Christine Hudak and Matt Baldassarre
ECE 110 Lab ABC
12/3/14

Final Project Lab Report

Introduction

Problem Description

The purpose of our design was to build a car that would be able to follow a path made of white tape and stop before it hits the obstacle at the end of the path. We sought to do this through the use of two QRE1113 Line Sensor Breakouts located close to the ground in front of the chassis of the car, an HR-SR04 UltraSonic sensor located at the front of the car, and the car we built for Lab 8 that could follow a wall along with all components included in that challenge.

Design Concept

The Line Sensor Breakouts are able to detect whether or not the car is over a white line, or the black tabletop – we utilized two sensors and placed them side by side in the center of the front of the car, and each Line Sensor is connected to an individual wheel. We did this so that the sensors could detect a turn in the path by producing a different voltage (high or low) and allow the sensor to alter the duty cycle of the wheel, changing the speed so that the car can detect if it drives over white line and then make the appropriate correction. The UltraSonic sensor is located at the front of the car and is able to detect once the car is a distance of 3cm from an object, knowing this we integrated this sensor into our existing motor-PWM circuit so it would signal the Arduino a certain distance from the object. We wanted the UltraSonic sensor to detect objects that impede our car's path – the logical choice for the placement of this sensor was at the front of the front of the car.

Analysis of Components

Line Sensor Breakout Characterization (Description)

We characterized the Line Sensor Breakout in a similar fashion to how we characterized the flex sensors in lab 8, however we utilized the built in PWM in the Arduino and the ability of the Arduino to read the values from the sensors, that rather than directly hook our sensor up to the Digital Multimeter. In order to get the values for the line sensor characterization we held the sensors approximately 3mm away from the black table top, and then 3mm away from a white tape line. Since we wanted to convert the Arduino reading into volts we multiplied the sensor value by the ratio of the maximum voltage (5 Volts) to the maximum Arduino reading (1023).

[Arduino reading in volts = (5V/1023)*initial Arduino reading] 

Line Sensor Breakout Characterization (Values)

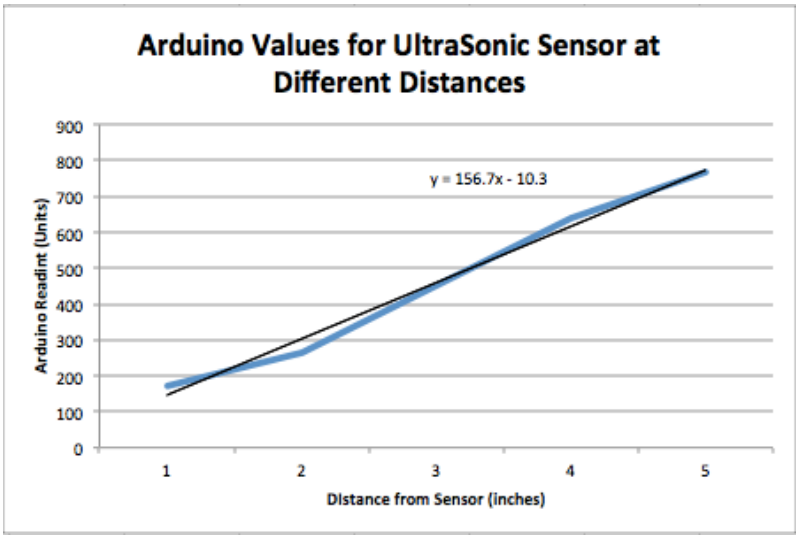
Input	Arduino Reading	Voltage Conversion (V)	Binary Conversion
Sensor connected to pin A0			
Black	68	0.31	LOW
White	1015	4.96	HIGH
Sensor connected to pin A1			
Black	57	0.28	LOW
White	1012	4.95	HIGH

UltraSonic Sensor Characterization (Description)

The UltraSonic Sensor Characterization was a bit trickier since we had both an OUTPUT and an INPUT, unlike the Line Sensor Breakout, which only has an OUTPUT (the value read by the sensor). We had to modify the code from our Line Sensor Characterization in order to account for this. In order to determine the different values for the sensor, we put a ruler (unfortunately all that we had was a small ruler in inches) next to our sensor and moved a notebook closer by increments of an inch (and a half inch at the end) and recorded the values read by the Arduino. For the purpose of our experiment we were interested in the values for 1-2 inches (since the sensor works best at distances under 3 cm, and since an inch is 2.54 cm, 1-2 inches is the around the value we want the sensor to detect for.

