

Final Project Report

E3390 Electronic Circuits Design Lab

The Seeing Natcar

Peter Fredrickson
Federico Garcia
Antonio Gellineau
Steven Mon

Submitted in partial fulfillment of the requirements for the
Bachelor of Science Degree

December 16th , 2008

Department of Electrical Engineering
Columbia University

Table of Contents

- 1) Executive Summary
- 2) Block Diagram, Design Targets, Specifications
- 3) Individual Block Descriptions
- 4) Bill of Materials
- 5) Health, Safety, and Environmental Issues
- 6) Final Gantt Chart
- 7) Assessment of the Course
- 8) Appendix

1. Executive Summary

The aim of the project is to build an autonomous car that can be entered into the NATCAR competition held in University of California – Davis. The challenge of the competition is to create a car capable of racing a track autonomously. The track is comprised of a wire with an AC signal beneath white tape on a dull carpet. The car can use any means within budget to follow the track as quickly as possible without going out of bounds.

Most teams employ inductors in order to track the AC signal in the wire. This design works well as small variations in the signal employ good feedback towards controlling the car. However, for our project we wanted to attempt a different and interesting approach that had the potential to improve on this design. Our goal was to create a “Seeing NATCAR”; a car that would track the difference in the color of the tape versus the carpet.

We accomplish this by using IR photodiodes and photosensitive BJTs in a reflective package in order to “see” the track. This is then analyzed by the A/D converters of the microprocessor which sends control signals to the circuitry which powers the car.

The final phase of this project would be to produce a car that is competitive at UC Davis.

2. Block Diagram, Design Targets, Specifications

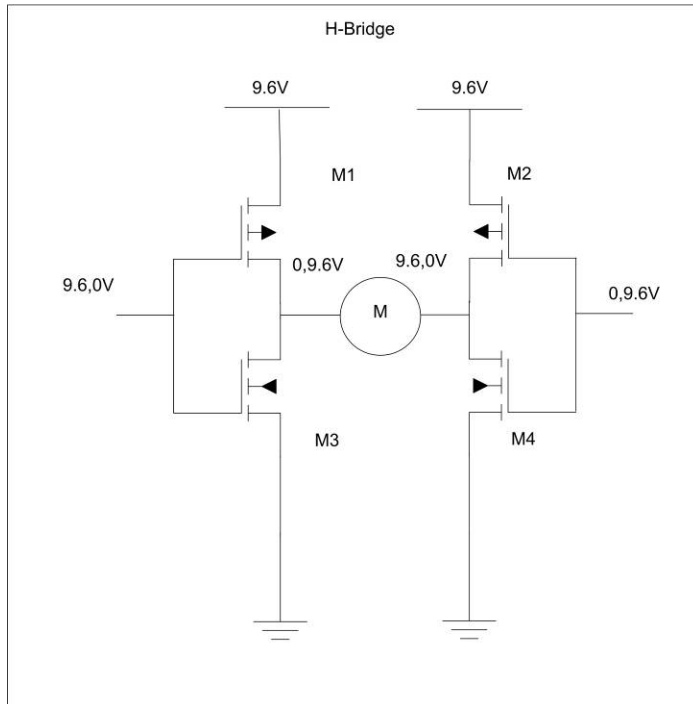


Figure 1 - H Bridge

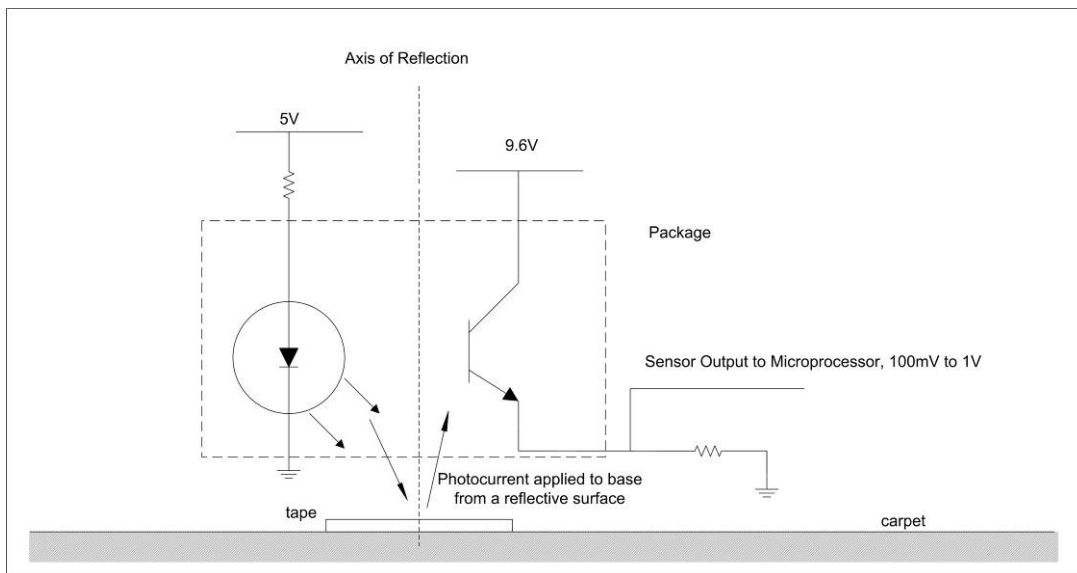


Figure 2 - Sensor Circuit

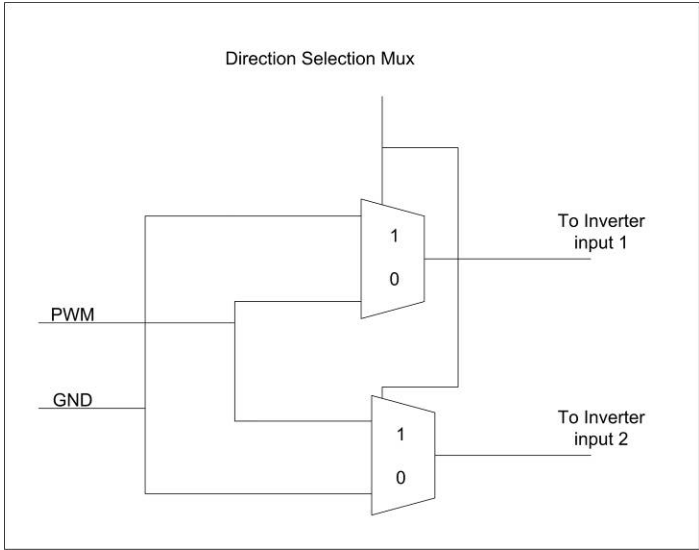


Figure 3 - MUX Logic (Direction Selection-Left/Right)

