ECE496Y Individual Progress Report Evaluation Form

Student				
	Project ID: 2011317	**Project Title:** Vehicle blind spot assist		
	Student Name: Tanzim Mokammel		**Supervisor:** M.Stickel	
	Section # 7	**Administrator:** R.Gillett		
	Contact hours per month with supervisor :	4	**Optimal # contact hours per month:** 4	

Administrator's Evaluation (Provide a rating for each section)	Excellent	Good	Adequate	Marginal	Poor/Missing	**Comments** *(also see comments in report)*
Progress: execution of work plan, significance of reported work Problems(circle): *claims not justified, status of tasks unclear*						
Method: Sound engineering practices reflected in actions, decisions, and testing. Changes reflect good project management. *Problems(circle): actions, decisions, testing*						
Reported Results: Quality and completeness of documentation *Problems(circle): inadequate documentation or analysis, inadequate test results, completion of task not verifiable*						
Overall Quality of Document						
Presentation (circle problem areas) *Grammar, spelling, clarity, writing style, use of figures & tables*						
Content (circle problem areas) *Organization, substance, references, Gantt chart incomplete*						
Other problems (circle): *late submission, other (specify)*						

Administrator's grade (/10):	**Section average (/10):**	**Administrator's signature:**

Note to supervisors: Your evaluation was done online. There is no need to return anything to the administrator.

The Edward S. Rogers Sr. Department of Electrical and Computer Engineering

University of Toronto

ECE496Y Design Project Course

Individual Progress Report

Title: Vehicle blind spot assist

Project I.D.#:	2011317
Team Member:	Tanzim Mokammel
Supervisor:	M. Stickel
Section #:	7
Course Administrator:	R. Gillett
Date:	Jan 16, 2012

Executive Summary:

The project involves the design of a device that assists users to safely change lanes by helping them determine dangerous situations. It also prevents accidents by locking the steering on the user vehicle to halt users from steering into an obstacle. The design consists of an input module, an output module, and an automation module. I am directly responsible for the input module, but I am also assisting with components of the output, and automation modules.

At the moment, considerable progress has been made on the basic functionality of each module, but there is a lot of integration and verification that needs to be completed. The input module consists of a sensor, and the microcontroller, and is responsible for calculating the distance and speed of an obstacle, in addition to predicting the amount of time needed for the obstacle to reach the blind spot. The sensor is able to measure the distance of an obstacle. An algorithm has been implemented in the microcontroller to use an array of the distance value in calculating the speed of the obstacle as well. The speed calculations have been verified to be accurate with less than 11% error while the sensor remains stationary. Code for the time calculation has been written, but yet to be verified.

The output module consists of an LCD screen, and an LED light. The LCD screen has been installed, and integrated with the input module to accurately display the microcontroller's calculations. The LED light circuit has been built, but the code specifying frequency of blinking needs to written and tested.

Work on the automation module has been initiated. Currently, the forward and backward movement of the user vehicle can be controlled. Steering control is the immediate next step.

The team is currently behind the initial schedule. In addition to completing the tasks mentioned above, we need to install the integrated input and output module to the user vehicle, and test the obstacle detection and anticipation functionality. Valikhan and I will be working closely to achieve this. Hani will be working to obtain full control of the user vehicle, and therefore implement the counter steering mechanism. A new schedule has been set to ensure project completion in time, without sacrificing initially intended features. However, we are experiencing higher than expected error and lower than expected range from the sensor. As a result, acceptance criteria for the design will have to be redefined.

Table of Contents

Summary of Tasks: 1

Detailed Explanation of Tasks: 2

1 & 3: Research input methods (sensors) ; Research processing methods (Microcontroller) 2

5: Program the Arduino to retrieve input from the sensors 3

7: Integrate the input and output components to provide user feedback according to the microcontroller's calculations 4

9 : Design an algorithm to detect approaching vehicles and calculate their relative speed 5

12: Achieve full control of the RC vehicle through the Arduino 7

Progress Assessment/ Conclusion: 9

Initial References 9

Appendix A: Detailed Data on the Input/output Module 10

A1. Current Code for Input/output Module 10

A2. Picture of the Physical Input / Output Circuit 14

A3: Serial Monitor 14

Appendix B: Details on Motor Control 15

B1. Code to Control Forward Motor Speed 15

B2. Example Output of the Motor Control Circuit 15

Appendix C: Initial Research 16

Appendix D: Speed Measurement Tests with Stationary Sensor 18

D1. Initial Tests 18

D2. Further Tests 19

D3. Comparison with Actual Speeds 22

Summary of Tasks:

The following is a summary of tasks:

Task #	Task Title	Responsible (R)/ Assisting (A)	Category	Status	Old Completion Date	New Completion Date
1	Research input methods (sensors)	R	Old	Complete	Sep 15, 2011	Aug 20, 2011
3	Research processing methods (Picaxe)	A	Old	Complete	Sep 15, 2011	Aug 20, 2011
5	Program the Arduino to retrieve input from the sensors	R	Old	Complete	Oct 20, 2011	October 12, 2011
7	Integrate the input and output components to provide user feedback according to the microcontroller's calculations	R	Old	Delayed	Nov 10, 2011	January 29, 2011
9	Design an algorithm to detect approaching vehicles and calculate their relative speed	R	Old	Delayed	Nov 15, 2011	January 29, 2011
12	Achieve full control of the RC vehicle through the Arduino	A	Old	In progress	January 25, 2012	January 29, 2011

Detailed Explanation of Tasks:

The following tables explain the details of each listed task:

1 & 3: Research input methods (sensors) ; Research processing methods (Microcontroller)

- Category: Old
- Old Completion Date: Sep 15, 2011
- Date completed: August 20, 2011

Responsible/Assistant : Responsible for Certain Parts (See Actions)

Partners: Hani Hadidi, Valikhan Kuparov

Status at start of reporting period:

- Started

Status at end of reporting period:

- Completed

Actions

- I was responsible for researching ultrasonic sensors. I had researched various models ultrasonic sensors, and how they may be ideal for our project. I had researched mainly on the following models:
  - Maxbotix LV-EZ1, SRF05, DIY Ultrasonic Sensor
  - I had also researched an infrared sensor (Sharp IR Rangefinder) for possible secondary applications as shown in Appendix C
- For the microcontroller, I was responsible for researching the Picaxe microprocessor. I used various web sources to gather knowledge on this microcontroller, and evaluated its strengths and weaknesses in relation to out design. I summarized my findings for the team in the document shown in Appendix C.

Decisions

- The research of the sensors, and microcontroller was split up between the team by type. This allowed each of us to effectively and quickly learn about one alternative, and then discuss the pros and cons of each to decide on the type of sensor or microcontroller to use. As a consequence, we were able to move on the design stage from the research stage.

