UNIVERSITY OF CALIFORNIA, IRVINE

Line Following RC Car

Automated Path Follower

EECS 129B – Winter 2008
Professor Klefstad
Kevin Fong

The line following RC car provides a model for the potential implementation of unassisted steering in full sized automotives. In this model, a microcontroller simultaneously polls a set of photocells to determine the output to a stepper motor for differential steering. The photocells themselves do nothing but with the integration of highly contrasting surfaces it is able to discern a path to follow. The microcontroller also controls a rear drive DC motor which outputs pulse width modulated values to propel and control the speed of the vehicle.

Table of Contents

1) Abstract 1
2) List of Illustrations 3
3) Project Description 4
4) System Level Block Diagram 5
5) Circuit Level Block Diagram 6
   - Arduino Pin Mapping 6
   - SN754010NE Pin Mapping 7
   - Sensor Array Pin Mapping 8
6) Software Description 9
7) System Test Plan 14
8) Cost Sheet 15
9) Summary 16

List of Illustrations

Figure 1: Steering mechanism 4
Figure 2: System level block diagram 5
Figure 3: Arduino microcontroller 6
Figure 4: H-bridge circuit 7
Figure 5: Sensor array circuit 8
Figure 6: Software Diagram 9

3) Project Description

The purpose of this project is to provide a primitive model for autonomous steering and drive in cars. From the knowledge I learned last quarter on the programmability of microcontrollers with electronic components this project incorporates many of the points we focused on. The aim of this project is planned and designed for a first time, open ended project and to gain laboratory experience on working on a self managed project. Based around the Arduino microcontroller and components used during the first half of the course I came up with an idea to extend the functionality of the labs into a scenario that could be applied in the real world.

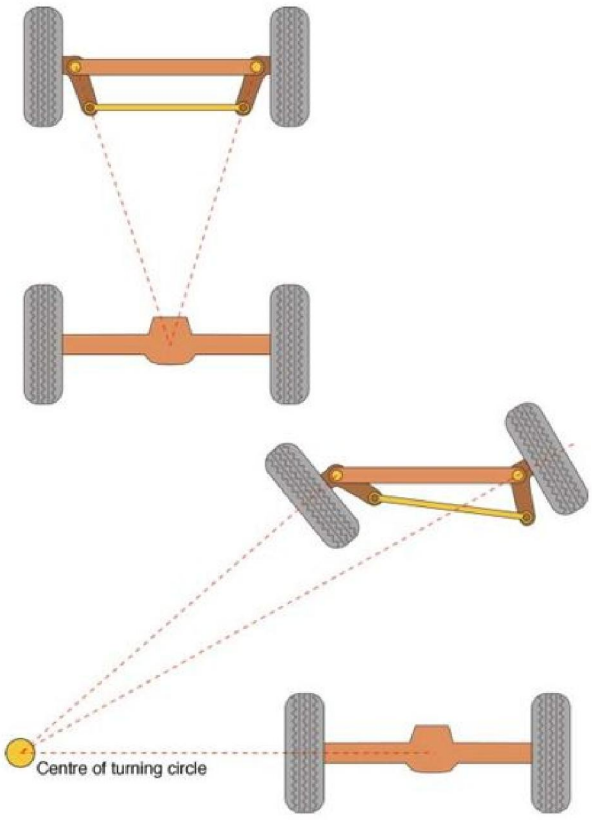

Figure 1: Steering mechanism

The line following RC car contains some elements and physics found in real cars. Four tires driven by a rear wheel drive and a limited turning radius controlled by a front steering axle mimic basic drive and steering functions of real cars shown in Figure 1. While this is a slight disadvantage in the minimal turning radius of the vehicle compared to a two wheeled vehicle, it more closely resembles real world physics of cars.

