The Edward S. Rogers Sr. Department of

Electrical and Computer Engineering

University of Toronto

ECE496Y Design Project Course

Group Final Report

Title: Vehicle Blind Spot Assist		
Project I.D.#:	2011317	
Team members: (Select one member to be the main contact. Mark with '*')	Name:	Email:
	Tanzim Mokammel*	tanzim.mokammel@utoronto.ca
	Valikhan Kuparov	valikhan.kuparov@utoronto.ca
	Hani Hadidi	Hani.hadidi@utoronto.ca
Supervisor:	M. Stickel	
Section #:	7	
Administrator:	R. Gillett	
Submission Date:	March 22, 2012	

Group Final Report Attribution Table

This table should be filled out to accurately reflect who contributed to each section of the report and what they contributed. Provide a **column** for each student, a **row** for each major section of the report, and the appropriate codes (e.g. 'RD, MR') in each of the necessary **cells** in the table. You may expand the table, inserting rows as needed, but you should not require more than two pages. The original completed and signed form must be included in the hardcopies of the final report. Please make a copy of it for your own reference.

Section	**Student Names**			
	Tanzim	Hani	Valikhan	
Executive Summary	RD	ET		
Project Description	MR	RD	ET	
Final Design	RD, MR	ET	ET	
Validation and acceptance test	ET	ET	RD, MR	
Summary and conclusion	RD, MR	ET	ET	
All	FP	CM	FP	

Abbreviation Codes:

Fill in abbreviations for roles for each of the required content elements. You do not have to fill in every cell. The "**All**" row refers to the complete report and should indicate who was responsible for the final compilation and final read through of the completed document.

Signatures

By signing below, you verify that you have read the attribution table and agree that it accurately reflects your contribution to this document.

Name Tanzim Mokammel **Signature** **Date:** March 22, 2012

Name Valikhan Kuparov **Signature** **Date:** March 22, 2012

Name Hani Hadidi **Signature** **Date:** March 22, 2012

Voluntary Document Release Consent Form[1]

To all ECE496 students:

To better help future students, we would like to provide examples that are drawn from excerpts of past student reports. The examples will be used to illustrate general communication principles as well as how the document guidelines can be applied to a variety of categories of design projects (e.g. electronics, computer, software, networking, research).

Any material chosen for the examples will be altered so that all names are removed. In addition, where possible, much of the technical details will also be removed so that the structure or presentation style are highlighted rather than the original technical content. These examples will be made available to students on the course website, and in general may be accessible by the public. The original reports will not be released but will be accessible only to the course instructors and administrative staff.

Participation is completely voluntary and students may refuse to participate or may withdraw their permission at any time. Reports will only be used with the signed consent of all team members. Participating will have no influence on the grading of your work and there is no penalty for not taking part.

If your group agrees to take part, please have all members sign the bottom of this form. The original completed and signed form should be included in the hardcopies of the final report.

Sincerely,

Phil Anderson

ECE496Y Course Coordinator

Consent Statement

We verify that we have read the above letter and are giving permission for the ECE496 course coordinator to use our reports as outlined above.

Project ID: __2011317______ Project Title: __Vehicle blind spot assist__________

Supervisor: ___M. Stickkel___ Administrator: _____R. Gillett_________________

Name	Tanzim Mokammel	**Signature**		**Date:**	March 22, 2012
Name	Valikhan Kuparov	**Signature**		**Date:**	March 22, 2012
Name	Hani Hadidi	**Signature**		**Date:**	March 22, 2012

[1] This form will be detached from the hardcopy of the final report. Please make sure you have nothing printed on the back page.

ECE496Y Final Report Individual Evaluation Form (one sheet per student)

Student			
Project ID: 2011317	**Project Title:** Vehicle Blind Spot Assist		
Student Name: Tanzim Mokammel		**Supervisor:** M. Stickel	
Section: 7	**Administrator:** R. Gillett		

Administrator's Evaluation Provide a rating for each section. Circle to indicate problem areas.	Excellent	Good	Adequate	Marginal	Poor/unclear	**Comments** *(also see comments in report)*
Document						
Introduction: clear background, motivation, goals, requirements						
Final Design: system diagram, system and module level descriptions, assessment of strengths and weaknesses						
Testing and Verification: adequate documentation, discussion of results, comparison of results to requirements						
Summary and Conclusions: summary of accomplishments and challenges, success in achieving project goals						
Presentation: clear writing, grammar, organization, use of tables, diagrams, figures						
Project						
Final outcome: success in achieving stated project goals, technical complexity						
Individual effort and contributions						

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

General Comments:

Note: Supervisor final evaluations are done online. See Supervisor > Final Evaluation on the course menu at http://int.ece.utoronto.ca/ece496.xxyy/index.html, where xxyy are the years (ex 0809 or 0910)

ECE496Y Final Report Individual Evaluation Form (one sheet per student)

Student			
Project ID: 2011317	**Project Title:** Vehicle Blind Spot Assist		
Student Name: Valikhan Kuparov		**Supervisor:** M. Stickel	
Section: 7	**Administrator:** R. Gillett		

Administrator's Evaluation

Provide a rating for each section. Circle to indicate problem areas.

Comments *(also see comments in report)*

Document	Excellent	Good	Adequate	Marginal	Poor/unclear
Introduction: clear background, motivation, goals, requirements					
Final Design: system diagram, system and module level descriptions, assessment of strengths and weaknesses					
Testing and Verification: adequate documentation, discussion of results, comparison of results to requirements					
Summary and Conclusions: summary of accomplishments and challenges, success in achieving project goals					
Presentation: clear writing, grammar, organization, use of tables, diagrams, figures					
Project					
Final outcome: success in achieving stated project goals, technical complexity					
Individual effort and contributions					

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

General Comments:

Note: Supervisor final evaluations are done online. See Supervisor > Final Evaluation on the course menu at http://int.ece.utoronto.ca/ece496.xxyy/index.html, where xxyy are the years (ex 0809 or 0910)

ECE496Y Final Report Individual Evaluation Form (one sheet per student)

Student			
Project ID: 2011317	**Project Title:** Vehicle Blind Spot Assist		
Student Name: Hani Hadidi		**Supervisor:** M. Stickel	
Section: 7	**Administrator:** R. Gillett		

Administrator's Evaluation

Provide a rating for each section. Circle to indicate problem areas.

Comments *(also see comments in report)*

Document	Excellent	Good	Adequate	Marginal	Poor/unclear
Introduction: clear background, motivation, goals, requirements					
Final Design: system diagram, system and module level descriptions, assessment of strengths and weaknesses					
Testing and Verification: adequate documentation, discussion of results, comparison of results to requirements					
Summary and Conclusions: summary of accomplishments and challenges, success in achieving project goals					
Presentation: clear writing, grammar, organization, use of tables, diagrams, figures					
Project					
Final outcome: success in achieving stated project goals, technical complexity					
Individual effort and contributions					

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

General Comments:

Note: Supervisor final evaluations are done online. See Supervisor > Final Evaluation on the course menu at http://int.ece.utoronto.ca/ece496.xxyy/index.html, where xxyy are the years (ex 0809 or 0910)

Executive Summary

Vehicle accidents cause significant damage to one's health, and finances. One factor contributing to such accidents arises from users being unaware of obstacles in their blind spot. As a result, lane changing becomes a dangerous maneuver. To account for this issue, we are prototyping a system that will warn users of obstacles present, or approaching their vehicle's blind spot, as well as preventing them from making unsafe lane changes.

The design is intended to be used on a scaled down remote controlled vehicle. It consists of four main modules: obstacle detection, output, automation, and a remote graphical user interface (GUI). The obstacle detection module uses two sensors, and a microcontroller to identify obstacles present or approaching the user vehicle's blind spot. The microcontroller calculates the obstacle's position, relative speed, and the amount of time required for it to reach the blind spot (if not already present) using the sensor's distance values, and the trigger rate. Finally, the automation module allows the microcontroller to assume control of the user vehicle. The purpose of this module is to prevent collisions by locking the steering of the user vehicle towards the obstacle. The GUI allows remote operation of the system for demonstration and verification. It plots the microcontroller's calculations in real-time in addition to mirroring the onboard Light Emitting Diode (LED), and Liquid Crystal Display (LCD). It also allows for full remote control over the speed, and direction of the user vehicle.

Module level, as well as system level tests was run to verify the design. Based on these tests, the system meets the initially set requirements. It successfully determines the danger associated with a possible lane change, and warns the user accordingly. If necessary, it activates the steering lock to prevent a collision.

Future iterations of the design can be improved by replacing the current sensors with more advanced alternatives, as many of the system's shortcomings stem from the noise on the sensor data. This issue was compensated with an averaging algorithm, but the system's response time was compromised as a result. The automation module can also be improved by introducing counter-steering mechanisms as opposed to steering-locks.