Testing & Verification, Final Results

- Not applicable

5: Program the Arduino to retrieve input from the sensors

- **Description***:* Design the circuit, and write the driver software to collect distance data from the ultrasonic sensor
- **Category**: Old
- **Old Completion Date:** October 20, 2011
- **Date completed or New Completion Date:** October 12, 2011

Responsible/Assisting : Responsible

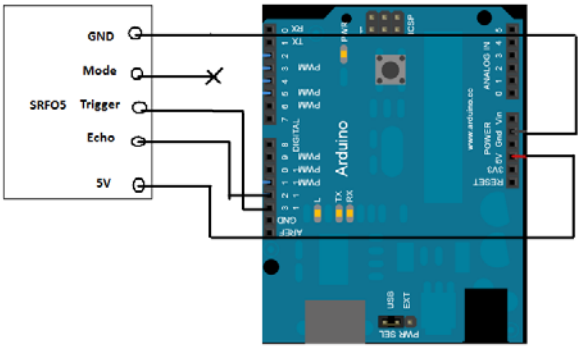
Partners: Valikhan Kuparov

Status at start of reporting period: Not started

Status at end of reporting period: Completed

Actions

- Using the technical specifications of the SRF05, the following circuit was constructed to allow interfacing between the Arduino and the SRF05 ultrasonic sensor:



- Code was written to drive the sensor w/ the microcontroller:
  - Instruct the sensor to send an ultrasonic pulse at intervals specified on the data sheet
  - The sensor returns a value using time of flight calculations. This value is converted using the specified rate to determine the distance between the sensor and the object in centimeters.
  - The distance is then displayed to the debug serial monitor.
  - Code Fragment to initiate the sensor (complete code is included in Appendix A)

```
digitalWrite (pin_trigger, HIGH); //drive trigger pin (10ms)
delayMicroseconds(TRIGGER_DELAY);//10ms wait
digitalWrite (pin_trigger, LOW); //stop sending pulses
pulseTime=pulseIn(pin_echo, HIGH); //get the returned pulse from srf05
```

Decisions

- There are 2 ways to interface with the SRF05. One method allows for 2 pins for trigger and echo (mode 1), and the other allows for a single pin for both trigger and echo (mode 2). We used mode 1 since its much simpler to program. In Mode 1, we can monitor 2 separate pins to verify the operation of the trigger, and echo signal, therefore there is no risk of interference. This allows easier control over the sensor, and therefore we picked this method. There are no drawbacks of this method for the scope of this project. Since, the distance measurements are the essence of the obstacle detection and anticipation module, we were able to complete it quickly, and begin working on relative speed calculations.

Testing & Verification, Final Results

- A box placed at known distances from the sensors, and the sensor's distance value was compared against the known value. The sensor is able to provide accurate measurements rounded to the nearest centimeter, for up to 3 meters. However, this doesn't meet our requirement of +-2.5% error, unless the object is at least 20 cm away. Since we are limited by the sensor's abilities, we will have to redefine out acceptance parameters to account for this issue. We are also considering having a second sensor to handle the cases when the objects are closer than a certain distance.

7: Integrate the input and output components to provide user feedback according to the microcontroller's calculations

- **Category**: Old
- **Old Completion Date:** November 10, 2011
- **Date completed or New Completion Date:** January 29, 2012

Responsible/Assisting: Responsible

Partners: Valikhan Kuparov

Status at start of reporting period:

- Not Completed

Status at end of reporting period:

- Delayed
- The distance and speed are displayed on the LCD screen. However, the time left to change lanes still needs to be displayed. In addition, the blinking LED lights indicating levels of danger still need to be implemented.

Actions

- I had assisted in this task by ensuring the correct variables representing the speed and distance calculated using the microcontroller and the sensor were being passed on to the LCD module. Here is a code snippet demonstrating this:

```
lcd.clear();
    // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 0);
  // print distance from obstacle

  lcd.print(current_distance);
  //lcd.print("");

  lcd.setCursor(0, 1);
  lcd.print(average speed);
```

- An example of the output can be viewed in Appendix A2.

Decisions

- This milestone was put on hold to work on the speed calculation algorithm. Previously, I had experimented with the circuitry, and the code required to blink an LED at different frequencies based on an input. I can simply transfer that module to work with our design. Due to the importance of the speed detection module, more focus was placed on that. As a result, this milestone has been delayed.

Testing & Verification, Final Results

- Values on the LCD were compared against the values on the serial monitor for verification. In additions, values on the LCD were compared against measured values. At the moment, we have an LCD module that accurately duplicates the microcontroller's calculations as targeted. Appendix A3 contains a screenshot showing the serial monitor used for testing.

9 : Design an algorithm to detect approaching vehicles and calculate their relative speed

- **Category**: Old
- **Old Completion Date:** November 15, 2011
- **Date completed or New Completion Date:** January 29, 2012

Responsible/Assisting : Responsible

Partners: Valikhan Kuparov

Status at start of reporting period:

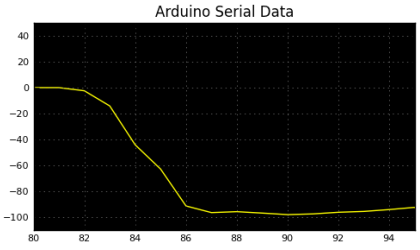
- Not started

Status at end of reporting period:

- Delayed
- Completed for the case of a stationary sensor (stationary user vehicle), needs to be verified when both vehicles are in motion

Actions

- Using the distance values provided from the sensor, I designed an algorithm to calculate the average speed of an obstacle. This was done by storing the distance values in an array, and then finding the difference in distance b/w consecutive array elements. An internal clock was used to measure the time difference b/w two sampling instances. Using the difference in time and distance, an instantaneous speed value was calculated. These instantaneous speeds are continually averaged to provide a meaningful speed value. The averaging factor was optimized through testing as shown in Appendix D2.
- I set up an external application plot the Arduino's serial data in real time. This was used extensively during verification.
- The following is a Speed vs. Time plot captured using the plotting application. The initial drop represents the car coming into the stationary sensor's field of vision. Since the speed is continuously being average it takes some time to settles to the accurate average speed of the car. Note that the values are negative since the obstacle is approaching the sensor. Extensive detail can be found in Appendix D.



x-axis: Sample Point y-axis: speed of obstacle in cm/s

- The following sample code demonstrates the aforementioned algorithm. Please refer to Appendix A1 to see the full code.

```
//do speed calculations and averaging
  delta_d = current_distance - old_distance; //difference b/w last two distances
  if (abs(delta_d) > MAX_DELTA_D ) {
      delta_d=0;

  }
  else {
    inst_speed= float(delta_d)/((float(elapsedTime))/float(1000000.00));
    total_speed=total_speed-speeds[s_index]; //get rid of last value in array
    speeds [s_index] = inst_speed;
    total_speed=total_speed+speeds[s_index];

    s_index +=1;

    if (s_index >= NUM_SPEED_READINGS ) {
      s_index=0;
    }

    average_speed=total_speed/float(NUM_SPEED_READINGS);
```

Decisions

- A lot of importance was placed on this milestone since it is essential to the obstacle anticipation module. As a result, work on the output module was put off for later. Also, progress on this module has been delayed since extensive testing is needed. Consequently, integration between the output and input module is also delayed. However, the first component of the automation module (control of the RC through the Arduino) is not affected since we have a second Arduino, and RC car.