To automate the steering I will make use of the experience gained from using photocells to control the brightness of LEDs, motor speed, and speaker volume to constructing a comparator circuit. The idea behind a comparator circuit for line following comes from the properties of light. Lighter surfaces reflect more light than dark surfaces and photocells are capable of seeing incoming light intensities and in a sense provide the vehicle to handle rough binary imaging. In conjunction with the photocells, LEDs will be accommodating each one that is implemented to provide a constant source of light for each respective photocell. To eliminate ambient light interference, each photocell and LED pair will be contained and isolated. The resulting values gathered from each photocell should then be fairly constant on any given surface provided that the surface is uniform. Being able to distinguish between light and dark colors is the key of my particular project to establish a way to follow a high contrast line compared to the surface it is on.

To control the steering the vehicle I will use a H-bridge circuit to turn the DC motor counter-clockwise. The orientation of the motor will be in a way that clockwise rotation shifts the steering axle to the right, allowing right turns, and counter-clockwise rotation shifts the steering axle to the left, allowing left turns. To control the speed of the rear DC motor for driving, a transistor must be used to supply a higher

current and voltage than what the microcontroller can supply to provide enough torque for gear rotation. Coupling this with a pulse modulated signal, a discrete, intermediate value within an 8-bit range can throttle down the voltage and current supplied to the rear motor. The variation of speeds resulting from varying the voltage and current can be used with the previously provided fact that the turning radius is fairly large thus minimizing the turning radius variation using low speed propulsion should be implemented.

4) System Level Block Diagram

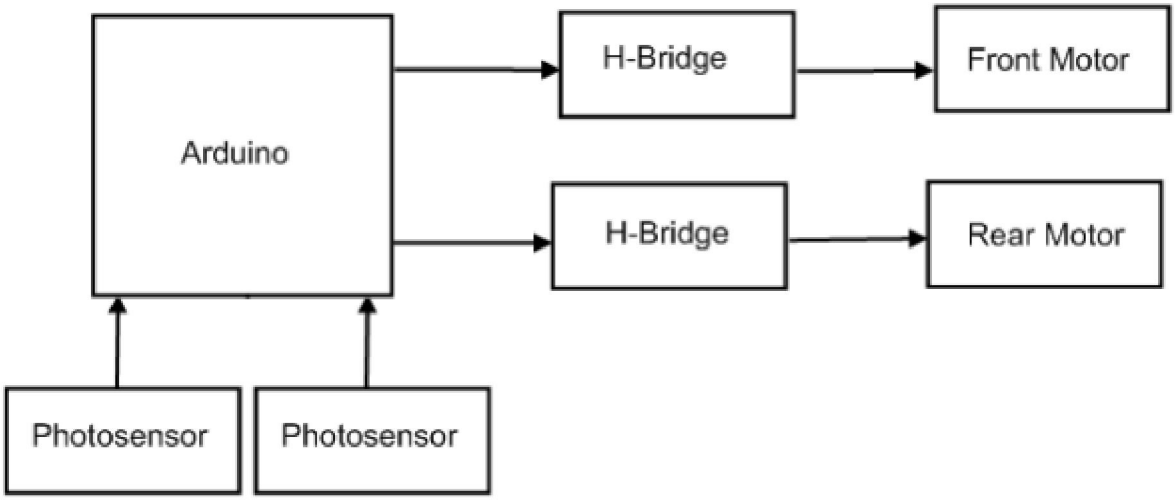

Figure 2: System level block diagram

- Microcontroller simultaneously polls a set of photocells.
- Microcontroller provides a signal to the motors that drive the car's steering and drivetrain.
- Photocells in parallel to a resistance create potentiometers to sense light intensities.
- Values of the light intensities are fed back into microcontroller.
- DC motors provide steering and rear wheel drive for the car based on feedback values from photocells.
- H-bridges aids DC motors for clockwise and counter-clockwise rotation as well as supply ample voltage and current that the microcontroller is unable to provide.
- On the front (steering) motor, the rotation moves set of gears that pivot the steering axle.
- On the rear (drive) motor, the H-bridge is able to control the voltage outputted into the motor to control speed.

