An Arduino Controlled Autonomous Track Following Vehicle

ECE 110: Introduction to Electrical and Computer Engineering
Final Design Challenge Lab Report

Ali K. Afridi

December 11, 2012

1 Introduction

The objective of this project was to build a car that can navigate a track by following a white tape track, which had been set down on a black table. The car must be completely autonomous meaning that it should be able to navigate the entire track without the help of any external forces. The track has several features that the car must be able to navigate, these features include curves, zig-zags, right angle turns and splits. The car will not encounter any turns greater than 90 degrees. At splits the car will either have to turn left or right depending on whether the tape at the split is grey (turn right) or white (turn left). The car must come to a complete stop when it arrives at the end of the track, which is made up of a large white square of tape. This vehicle can either work by using purely hardware or by using an Arduino microcontroller, which has been programmed by C.

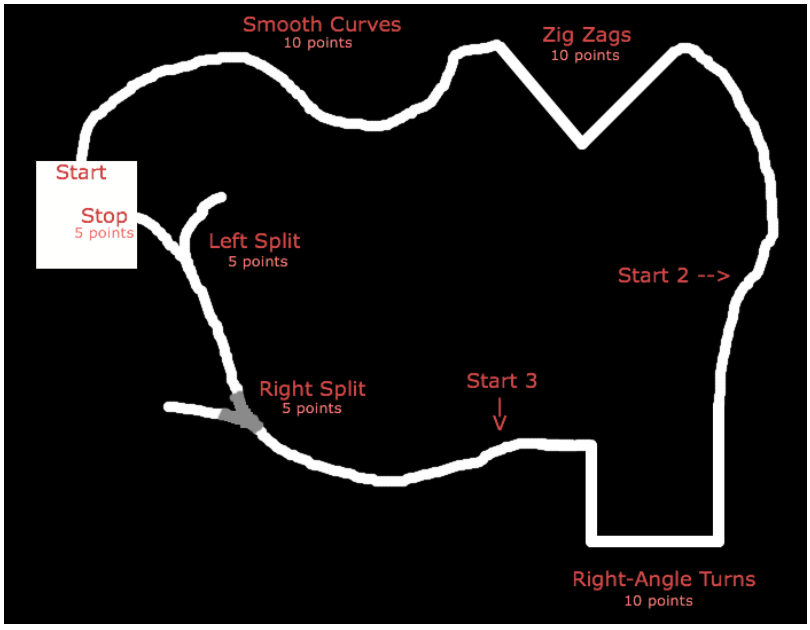

Figure 1: Example Track with Points Breakdown

Figure 1 shows a diagram of the track that the car was required to navigate. On the test day you are allowed to have 5 trials on the final track, and you can combine points from several trials. There are three different starting locations and you can decide where you would like to place the car at the beginning of the trial. Along with the track are labels showing what each obstacle is, as well as how many points each part of the track was worth for the final score. For the splits, there was a combined points of 10 points for navigating both splits, although it is possible to get 2.5 points each (5 total) just for showing that the car can detect the split, either by using LED or using serial reads. In addition to the points shown in the figure you can also get an additional 5 points if the car can navigate everything from the start to the stop in a single run. This means that it is possible to get as many as 50 points out of 40.