Testing & Verification, Final Results

- Using the external plotting application, we plotted the speed of our obstacle as it approached the stationary sensor. During the testing, the speed averaging factor, and delay between subsequent sensor pulses was varied. These parameters affect the noise in the speed values, and the sensor's ability to react to changing speeds. Analysis of the resulting plots allowed us to find the right balance between a fast response, and an acceptable amount of noise. The stable portions of the plots were compared to known constant speed of the car, and it was seen that the percent error is slightly above 5%. Please refer to Appendix D for extensive detail regarding the verification procedure. At the moment, the anticipation module can somewhat accurately detect speeds while the sensor remains stationary.

12: Achieve full control of the RC vehicle through the Arduino

- **Category:** Old
- **Old Completion Date:** January 25, 2012
- **Date completed or New Completion Date:** January 29, 2012

Responsible/Assisting : Assisting

Partners: Hani Hadidi

Status at start of reporting period:

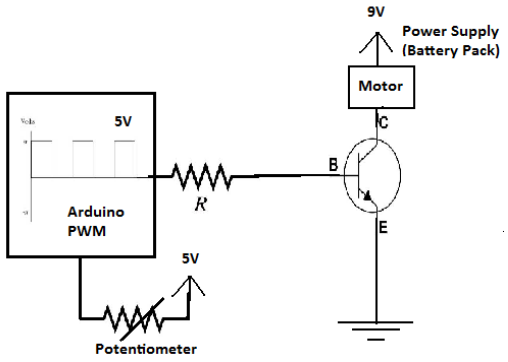
- Not completed

Status at end of reporting period:

- In progress
- Rotation of the motor for forward and backward movement is controllable, but steering control still need to be implemented and tested.

Actions

- I was assisting Hani complete this task. I've helped with the research of an external motor driver (Transistors), since the Arduino is not able drive a motor by itself. Using the research, I helped design and test the following circuit. Speed control of the forward movement is possible by varying the resistance of the potentiometer, which changes the duty cycle of the PWM, generating varying DC voltages:



- I've also written the code to generate the PWM signal needed to regulate the voltage, and thus the speed of the motor. My code monitors the voltage across a potentiometer, and converts that value to a duty cycle, which in turn controls the speed of the motor. Here is a snippet of the code. Appendix B contains further details on the operation of the motor.

```
void loop () {
  pot_val = analogRead(pot_pin);    // read the input pin
  pwm_val=pot_val/4;
  analogWrite (forward_Pin, pwm_val);
  Serial.println(pwm_val);             // debug value
  delay (1000);
}
```

Decisions

- Initially, the circuit BJT circuit shown earlier was to be duplicated, and used for steering, and backward control. However, this was decided to be inefficient, time consuming and expensive. As a result, an H-Bridge motor driver was used instead of the BJT. The H bridge driver is able to control the steering and the speed of the car through a single chip. We had to discard the BJT circuit, and therefore constructing it was somewhat wasteful. However, the PWM control circuitry with the potentiometer is still applicable to the H-Bridge circuit.

Testing & Verification, Final Results

- With the help of Hani, I have verified that the PWM signal is properly generated by connecting the signal to a scope, and observing it while varying the duty cycle. Once the overall circuit was constructed, I've verified that the pot's resistance corresponds to the duty cycle of the PWM, and the speed of the motor is properly regulated. The following is an example of the output seen at the Collector node, which is delivered to the motor:

Progress Assessment/ Conclusion:

So far, I have worked the following essential components of the device:

- Distance calculations
- Speed Calculations
- Communication b/w input and output module
- Speed control of the user vehicle's motor

I am limited by the sensor's ability to further improve its accuracy in measuring distance. Accuracy of speed calculations also need to be improved, but due to the shortage of time, further tests cannot be run on a stationary user vehicle. My first priority is to mount the sensor on the user vehicle, and verify the speed calculations. If necessary, I will attempt to the sensor's accuracy in measuring speeds. My next priority is to complete the time calculations. I have already written the code for it, but I need to verify it once the sensor is mounted. Once complete, I will find the optimal placement of the sensor to cover the blind spot region. I may need an additional sensor to aid in this.

I have made considerable progress on basic operations of the device, but there is much integration, and verification to be done. As a team, we are behind schedule, and I, including the rest of the team will have to work harder this term to complete the design in time without sacrificing any features.

Initial References

[1] Transport Canada, "Canadian Motor Vehicle Traffic Collision Statistics: 2009", Internet: http://www.tc.gc.ca/eng/roadsafety/resources-researchstats-menu-847.htm, Jun. 01, 2011 [Sep.11, 2011].

[2] Unknown, "What Causes Car Accidents? ", Internet: http://www.smartmotorist.com/traffic-and-safety-guideline/what-causes-car-accidents.html, Apr. 13, 2008 [Sep. 05, 2011].

[3] Unknown. "Lane Change and Lane Departure Warning". Internet: http://www.bmw.com/com/en/newvehicles/5series/sedan_active_hybrid/2011/showroom/safety/lane_departure_warning.html#t=l, Unknown date [Sep. 15, 2011].

[4] Emily Clark. "Active care safety features a top priority according to new research". *Gizmag*. Available: http://www.gizmag.com/go/8186/, *Oct. 16, 2007* [Sept 15, 2011].

[5] Unknown. "Blind Spot Information System (BLIS) with Cross-Traffic Alert". Internet: http://media.ford.com/images/10031/BLIS.pdf, Jun. 7, 2011 [Sep. 15, 2011].

[6] Ministry of Transportation of Ontario, "A guide to oversize/overweight vehicles and loads in Ontario" Internet: http://www.mto.gov.on.ca/english/trucks/oversize/guide.shtml, [Oct. 10, 2011].

[7] Y. Kolesnikov. "Blind spot". *Russian Bazaar.* 23(581). Available: http://www.russian-bazaar.com/Article.aspx?ArticleID=10506", Jun. 13, 2007 [Sep. 20, 2011].

[8] Unknown. "SRF05 - Ultra-Sonic Ranger: Technical Specification". Internet: http://www.robot-electronics.co.uk/htm/srf05tech.htm, [Jul. 25, 2011].

Appendices:

Appendix A: Detailed Data on the Input/output Module

This appendix contains details regarding the sensors, microcontroller, and the LCD screen.

A1. Current Code for Input/output Module

The code below performs the following functions. In addition to writing the code for the bolded components, I worked on the integration of the overall code.