Group Highlights

The project is broken down into 3 distinct modules:

1: Obstacle Detection Module

Tanzim Mokammel (TM), with the assistance of Valikhan Kuparov (VK) were responsible for this module. The purpose of this module is to utilize the distance data from the sensors, and use it to perform relative speed, and time remaining calculations for determining danger levels, and notifying the user. The module is able to adequately perform these calculations within a certain degree of error (as shown in test results). A major challenge in this module was the treatment of noise on the sensor data. To overcome this, TM and VK implemented various averaging and sample and hold algorithms. The result is much more stable, though somewhat inaccurate and slow calculation of desired values. The module has been tested in cases where the sensor remains stationary, as well as where the sensor is mounted on the user vehicle. The mounting of the sensor required more aggressive averaging, and thus slower acquisition of data.

2. Output Module

VK, with the assistance of TM is responsible for this module. The LCD display is fully functional, and is able to display the distance, relative speed, and time remaining calculations from the microcontroller. The LED light has also been implemented, but it will use varying brightness levels, as opposed to blinking frequencies to represent danger levels. The reason for this is that blinking the LED requires delays to be added to the operation of the microcontroller, and this adversely affects the data acquisition of the sensor. In addition to the LCD screen and the LED, TM has designed an external GUI on a personal computer (PC) to remotely display all the relevant information to the user.

3. Automation Module

Hani Hadidi (HH), with the assistance of TM is responsible for this module. Complete control of the user vehicle has been implemented locally through the slave Arduino. In addition, the user is able to remotely control the car through the designed GUI. The module has also been integrated with the obstacle detection/anticipation module to have an operating steering lock function based on the danger level.

Individual Contribution – Tanzim

My overall responsibility was to develop the obstacle anticipation and detection module. Initially, I researched various models of ultrasonic sensors, as well as types of microcontrollers. Once a sensor and microcontroller were chosen, I implemented the circuitry, and wrote the software required to drive the sensor, and gather distance data from it. With the help of Valikhan Kuparov, I have developed the software required to use the distance data in performing relative speed, as well as time remaining calculations. Developing the speed and time calculation algorithms took multiple iterations due to noise issues from the sensor. In the end, I was able to fine tune the algorithm to achieve satisfactory results as demonstrated by the test results.

In addition to working on the obstacle detection module, I recognized that the initially planned implementation of the output module was not sufficient. Therefore, I worked on developing the external GUI using Processing. The GUI is used to gives the user real-time control of the user vehicle, while displaying real-time plots of the distance, speed, and time calculations. The GUI's main purpose is meant to demonstrate the project, but it has also been instrumental in debug, and testing.

I have also contributed to certain components of the automation module. The GUI I developed communicates with Hani Hadidi's control mechanism to provide remote control of the user vehicle using a personal computer (PC). I had assisted in the initial development of the automation module by designing a transistor driven circuitry to control the speed of the motor. Although a better design was later used, the initial design helped us move on the final design using a half-bridge motor driver. I had also assisted Hani debug the motor driver code, and assisted in implementing the steering lock feature by designing the danger look-up table, and integrating the motor driver code with the obstacle detection/anticipation code.

In summary, my main contributions were the obstacle detection/anticipation module, and the external GUI. In addition to this, I have assisted with the automation module, and the overall integration of the final design.

Individual Contribution - Valikhan

My main responsibilities for this project included designing, making and implementing the output module. At early stages the components of the output module were chosen. After the analysis of pros and cons of the different options, an implementation of the LED light and the LCD display was selected. This choice best provides informative, intuitive and numerical feedback to user driver. Some of my responsibilities for the output module consist of:

- ✓ Choosing output technology
- ✓ Choosing location and angles of placement on the user car in order to cover the blind spot
- ✓ Integrating the system of the LED light and LCD display with the microcontroller as well as writing the code

In addition to the output module, I helped Tanzim to implement the Obstacle detection/anticipation module. Here I assisted him with overall design of algorithm to calculate a relative speed and a remaining time. Once algorithm was completed, we started writing the code for microcontroller. At the same time we worked on circuitry construction. After circuitry setup and coding was accomplished we run series of tests to verify desired performance of the system, and where necessary to improve it.

Task	Description
Output Module	• Implement LED light and LCD display • Wrote a code for Modularized the code into individual functions
Obstacle detection/anticipation	• Chose sensor technology (Infrared) and model (Sharp GP2Y0A21YK) • Planned the placement and orientation of sensors • Assisted integrating the module • Assisted designing algorithm and writing a code
Testing	• Came up with test cases plan • Conducted testing for three modules and integrated system

Individual Contribution - Hani

My main contributions to the project involved making, implementing and designing the automation module. Few of the main functionalities for the automation include:

- Simulating Driving
- Steering lock mechanism
- Controlling relative speed
- Verify anticipation module

Before deciding on the actual module, one very key component needed to be chosen which potentially affects all other systems, the microcontroller and the remote control cars (RC-cars) for both the obstacle and the user vehicles. The microcontroller that was chosen to accommodate all the modules, including the RC-cars was the Arduino microcontroller.

In addition to the automation module, I also worked in assisting my team members with various tests while performing automation module testing as well as during system integration. The table below is a summary of my contributions with a brief description.

Task	Description
Microcontroller	• Designing the hardware and software to fully control the RC-car after taking out the initial PCB installed
Automation Module	• Designing the circuit and writing the code for the Arduino to communicate with the RC-car motors • Controlling the speed of the RC-car by modifying the PWM signal
Interface	• Designing and implementing the code for user interface (keyboard) • Controlling the speed according to various percentages of the PWM-signal
Integration/ Testing	• Integrating the hardware from the anticipation and detection modules, with the automation module onto one breadboard • Assisting the team with the testing to verify functionality • Debugging problems associated with the final integration

Table 1: Summary of tasks

Acknowledgements

With the completion of our design project and soon our term at the University, we would like to take this opportunity to thank the following individuals and resources for their support and contributions:

- Professor Micah Stickel, for constantly providing guidance and mentorship throughout the project.
- Professor Ross Gillett, for providing administrative support and feedback throughout the project.
- Professor Khoman Phang, for providing support during lectures that explained the guidelines for the different documentation throughout the semester.
- Mike Mehramiz, Design Center, for granting access to tools and other resources needed
- Lawrence Chan, Creatron, for providing most of the purchased components.

In addition to the individuals mentioned above, the completion and success of our project would not have been possible without the testing equipment and the facilities from the Electrical and Computer Engineering Department within the University of Toronto.

Table of Contents

1. Project Description

1.1 Background and Motivation (Tanzim Mokammel, Hani Hadidi)

Vehicle accidents cause physical injuries, mental trauma, financial loss, and in the most extreme cases: death. According to Statistics Canada, in 2009, there were 123,192 total recorded collisions in Canada. 2,011 of these were fatal [1]. In 1990, total number of collisions was 178,515; 3,445 of which were fatal [1]. As further indicated by Canadian accident statistics (Appendix D), the numbers have decreased dramatically. This decrease can be largely credited to advancements in vehicle safety technology, among other factors. Though there is a decrease, as the numbers indicate, accidents are still a significant issue.

It has been found that 95% of accidents are caused by bad driving behavior [2]. One common driving behavior is to not check the vehicle's blind spot while changing lanes, which leads to accidents during lane changes. To account for this issue, car manufacturers such as BMW [3], Volvo [4], and Ford [5] are introducing vehicle blind spot assist technologies in their new vehicle models. However, not every driver is able or wants to attain these vehicle models. Therefore, there is a need for a universal and affordable device that can aid users in becoming aware of obstacles in their blind spot. Advances to the currently available technology also need to be made to further prevent accidents in general.

1.2 Project Goal (Valikhan Kuparov)

The goal of the project is to aid in accident prevention through a device that will aid users identify and anticipate obstacles in their vehicle's blind spot. In addition, the device will be able attain vehicle control to prevent the user from steering into obstacles in the blind spot. In order to achieve this, the group will build a prototype to be tested on scaled down vehicles (i.e. remote controlled cars) under simulated road conditions.