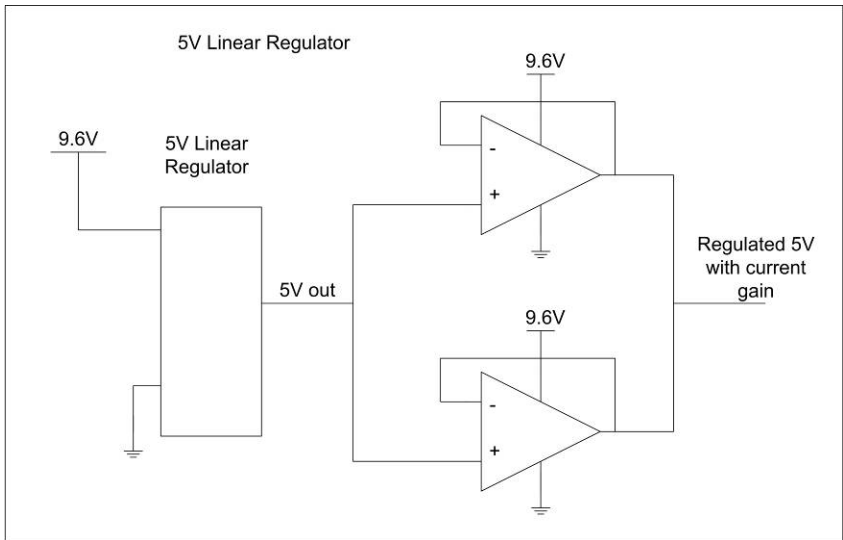


Figure 4 - 5V Voltage Regulator

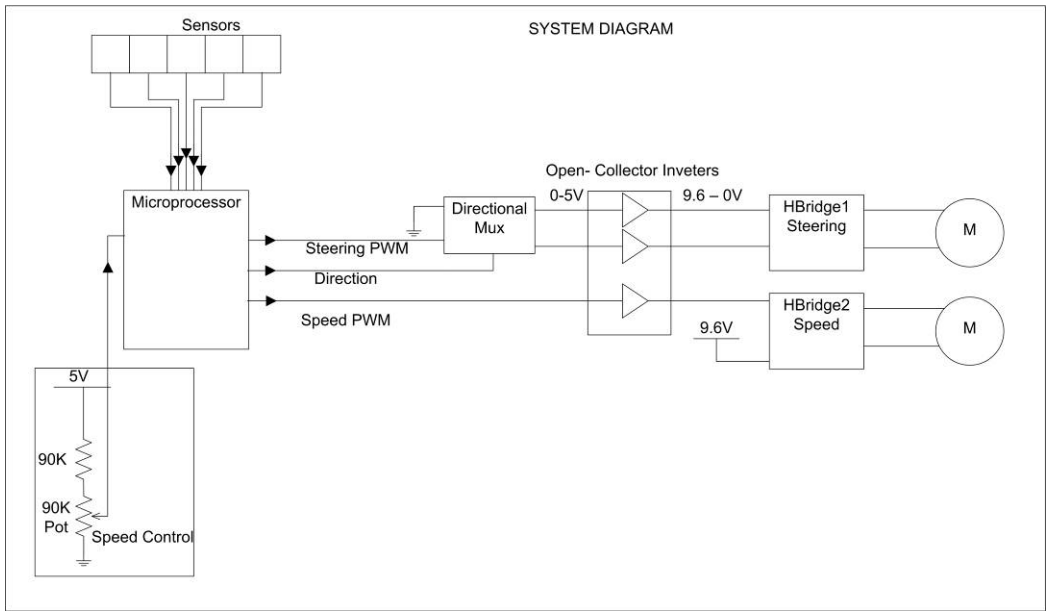


Figure 5 - Full Block Diagram

3. Individual Block Descriptions

i. Microcontroller

The microcontroller serves as the primary unit to take in all the sensor signals and output all control signals. For a variety of reasons, the specific chip chosen for this project is the PIC16F74 from MicroChip Inc. Free samples were obtained from MicroChip and the software and hardware necessary for programming was already available in the Mechatronics Lab. The microcontroller has a wide operating range and can tolerate input voltages from our DC Converter Circuit between 2.5V and 5.5V. It also runs at low power consumption, typically less than 2mA when at a 5V supply. The chip has 33 digital input/output pins available to use, as well as the ability to use 8-bit analog-to-digital conversion and output a Pulse Width Modulated signal.

We configured a full 8-pin port of the microcontroller to handle the analog inputs coming from the photosensors. Since we were handling low voltage inputs, one of the eight pins was set to the minimum reference voltage of 2.5V. With the minimum V_{Ref} of 2.5V and digital conversion values from 0 to 255, we had a digital resolution of approximately 10mV.

Before programming the microcontroller to manage the car in its entirety, we first designed two test programs. For the first one, we built a circuit and wrote a simple program to test the analog-to-digital conversion. The circuit successfully converted any of the chosen seven inputs and displayed the conversion on 8 output pins indicated with LEDs. Secondly, we verified that the two PWM output pins functioned at operating voltages and frequencies that would work for our control logic. With these checks complete, the PIC16F74 was ready to be programmed to interface with the full car.

The program to run the car went through several revisions during testing and experimentation with each revision adding some new functionality to the microcontroller. In the primary set up, the microcontroller sweeps through the five inputs from the photosensors and stores each value. It then uses conditional logic to compare the values and see which sensor had the highest response, and therefore would have been on top of the track. This

is then used to select a direction to go; either straight ahead, slightly left/right, or sharp left/right. Each direction has an associated duty cycle for speed, steering, and if necessary for turning, a digital output to control left/right. It was also programmed to reduce the speed while turning. This helped increase control and reduce the chances of running off the track.