UltraSonic Sensor Characterization (Values)

Input (inches)	Arduino Reading (Units)
5	770
4	638
3	452
2	267
1	172
.5	180



Design Considerations

After characterizing both sensors we discovered the exact level of sensitivity each sensor has, specifically that both need to be right up close to the either the object or the line that they are detecting (something that is a bit cumbersome for real life use). We realized that we needed to construct some sort of arm that would allow for the Line Sensor Breakouts to be under 3mm away from the surface it is driving over and properly detect when it drives over the white line. As for the UltraSonic Sensor, we saw that we needed to position it on the end of our breadboard,

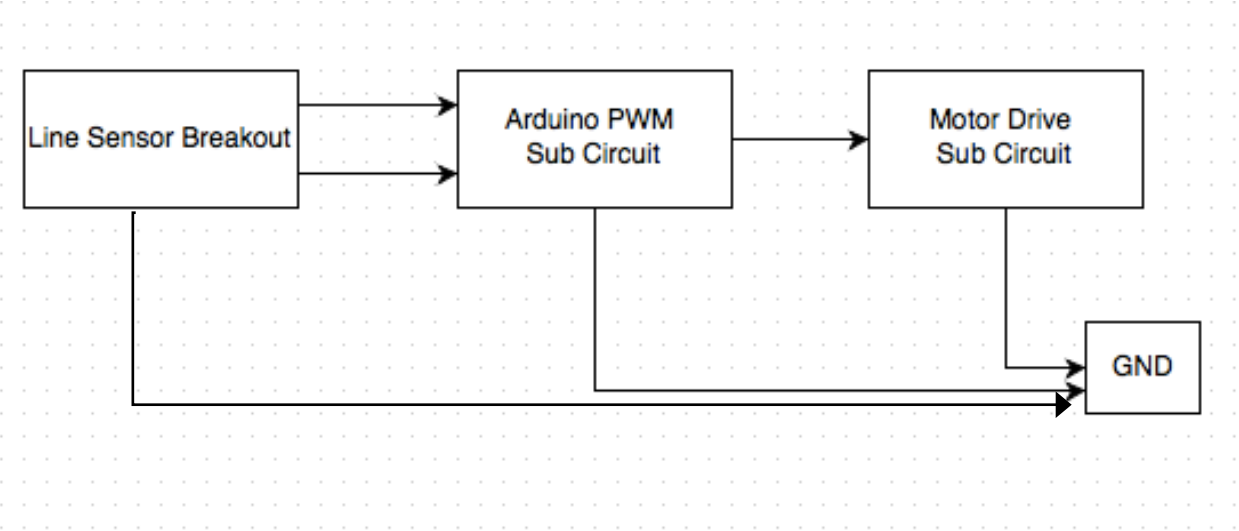
and the entire breadboard at the very front of our car in order to maximized our cars ability to detect objects impeding its path.

Our code takes in the values from both sensors and calculates a ratio based off of the black and white sensor values and the reading from the sensor which is then converted into a motor speed. [motor speed = (sensor value – White)*(double(255)/(Black – White)] This formula is similar to the voltage conversion we did and allows for the code to calculate a ratio dependent on the sensor inputs. The sensor then compares the value to see if it is negative (meaning the input is close to white or high) or very positive (the sensor is on black or low) and sets the speed accordingly. The UltraSonic sensor is coded to stop the motor if an object is a specified distance from the sensor.