1.3 Functional Requirements (Valikhan Kuparov)

1. The unit shall have sensors mounted on the sides (near the rear) of the user vehicle, such that it covers the defined blind spot region (Appendix F), and is able to obtain the distance to the obstacle and its position within the blind spot

2. The unit shall have a microcontroller to compute the following based on the inputs:

    a. The relative speed of the obstacle to the user vehicle within ±10% accuracy

    b. The amount of time needed for the approaching obstacle to appear in the blind spot, if not already there

3. The unit shall have a feedback mechanism to warn the user of potential danger based on the microcontroller's calculated relative speed and time allowed for a lane change (Appendix G, Appendix H)

4. The unit shall have an override mechanism that will prevent users from making unsafe lane changes by locking the steering of the user vehicle towards the obstacle under conditions outlined in Appendix H

5. The Unit shall adhere to the required technical parameters (Appendix E)

2. Final Design

2.1 System-Level Overview (Hani Hadidi)

The final design is composed of the four main modules:

- Obstacle Detection/Anticipation
- Output
- Automation
- Remote Graphical User Interface (GUI)

The diagram below outlines these modules:

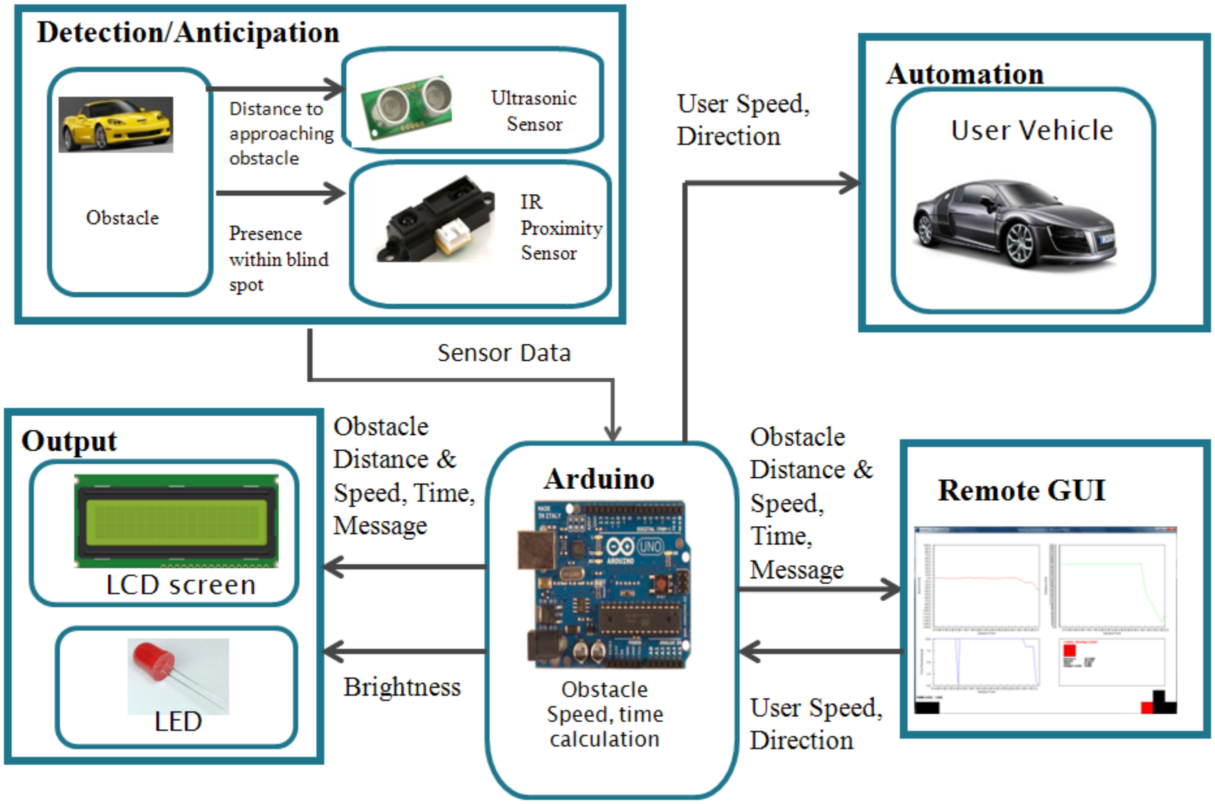


Figure 1: System Level Overview

The core component of the design is a microcontroller, as it is responsible for driving the sensors to gather data from the obstacles, and determine the safeness of the lane change. In addition, it will trigger the automation module in disabling the steering of the vehicle to prevent unsafe lane changes. The output will provide both qualitative and quantitative information to the driver regarding the situation. The remote GUI allows a user to visualize the microcontroller's calculations through graphs, and also lets them control the prototype user vehicle. Details of each module can be found in the following section:

2.2 Implementation (Tanzim Mokammel)

To implement and integrate the 4 modules, we designed 2 major parts. This section will describe these two parts, and the considerations that went into design of each.

1. Onboard Implementation
    a. Hardware Schematic

b. Description of Components

c. Additional Sensor and Considerations

d. Output Module and Considerations

e. Automation Module and Considerations

f. Onboard Software

2. Remote GUI for Control and Output

2.2a Onboard Implementation

This subsection describes the hardware and software within the onboard design.

Hardware Schematics

The following is the schematic for the hardware of the design:

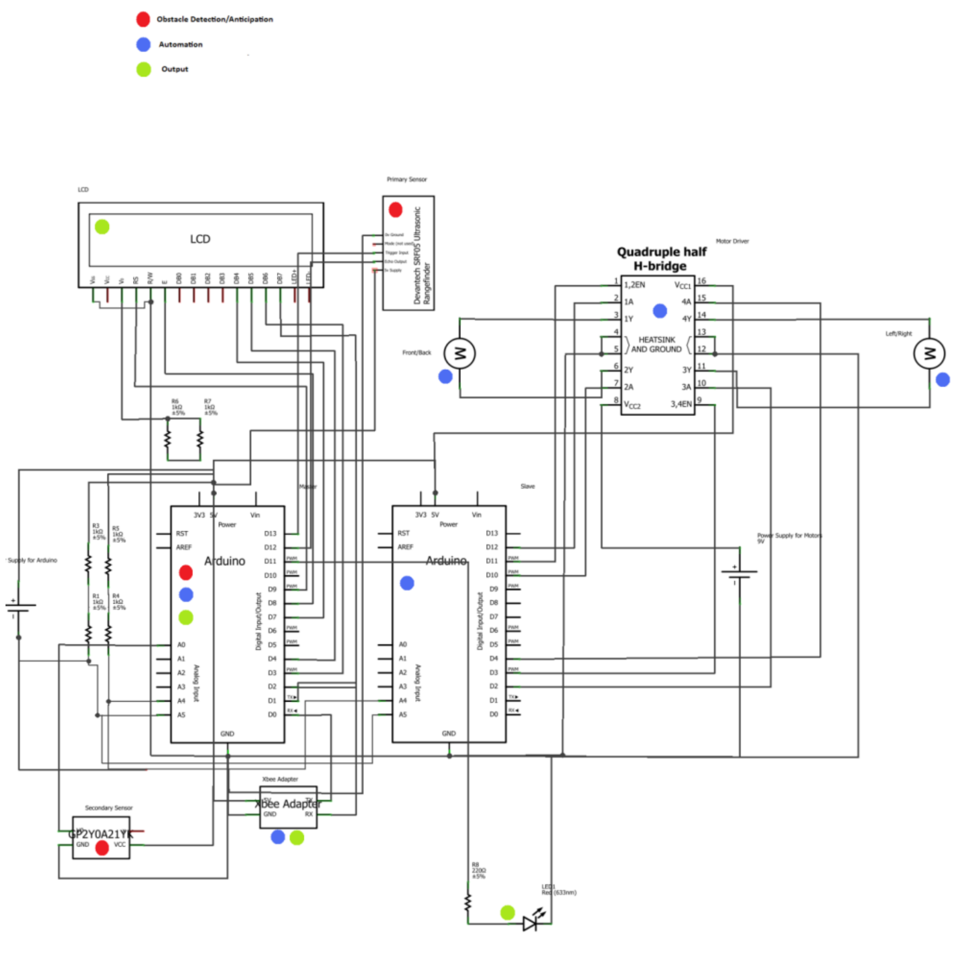


Figure 2: Hardware of the design

As seen, the main components of the hardware have been labeled according to the module they belong to.

Description of Components

The table below is to further outline the function of each of the major hardware components:

Component	Function	Module
Arduino (Master)	• Drive the sensor • Perform distance, speed, and time calculations • Receive commands from the user • Transmit command to Arduino (slave), receive speed levels from Arduino (slave) • Drive output (LCD, LED, External)	Obstacle Detection/Anticipation Output Automation
Arduino (Slave)	• Operate the motor driver according Arduino's (Master) commands • Track speed levels	Automation
Motor Driver (SN754410)	• Regulate the speed and direction of the two motors on the user vehicle (steering, front/back) according to Arduino's (Slave) commands	Automation
LED Light	• Indicate danger levels with varying brightness	Output
LCD Screen	• Provide summarized information to the user	Output
Ultrasonic (Primary) Sensor (SRF05)	• Gather "time of flight" data to be used for distance values for obstacle anticipation	Obstacle Detection/Anticipation
Infrared Proximity (Secondary) Sensor (GP2Y0A21YK)	• Detect obstacles in the blind spot when it is out of the primary sensor's range	Obstacle Detection/Anticipation
Xbee Wireless Module (Series 1)	• Gather wireless commands from user, and transmit sensor data wirelessly to be used in the GUI	Automation Output

Table 1: Arduino components and functionality

Additional Sensor and Considerations

One major revision in the final design from the proposed design is the addition of an extra sensor. The secondary sensor only covers the blind spot region of the car. This region is not fully covered by the primary sensor due to its placement. The primary sensor is placed to extend the range of the anticipation feature, and as a result and additional sensor is needed to fully cover the user's vehicles blind spot.