Two significant improvements were made for use in the final code. The first one involved implementing variable speed control with the use of an analog signal controlled by a potentiometer. The potentiometer uses one of the remaining two analog-to-digital pins and varies the voltage from 0V up to the reference at 2.5V. Rather than hard-coding the speed duty cycle, the value obtained from this pin is used as a controllable way for us to adjust the speed at any time. The speed is still programmed to decrease by 80% and 60% when making gentle and sharp turns, respectively.

The second adjustment used the one remaining analog pin and another potentiometer to again vary a voltage from 0V to 2.5V. This value was then used as a baseline threshold voltage that the inputs from the photosensors needed to pass. Before the car would make any adjustments, the threshold ensured it was recognizing the track itself and not just noise. This greatly improved the function of the car. Previously, if the vehicle did not turn sharply enough and went slightly past the track, it would move unpredictably in whichever sensor happened to be outputting the marginally greater signal. With this threshold, none of that noise is enough to trigger the microcontroller to change direction and it continues to turn in the direction it was attempting to return to the track. Once it passes over the track again, the sensor values will be large enough to begin triggering the controls and readjust itself on the track.

There still exist several possibilities for improvement in the programming. One feasible idea is to implement a timer variable that increments every time the microcontroller sweeps through and calculates all the analog-to-digital conversions. It would be reset whenever it detected a photosensor that surpassed the threshold to detect the track. However, if it made multiple sweeps without detecting any sensor above the threshold, the program could assume it has lost the track and go into a correction mode. At this point, we could either stop driving the car as a failsafe, or perhaps

even implement a sequence where it goes in reverse to try and return to the track. Our wiring of the car did not make use of any traveling in reverse because our primary concern was to follow a track quickly and at relatively competitive speeds. However, it would not be difficult to add this functionality and help the program more rigorously account for challenges that could come with extremely difficult tracks.

The code implemented in our final car is included in the Appendix.

ii. Directional Control Multiplexer

This functional block switches the inputs to the H-Bridge in order to change the direction of the DC motor which controls the steering of the car. One input is the Pulse Width Modulated signal which controls the amount of Power applied to the steering and the other is the input to the multiplexer which decides whether to switch the signal or not.

iii. Photosensors

Initially, the sensor design used five individual IR photoemitters and five individual photodetectors. However, it soon became apparent that they would not be suitable due to inherent variations in angle, manufacture and position. Therefore, we decided to replace them with the EE-SF5-B reflective photomicrosensors. These were chosen since they solved the issue of non-uniformity and that they contained a filter to help reduce interference from external light sources.

The IR LED of the EE-SF5-B emits IR light at an angle which is either reflected by the tape or absorbed by the dull carpet. The other side of the photosensors is an open base BJT transistor. The photocurrent turns on the BJT channel which creates a current to flow through the Resistor connected to the emitter. The Voltage created by these sensors is then read by the A/D convertors of the Microprocessor. The response of these sensors was 100mV to 1V on average. The resistance values were chosen from the included datasheet to balance response time versus the product of the current flowing through the BJT and the resistance in its emitter.

Also, the photosensors were mounted on an adjustable housing (Appendix Figure 1) allowed us to test the maximum distance between the sensors and the steering.

iv. H-Bridge

Controlling the car was easily the greatest challenge. The first design tried to control the car using switching relays and optical isolators but relays were not able to provide adequate switching speeds.

The second design employed H-Bridges as was recommended by the project's advisors. Two H-bridges were bought from Digikey (LMD18200T-ND). These were perfect for controlling DC motors but the voltage from the battery was simply not enough to adequately power the device. As a result, we had to abandon the chip.

Instead we constructed two H-bridges and the required control circuitry, which includes the **directional control multiplexor** and the **open collector inverters**, which are detailed in this section.

The H-Bridge used in this project was constructed from regular power MOSFETs in an inverting configuration and the inputs were buffered from TTL levels to the 9.6V and 0V rails in order to make sure the power FETs were in cut-off and current could only be conducted across a DC Motor.

v. Voltage Regulator

The car needed a voltage regulator since the car's battery had a nominal voltage output of 9.6V, but needs a 5V supply for the microcontroller and associated logic. Therefore, we used the LM723CN which was readily available in our projects lab. However, the regulator is not designed to power loads, so it could not directly provide current to the IR LED's, microprocessor and multiplexer. We therefore added a voltage follower implemented by a power operational amplifier. To ensure enough current was supplied for all of our control logic, we used two power operational amplifiers, both supplied directly from the battery.

vi. Speed Control

In our initial tests we determined that the speed of the car was too fast when the battery was fully charged for our sensors to detect the track while not very responsive while the battery was low. Instead of hard-coding a speed into the logic of the car a speed control was implemented.

Therefore, in our design is a potentiometer which controls the speed of the car. The car selects its speed proportional to the digital conversion of the analog voltage value across the potentiometer. It is set up to ensure that no more than 2.5V sets the value of the car as 2.5V is the max value of the A/D converter.

4. Bill of Materials

Part Name	Manufacturer/ Provider	Part Number	QTY	Unit Price	Final Price
Photodiodes	DigiKey	160-1032-ND	10	\$0.388	\$3.88
IR LED	DigiKey	1N6266-ND	10	\$1.32	\$13.20
PhotoMicrosensor	DigiKey	OR503-ND	15	\$5.21	\$78.15
Oscillators	DigiKey	535-9182-5-ND	5	\$0.73	\$3.64
H Bridge	DigiKey	LMD18200T-ND	2	\$14.14	\$28.28
Linear Regulator	Linear Technology	LT3008	2	\$1.71	\$3.42
Negative Linear Regulator	Linear Technology	LT1185	2	\$4.50	\$9.00
4-Pack AA Batteries for remote car testing	Energizer	NH15-2500	2	\$13.99	\$27.98
RC Cadillac Electric Car	Hobbytron	AA-69003-XQ031	2	\$31.50	\$63.00
Total Cost:					\$230.55

*Note: Shipping and tax not included in bill total

5. Health, Safety, and Environmental

a. Product Dangers

This product does not pose any danger if operated properly. However, the user should be aware that the power MOSFETs, if not biased properly, can have shoot through current which causes the MOSFETs to draw large currents and heat up the package to extreme temperatures. The car is properly biased so this does not happen, but if wires are reconnected incorrectly this scenario is possible. Therefore, one should always check to make the MOSFETs are connected properly to the battery, and even then avoid touching the MOSFETs shortly after operation or confining them without adequate cooling/ventilation.