5) Circuit Level Block Diagram

Arduino Pin Mapping

- Digital
  - Pin 3: to H-bridge2 leg 1
  - Pin 4: to H-bridge2 leg 2
  - Pin 5: to H-bridge1 leg 1
  - Pin 6: to H-bridge1 leg 2
  - Pin 7: to sensor LEDs
  - Pin 9: to H-bridge2 enable
  - Pin 10: to H-bridge1 enable
  - Pin 11: to left headlamp
  - Pin 12: to right headlamp
- Analog
  - Pin 0: to left potentiometer
  - Pin 1: to middle potentiometer
  - Pin 2: to right potentiometer

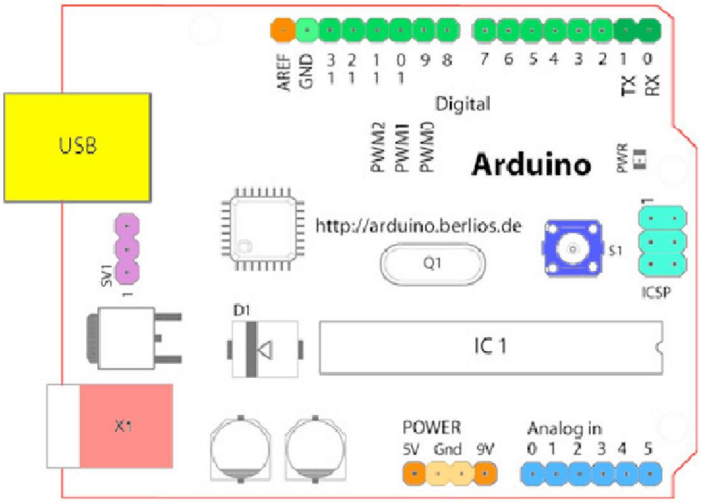

Figure 3: Arduino microcontroller

SN754010NE Pin Mapping

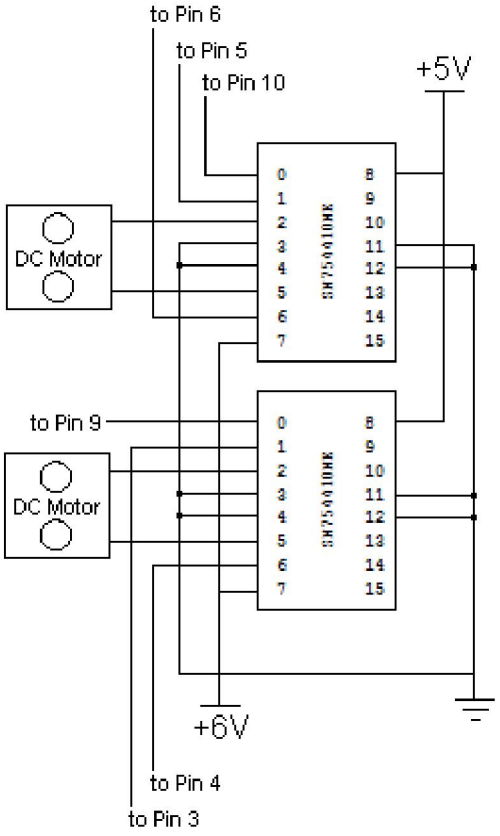

Figure 4: H-bridge circuit

Sensor Array Pin Mapping

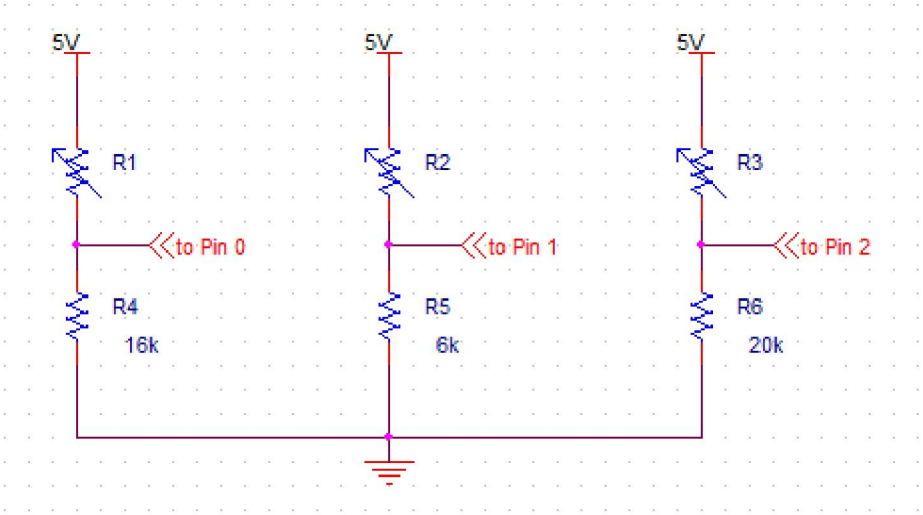

Figure 5: Sensor array circuit

6) Software Description

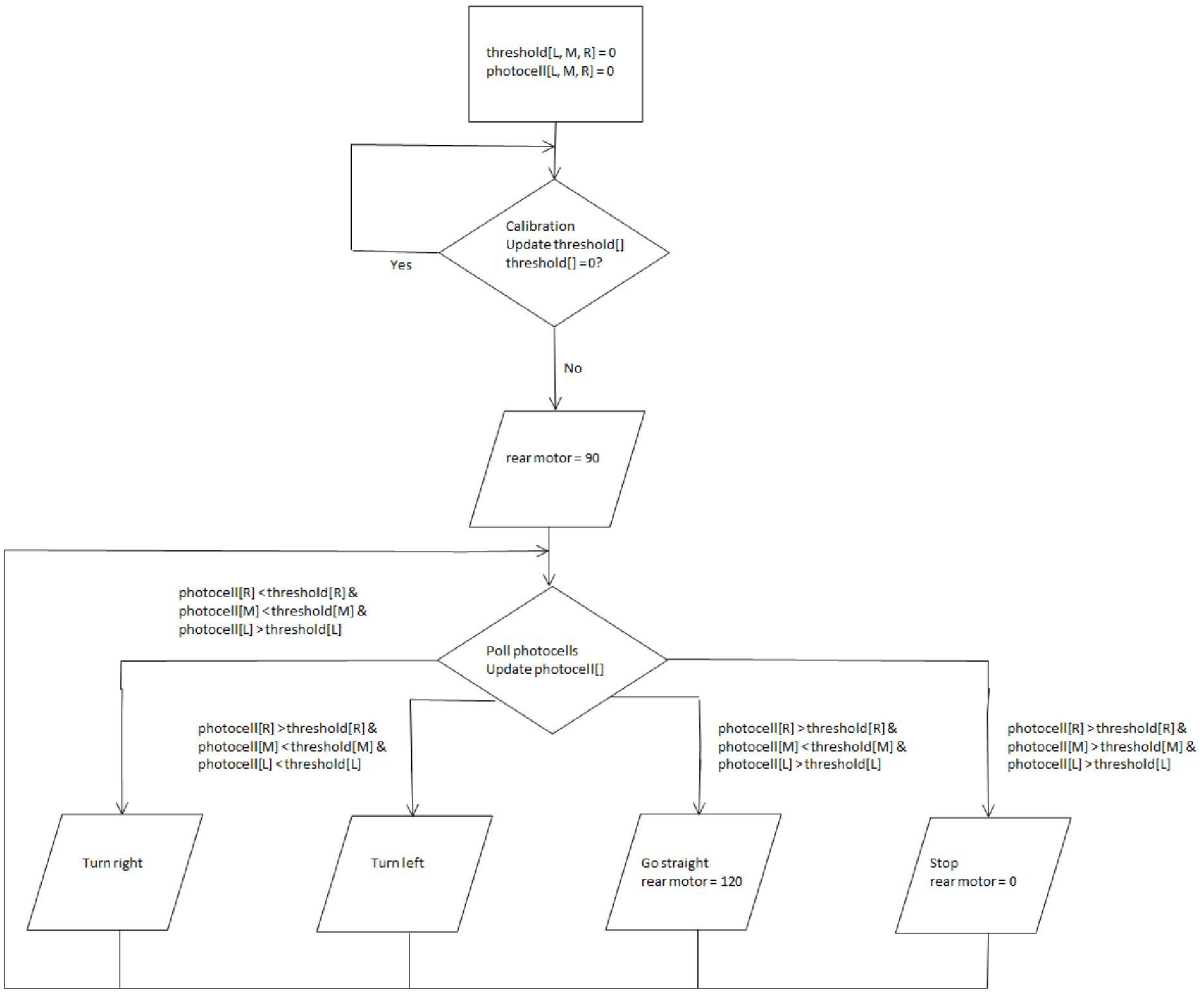

Figure 6: Software diagram

```
/* Line Following RC Car
  * Kevin Fong
  */

// IO pins
int frontMotor1Pin = 2;          // H-bridge2 leg 1
int frontMotor2Pin = 4;          // H-bridge2 leg 2
int frontMotorSpeedPin = 9;    // H-bridge enable pin
int rearMotor1Pin = 5;           // H-bridge1 leg 1
int rearMotor2Pin = 6;           // H-bridge1 leg 2
int rearMotorSpeedPin = 10;    // H-bridge enable pin

int sensorLedPin = 7;            // Sensor LED

int potLPin = 0;                  // Photocell pins
int potMPin = 1;
int potRPin = 2;

int lBlinker = 11;                // Left and right blinkers
int rBlinker = 12;
```

```
int ledPin = 13;                  // LED

// Initial values
int valL = 0;                     // Variable to store the value coming from the sensors
int valM = 0;
int valR = 0;

int threshL = 0;
int threshM = 0;
int threshR = 0;

int accL = 0;
int accM = 0;
int accR = 0;
int count = 0;

float darkPercentL = 0.96;        // Threshold for dark/light contrast (percent)
float darkPercentM = 0.93;        // *darkPercentX can be changed depending on surface color/contrast
float darkPercentR = 0.93;

void setup() {
  // Set up all pins for output
  pinMode(frontMotor1Pin, OUTPUT);
  pinMode(frontMotor2Pin, OUTPUT);
  pinMode(frontMotorSpeedPin, OUTPUT);
  pinMode(rearMotor1Pin, OUTPUT);
  pinMode(rearMotor2Pin, OUTPUT);