The situation can be seen in the following picture of the physical implementation of the circuit on the user vehicle:

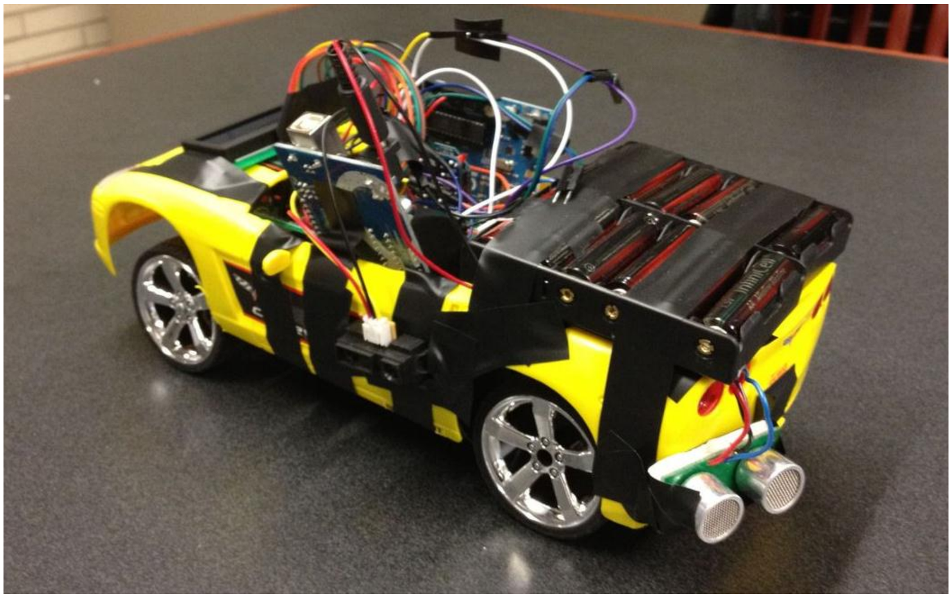


Figure 3: Circuit on the user vehicle

Output Module and Considerations

The LCD screen, and the LED are very important components of the design as it acts as the onboard part of the output module. It provides a summary of all the relevant information analyzed by the microcontroller, according to the various danger levels. The danger levels are determined by the time left calculations, and the presence of an obstacle within the blind spot. The following table outlines the behavior of the LCD screen, and the LED:

Situation	Sensor Involved	Time Left [s]	Danger Level	LCD Screen Output	LED Output	Steering Locked
No Danger	Primary	$t > 15$	0	Distance to obstacle, "No Danger"	Off	N
Obstacle Approaching	Primary	$11 < t < 15$	1	Distance to obstacle, relative speed, "Obstacle Approaching"	Brightness from 0% to 78%, depending on increasing distance	N
Obstacle Approaching	Primary	$6 < t < 11$	2			N
Obstacle Approaching	Primary	$3 < t < 6$	3			N
Obstacle Very Near	Primary	$t < 3$	4	Distance to obstacle, relative speed, "Caution"	Fully On	N
Obstacle at Blind Spot	Secondary	$t \leq 0$	5	"Danger, Check Blind Spot"	Fully On	Y

Table 2: Danger Levels

The LED behavior has been revised from the proposal since blinking the LED requires delay to be added to the software. This delay reduces the frequency of operation of the sensor, and degrades the overall performance of the system. As a result, the current implementation of the output module was designed to ensure intuitive feedback to the user, without sacrificing performance.

Automation Module and Considerations

The automation module is activated if a certain level of danger has been detected. As seen on Table 3, danger levels 4, and 5 activate the steering lock. The reason for this implementation is to maximize user control of the vehicle, but prevent collisions. There are dangers associated with this solution. Due to this feature, more serious frontal collisions may occur. These situations will have to be addressed in future iterations of the design.

Onboard Software

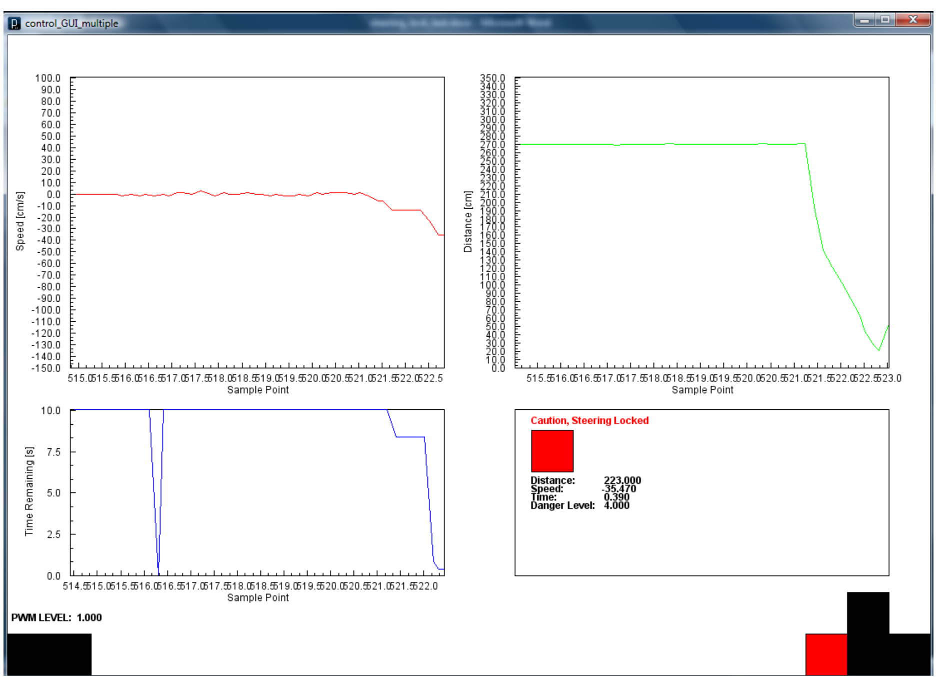
To drive all the hardware components in harmony in order for them to perform their required functions, software has been programmed into the two microcontrollers. Some main features of the software can be found in the "Function" column of the two Arduino components in Table 2. The software, with comments explaining the functions of the separate pieces of the code has been included in Appendix I.

2.2b Remote GUI for Control and Output

It was noted that the LCD screen wasn't a sound enough output needed for the debug, testing, and demonstration of the project. Therefore, a remote GUI was designed. The main functions of the GUI are:

- Remote control the user car for testing and demonstration
- Remotely output the microcontroller's (Arduino Master) output in a graphical form for debug, testing, and project demonstration

The following is a screenshot of the GUI:



The GUI consists of the following components:

- Real-time plots
    - Relative Speed
    - Distance
    - Time Remaining
- Summary Box mirroring a combination of the physical LED, and LCD screen; outlining information regarding danger levels, and instantaneous readings from Arduino Master
- Indication of the current speed level (motor PWM) of the users vehicle

- Virtual buttons indicating the following controls:
    - Accelerate
    - Reverse
    - Steer Left
    - Steer Right
    - Increase Speed Level
    - Decrease Speed Level

The GUI has been written in Processing, and the commented code can be found in Appendix I.

2.3 Assessment of Final Design

The final design meets the requirements set at the time of the proposal, in addition to introducing some new features. However, due to the limitations of the sensors, functionality of the system is compromised in certain situations. Based on the module level, and system level testing, these are the aspects of the design that were successful:

- In cases where the obstacle vehicle remains fully within the sensor's range, the system is able to accurately calculate the relative speed, distance, and time remaining as initially required

- Danger levels are calculated according to time remaining calculations, and activate the steering lock successfully

- Due to the addition of an external sensor, the blind spot is fully covered

- The system is able to anticipate obstacles in the blind spot due to the primary sensor having a large range. This feature differentiates the system from similar products.
- The LCD screen effectively displays the essential information needed for the user to make an informed decision regarding lane changes
- The GUI plots the calculations in real time, and help user's visualize the situation

However, there are various problems that need improving:

- Due to the implemented averaging algorithm, the response time of the system is slow. As a result, user feedback isn't instant. This may be dangerous if the obstacle vehicle is moving at a much faster speed than the user vehicle.
- The system doesn't behave well if the obstacle continuously enters and leaves the sensor's range. This is a major issue when the obstacle is within the edge of the sensor's range. This leads to unstable results, and unreliable user feedback.
- The steering lock feature is dangerous in real life situations
- The GUI, and the LCD screen has not been optimized to be user friendly
- Brightness, instead of frequency was used to indicate danger levels on the LED due to design restrictions. Brightness is often hard to differentiate, especially considering various lighting conditions

On a modular level, the following table describes the success of each module:

Module	Advantage	Disadvantage
Obstacle Detection/Anticipation	• Successfully calculates the distance, speed, and time remaining values	• Slow response time due to compensational averaging algorithm • Unpredictable behavior while the obstacle is on the edge of the sensor's range
Output	• Accurate representation of microcontroller's calculations, and	• Hard to distinguish brightness levels of the LED • The LCD screen can be

	meaningful warning messages	improved to provide more user friendly feedback
Automation	• Steering locks successfully according to time remaining calculations, and the presence of an obstacle in the blind spot	• Time remaining calculations are unreliable in non-ideal situations, and thus result in undesirable steering defects

Overall, the design can be considered a success. However, in the process, we discovered various features that can be improved in future iterations.