2 Design Documentation

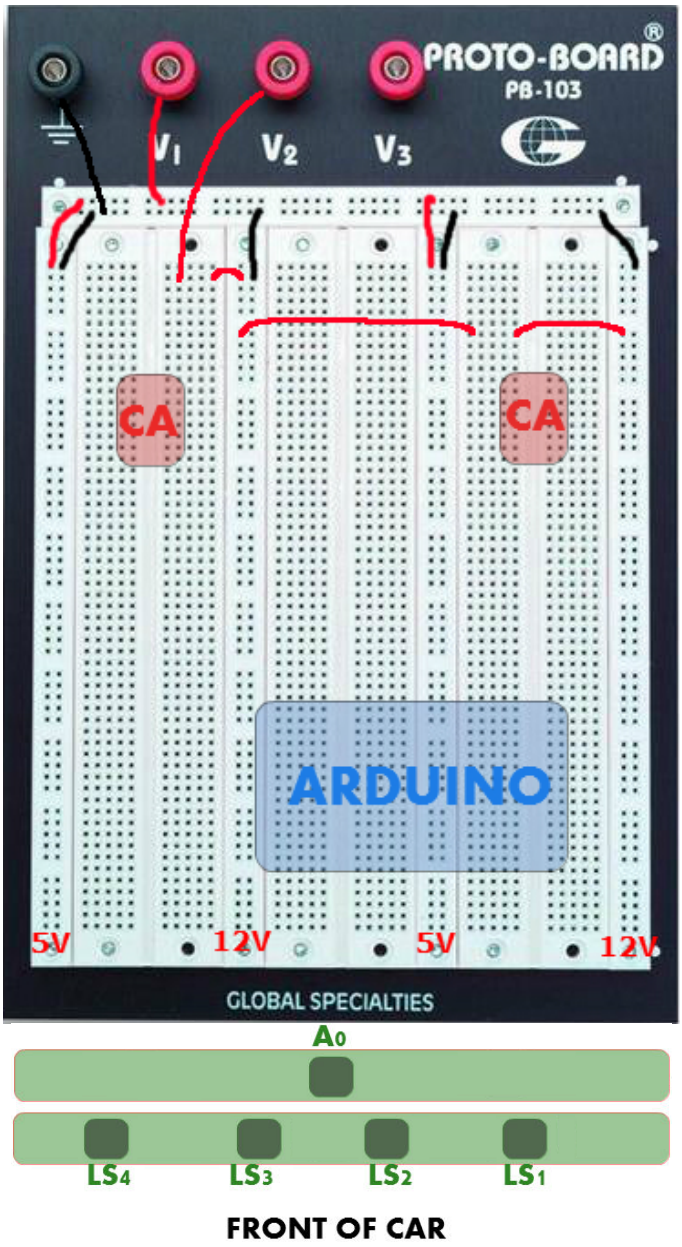

Figure 2: Board Layout and sensor placement

Figure 2 shows the placement of the Current-Amplifier (CA) modules and the arduino on the protoboard. It also shows how the wires are arranged on the protoboard to provide the proper voltage to the sensor bars, the arduino, and the CA modules. The sensor placement is also shown, LS1 is the sensor on the extreme left, and LS4 is the sensor on the extreme right when the car is taken as the reference frame. A0 is a sensor which is used during split to decide whether the car is to turn right or left. The voltages are arranged in the way that is shown because both the CA modules require 12 Volts (V_{motor}) while the arduino and sensor bar must be connected to 5V (V_{logic}).