- **Initiate the sensors**
- **Gather distance data from the sensors, and store a specified number of them in an array**
- **Calculate instantaneous speed using the difference in consecutive speed values, and the difference in Arduino's internal clock**
- Take the Arduino's calculations, and display the values on a LCD Screen
- Cause an LED to light with a certain brightness based on a specified variable

```
#include <LiquidCrystal.h>


const int NUM_READINGS =3;//number of readings for distance array (needs to be at least 2 for the speed calculations to work;
const int NUM_SPEED_READINGS =10;//number of readings for speed array [TEST CASE VARIABLE]
const int TIME_DELAY=15;//delay time after the entire process [TEST CASE VARIABLE]
const int TRIGGER_DELAY=10; //how long to wait for echo (don't change this from 10)
const int MAX_DELTA_D=40; //maximum allowed distance delta b/w two intervals to steady speed calculations

int total=0;
int a_index=0;
int current_index=0;
int average_distance=0;
int current_distance=0;
int distance [NUM_READINGS];

//for speed calculations
unsigned long startTime=0 ;                 // start time for stop watch
unsigned long elapsedTime=0;                  // elapsed time for stop watch
int old_distance=0;
int delta_d=0;
float inst_speed=0.0;
float average_speed=0.0;
float total_speed=0.0;
float speeds [NUM_SPEED_READINGS];
int s_index=0;

float time_left=0.0;    //time left until distance is zero



//hardware variables setup
//int pin_echo=2;  //srf05 echo pin (dig 2)
//int pin_trigger=3 ;  //srf05 trigger pin (dig 3)
//int pin_led_trigger=13; // (digital 13)

int pin_echo=12;  //srf05 echo pin (dig 2)
int pin_trigger=13 ;  //srf05 trigger pin (dig 3)

int pin_led_trigger=11;
int led_value=0;

//lcd definition
//LiquidCrystal lcd(12, 11, 7, 6, 5, 4);
```

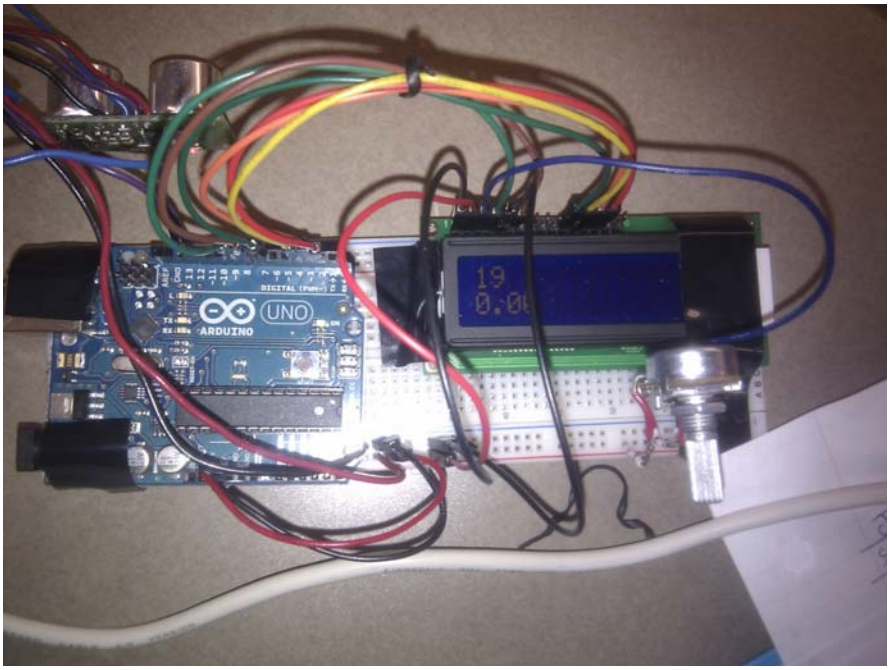
```
LiquidCrystal lcd(9, 8, 7, 4, 3, 2);

unsigned long pulseTime=0;  //stores pulse in microseconds


//setup
void setup () {
   pinMode (pin_trigger, OUTPUT);
   pinMode (pin_echo, INPUT);

   //LED setup
   pinMode (pin_led_trigger, OUTPUT); //sets dig. pin as output

   lcd.begin(16, 2); //sets up the lcd

   for (int cur_reading = 0; cur_reading < NUM_READINGS ; cur_reading ++) {
    distance [cur_reading] = 0;
  }

  //initialize speed array
   for (int cur_reading = 0; cur_reading < NUM_SPEED_READINGS ; cur_reading ++) {
    speeds [cur_reading] = 0.0;
  }
  Serial.begin(9600);
}



//external function to avoid rounding errors:
//found from:
//http://www.arduino.cc/playground/Code/PrintFloats
// printFloat prints out the float 'value' rounded to 'places' places after the decimal point
void printFloat(float value, int places) {
  // this is used to cast digits
  int digit;
  float tens = 0.1;
  int tenscount = 0;
  int i;
  float tempfloat = value;

    // make sure we round properly. this could use pow from <math.h>, but doesn't seem worth the import
  // if this rounding step isn't here, the value  54.321 prints as 54.3209

  // calculate rounding term d:   0.5/pow(10,places)
  float d = 0.5;
  if (value < 0)
    d *= -1.0;
  // divide by ten for each decimal place
  for (i = 0; i < places; i++)
    d/= 10.0;
  // this small addition, combined with truncation will round our values properly
  tempfloat +=  d;

  // first get value tens to be the large power of ten less than value
  // tenscount isn't necessary but it would be useful if you wanted to know after this how many chars the number will take

  if (value < 0)
    tempfloat *= -1.0;
  while ((tens * 10.0) <= tempfloat) {
    tens *= 10.0;
    tenscount += 1;
  }


  // write out the negative if needed
  if (value < 0)
    Serial.print('-');

  if (tenscount == 0)
    Serial.print(0, DEC);
```

```
  for (i=0; i< tenscount; i++) {
    digit = (int) (tempfloat/tens);
    Serial.print(digit, DEC);
    tempfloat = tempfloat - ((float)digit * tens);
    tens /= 10.0;
  }

  // if no places after decimal, stop now and return
  if (places <= 0)
    return;

  // otherwise, write the point and continue on
  Serial.print('.');

  // now write out each decimal place by shifting digits one by one into the ones place and writing the truncated value
  for (i = 0; i < places; i++) {
    tempfloat *= 10.0;
    digit = (int) tempfloat;
    Serial.print(digit,DEC);
    // once written, subtract off that digit
    tempfloat = tempfloat - (float) digit;
  }
}




void loop () {

  digitalWrite (pin_trigger, HIGH); //drive trigger pin (10ms)
  delayMicroseconds(TRIGGER_DELAY);//10ms wait
  digitalWrite (pin_trigger, LOW); //stop sending pulses



  pulseTime=pulseIn(pin_echo, HIGH); //get the returned pulse from srf05

  elapsedTime = micros() - startTime;
  startTime = micros();
  //float_t=float(elapsedTime)/float(1000000);



  current_distance = pulseTime/58; //conversion ratio according to spec sheet
  total=total-distance[a_index];
  distance[a_index]=current_distance;  //store current value of distance in distance array
  total=total+distance[a_index];



  if (a_index==0) {
    old_distance=distance[NUM_READINGS-1];
  }
  else {
    old_distance=distance[a_index-1];
  }

  a_index += 1;   //INCREMENT COUNTER

  if (a_index >= NUM_READINGS ) {
    a_index=0;
  }


  average_distance=total/NUM_READINGS;

  if (average_distance < 30) {
    led_value = 30 - average_distance;        // this means the smaller the distance the brighterthe LED.
```

```
 }

  analogWrite(pin_led_trigger, led_value);

 //do speed calculations and averaging
 delta_d = current_distance - old_distance; //difference b/w last two distances
 if (abs(delta_d) > MAX_DELTA_D ) {
    delta_d=0;

 }
 else {
   inst_speed= float(delta_d)/((float(elapsedTime))/float(1000000.00));
   total_speed=total_speed-speeds[s_index]; //get rid of last value in array
   speeds [s_index] = inst_speed;
   total_speed=total_speed+speeds[s_index];

   s_index +=1;

   if (s_index >= NUM_SPEED_READINGS ) {
     s_index=0;
   }

   average_speed=total_speed/float(NUM_SPEED_READINGS);

   //time left caculations

   if (average_speed != 0) {
     time_left = float(current_distance)/average_speed;
   }
   else {
     time_left=100;
   }

 }
 //if (average_speed != 0) {
   //Serial.println(current_distance, DEC);
   //Serial.print("     ");
  //printFloat(average_speed, 7);

  //printFloat(time_left, 5);
 //}

 //Serial.print("average_speed: ");
 printFloat(time_left, 7);
 //Serial.print("-----");
 Serial.println();

lcd.clear();
   // set the cursor to column 0, line 1
 // (note: line 1 is the second row, since counting begins with 0):
 lcd.setCursor(0, 0);
 // print distance from obstacle

 lcd.print(current_distance);
 //lcd.print("");

 lcd.setCursor(0, 1);
 lcd.print(average_speed);

 delay (TIME_DELAY);


}
```