3. Validation and Acceptance Test (Valikhan Kuparov, Hani Hadidi)

3.1 Module-Level Test Results

In this section, test results for performance of each module are described. Test cases were designed based on the project requirements that were initially set. These test cases can be found within Validation and acceptance outlined in Appendix B.

3.1.1 Obstacle Detection/Anticipation:

3.1.1.a Blind Spot Coverage

Requirement	Target specification	Final Result	Compliance (Pass/Fail)	Comments and Documentation
1. 1 Sensor's coverage of blind spot	Coverage of entire blind spot as defined in Appendix BS	All spots of blind region are covered	Pass	Positioning of the sensors affect the coverage of the blind spot. Incorrect placement results in a "dead" region

As mentioned in the previous sections, our team decided to place an IR sensor at the side of user car, since the SRF05 sensor did not provide full coverage of the blind spot.

Mounting of side IR sensor and back SRF05 sensor on the RC car plays a critical role in blind spot coverage. Improper placement causes a "dead" region, where neither the IR sensor, nor the

SRF05 sensors are able to detect obstacles present in the blind spot. This is due to the angle and range specifications of the sensors. While adjusting the placement of the sensors at different locations and different angles, our team ensured entire blind spot is covered. Photos of the final positioning of the sensors are demonstrated in Appendix J.

3.1.1. b Distance Measurements

Requirement	Target specification	Final Result	Compliance (Pass/Fail)	Comments and Documentation
1.2 Accurate distance measurements of side IR sensor	Computed distances should match measured distances within ±2.5%	Max 2.5% Error	Pass	- Placed obstacle car at 5, 10, 20 cm away from side IR sensor - Tested distances computed by side IR sensor - All measurements accurately correspond to actual distances
1.2 Accurate distance measurements of back SRF05 sensor		Max 1% Error	Pass	- Placed obstacle car at 0.05, 1, 2, 3 m away from sensor - Tested distances computed by back SRF05 sensor - All measurements accurately correspond to actual distances

Test results show that both sensors provide a high degree of accuracy. However the back SRF05 sensor was not able to accurately detect an object car which is smaller in size compared to user car for distances more than 1.5 m. Therefore, our team decided to enlarge the size of obstacle car by placing s paper box on top of the obstacle car. This paper box has negligible impact on dynamics of obstacle car, but helps the back SRF05 sensor improve its detection accuracy. Detailed results for both sensor testing can be found in Appendix K.

3.1.1.c Relative speed and remaining time calculation

Requirement	Target specification	Final Result	Compliance (Pass/Fail)	Comments and Documentation
2.a The relative speed of the obstacle to the user vehicle	Error < 10%	Max 5% Error	Pass	Two cases were tested: -Stationary user car & approaching obstacle -Both user car and obstacle car in motion -To improve performance a new algorithm was introduced
2.b Accurate time remaining to change lane safely	Error < 12%	NA	Pass	- The accuracy of distance measurements and time calculations guarantee the accuracy of the remaining time calculations

When the both cars are in motion, the difference between the actual speed and the computed one could exceed 30%. Therefore our group had to improve the Obstacle Detection/Anticipation module in order to satisfy target specifications. The methods used to solve the problem are:

- Change the averaging factor and time delay
- Error correction algorithm

Detailed explanation of both methods and test results for relative speed calculations are provided in Appendix L.

3.1.2 Output Module

Requirement	Target specification	Final Result	Compliance (Pass/Fail)	Comments and Documentation
3. Warn user about potential	1. LCD display contains three values: distance to obstacle;	Displays three values according to the	Pass	- LCD display could consist up to 32 characters. Thus, the message should be

danger	relative speed, and message to user according to Look-up table	situation		short but informative - Does not affect other functionality
	2. LED changes brightness according to distance to obstacle	LED brightness adjusts according to distance	Pass	- LED brightness is increased as the distance is reduced - Does not affect other functionality

Initially our team aimed to display the distance to obstacle, relative speed and time remaining to change lane safely. However, testing showed that the time remaining value was not very informative to users due to how rapidly it may change. Instead, our team decided to display the remaining time on the GUI in the form of a rolling graph. On the onboard display, a warning message consisting of 4 levels of danger was displayed instead:

- “NO DANGER” – means no obstacle present in blind spot / lane can be changed safely
- “APPROACHING” – indicates detection of approaching car, warns user to check side mirrors
- “CAUTION” – obstacle car reached very close distance to user car. Steering lock is activated
- “CHECK BLIND SPOT, DANGER” – obstacle car present in blind spot. Lock steering is activated

Danger levels above correspond to the look-up table (Appendix H). All danger levels were simulated. In all cases, the LED behaved according to look-up table, and the LCD screen displayed correct values and messages. Results can be found in Appendix M.

3.1.3 Automation Module

Requirement	Target specification	Final Result	Compliance (Pass/Fail)	Comments and Documentation
4. Prevent user from making unsafe lane change	Lock steering towards obstacle present lane if danger exists	Fully automated lock steering achieved	Pass	Two cases corresponding to danger level 4 and 5 (Appendix LUT) were simulated. In both cases desired behavior of automation is observed

To simulate the test case for danger level 4 (not sufficient time remaining to change lane), our team fabricated a very small remaining time value in the microcontroller. Correspondingly, microcontroller always decided to lock steering to the left.

To simulate danger level 5 (obstacle present in blind spot), an obstacle car was placed at blind spot region. As a result, we were not able to steer the user car towards the obstacle.

Test results for both simulations are given in Appendix N.

3.2 System-Level Test Result (Valikhan Kuparov, dHani Hadidi)

Previous sections demonstrated the three modules performing well individually. On a system level, all of the modules were integrated and tested against safety requirements. For the purpose of validating the system with requirements 2 & 3, the tests were setup as follows. Observations from the test are also recorded below. Images that show the test environments and results are attached in Appendix O.

Test 1: Overall performance of integrated system	
Objective:	Ensure that integrated system demonstrates good performance and satisfies all requirements
Date:	March 20th, 2012
Location:	3rd floor hall, Bahen Center

Test Setup	- Obstacle car and user car run with different known speeds - Compare the microcontroller's calculated relative speed with the measured relative speed - Observe feedback provided by LCD and LED
Feedback and Observations	- Satisfactory test results are obtained as all requirements are met. - Oscillatory and unexpected results occur when cars don't travel on a straight path

As the automation module relies on the remaining components, its successful operation is an indication of the successful operation of the integrated system. A second test was therefore conducted under the same conditions as Test 1 with the aim of verifying the integrated system through the automation module. The following table summarizes test:

Test 2: Lock steering while both cars in motion	
Objective:	Device should enable lock steering for danger levels 4 and 5 (see Appendix LUT)
Date:	March 20th, 2012
Location:	3rd floor hall, Bahen Center
Test Setup	- Obstacle car and user car run with different known speeds - Relative speed, distance and remaining time plots are observed - For cases where time remaining was below the threshold (refer to Appendix LUT) , or an obstacle was already present in the blind spot, successful steering lock was observed
Feedback and Observations	- User is unable to collide into obstacle as the steering is locked for danger levels 4 & 5

Summary and Conclusion (Tanzim Mokammel)

The goal of the project was to aid in accident prevention through a device aiding users identify and anticipate obstacles in their vehicle's blind spot. In addition, the device was to attain vehicle control and prevent the user from steering into obstacles in the blind spot. The prototype device was to be verified on scaled down vehicles under simulated road conditions.

As planned, the device was subject to a range of modular, and system level testing. The results of these tests verified the success of the device, and the ideas that were behind the final implementation.

The main idea behind the implementation was to use simple ultrasonic sensors, and standard microcontrollers in calculating relative speeds between objects. This objective was successful, and it allowed us to reach the overall goal of determining the danger associated with lane changes according the amount of time the user has to perform the maneuver.

Though our design was implemented on scaled down remote controlled vehicles, the obstacle detection and anticipation module can be easily expanded to practical situations. With more advanced sensors, the implemented software and hardware can easily be ported to be used in real life vehicles. However, much more rigorous testing will need to be completed, and detection algorithms need to be improved for the design to become reliable enough for real-life applications.

Though the device is meant to be used for automobiles, components of the project can be used for various other purposes. The speed detector for example has many applications within law enforcement, organized sports, and science. The wireless component of the project can be applied to control remote vehicles, and robots. In addition, the real-time plotting tools developed during the project can be used for a wide range of applications within science, mathematics and engineering.

In future, the design can be improved by replacing the current sensors with more advanced alternatives. In doing so, the need for sluggish averaging algorithms will be eliminated, resulting in a more reliable device. The software itself can also be enhanced to improve response times, and account for anomalies in sensor readings. The automation module can also be further enhanced. The steering lock feature is rather dangerous in real situations, and can be substituted with counter-steering mechanisms. In addition, sensors can be placed at the front of the vehicles to check whether a more serious frontal collision is possible due to the steering lock. In such situations, the automation module should avoid the more serious accident.

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].