  pinMode(rearMotorSpeedPin, OUTPUT);
  pinMode(sensorLedPin, OUTPUT);
  pinMode(lBlinker, OUTPUT);
  pinMode(rBlinker, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // Set speedPins high for front motor so that it can turn on while low for rear motor to prevent it from jumping
  digitalWrite(frontMotorSpeedPin, HIGH);
  digitalWrite(rearMotorSpeedPin, LOW);

  // Sets the sensor LED high
  digitalWrite(sensorLedPin, HIGH);

  // blink the LED 3 times. This should happen only once.
  // if you see the LED blink three times, it means that the module
  // reset itself,. probably because the motor caused a brownout
  // or a short.
  blink(ledPin, 3, 100);

  Serial.begin(9600);             // use the serial port to send the values back to the computer

  // Sequence lets user see that motor is functional and power supply is connected
  // All low sequences ensure that at no given point in time that a short does not exist within the H-bridge
  delay(500);
  digitalWrite(frontMotor1Pin, LOW);      // set leg 1 of the H-bridge2 low
  digitalWrite(frontMotor2Pin, LOW);      // set leg 2 of the H-bridge2 low
  delay(500);
  digitalWrite(frontMotor1Pin, LOW);      // set leg 1 of the H-bridge2 low
  digitalWrite(frontMotor2Pin, HIGH);    // set leg 2 of the H-bridge2 high
  delay(500);
```

```
  digitalWrite(frontMotor1Pin, LOW);      // set leg 1 of the H-bridge2 low
  digitalWrite(frontMotor2Pin, LOW);      // set leg 2 of the H-bridge2 low
  delay(500);
  digitalWrite(frontMotor1Pin, HIGH);     // set leg 1 of the H-bridge2 high
  digitalWrite(frontMotor2Pin, LOW);      // set leg 2 of the H-bridge2 low
  delay(500);
  digitalWrite(frontMotor1Pin, LOW);      // set leg 1 of the H-bridge2 low
  digitalWrite(frontMotor2Pin, LOW);      // set leg 2 of the H-bridge2 low

  delay(5000);                  // Gives time for user to turn on car and set it down on a surface
  calibrateThreshholds();       // Runs calibration of surface light intensity and estimates black line threshold value
  delay(3000);
  blink(lBlinker, 3, 500);      // Blink headlight sequence to notify user of successful threshold calibration