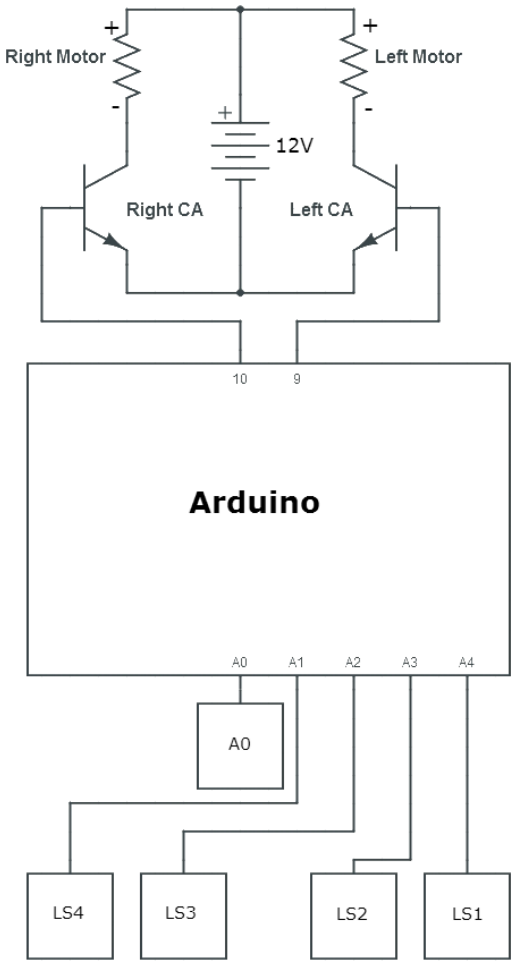

Figure 3: Block Diagram of Circuit

Figure 3 shows a block diagram of the circuit that was used for the final design. Each of the 5 sensors were attached to analog sensors of the arduino. The outputs of pin 9 and 10 were then attached to the CA modules which controlled whether or not the motor actually ran. LS1, LS2, LS3, LS4 and A0 will all output analog values to the arduino which will take in these analog values as input. Then according to the logic which has been programmed into the arduino will decide the values for pin 9 and pin 10. If one of the output pins of the arduino (pins 9 and 10) outputs a 0, the motor on that side of the car will not run. For example if pin 9 has an output of 0, the left motor will be off. However if pin 9 or 10 have an output signal with a duty cycle that is above a certain threshold (~20% for my car) the motor on that side will run.

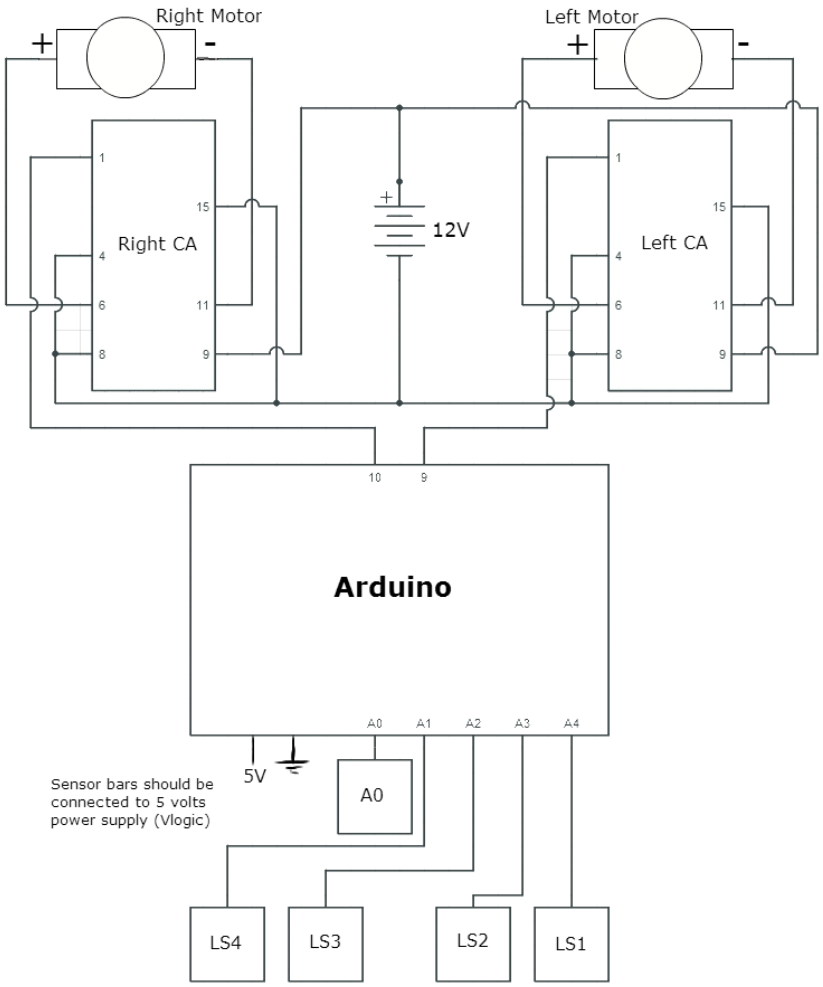

Figure 4: Wiring Diagram of circuit

Figure 4 is a wiring diagram for the circuit, which should be useful to anyone that would like to replicate the design. Pretty much every part of the circuitry is shown in the wiring diagram, but you must make sure that you attached the sensor bars to the 5 volts that is Vlogic. The arduino must also be connected to 5V. Also since the arduino should be used with a different type of CA module from the ones we have used in previous labs, the CA module should be connected the way that is described in this wiring diagram.