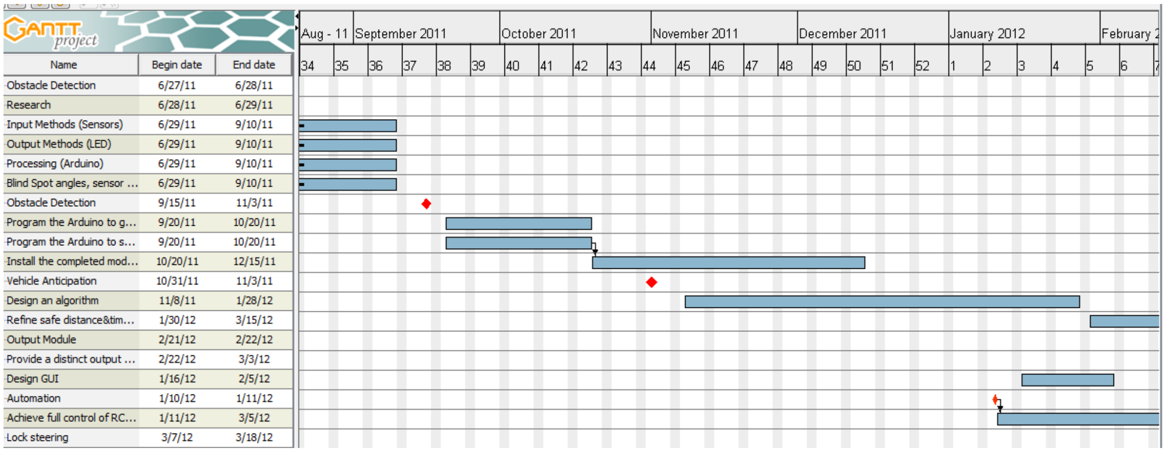
[8] Unknown. "SRF05 - Ultra-Sonic Ranger: Technical Specification". Internet: http://www.robot-electronics.co.uk/htm/srf05tech.htm, [Jul. 25, 2011].

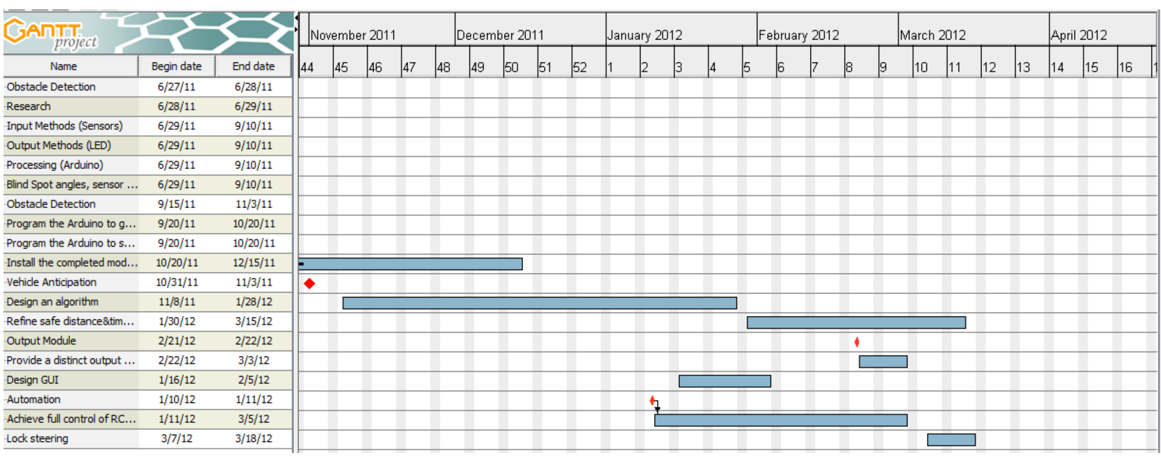
Ontario" Internet: http://www.mto.gov.on.ca/english/trucks/oversize/guide.shtml, [Oct. 10, 2011].

Appendices

Appendix A: Gantt chart (Valikhan Kuparov)

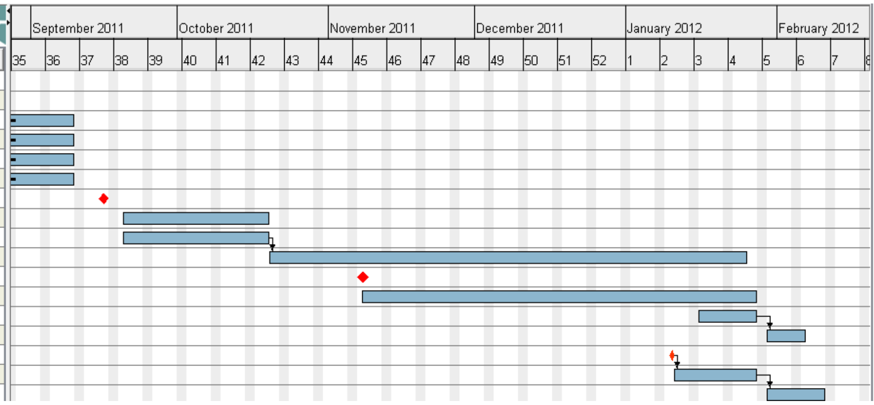
Gantt Chart (final)



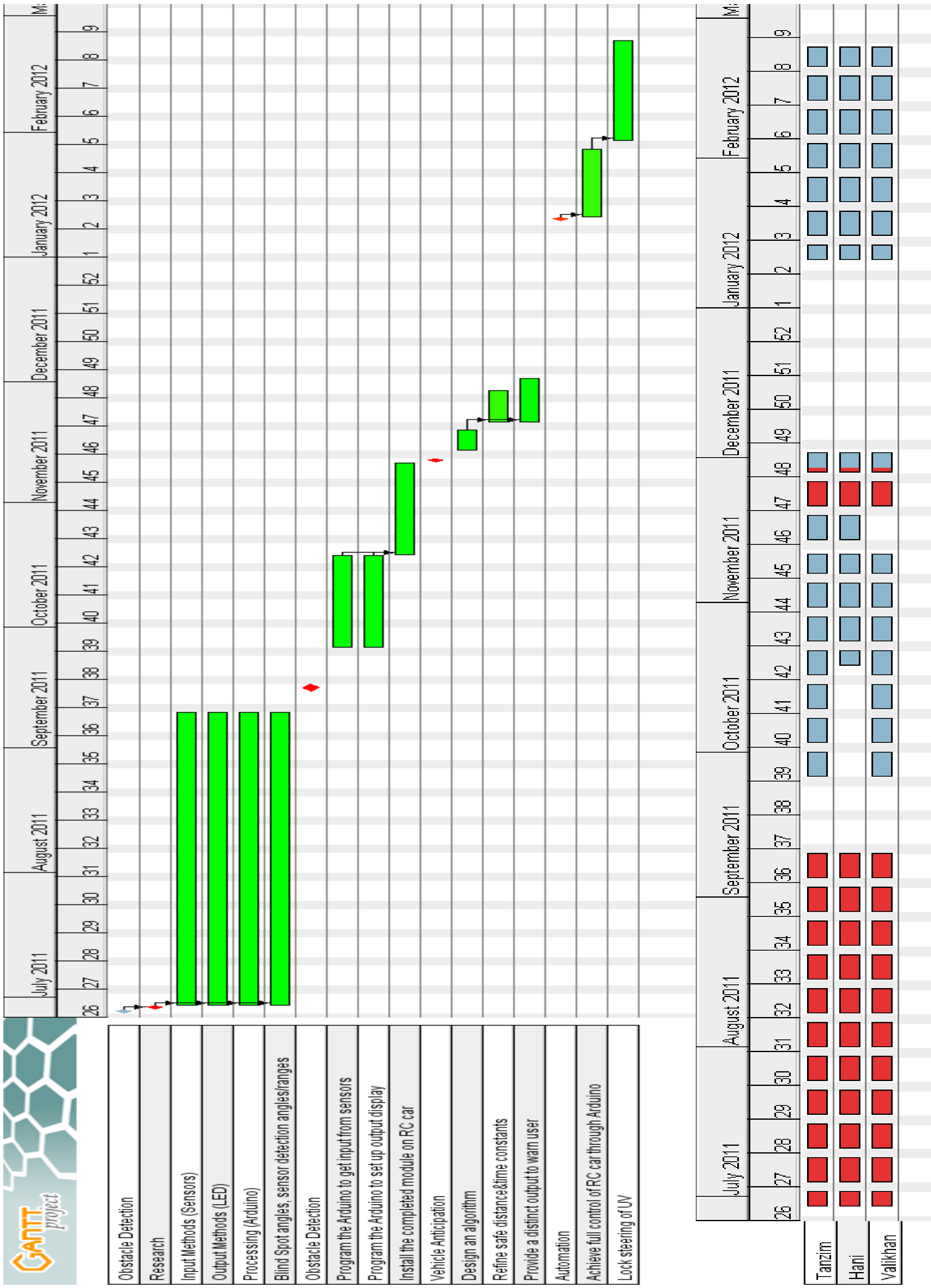


Gannt Chart (Progress Report)