  blink(rBlinker, 3, 500);
}

void loop() {

  valL = analogRead(potLPin);       // read the value from the sensor
  valM = analogRead(potMPin);
  valR = analogRead(potRPin);


  Serial.print(valL);               // print the value to the serial port
  Serial.print("\t");
  Serial.print(valM);
  Serial.print("\t");
  Serial.print(valR);
  Serial.println();

  // rear motor drive
  digitalWrite(rearMotor1Pin, HIGH);      // Set leg 1 of the H-bridge1 high
  digitalWrite(rearMotor2Pin, LOW);       // Set leg 2 of the H-bridge1 low
  analogWrite(rearMotorSpeedPin, 90);     // Write to speed pin to change speed

  // front motor steering
  // Blinkers provide debugging for steering
  if(valR < threshR && valM < threshM && valL > threshL)
  {
    digitalWrite(lBlinker, LOW);
    digitalWrite(rBlinker, HIGH);
    turnRight();
  }

  else if(valR > threshR    && valM < thresh && valL < threshL)
  {
    digitalWrite(lBlinker, HIGH);
    digitalWrite(rBlinker, LOW);
    turnLeft();
  }

  }
  else if(valR > threshR && valM > threshM && valL > threshL)
  {
    digitalWrite(lBlinker, LOW);
    digitalWrite(rBlinker, LOW);
    stop();
  }
```

```
    else
    {
      digitalWrite(lBlinker, HIGH);
      digitalWrite(rBlinker, HIGH);
      digitalWrite(rearMotor1Pin, HIGH);      // Set leg 1 of the H-bridge1 high
      digitalWrite(rearMotor2Pin, LOW);       // Set leg 2 of the H-bridge1 low
      analogWrite(rearMotorSpeedPin, 120);    // Write to speed pin to change speed
    }
}

/*
   blinks an LED
 */
void blink(int whatPin, int howManyTimes, int milliSecs) {
    int i = 0;
    for ( i = 0; i < howManyTimes; i++) {
      digitalWrite(whatPin, HIGH);
      delay(milliSecs/2);
      digitalWrite(whatPin, LOW);
      delay(milliSecs/2);
    }
}

void turnLeft() {
    digitalWrite(frontMotor1Pin, HIGH);      // set leg 1 of the H-bridge2 low
    digitalWrite(frontMotor2Pin, LOW);    // set leg 2 of the H-bridge2 high
    delay(100);
    digitalWrite(rearMotor1Pin, HIGH);      // Set leg 1 of the H-bridge1 high
    digitalWrite(rearMotor2Pin, LOW);       // Set leg 2 of the H-bridge1 low