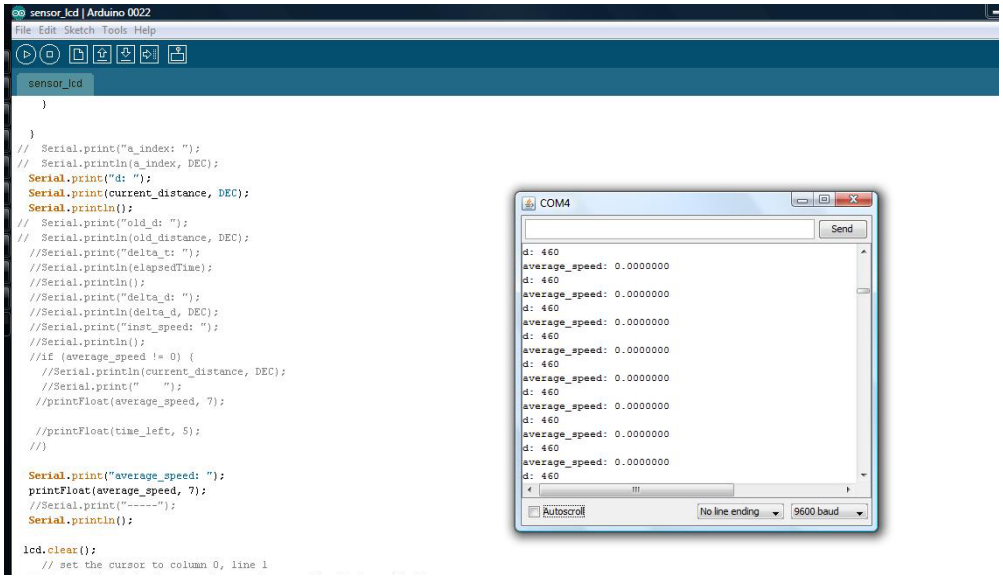
A2. Picture of the Physical Input / Output Circuit

The following picture shows the operational circuit containing the input/output module. The sensor is measuring a stationary object that is 19cm away.



A3: Serial Monitor

The following a screenshot of the serial monitor (COM4) used for debug. The code used to print variables on it is demonstrated on the left.



Appendix B: Details on Motor Control

This appendix contains the details of the circuit designed to control forward speed of the motor.

B1. Code to Control Forward Motor Speed

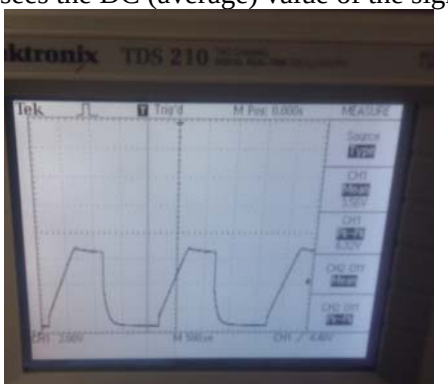
The following code:

- Monitors the value of the potentiometer
- Sets the duty cycle of the PWM signal according to the potentiometer's value

```
int pot_pin = 2;   //pin for the pot
int pot_val=0;//variable to store pot value
int pwm_val=0;
int forward_Pin = 9;  //PWM output for forward motor movement
int backward_Pin = 10; //PWM output for forward motor movement

void setup () {
  pinMode(forward_Pin, OUTPUT);
  pinMode(backward_Pin, OUTPUT);
  Serial.begin(9600);          //  setup serial

}

void loop () {
  pot_val = analogRead(pot_pin);    // read the input pin
  pwm_val=pot_val/4;
  analogWrite (forward_Pin, pwm_val);
  Serial.println(pwm_val);            // debug value
  delay (1000);
}
```

B2. Example Output of the Motor Control Circuit

The following is an example of the signal seen by the motor from the output. The duty cycle is set to 50%. The Square wave pattern represents the switching between 0V, and Vdd (~9V). Note that the motor sees the DC (average) value of the signal. The average value can be varied by varying the duty cycle.



Appendix C: Initial Research

Research was done by the team over the summer, and summaries of were written and stored in Google Docs. The following are some of my contributions:

Ultrasonic Sensors
July 25, 2011
Cheap Ultrasonic Range Finder (DIY Instructions + Code):

- We may be able to build this ourselves for our prototype
- http://www.micro-examples.com/public/microex-navig/doc/090-ultrasonic-ranger.html
    - Pros:
        - cheap
        - may be challenging and fun to build
        - source code is given
    - Cons:
        - extra work
        - only 200 cm range

Ultrasonic Range Finder - Maxbotix LV-EZ1

- Documentation, and demo can be found at the following link:
    - http://www.sparkfun.com/products/639
- We can place this sensor at certain angles to check if anything is in the blind spot of a car
- we can pick a variation of this product to vary the beam angle necessary for our application
- as shown in the demo, the out is an led display, we should be able to easily get this into a computer, and perform calculations to notify the user if something is in their blind spot
- Here is the companies sensor selection guide. Can be helpful if we have time to extend the project to be used on real cars:
    - http://www.sparkfun.com/datasheets/Sensors/Proximity/Sensor_Selection_Guide.pdf
- We may be able to place a few of these sensors around the car, and then design an algorithm to map out the car's surroundings, then decide whether or not it is safe to change lanes
- Pros
    - easy to use
    - very small, can fit on a mid sized RC car
    - company has good documentation we can use to get started, and continue
    - affordable ($25 each)
- Cons
    - uses RS232 interface instead of TTL (standard)

Robot Wall Racers (SRF05)