Name	Begin date	End date
Obstacle Detection	6/27/11	6/28/11
Research	6/28/11	6/29/11
Input Methods (Sensors)	6/29/11	9/10/11
Output Methods (LED)	6/29/11	9/10/11
Processing (Arduino)	6/29/11	9/10/11
Blind Spot angles, sensor ...	6/29/11	9/10/11
Obstacle Detection	9/15/11	11/3/11
Program the Arduino to g...	9/20/11	10/20/11
Program the Arduino to s...	9/20/11	10/20/11
Install the completed mod...	10/20/11	1/26/12
Vehicle Anticipation	11/7/11	11/10/11
Design an algorithm	11/8/11	1/28/12
Provide a distinct output ...	1/16/12	1/28/12
Refine safe distance&tim...	1/30/12	2/7/12
Automation	1/10/12	1/11/12
Achieve full control of RC...	1/11/12	1/28/12
New task_32	1/30/12	2/11/12



Gannt Chart (proposal)



Appendix B: Validation and Acceptance test (Valikhan Kuparov)

The test plan for individual modules of the design is outlined in the following table:

Module	Requirements	Test	Acceptance Criteria
Obstacle detection/anticipation	Sensors should measure distances accurately	Place obstacles at 4 different distances(3cm, 1m, 2m, 3m) away from the sensors	Computed distances match the measured distances within ±2.5% error
	Blind spot Coverage	Mount the sensors on the remote controlled (RC) car to cover the defined blind spot (Appendix F); place obstacles within it	The sensors detect the presence of the obstacle within the defined range
	Microcontroller should compute relative speed based on distance values	Mount sensors to the RC car; place an obstacle and the RC car in motion with a known relative speed	The microcontroller's calculated speed values match the known value within ±5% error
	Obstacle Anticipation	Set up situations where both distance and relative speeds are known, and perform allowed time calculations with the microcontroller	The microcontroller's time values match the calculated ones within ±12% error
Output	The Light Emitting Diodes (LED) and the display should provide accurate feedback to the user	Observe the feedback response of the LED and the Liquid Crystal Display (LCD) display	The LED should behave according to the defined look-up table (Appendix H); the display should contain calculated values of distance to obstacle; relative speed, and time left for a lane change
Automation	The steering of the RC car should lock when necessary	Simulate the extremely dangerous, and very unsafe cases (Appendix H)	The steering of the user car is irresponsive in the direction of the obstacle

Appendix C: Initial Project Requirements (Hani Hadidi)

1. The unit shall have sensors mounted on the sides (near the rear) of the user vehicle, such that it covers the defined blind spot region (Appendix F), and is able to obtain the distance to the obstacle and its position within the blind spot
2. The unit shall have a microcontroller to compute the following based on the inputs:
    a. The relative speed of the obstacle to the user vehicle within ±10% accuracy
    b. The amount of time needed for the approaching obstacle to appear in the blind spot, if not already there
3. The unit shall have a feedback mechanism to warn the user of potential danger based on the microcontroller's calculated relative speed and time allowed for a lane change (Appendix G, Appendix H)
4. The unit shall have an override mechanism that will prevent users from making unsafe lane changes by locking the steering of the user vehicle towards the obstacle under conditions outlined in Appendix H
5. The Unit shall adhere to the required technical parameters (Appendix E)

Constraints

1. The unit must cost less than $500
2. The unit must weigh less than 1kg
3. The unit must not impede the user's view of the road
4. The user vehicle must not exceed the maximum length of 12.5 m, and a maximum width of 2.6m (to be scaled down by the vehicle prototype ratio) with the unit attached [6]

Objectives

1. The unit's input sensor shall work correctly regardless of light and weather conditions
2. The blind spot detection component of the unit (no vehicle override) should be a standalone device
3. The microcontroller should be hidden from the user's view

Appendix D: Statistics on Collisions and Casualties (Hani Hadidi)

The following table has been extracted from [1], and it displays statistics on vehicle accidents in Canada from 1990-2009:

Collisions and Casualties 1990–2009

Year	Collisions		Victims		
	Fatal[1]	Personal Injury[2]	Fatalities[3]	Serious Injuries[4]	Injuries[5] (Total)
1990	3,445	178,515	3,963	25,183	262,680
1991	3,225	170,693	3,690	26,035	249,217
1992	3,073	169,640	3,501	25,521	249,823
1993	3,121	168,106	3,615	23,902	247,593
1994	2,837	164,642	3,230	22,830	241,899
1995	2,817	161,950	3,313	21,494	238,458
1996	2,740	153,944	3,129	18,734	227,283
1997	2,660	147,549	3,076	17,294	217,401
1998	2,583	145,615	2,919	16,197	213,319
1999	2,632	148,683	2,980	16,187	218,457
2000	2,547	153,300	2,903	15,583	222,869
2001	2,413	148,996	2,756	15,285	216,489
2002*	2,583	153,859	2,921	15,907	222,707
2003	2,489	150,545	2,779	15,125	216,210
2004	2,436	145,248	2,731	15,591	206,229
2005	2,551	145,603	2,898	15,814	204,764
2006	2,599	142,531	2,884	15,885	199,994
2007	2,462	138,632	2,761	14,236	192,762
2008	2,182[r]	127,634[r]	2,419	12,591[r]	176,433[r]
2009	2,011	123,192	2,209	11,451	172,883

Table 3: StatsCan Statistics on Vehicle Accidents

The following graphs plot the boldly outlined statistics:

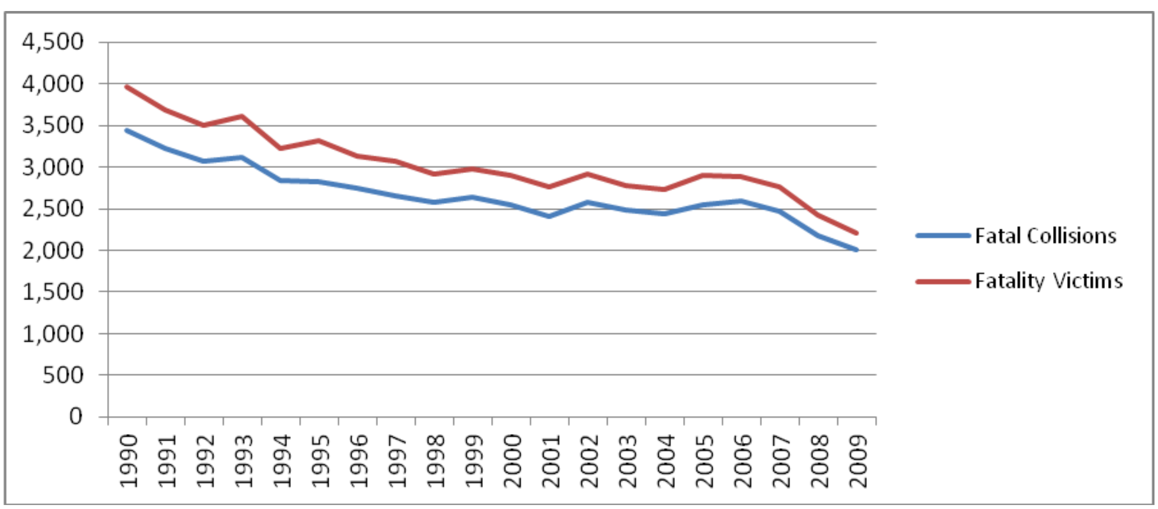


Figure 4: Fatal Collisions and Fatality Victims (1990-2009)

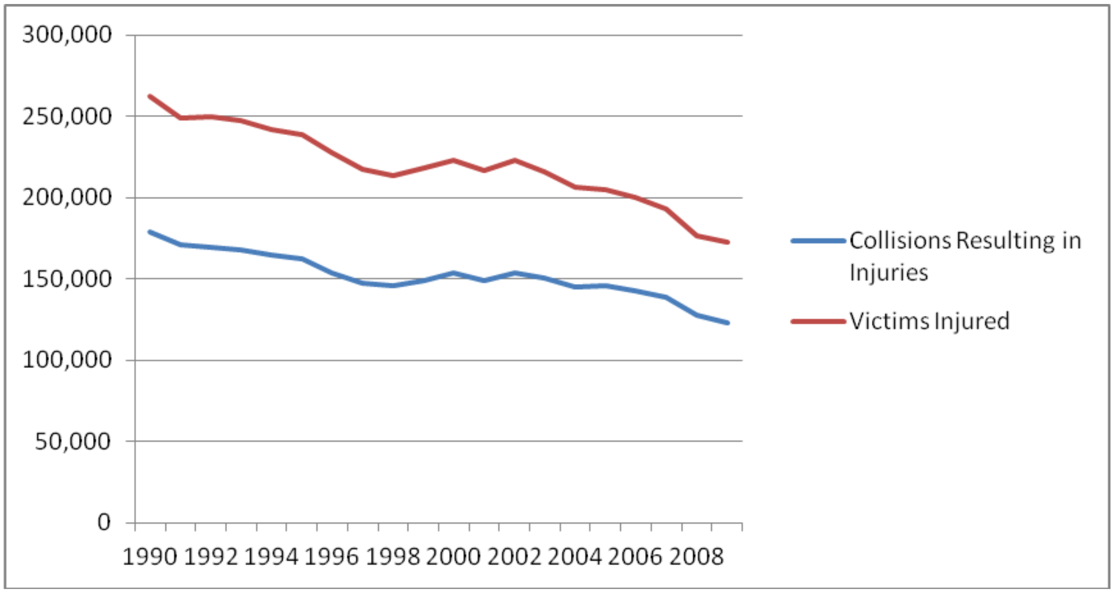


Figure 5: Collisions Resulting in Injury and Victims Injured (1990-2009)

.