3 Design Philosophy

The car that I designed uses a tape avoiding circuit along with a few added features to navigate the track. The tape avoiding circuit works so that the car will run when the sensors see black and will stop when the sensors see white. If both LS2 and LS3 see black the car will run straight, and if LS1, LS2, LS3, and LS4 see white then the car will stop. If LS2 sees white while LS3 sees black then it means that the left motor will switch off, while the right motor will stay on. The car will then turn to the left, LS2 starts to see black again. The opposite happens when LS3 sees white. If both LS2 and LS3 see white and LS1 and LS4 are black, then a split is detected and the value of A0 is checked to see whether the vehicle should turn left or right. Sensors LS1 and LS4 are used to detect 90 degree turns. The arduino will check the sensors and then call a method to carry out the task appropriate for the given inputs. The table below shows a truth table of the task that will be done for different types of sensor values. All the logic is done by an Arduino Uno board which has been programmed in C. The only other module on the protoboard are the two Current-Amplifier modules which run the motors of the car.

LS1	LS2	LS4	LS5	Function
0	0	0	0	Straight
0	0	0	1	Sharper Right
0	0	1	0	Right
0	0	1	1	Sharper Right
0	1	0	0	Left
0	1	0	1	x
0	1	1	0	(Depends on Analog)*
0	1	1	1	x
1	0	0	0	Sharper Left
1	0	0	1	x
1	0	1	0	x
1	0	1	1	x
1	1	0	0	Sharper Left
1	1	0	1	x
1	1	1	0	x
1	1	1	1	Halt

* If Analog is between 100 and 315 the car turns right, otherwise it turns left

Since I was having trouble with having the car read a digital high for white and grey and a low for black I decided to implement a car which uses analog input values instead. The sensor bars that I had been using were not going high enough, so the sensors would always read digital high, even for the black table. By using analog values I could easily adjust the value for which the car sees high, for any table. Although sensor height is still important, this really helped me with the problem of my digital inputs always reading high.