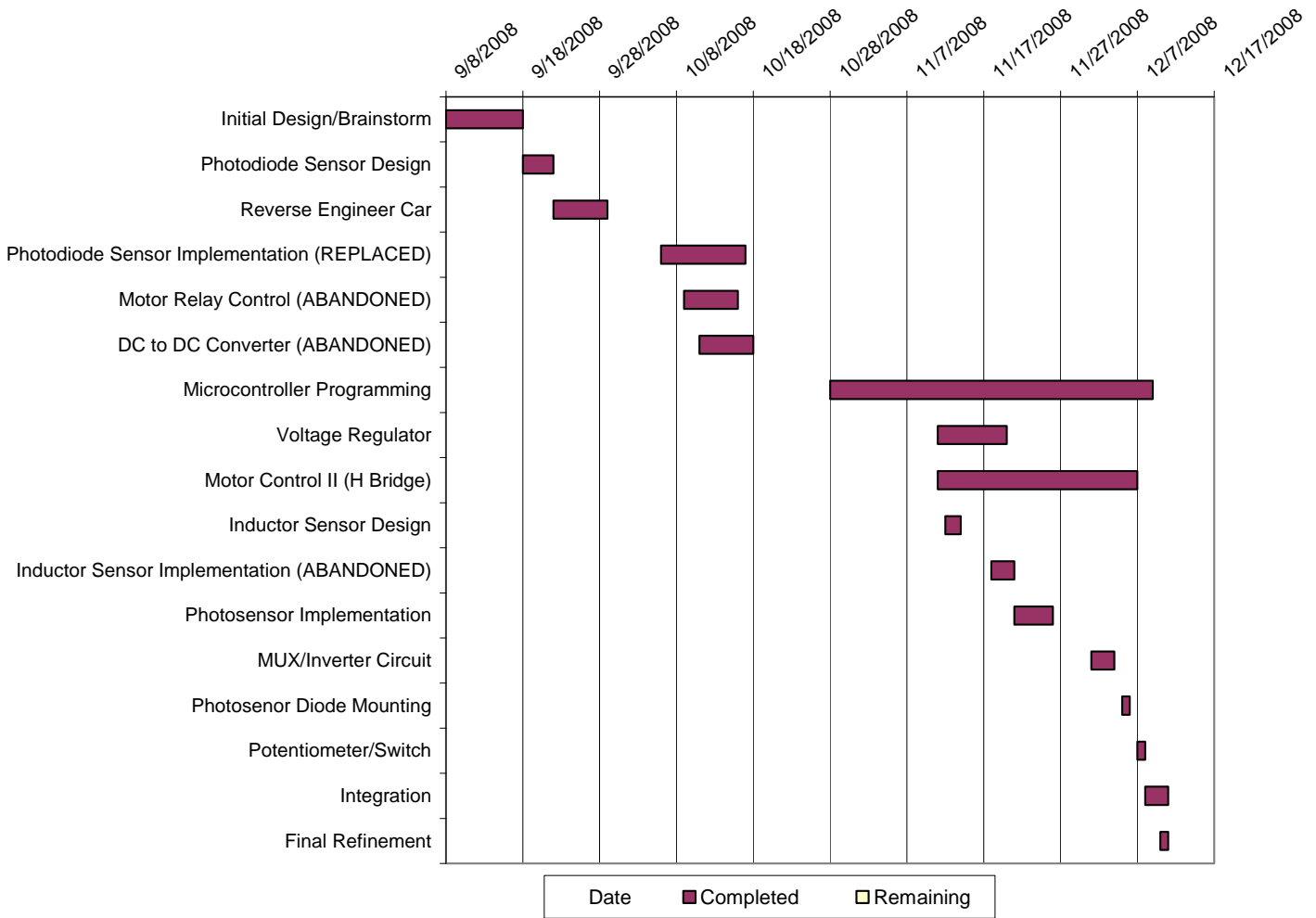
b. Health Hazards

In normal operation, only notable health hazard would take place if someone were to recklessly operate the car in an uncontrolled testing area. If the car was set to maximum speed and ran into an unaware individual, it may cause some very minor injuries. This hazard is easily managed by making sure to operate the car in a controlled track environment with a barrier between the area of operation and any unaware bystanders.

c. Environmental Hazards

- i. This product conforms to FCC standards
- ii. There is a potential shock danger if the wiring is tampered with or the battery/battery charger is disassembled.

6. Final Gantt Chart



7. Assessment of the Course

Our group believes the main and critical component of a course such as this one is good advising. This was handled well and whenever our group ran into problems with our design or implementation, the assistance and advice we received greatly helped the project to succeed.

With this in mind, the only shortcomings of this course would be the space and equipment. The space was an issue since there were probably seven or so groups trying to use the same projects lab.

Equipment was an issue because it was very hard to find parts without assistance. Also, one of the main difficulties we encountered came from a faulty breadboard. The tape backing was removed to connect the board to platform of the car but it could not hold up to connections. (See Figure 2)

Since new space has been allocated to Senior Design, we are confident that our criticisms have been acknowledged and that work is being done to improve the course.

8. Appendix

Microcontroller Code

```
#include<pic.h>
    __CONFIG(11111110110010);
/*****
Natcar5.c
```

This program uses the PIC16F74 Microcontroller. It takes in 5 analog signals from photodiode sensors and takes the AtoD conversion of each. By finding the highest value of those signals, it decides a direction to turn if any, and a speed to drive the car at. The steering is controlled by a PWM wave output at 19.53kHz as well as a single digital signal to control the direction. The driving is controlled by the second PWM wave, also at 19.53kHz with an independently controllable duty cycle.