- Car is able to detect walls/obstacles, and make turns accordingly, we can use the same technology to detect cars in blind spot, and we can even extend the idea to detect cars in front or behind
    - http://letsmakerobots.com/node/696
- This is the sensor he used (**SRF05)** :
    - http://www.robot-electronics.co.uk/htm/srf05tech.htm
- Here is a demo of the SRF05 being used with an Arduino:
    - http://luckylarry.co.uk/arduino-projects/arduino-sonic-range-finder-with-srf05/
- I am thinking we can use this technology to extend our project
    - If a vehicle is in the blind spot, cut off steering in that direction
    - if a vehicle is in front, apply brakes
- Pros:
    - 3-4m range, good for anticipating cars in blind spot
    - small, should be easy to place
    - multiple projects have already been made with it, we can use their steps
    - the sensor is apparently really popular, we can find lots of guides/code etc online
    - $30
- Cons
    - unsure if it will work well with both cars in motion

Current Blind Spot Assists

- Audi
  - Uses sensors mounted on the rear bumper to detect oncoming vehicles in the adjacent lanes
- Volvo
  - takes 25 pictures/s
  - Doesn't work in harsh weather conditions (fog, rain)
- http://www.tsfbcaa.com/pdf/BlindSpot.pdf

Other Options
Infrared Sensors

- Sharp IR Rangefinder
  - http://www.societyofrobots.com/sensors_sharpirrange.shtml
  - Uses infrared light and triangulation to measure distance
  -
  - Pros:
    - $10 - $20
    - low power consumption
    - immune to interference from ambient light
    - very accurate
    - has a football shaped sensing area, as opposed to a cone (perhaps good for detecting vehicles in blind spot)
    - **good at detecting moving objects!**
  - Cons
    - very narrow bean width
      - have to directly point at the object

doesn't work when object is too close, thus it cannot be used for blind spot check, but possibly to anticipate the car going towards the blind spot

cross interference is possible, thus multple sensors is an issue

Picaxe

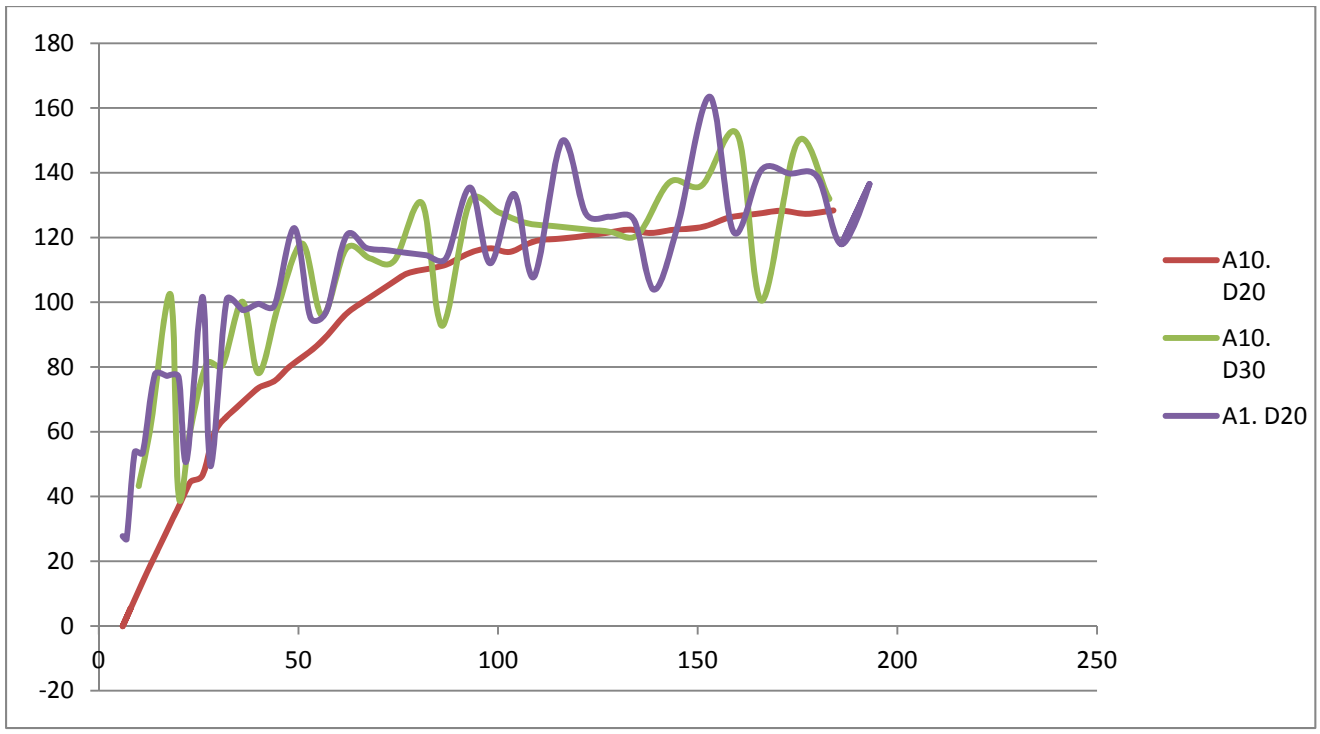
- Main site:
  - http://www.rev-ed.co.uk/picaxe/

- Advantages
  - comes with a free software that allows "flowcharting"
    - no programming necessary
  - is able to program very quickly
  - since it is aimed towards a younger audience, programming it ma be a very easy task
- Picaxe products
  - http://www.rev-ed.co.uk/docs/cat_01.pdf
- Various picaxe projects:
  - http://www.instructables.com/group/PICAXE%20Projects/
  - http://letsmakerobots.com/taxonomy/term/97
- The robot wall racer is built using Picaxe, we can follow this tutorial if we wanted to use the Picaxe uController
- Here is the Picaxe manual
  - http://www.rev-ed.co.uk/docs/picaxe_manual1.pdf
  - refer to pages 7-12 for technical specs, and layout diagrams
  - We can even use this document for general reference. There seems to be a lot of information on micro-controllers beginners (i.e: page 12)
- Disadvantages
  - This device seems to be aimed towards children. Therefore, I expect it to lack advanced functionality

Appendix D: Speed Measurement Tests with Stationary Sensor

D1. Initial Tests

The following graphs represent the speed of the obstacle against time as it moves away from the sensor. The data was collected from the Arduino's serial monitor, and then exported to Excel to create the following graphs. Note that this test captures the speed as the obstacle is moving away from the sensor, and therefore, we have positive speeds. The averaging factor (A), and delay time (D) are varied.



Y-axis: Speed of Obstacle in cm/s

X-axis: Sample Point

A=number of stored instantaneous speed averages;

D= delay between consecutive pulses;

Using this initial data, we got an estimation of the A, and D required to produce an acceptable output. Therefore, further tests were done to check the outputs near the range of A=10,D=10.