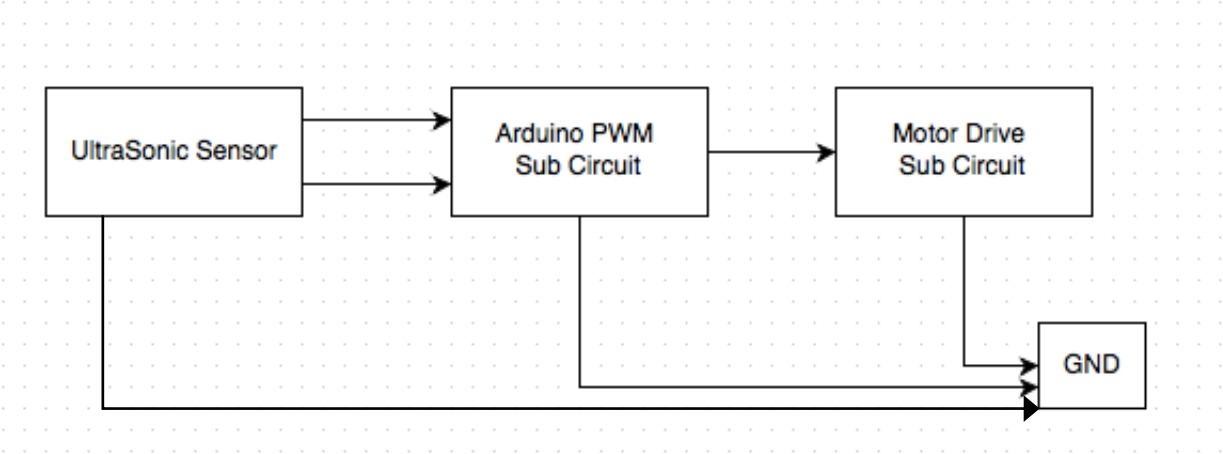
Design Description

Block Diagram

Line Sensor Breakout



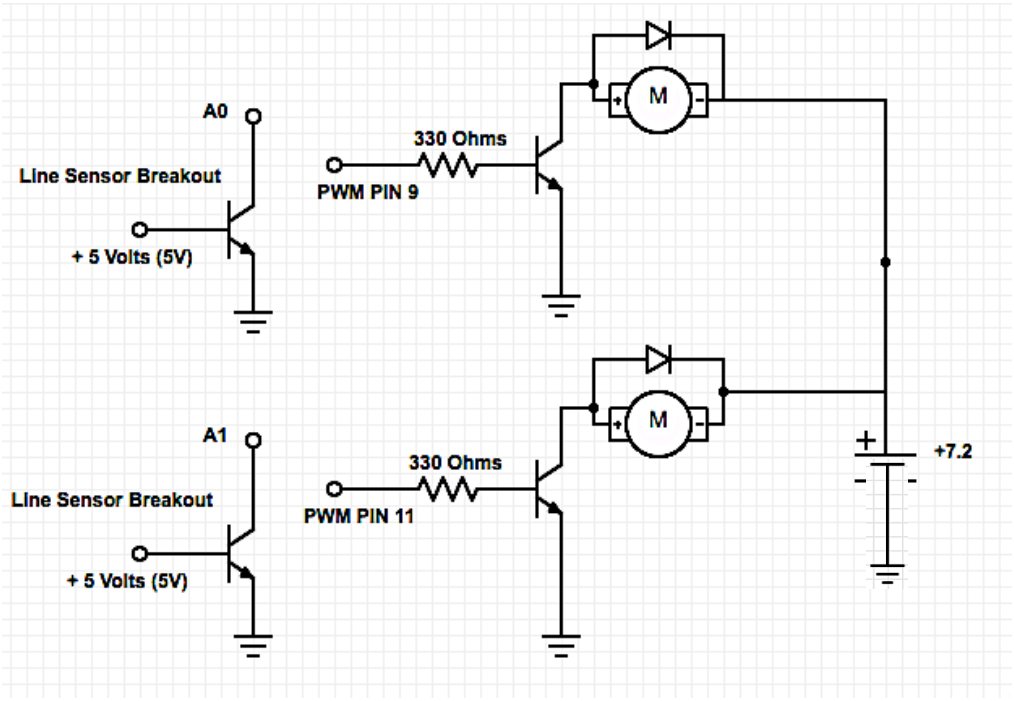
UltraSonic Sensor



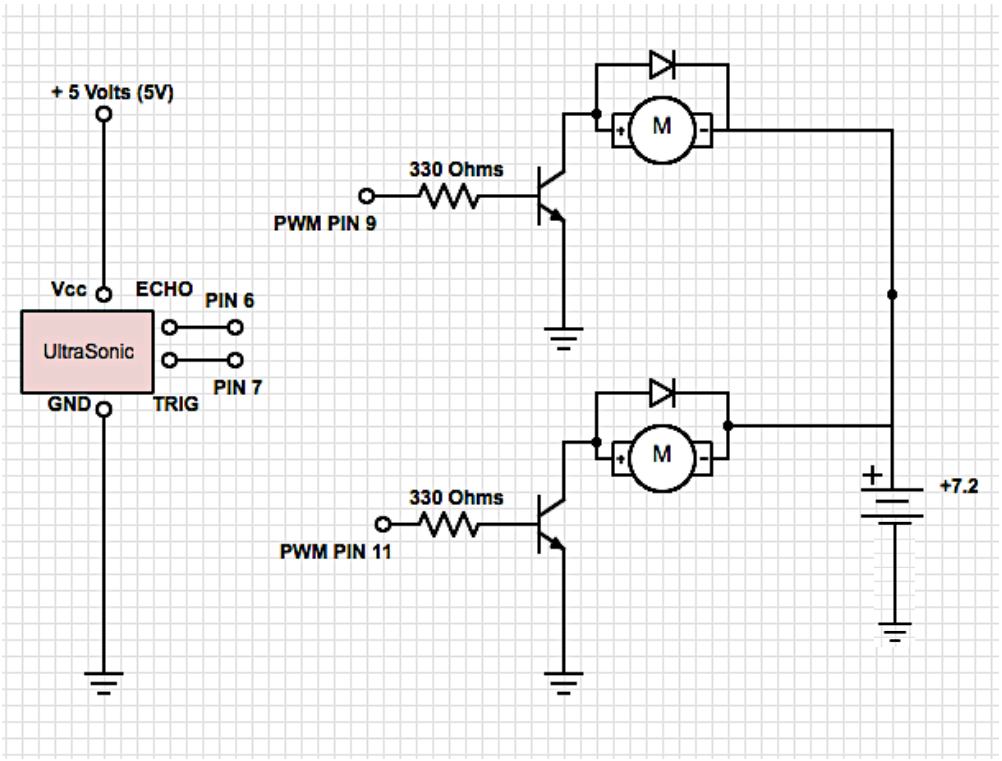
Circuit Schematics

Line Sensor Breakout

Note: there was not a symbol for a line sensor breakout, and since it's functionality for the purpose of our experiment is like a transistor I used the symbol for a BJT Transistor and labeled it as a line sensor breakout.