Port C bits will be defined as follows:

- 0-Digital Output-Controls the direction of the steering
- 1-CCP2:PWM Output to control steering
- 2-CCP1:PWM Output to control driving speed

The analog signals will be defined as follows:

- AN0(Pin2)-Signal from far left sensor
- AN1(Pin3)-Signal from close left sensor
- AN2(Pin4)-Signal from center sensor
- AN3/Vref(Pin5)-Reference voltage set to 2.5V
- AN4(Pin7)-Signal from close right sensor
- AN5(Pin8)-Signal from far right sensor
- AN6(Pin9)-Signal from a potentiometer to control speed
- AN7(Pin10)-Signal from a potentiometer to control ambient threshold

```
*****/
```

```
/*Variable declarations */
```

```
#define PORTBIT(adr,bit)      ((unsigned) (&adr)*8+(bit))    //Gives descriptive
                                                                    //names of bits
```

```
static bit   Direc          @ PORTBIT(PORTC,0);           // Map bit 0 of PORTC->
                                                                    // control direction
```

```
char Temp;                // Variable for AtoD delay loop
char L2;                  // Variable for far left sensor
char L1;                  // Variable for close left sensor
char Center;             // Variable for center sensor
char R1;                  // Variable for close right sensor
char R2;                  // Variable for far right sensor
unsigned char Pot;        // Variable for Potentiometer
unsigned char Amb;        // Variable for ambient threshold
```



```

void SetupDelay (void) // Delay Loop for AtoD Aquisition
{
    for (Temp=1; Temp > 0; Temp--) {} // ~17 us delay
}

void AcquireAtoD (void)
{
    ADCON0=0b01000001; // select 8* oscillator, analog input 0,
                        // turn on

    SetupDelay();
    ADGO = 1; // Start A/D
    while (ADGO== 1) {} // Waiting for conversion to finish
    L2 = ADRES; // Store value for far left sensor

    ADCON0=0b01001001; // select 8* oscillator, analog input 1,
                        // turn on

    SetupDelay();
    ADGO = 1; // Start A/D
    while (ADGO== 1) {} // Waiting for conversion to finish
    L1 = ADRES; // Store value for close left sensor

    ADCON0=0b01010001; // select 8* oscillator, analog input 2,
                        // turn on

    SetupDelay();
    ADGO = 1; // Start A/D
    while (ADGO== 1) {} // Waiting for conversion to finish
    Center = ADRES; // Store value for center sensor

    ADCON0=0b01100001; // select 8* oscillator, analog input 4,
                        // turn on

    SetupDelay();
    ADGO = 1; // Start A/D
    while (ADGO== 1) {} // Waiting for conversion to finish
    R1 = ADRES; // Store value for close right sensor

    ADCON0=0b01101001; // select 8* oscillator, analog input 5,
                        // turn on

    SetupDelay();
    ADGO = 1; // Start A/D
    while (ADGO== 1) {} // Waiting for conversion to finish
    R2 = ADRES; // Store value for far right sensor

    ADCON0=0b01110001; // select 8* oscillator, analog input 6,
                        // turn on

    SetupDelay();
    ADGO = 1; // Start A/D
    while (ADGO== 1) {} // Waiting for conversion to finish
    Pot = ADRES; // Store value for speed control

    ADCON0=0b01111001; // select 8* oscillator, analog input 7,
                        // turn on

    SetupDelay();
    ADGO = 1; // Start A/D
    while (ADGO== 1) {} // Waiting for conversion to finish
    Amb = ADRES; // Store value for ambient threshold
}