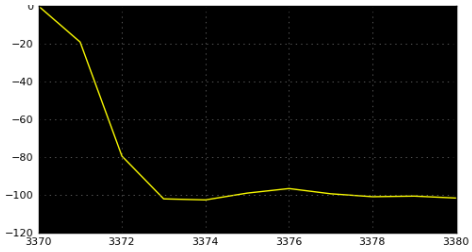
D2. Further Tests

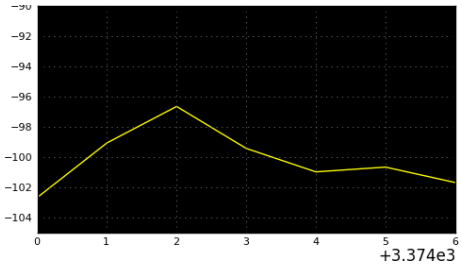
The following cases were tested:

A5, D15	A10, D15	A15, D15
A5, D20	A10, D20	A15, D20
A5, D25	A10, D25	A15, D25

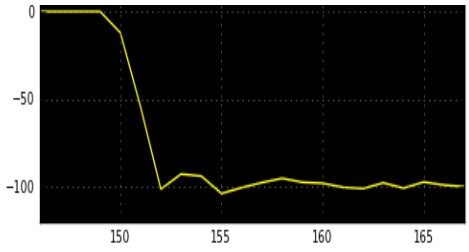
To aid in the collection of data, I researched online, and found an open-source application to plot the Arduino's serial data in real time. Using this tool, we obtained the following plots for the aforementioned test cases. Note that these plots represent the obstacle approaching the sensor, and as a result, the speeds are negative. The left graph is a capture of the entire time the car is in the sensor's range, and the right graph is zoomed in look at the flat portion representing the constant average speed of the car. The falling slope represents the time needed for the sensor to adjust to the speed of the car. In an ideal case, the drop would be instant, since sensor reads 0 until the car is in the sensor's range of view. However, since the calculated speed is a moving average, there is a fall time.

