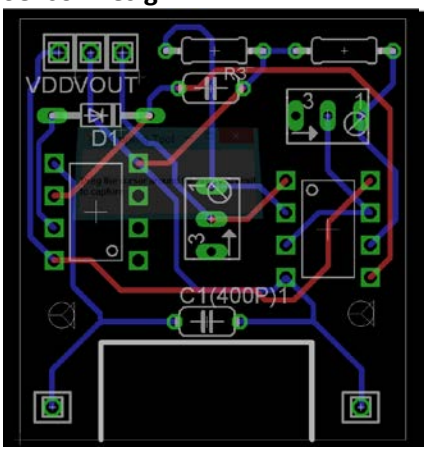
-No PCBs for Arduino Nano. We had to breadboard the Arduino Nano.

Remodeled PCBs (For NatCar)

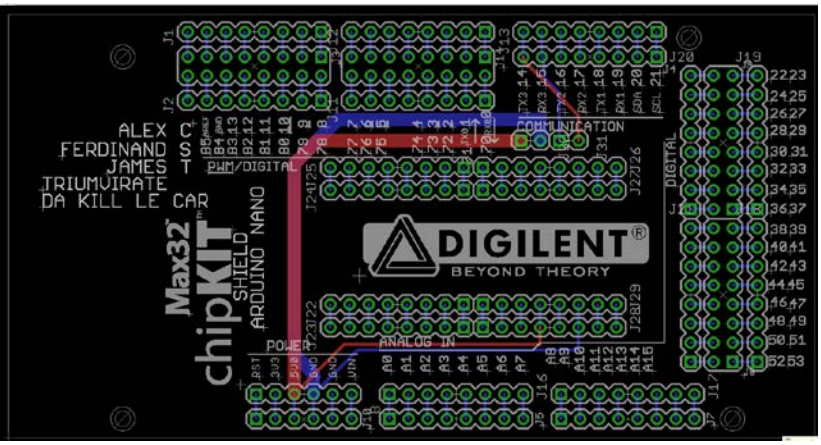
We have determined the problem for the designs and therefore we modified the designs slightly based on the errors that we recognized in our previous design. Unfortunately, due to time constraints, the

PCBs cannot be fabricated on time for the race. However, if we had more time, these are the designs that will be used.

Sensor Design



Breakout board incorporating Arduino Nano and Bluetooth



Sensor System (Modified by Ferdinand)

Changes from previous design:

- Reoriented the sensor potentiometer to fit the op-amp socket and the trim pots.
- Moved the capacitor across the inductors so it can be placed without bending the leads.
- Rerouted the wires VDD, output, and GND to an easy to solder and replace location.
- Removed the 10k ohm resistor from the layout.

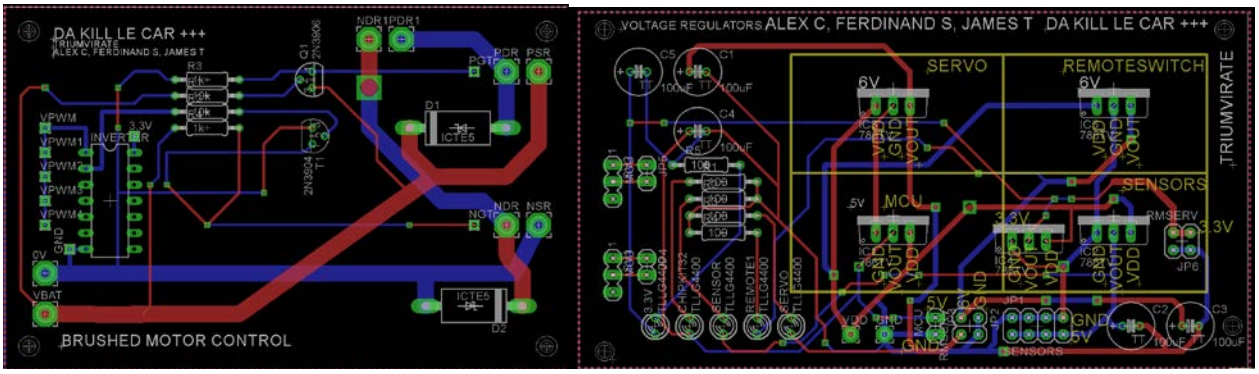
Breakout Board (Modified by Ferdinand)

Changes from previous design:

- Slightly larger and longer area.
- Added pins on the sides of the microcontroller for easier soldering.
- Added a slot for the Arduino Nano in the center of the board wired to the Arduino power pins.
- Added a slot for a Bluetooth wireless module wired to the power pin of the Arduino board.