```

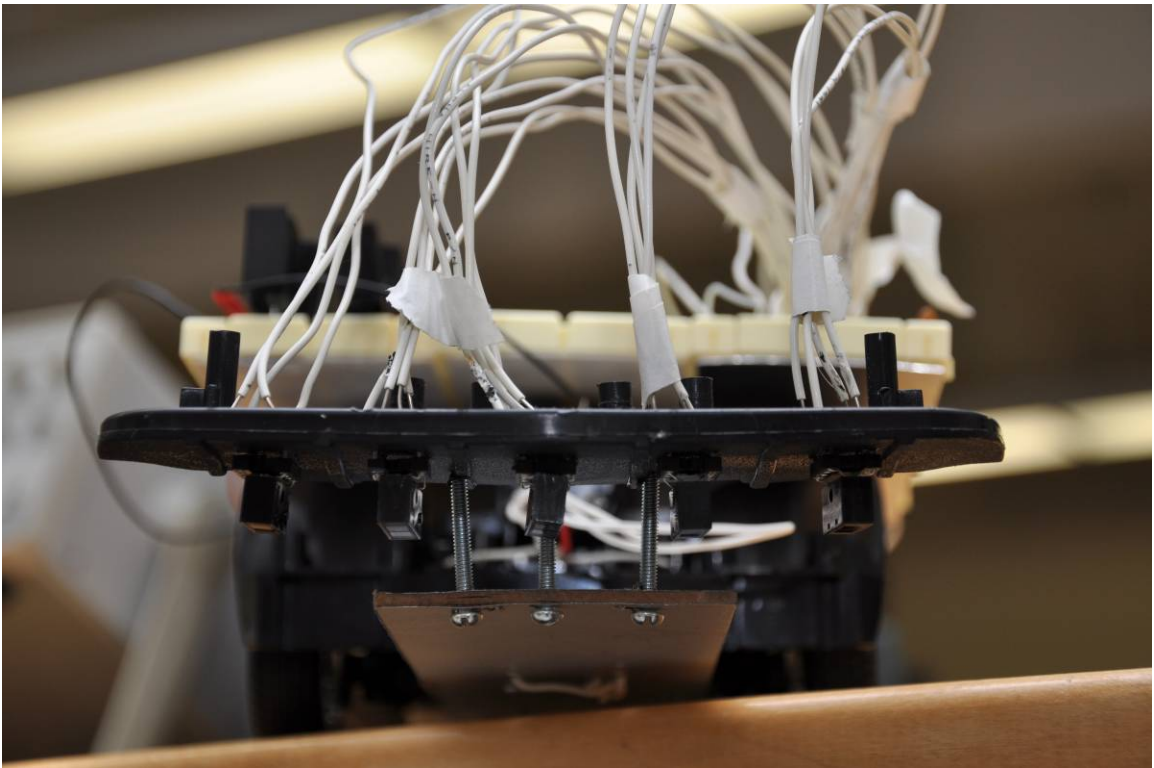
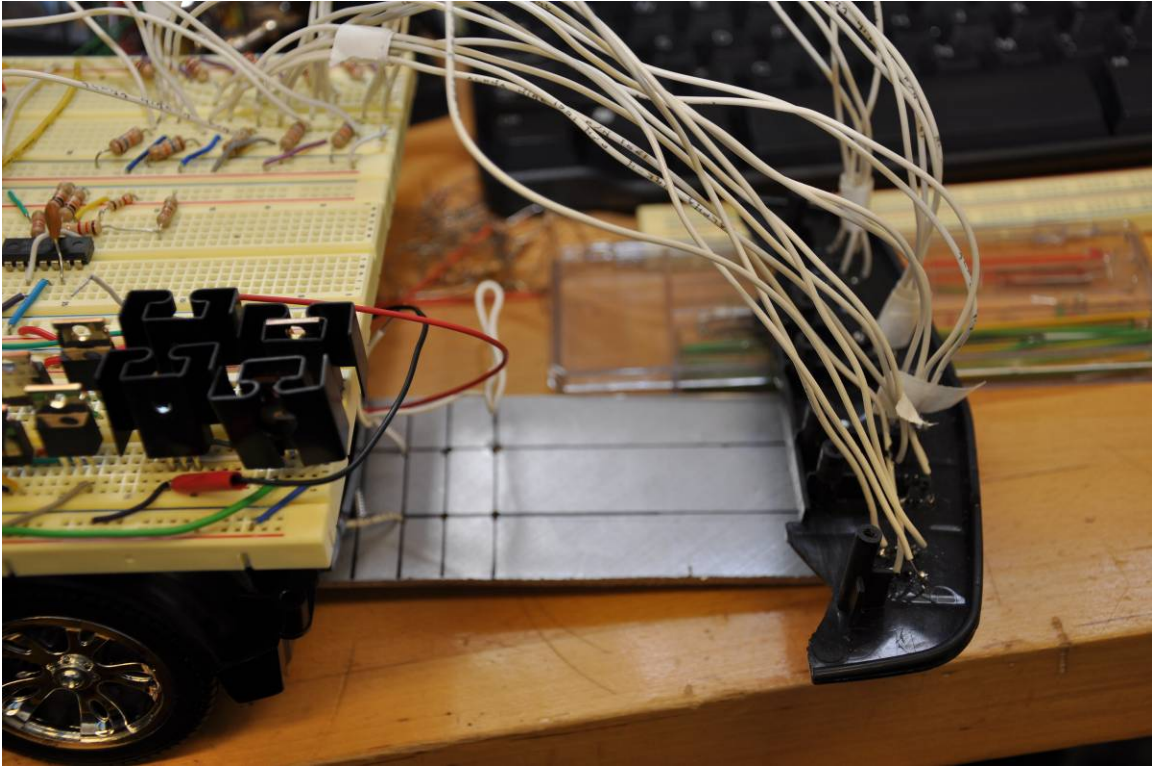