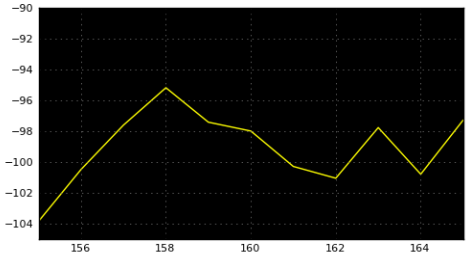
A5, D15



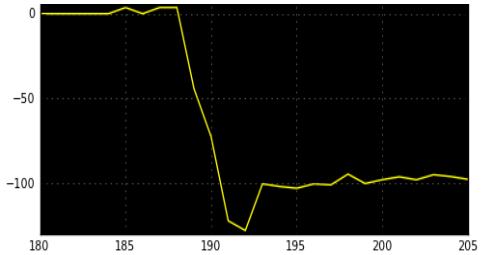


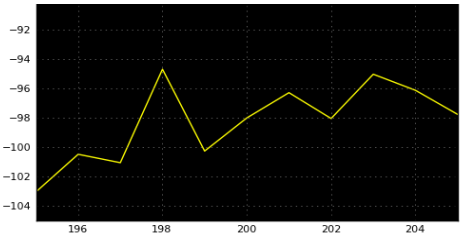
A5, D20



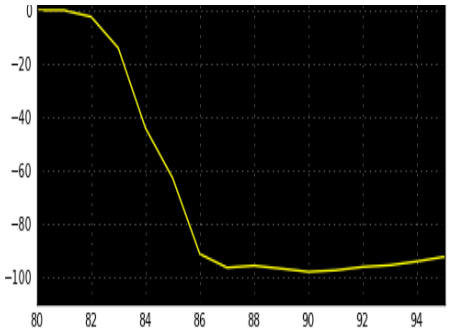


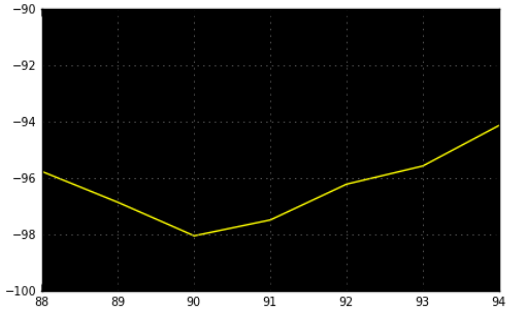
A5, D25



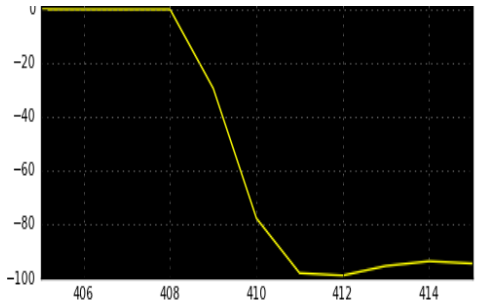


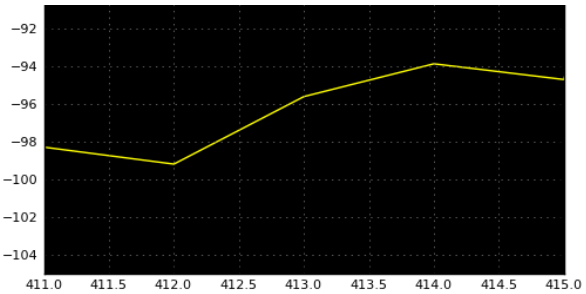
A10, D15



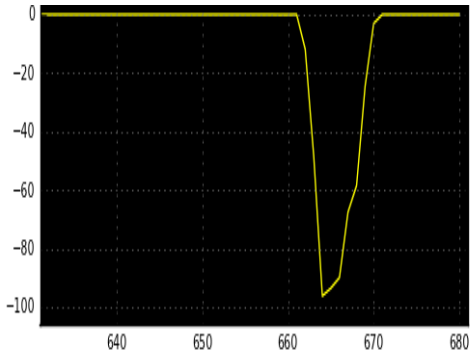


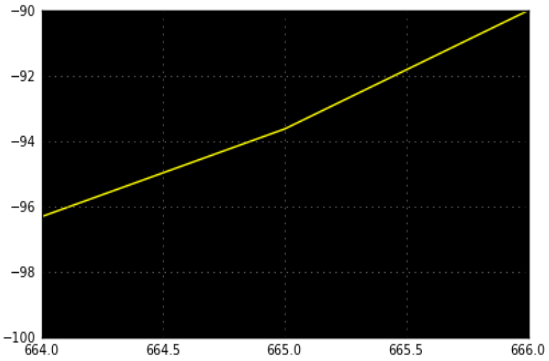
A10, D20



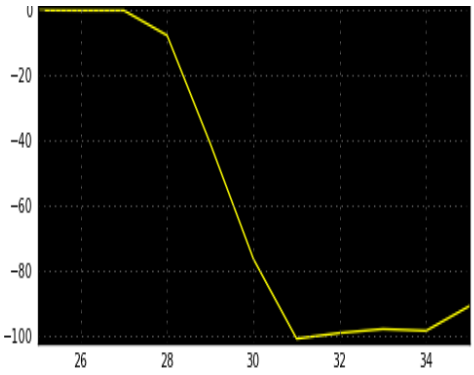


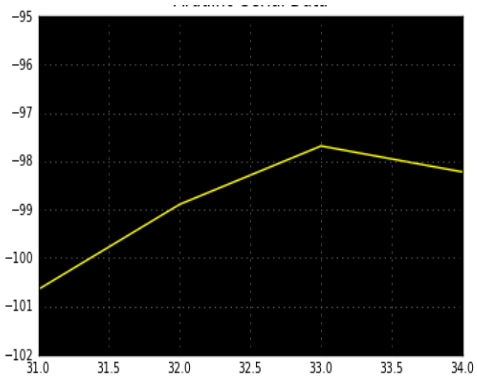
A10, D25



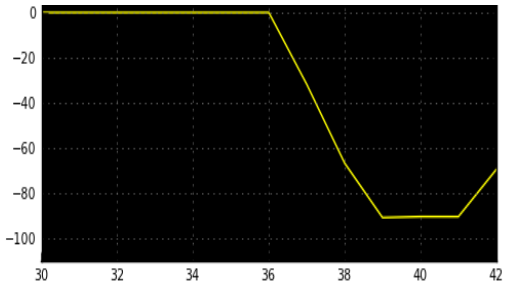


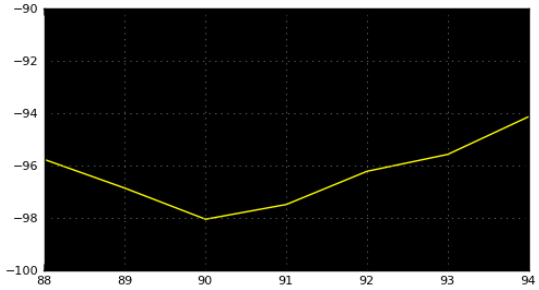
A15,D15



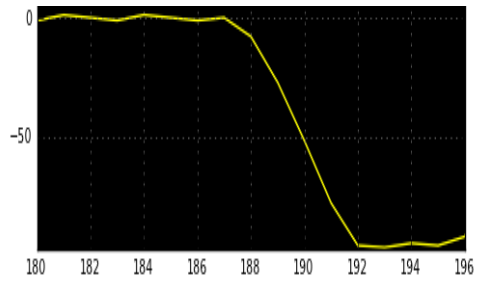


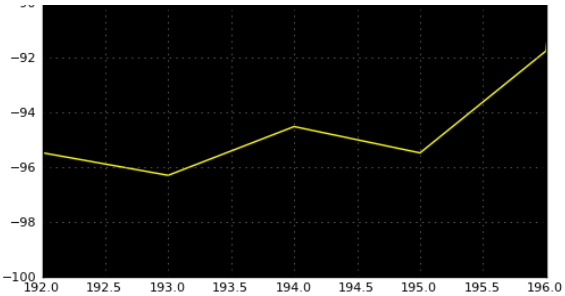
A15, D20





A15, D25





After analyzing the plots, we have decided to use A10,D20. It provides the smooth output, while maintaining a fast response time. This is desired since output must be steady to make sense to the user, and at the same time, it must be able to pick up sudden changes in speeds.

It is important to note that our decision may change once we mount the sensors on the user vehicle. The vibration of the user vehicle will introduce further noise, and we may need to experiment with the parameters again to ensure a desirable, while accurate response.

D3. Comparison with Actual Speeds

We have tested our sensor's accuracy by measuring the speed of the obstacle RC car, and then comparing it with the flat portion A10, D20 plot. The speed of the obstacle was varied by running it with:

- 2 Batteries
- 3 Batteries

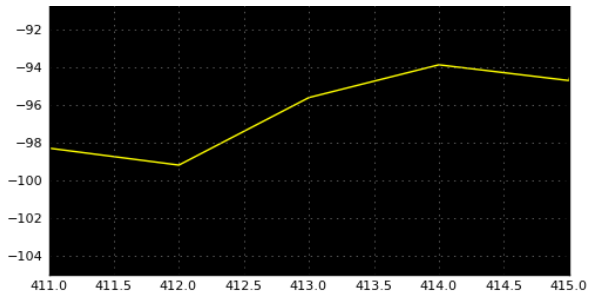
The actual average speeds of the obstacle were measured by observing how long it takes to travel a known distance. The results are summarized below:

2 Batteries

Trial	Distance (cm)	Time Taken (s)	Speed (cm/s)
1	300	3.12	96.15
2	300	3.28	91.46
3	300	3.22	93.16
4	300	3.27	91.7
5	300	3.31	90.63

Average Speed = 92.6 cm/s

The flat portion of the A10, D20 plots shows:



We obtain the following values from the plot:

Max Speed = 99.4 cm/s

Min Speed = 94.0 cm/s

Mean Value = (99.4+94.0)/2 = 96.7 cm/s

Now we can perform the % error calculations:

Max % Error = [(99.4-92.62)/92.6] *100% = 7.32%

Average % Error = [(96.7-92.6)/92.6] *100% = 4.43%

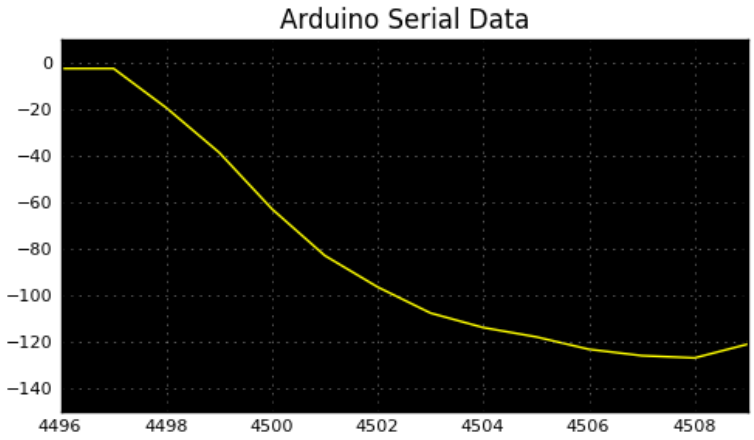
This is above our constraint of 5% allowable error by a small margin.

3 Batteries:

Trial	Distance (cm)	Time Taken (s)	Speed (cm/s)
1	360	2.82	127.66
2	360	2.48	145.16
3	360	2.60	138.46
4	360	2.68	134.32
5	360	2.77	129.96

Average Speed = 135 cm/s

Unfortunately, we do not have the zoomed in plot for this case. However, the normal plot for the A10, D20 case shows:



The plot stabilizes at around sample point 4506. Using the period between 4506, and 4510, we calculate:

Max Speed = 128 cm/s

Min Speed = 121 cm/s

Mean Value = (128+121)/2 = 125 cm/s

Now we can perform the % error calculations:

Max % Error = [(121-135)/135] *100% = 10.4%

Average % Error = [(125-135)/125] *100% = 7.41%

This is above our constraint of 5% allowable error.

As a summary:

Batteries	**Avg. Measured Speed (cm/s)**	**Mean Value from Sensor's Data Plots (cm/s)**	**Max. % Error**
2	**92.6**	**96.7**	**6.82**
3	**135**	**125**	**10.4**