    analogWrite(rearMotorSpeedPin, 90);    // Write to speed pin to change speed
    delay(100);
    digitalWrite(frontMotor1Pin, LOW);      // set leg 1 of the H-bridge2 low
    digitalWrite(frontMotor2Pin, LOW);      // set leg 2 of the H-bridge2 low
    delay(100);
}

void turnRight() {
    digitalWrite(frontMotor1Pin, LOW);    // set leg 1 of the H-bridge2 high
    digitalWrite(frontMotor2Pin, HIGH);      // set leg 2 of the H-bridge2 low
    delay(100);
    digitalWrite(rearMotor1Pin, HIGH);      // Set leg 1 of the H-bridge1 high
    digitalWrite(rearMotor2Pin, LOW);       // Set leg 2 of the H-bridge1 low
    analogWrite(rearMotorSpeedPin, 90);    // Write to speed pin to change speed
    delay(100);
    digitalWrite(frontMotor1Pin, LOW);      // set leg 1 of the H-bridge2 low
    digitalWrite(frontMotor2Pin, LOW);      // set leg 2 of the H-bridge2 low
    delay(100);
}

void stop() {
    digitalWrite(frontMotor1Pin, LOW);      // set leg 1 of the H-bridge2 low
    digitalWrite(frontMotor2Pin, LOW);      // set leg 2 of the H-bridge2 low
    delay(100);
    digitalWrite(rearMotor1Pin, LOW);       // Set leg 1 of the H-bridge1 low
    digitalWrite(rearMotor2Pin, HIGH);      // Set leg 2 of the H-bridge1 HIGH
    analogWrite(rearMotorSpeedPin, 0);    // Write to speed pin to change speed
    delay(100);
}
```

```
void calibrateThreshholds() {
   for(int i = 0; i < 50; i++) {
      accL += analogRead(potLPin);        // read the value from the sensor
      accM += analogRead(potMPin);
      accR += analogRead(potRPin);
   }

   Serial.println("Averages");
   Serial.print(accL/50);
   Serial.print("\t");
   Serial.print(accM/50);
   Serial.print("\t");
   Serial.print(accR/50);
   Serial.println();

   threshL = (accL/50)*darkPercent;
   threshM = (accM/50)*darkPercent;
   threshR = (accR/50)*darkPercent;

   Serial.println("Thresholds");
   Serial.println("Left\tMiddle\tRight");                    // print the value to the serial port
   Serial.print(threshL);
   Serial.print("\t");
   Serial.print(threshM);
   Serial.print("\t");
   Serial.print(threshR);
   Serial.println();
}
```