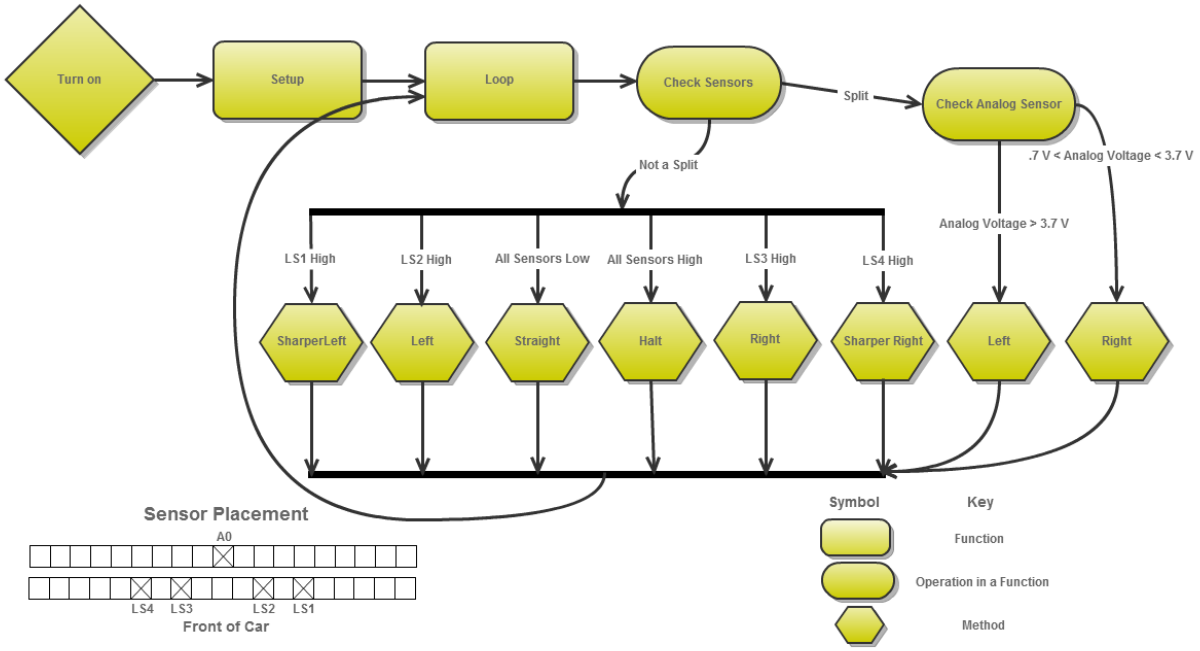

Figure 5: Code Flowchart

Figure 5 above is a flowchart showing the basic process that the code that I wrote goes through to implement the track navigation. After the arduino is turned on, the program will run a function called Setup() within the code. Here the pins in use will be assigned as either input or output. The program will then run a loop() function which will continue to run until power is turned off. The loop() function has several if, elseif, else conditions which determine what task needs to be done, and then calls on a method to actually do operation the car requires to follow the car. For split navigation, the two analog sensors are read and then depending on what the value is, the car will either turn left or right. Below is a list of the methods, as well as what they do:

- Straight() - go straight
    - Both Motors on
- Right() - turn right
    - Left motor on, right motor off.
- Left() - turn left
    - Right motor on, left motor off.
- Sharper_Right() - turn right at a right angle turn

    - Left motor on, right motor off
    - Stays on until LS2 (inner left sensor) is high
- Sharper_Left()
    - Right motor on, left motor off.
    - Stays on until LS4 (inner right sensor) is high
- Halt() - stop car
    - Both Motors off.

4 Results

My car worked perfectly on the practice track, navigating the entire track and going around the entire course without any problems. However it did not have the same success on the actual track. On the first trial the car successfully showed the stop mechanism, but when it was turned around the car did not even start. The same happened on the second trial, even though in between the trials I had made sure all the values I had entered were calibrated for the final track. Between the second and third trial I increased the speed of both motors, and during this trial the car was able to successfully navigate the smooth curves as well as the zig-zags. However it went off the track before it could do the right-angle turns. For my fourth trial, I decided not to change anything and started the car just before the right angle turns, and the car successfully navigated 2 of the 3 turns. The one that it was unable to do was the left 90 degree turn. However at this point I realized that the big issue with my car was that the left motor was running faster than the right one, which was one of the reasons I was not getting the results that I wanted. The car would always turn correctly when the turns were supposed to be towards the right because the left motor would turn fast enough. However on left turns the right motor would not turn as well because it was going slower. I should have changed the value of the speed for the two motors, however I only had five minutes left for my final trial and so I decided to try the split navigation with the car as it was. The car navigated the grey split correctly and turned right, however for the next split it started turning but stopped, which I believe is because the motor speeds should have been different so the car could do the left turn better. The duty cycle of the right motor should have been increased to compensate for the left motors faster speed.

5 Future Work

Although the car was able to navigate the track pretty well in all the trials and worked pretty effectively there would be a few things that could be added to the circuitry to improve the performance. If I had had more time one of the first things I would have done is add an additional method which would cause the car to go backwards if all 5 sensors detected black, because this would mean that the car was no longer following the track. This would be a backup mechanism, which would cause the car to reverse if it lost the tape, and this could be a useful strategy. If I was to go back and do it again I would probably challenge myself more by trying to hard-wire the car and not use Arduino because I've already programmed before but I haven't really played around with all the chips and devices we used in lab before this course. One last thing that I might change if I was to redo the project was to have a partner that I worked with because it seemed like other groups which had more than one person were able to effectively

work together to divide up the work, and having two minds would be much better than one especially when troubleshooting.

6 Conclusion

Although the car had a few glitches which caused it to get a 40.5 on the test day rather than the maximum 50, it is still a good design which, with a few minor changes, could be a very effective tape avoiding track follower.

7 Acknowledgements

The flowchart for the code was made using gliffy.com. The circuit schematics were made using circuitlab.com, and were then edited in the editor on pixlr.com. All other diagrams were made using the online image editor on pixlr.com.