Figure 1: Extension for sensors

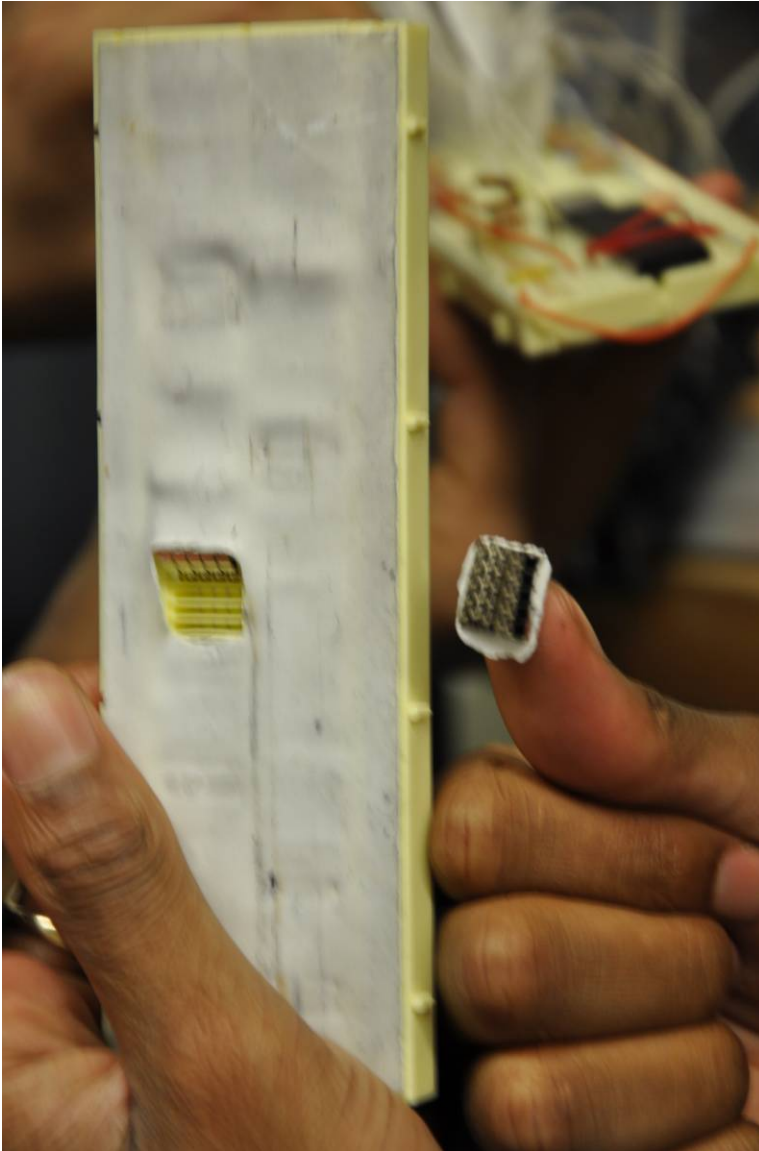


Figure 2: Breadboard without backing

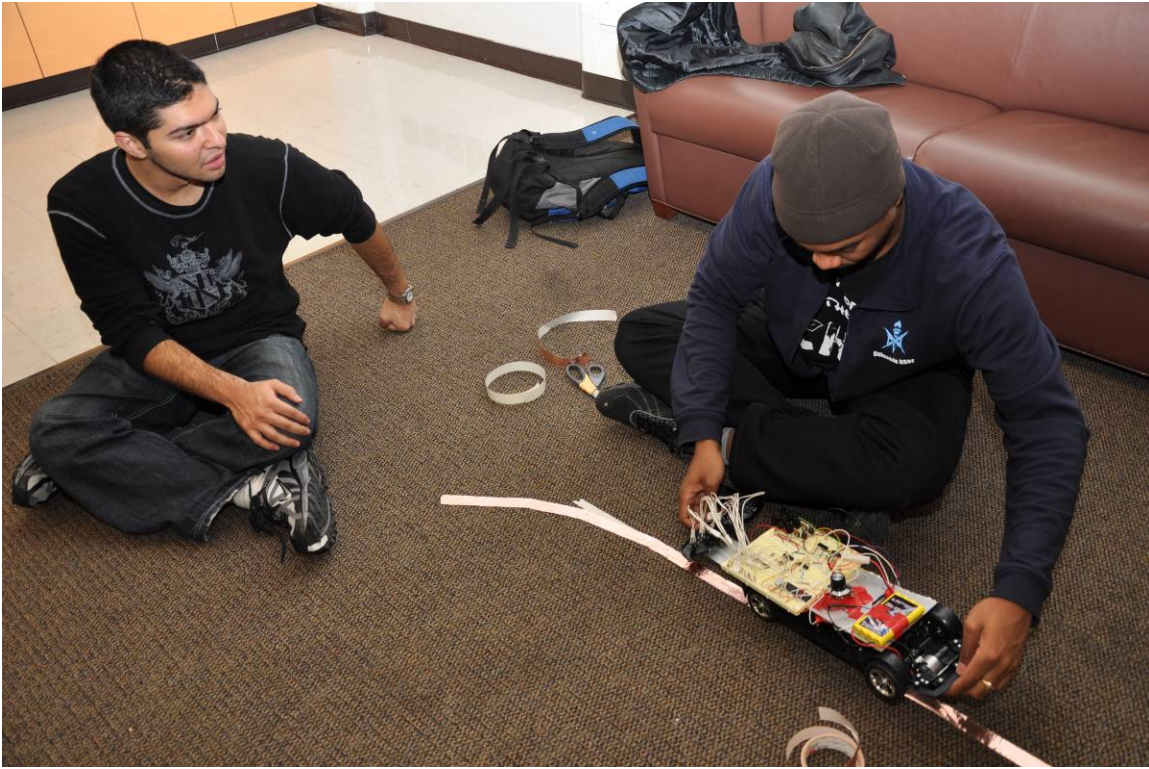


Figure 3: Track Construction