7) System Test Plan

Test #	Description of Set-up	Input or Stimulus	Expected Behavior
1	Simplest course on high contrast floor	Straight line	Go straight
2	Simple course on high contrast floor	Gentle curves with straight lines (loop)	Either turn left or right and realign itself
3	Impossible course on high contrast floor	Approximately right angled turns	Differential steering does not support very sharp turns. Car should end up lost.
4	Any simple course on high contrast floor	Excessive ambient light	Should not affect car behavior
5	Any simple course on high contrast floor	One or more sensors read wrong data	Code does not support normalization yet. Erratic values may confuse stepper motor.
6	Low contrast floor	Any line	Photocell may not be sensitive enough to produce a high distribution of values. Without normalization, car will be left constant in one position.

8) Cost Sheet

Part Number/Name	Unit Cost	Number of Units	Total Cost
Arduino	$34.95	1	$34.95
RC Car	$19.99	1	$19.99
Breadboard	$15.95	1	$15.95
Photocell (5 pack)	$2.99	1	$2.99
LED	$1.49	3	$4.47
Resistors	$0.01	7	$0.07
H-Bridge Motor Driver 1A	$2.35	2	$4.70
TIP120 NPN-D Transistor	$1.59	1	$1.59
Wire	$5.49	1	$5.49
PC Board	$1.99	1	$1.99
PCB Standoff Screws	$2.99	1	$2.99
9V Battery Connector	$1.99	1	$1.99
9V Battery	$3.99	1	$3.99
AA Battery (4 pack)	$3.99	1	$3.99
			$105.16