Appendix: Vehicle Arduino Code

```
/****************************************
ECE 110 Final Project
Author: Ali Afridi
Start Date: 11/13/2012
Last Edited : 12/10/2012
****************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//Inputs
#define LS1 A4
#define LS2 A3
#define LS4 A2
#define LS5 A1

//Outputs
#define LM 9 //Left motor
#define RM 10 //Right motor

#define white_val 315 //Value at which white tape is detected
#define outer_white_val 250 //Value at which white tape is detected
#define turn_val 260 //Value at which white tape is detected
#define split_val 90 //Value at which white tape is detected
#define motor_speed 95 //Value at which white tape is detected

void setup()
{
 Serial.begin(9600);
 int i;
 for (i=2; i<7; i++) //pins 2 through 6 are now inputs
 {
  pinMode(i,INPUT);
 }
 for (i=9; i<13; i++) //pins 7 through 9 are now outputs
 {
  pinMode(i,OUTPUT);
 }
}
```

```
void loop()
{
Serial.println("    ");
 if((analogRead(LS1) < white_val || analogRead(LS5)<white_val) && analogRead(LS2) >
split_val && analogRead(LS4) > split_val ) //Split, Both center sensors see white
 {
  //Implement Split Circuit
  int color_val1 = analogRead(0);
    Serial.println("Split");
  if((color_val1 < 340)) //grey (turn right)
  {
   right();
  }
  else
  {
   left();
  }
 }
 else if(analogRead(LS1) < white_val && analogRead(LS2) < white_val && analogRead(LS4)
< white_val && analogRead(LS5) < white_val) //All sensors see black
 {
  straight();
 }
 else if(analogRead(LS1) < white_val && analogRead(LS2) < white_val && analogRead(LS4)
> white_val && analogRead(LS5) < white_val) //LS4 sees white, turn right
 {
  right();
 }
 else if(analogRead(LS1) < white_val && analogRead(LS2) < white_val && analogRead(LS4)
> white_val && analogRead(LS5) > outer_white_val) //LS4 and LS5 see white, right angle turn
right
 {
  sharper_right();
 }
  else if(analogRead(LS1) < white_val && analogRead(LS2) > white_val && analogRead(LS4)
> white_val && analogRead(LS5) > outer_white_val) //LS4 and LS5 see white, right angle turn
right
 {
  sharper_right();
```

```
 }
 else if(analogRead(LS1) < white_val && analogRead(LS2) > white_val && analogRead(LS4)
< white_val && analogRead(LS5) < white_val) //LS2 sees white, Turn Left
 {
  left();
 }
 else if(analogRead(LS1) > outer_white_val && analogRead(LS2) > white_val &&
analogRead(LS4) < white_val && analogRead(LS5) < white_val) //LS1 and LS2 see white,
Right angle turn left
 {
  sharper_left();
 }
  else if(analogRead(LS1) > outer_white_val && analogRead(LS2) > white_val &&
analogRead(LS4) > white_val && analogRead(LS5) < white_val) //LS1 and LS2 see white,
Right angle turn left
 {
  sharper_left();
 }
 else if(analogRead(LS1) > white_val && analogRead(LS2) > white_val && analogRead(LS4)
> white_val && analogRead(LS5) > white_val)//all sensors see white, stop
 {
  halt();
 }
 else //something else happened, which shouldn't have happened
 {
  digitalWrite(13, HIGH);
  halt();
 }
}
//Functions
void straight()
{
 analogWrite(9, motor_speed);
 analogWrite(10, motor_speed);
}
void left()
{
 analogWrite(9, speed0);
 analogWrite(10, motor_speed);
}
```

```
void right()
{
 analogWrite(9, motor_speed);
 analogWrite(10, speed0);
}
void sharper_left()
{
 while (analogRead(LS4) < turn_val)
 {
 analogWrite(9, speed0);
 analogWrite(10, speed80);
 }
}
void sharper_right()
{
 while (analogRead(LS2) < turn_val)
 {
 analogWrite(9, speed80);
 analogWrite(10,speed0);
 }
}
void halt()
{
 analogWrite( 9, speed0 );
 analogWrite( 10, speed0 );
}
```