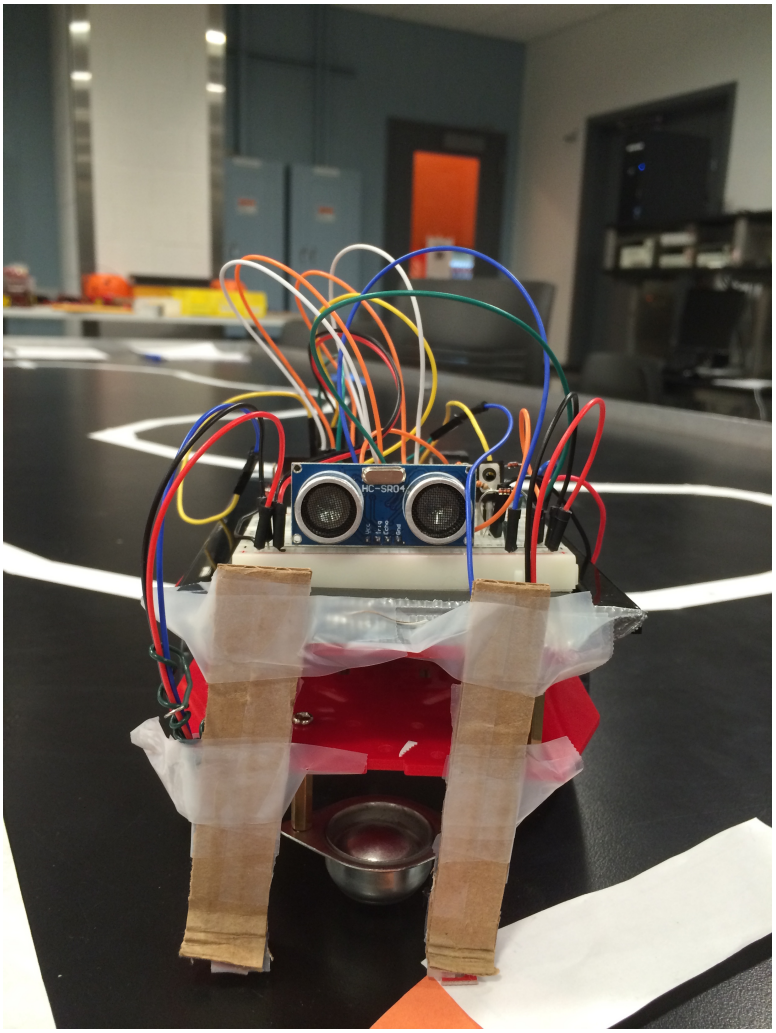
UltraSonic Sensor



Physical Mechanical Construction

We attached our Line Sensor Breakouts to the front of the car by cutting strips of cardboard and attaching them to our chassis through the use of duct tape. They are positioned at the front of the car in between the wheels close together. Initially we wanted the car to "follow" the white line and for the sensors to be right next to each other, but we found the car was unable to make many of the turns this way, so we adjusted our code so to that the car will correct itself whenever a sensor goes over a white strip.

We attached the UltraSonic Sensor directly to our breadboard on the top of our car. The breadboard was conveniently located at the front of our chassis and minimizes the distance between the UltraSonic Sensor and the object blocking its path.



Front view of our with the line sensor breakouts and Ultra Sonic sensor

Conclusion

Lessons Learned

The biggest challenge we faced was when we were figuring out how to adjust the Lab 8A code to suit the input from the Line Sensor Breakouts. Initially our problem was that our speed was too fast for our car to go throughout the track. We kept going back to the lab 8A code and systematically adjusted the variable in our code called "motorSpeed" for both wheels until we reached a slow enough speed that the car would be able to follow the track without crashing into objects at full speed. Additionally, we could not understand why the wheels would do the opposite of what we wanted it to do. Rather than speed up on white, the wheels would speed up on black. The resolution to this problem was to switch the values for black and white inputs, or swapped the values for each throughout the rest of our code. Overall we learned that it is necessary to fully test and adjust code before finally outputting a product. Additionally, we discovered that the Arduino is an incredibly powerful and helpful tool.

Self-Assessment

With respect to the goals we set out for the line sensor breakouts, we were able to reach all of those goals. Our car is able to follow a white track on a black surface by correcting the path once the sensor goes over a white line. Our UltraSonic sensor however is a less successful story, we were able to characterize the sensor and integrate it into the PWM-Motor circuit. But the car does not response to the input from the sensor – we believe that this is the result of our code, and are looking into rectifying the issue.