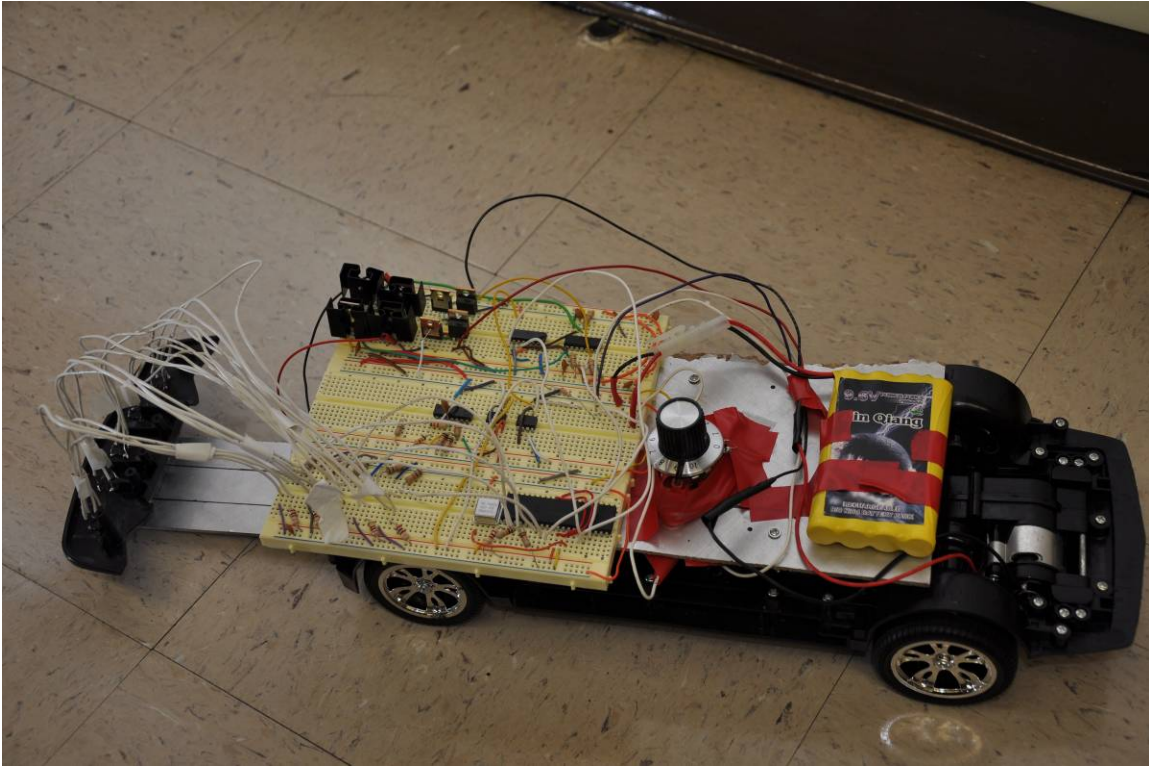


Figure 4: Top View

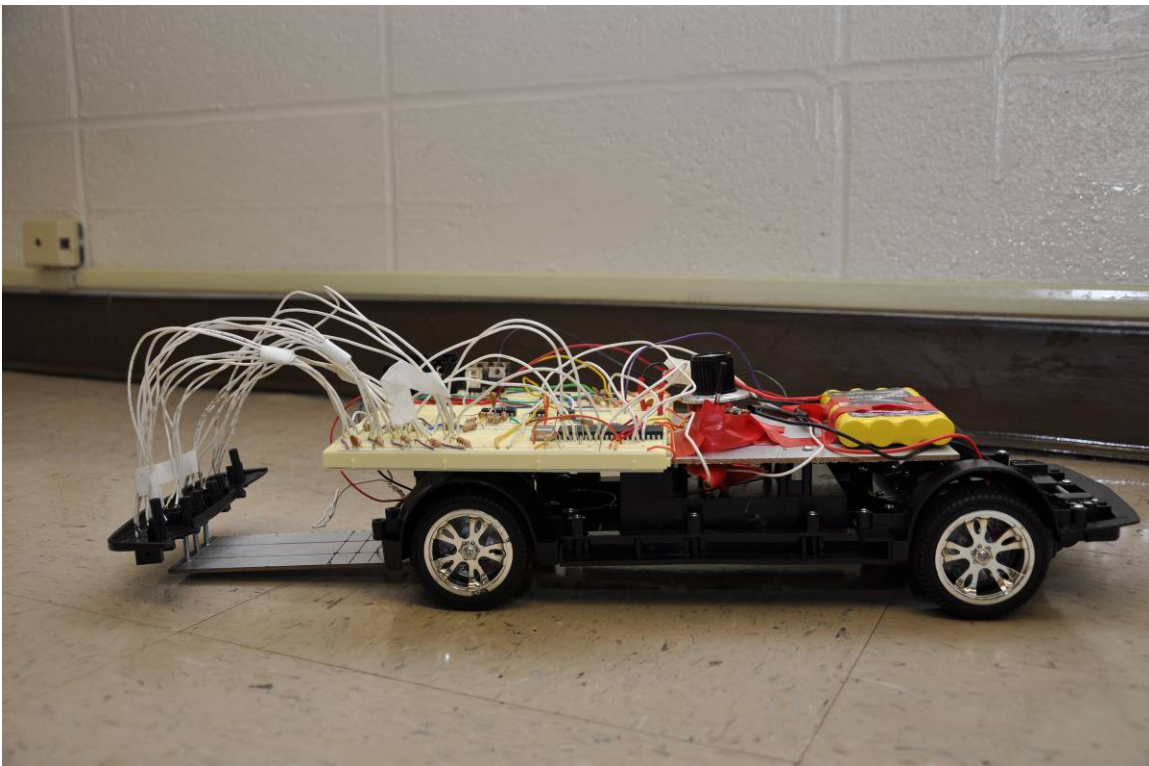


Figure 5: Profile View

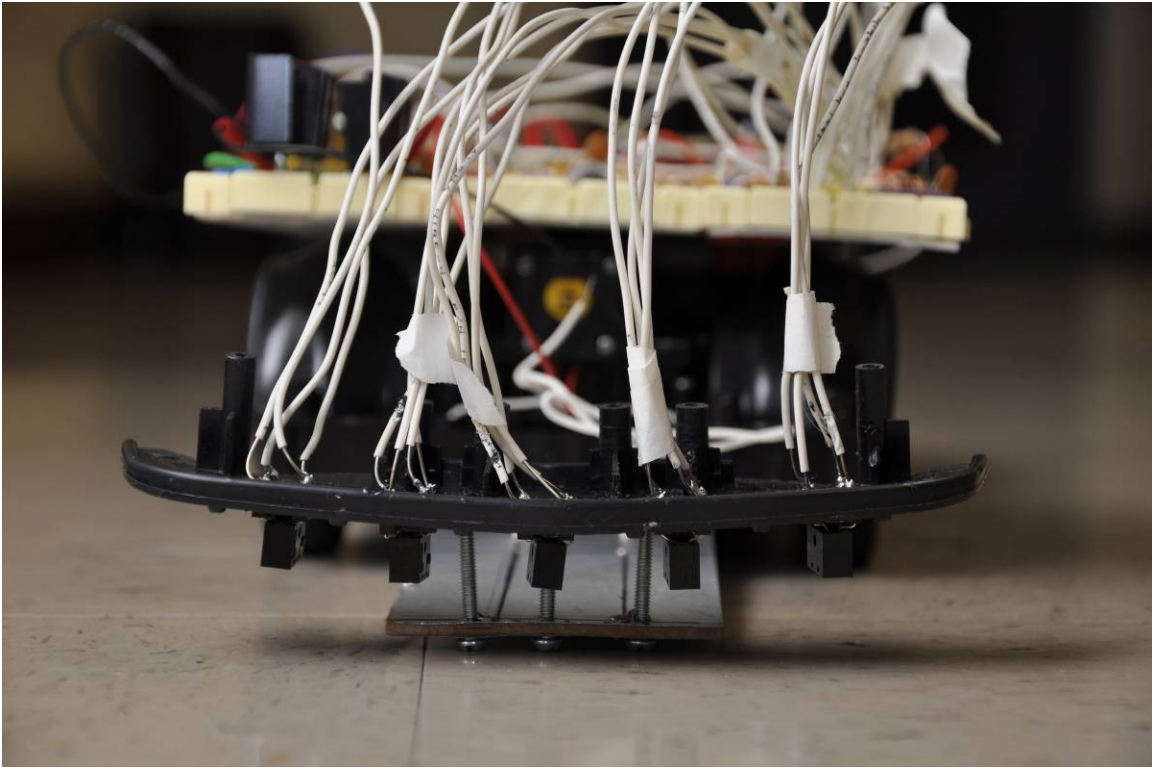


Figure 6: Front View