Brushed Motor Control

Voltage Regulators



Brushed Motor Control

Changes from previous design:

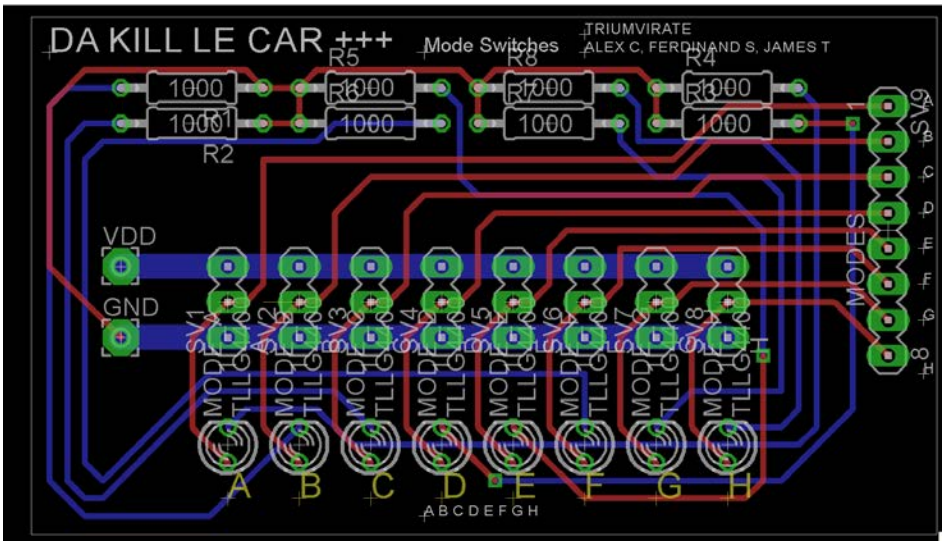
- Rerouted the pull-up logic.
- Moved the location of the wires to the motor.
- Incorporated an inverter to the design of the PCB.

Voltage Regulators

Changes from previous design:

- Added a slot for the 3.3V regulator.
- Added a slot for the switches.
- Added one more LED status for the 3.3V regulator.
- Spaced the VDD and GND pads apart for inserting a clamping device for larger wire use.

Mode Switches



This is strictly for NatCar, this mode switches will be used to toggle the speeds of the car rather than uploading. Depending on which switches are on, this will change the speed mode of the car when the

car is reset and greatly simplifies the race time. LEDs are shown which will indicate which switches are currently on. We can also use this for debugging purposes. However, as we could not fabricate the PCBs, we resorted to using a perfboard which will also do the same thing. We believed that this design would have an impact on the aesthetics of the car and also saves more space on top of the car platform.

Timeline of the Project

Task	Start	Duration (Days)	End
Research Servo	7/1/2013	5	7/6/2013
Research Sensor	11/1/2013	30	12/1/2013
Research Motor Control	11/5/2013	20	11/25/2013
Testing Sensor	11/12/2013	7	11/19/2013
Testing Motor	1/14/2014	2	1/16/2014
Testing Servo	1/14/2014	5	1/19/2014
Testing Wireless Switch	2/5/2014	20	2/25/2014
Testing Microcontroller	1/14/2014	10	1/24/2014
PCB Making	1/29/2014	5	2/3/2014
Build Prototype Car	2/25/2014	10	3/7/2014
Assembling Car	3/29/2014	4	4/3/2014

Budgeting

Parts	Vendor	Quantity	Price	Q*P
Schottky Diodes	Mouser	10	0.29	2.9
Trim Pot 50k	Mouser	20	0.778	15.56
10mH Inductor	Mouser	15	1.13	16.95
Power NMOSFET	Mouser	6	2.6	15.6
Power PMOSFET	Mouser	6	4.59	27.54
6 V Voltage Regulation	Mouser	10	0.445	4.45
5 V Voltage Regulation	Mouser	5	0.69	3.45
Schottky Diodes (H-Bridge)	Mouser	12	0.642	7.7
TT-01E Chasis 1/10 Scale	Tower Hobbies	1	189.96	189.96
Servo 1/10 Scale	San Diego RC	1	85	85
Brushless 1/10 Scale 17.5T Motor	A Main Hobbies	1	77.49	77.49
5000mAH Gens Ace 50C LiPo	HobbyPartz	1	47.5	47.5
Wireless Remote & Receiver	San Diego RC	1	100	100
Acryllic Plexiglass	Ridout Plastics	1	10	10
Grand Total				604.1

Conclusion

The car successfully worked to the speed of 6.1 ft/sec during the IEEE Grand PriEEE race, and we will also participate in the NatCar competition at UC Davis this May 24th, 2014. In retrospect, however, we believed that it would have been better if we used the line scan camera approach since it eliminates the need for using bulky equipment and can be tested anywhere. Also, smaller cars would have been better since there will be less consumption of power. Nevertheless, our team successfully finished the track even though there are some things that we were not able to do such as the implementation of learning algorithm, brushless motor, gyroscope, and wireless feedback.