Code Appendix

Arduino Code for Line Sensor Characterization (Modified from Lab 5)

```
void  setup()  {
   Serial.begin(9600);
   pinMode(A0, OUTPUT); //Changed all A0 -> A1when testing sensor connected to A1
  }

void loop() {
   int sensorVaule0 = analogRead(A0);
   float voltage = sensorValue*(5.0/1023.0); //Converts the Arduino reading to volts
   Serial.println(sensorValue);
   Serial.println(voltage);

}
```

Arduino Code of UltraSonic Sensor Characterization (Modified from above and posted tutorial)

```
void  setup()  {
   Serial.begin(9600);
}

void loop() {

   long duration;
   pinMode(7, OUTPUT);
   digitalWrite(7, LOW);
   delayMicroseconds(2);
   digitalWrite(7, HIGH);
   delayMicroseconds(5);
   digitalWrite(7, LOW); //has the trigger pin (7) output a signal wave

   pinMode(6, INPUT); //reads the signal returned from pin 6
   duration = pulseIn(6, HIGH); //stores the value

   Serial.println(duration);

}
```

Arduino Code for Line Sensor-PWM Circuit (Modified from Lab 8A)

```
int  sensorPin0  =  A0;
int  PWMpin0  =  9;
```

```
float White0 = 1015; // max value read from the line sensor at pin A0
float Black0 = 68; // min value read from the line sensor at pin A0

int sensorPin1 = A1;
int PWMpin1 = 11;
float White1 = 1012; // max value read from the line sensor at pin A1
float Black1 = 57; // min value read from the line sensor at pin A1

 void setup() {
  pinMode(sensorPin0, INPUT); // set pin A0 as input
  pinMode(sensorPin1, INPUT); // set pin A1 as input
  Serial.begin(9600);
 }

 void loop() {
  int sensorValue0 = analogRead(sensorPin0); // read from pin A0
  int motorSpeed0 = (sensorValue0 – White0)*(double(255)/(Black1 - White1));

  if (motorSpeed1 < 0) {
  motorSpeed1 = 75;
  }
  if (motorSpeed1 > 175) {
  motorSpeed1 = 100;
  }

  int sensorValue1 = analogRead(sensorPin1); // read from pin A1
  int motorSpeed1 = (sensorValue1 – White1)*(double(255)/(Black1 – White1));

  if (motorSpeed1 < 0) {
  motorSpeed1 = 75;
  }
  if (motorSpeed1 > 175) {
  motorSpeed1 = 100;
  }
}
```

Arduino Code of Ultrasonic Sensor-PWM Circuit (Modified from Lab 8A and Characterization)

```
int sensorPin0 = A0;
int PWMpin0 = 9;
float White0 = 1015; // max value read from the line sensor at pin A0
float Black0 = 68; // min value read from the line sensor at pin A0
```

```
int sensorPin1 = A1;
int PWMpin1 = 11;
float White1 = 1012; // max value read from the line sensor at pin A1
float Black1 = 57; // min value read from the line sensor at pin A1

 void setup() {
  pinMode(sensorPin0, INPUT); // set pin A0 as input
  pinMode(sensorPin1, INPUT); // set pin A1 as input
  Serial.begin(9600);
 }

 void loop() {
  int sensorValue0 = analogRead(sensorPin0); // read from pin A0
  int motorSpeed0 = (sensorValue0 – White0)*(double(255)/(Black1 - White1));

  if (motorSpeed1 < 0) {
  motorSpeed1 = 75;
  }
  if (motorSpeed1 > 175) {
  motorSpeed1 = 100;
  }

  int sensorValue1 = analogRead(sensorPin1); // read from pin A1
  int motorSpeed1 = (sensorValue1 – White1)*(double(255)/(Black1 – White1));

  if (motorSpeed1 < 0) {
  motorSpeed1 = 75;
  }
  if (motorSpeed1 > 175) {
  motorSpeed1 = 100;
  }

   long duration;
   pinMode(7, OUTPUT);
   digitalWrite(7, LOW);
   delayMicroseconds(2);
   digitalWrite(7, HIGH);
   delayMicroseconds(5);
   digitalWrite(7, LOW); //has the trigger pin (7) output a signal wave

   pinMode(6, INPUT); //reads the signal returned from pin 6
   duration = pulseIn(6, HIGH); //stores the value
```

```
    if (duration < 200) {
    motorSpeed0 = 0;
    motorSpeed1 = 0;
    }

}
```

References:

https://courses.engr.illinois.edu/ece110/content/labs/

http://www.digikey.com/schemeit