9) Summary

The results of my project was very good at first. I started out by first purchasing a relatively cheap RC car and stripping it down to its barest quintessential; chassis, wheels, motors, and steering axle. Then I planned to base the movement of the vehicle using its DC motors. Since no technical specifications were noted on any manual or website I had to get a rough estimate on its ratings. Using its integrated circuit and a multimeter, I was able to derive its operating voltage and current needed to supply enough torque to the pinion and the gear, roughly 5~6V at 0.5A. The battery compartment houses 4 AA batteries of 1.6V each and able to supply 1A*hour. From the gearing of the motors the power supply is more than enough to power the two, front and rear, motors at the expense of some speed for higher torque. Next builds comprised of individual circuits that would be coupled to the microcontroller consisting of the sensor array and H-bridge circuits for bidirectional motor functionality. SN754010NE's were chosen as the H-bridge ICs because of their low cost and specifications. It is able to handle voltages between 4.5~36V and supply 1A of current. CDS photocells were chosen for the sensor circuits provided that they are low cost and efficient at sensing light intensities. The bulk pack purchased did not state any specifications of the photocells and came in a variety of sizes and designs. Measuring the average resistances of each photocell in varying lightings helped distinguish the faulty units from the working ones and the ones with a large variance in resistance were chosen for the sensor array. I tried three variations of the sensor array by putting low, medium, and high resistances in series for each photocell to get the largest variance in voltage division for a supplied 5V from the microcontroller. The key was to optimize the size and value of the secondary resistances using only 1k and 47k Ohm resistors. In order to mount the components and additional 9V power supply into the vehicle a large amount of plastic had to be shaven down with my tool of choice, a soldering iron. Programming the microcontroller to interface with the H-bridges to the motors worked without a hitch and I was able to control direction and speed for each motor simultaneously, relatively speaking. Timing of the holding torque for steering was chosen to allow the car to "grip" onto the line and prevent the vehicle from overshooting and getting lost. Selecting a threshold for the line proved to be quite difficult since it always seemed to change based on ambient lighting and surface color. Using the serial feature on the microcontroller for feedback of values from each potentiometer I was able to gather large amounts of data from various surfaces and lightings. Analyzing the data, I could determine the best threshold to be used per surface/lighting scenario but each test case proved to be too dynamic to settle for one setting. So I created a function in software that would calibrate the threshold of the line on a given surface by gathering a large amount of data, average the values, and estimate the value of the line to become the threshold. Of course it was not flawless but the amount of user data analysis could be reduced to only modifying three variables, threshold percentages, to be used for estimating the amount of light that would be reflected back into the photocell for the contrasting surfaces. Multiple tests per change including steering functionality, rear drive throttling, and sensor functionality were performed after each alteration of code and component change to reduce debugging time.

Problems faced throughout the build process were the inefficiency of power utilization of the vehicle. Below 5V supplied to the H-bridge circuit and the motors would not provide enough torque to propel the vehicle on a surface. The 6V power source was then always kept fresh for guaranteed optimal performance (torque to propel to the vehicle). The same problem occurred for the micro-

controller 9V power supply. A drop below 7V and degradation occurred in the sensor array. Potentiometer values became erratic and unreliable, so insufficient 9V batteries were swapped out for fresh ones. From what I observed as the build continued the physical composition of the car made it quite unreliable. The power supply housing for the batteries somehow failed in one or more of the four sections and has a slight connection break that stops the supply of power to the motors through the H-bridge circuit. This problem might be the biggest and most challenging problem because after this had occurred the motors had serious torque issues. Sometimes the vehicle could propel itself and sometimes it could not at the same speed regardless of battery conditions. I did not have a separate AA battery holder and did not trust myself in creating a makeshift one out of tin foil at the last minute so I am confident that the motor power supply is the biggest problem of my project.

Other implementations that I could have tried in my project were to use IR emitter and receiver pairs instead of LEDs and photocells to handle lower contrasting surfaces better. The problem with photocells is that it is a good comparator of highly contrasting uniform surfaces. IR emitters and receivers can also take values faster and handle patterned surfaces due to the wavelength of light that it only responds to. Another feature that could have been implemented was distance sensors to be able to detect objects and essentially programmed to slow down or halt the car. I would like to say that my project was a success in the times that it was fully functional and if I could have tried it again I would try to invest In a better RC car as a base.