Appendix E: Technical Requirements of the Device (Tanzim Mokammel)

The following table outlines the technical requirements to be met by the device:

Technical Requirements	Quantity
Sensor's frequency of acquisition	20 Hz
Sensor's range of acquisition and accuracy	3cm - 3 m, +-2.5% error
Sensor's angle of detection	See figure below [8]
Power	9 V (±2V) lithium-ion batteries

Table 4: Device's Technical Requirements

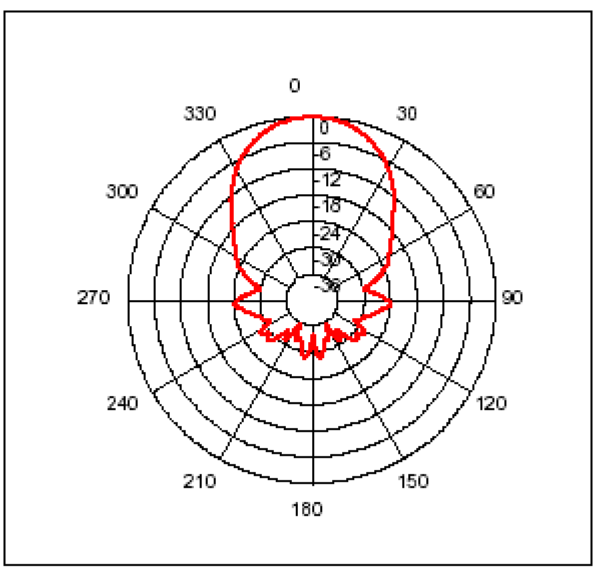


Table 8: Sensor's Angles of Detection [8]

Appendix F: Blind Spot Region (Valikhan Kuparov)

The following figure illustrates angles associated with left and right blind spot areas for a typical mid sized sedan [7]. However, for the purpose of the design project, these values will be scaled down according to the chosen RC car's sclaing factor defined on its specification sheet.

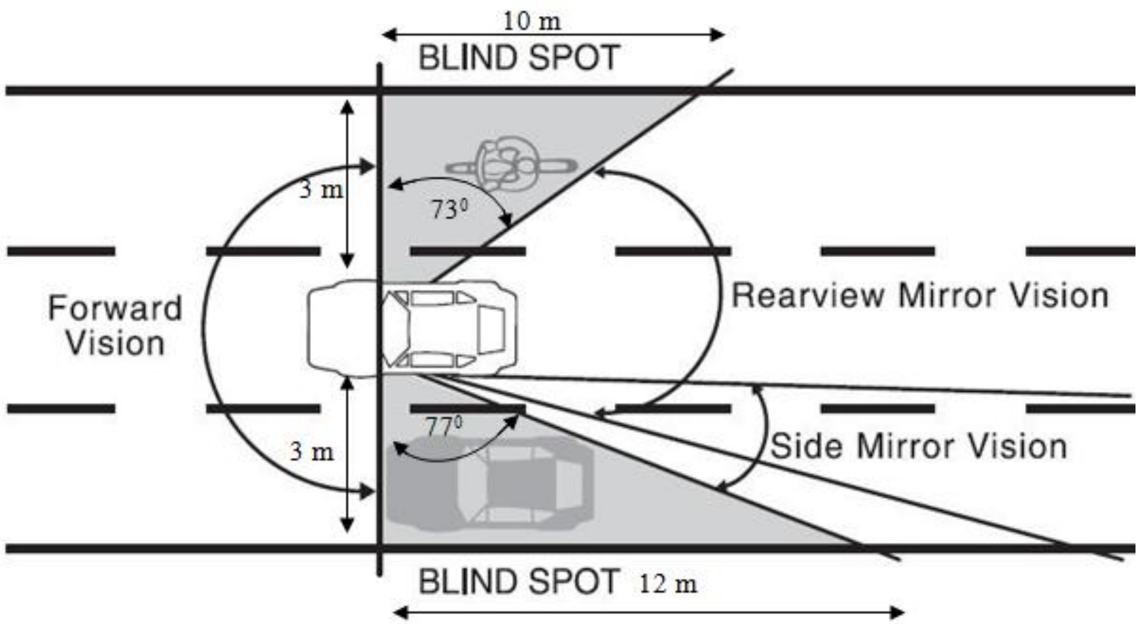


Figure 6: Defined Blind Spot Region

Appendix G: Calculations Performed by Microcontroller (Valikhan Kuparov)

In this appendix, an example is used to demonstrate the calculations perfomed by the micorcontroller in determining the time left for an approaching obstacle to reach the blind spot of the user vehcile (critical time).

Figure 4 illustrates the situation on the road before user vehicle A starts a lane change. Initially, two cars are in Position 1 with speeds V_a and V_b, respectively. At Position 1, the distance between the user vehicle (A) and obstacle (B) are equal to the distance (d): the detection range of sensor on the A. As B moves faster than the A, B takes critical time (t) to reach the blind spot of A. Position 2 illustrates this situation.

The process of finding the time required for a safe lane change is illustrated in Figure 5. A remains stationary, and B approaches with a known constant velocity V.

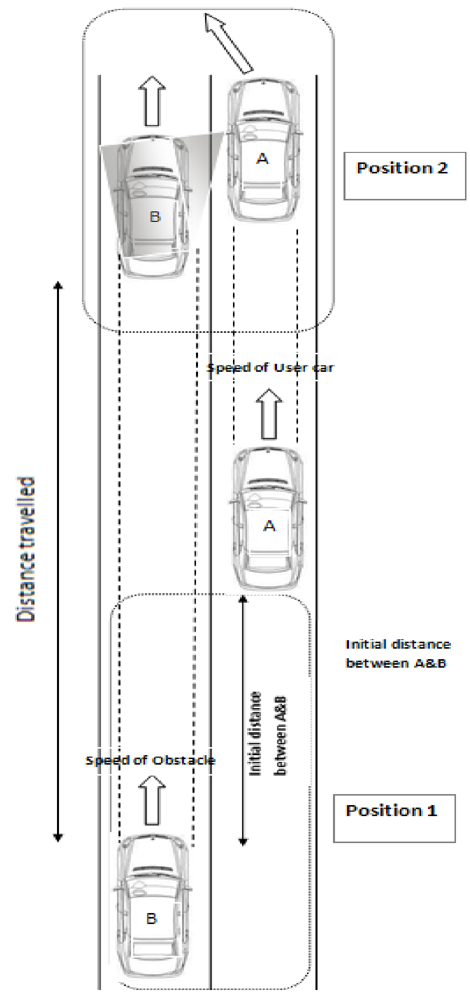


Figure 10: Obstacle Approaching the User Vehicle's Blind Spot

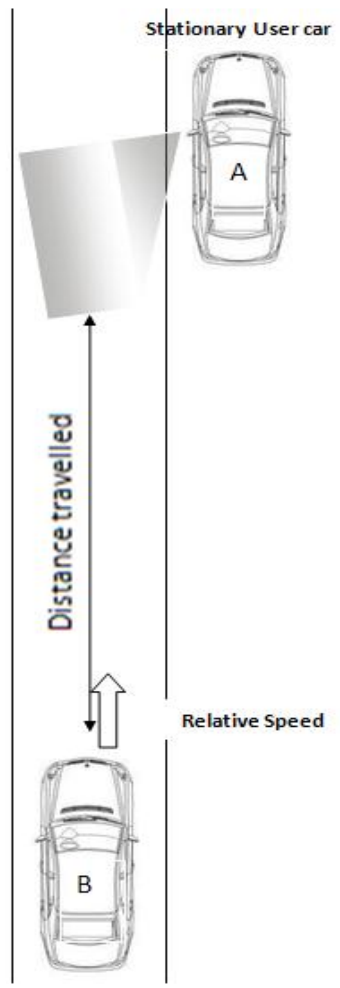


Figure 11: Considerations for Critical Time Calculations

For the sake of calculations, the following data is used outlining an extreme scenario based on actual vehicle speeds:

User vehicle (A) speed	100 km/h	28 m/s
Obstacle (B) speed	200 km/h	56 m/s
Relative speed b/w A and B	100 km/h	28 m/s

Table 5: Realistic Vehicle Speeds

Considering the limilations of the sensor (range, allowed trigger rate), the following table has been decided to be the the parameters for the most extreme case in the case of the remote controlled car. These values are used to calculate the critical time (t).

User vehicle (A) speed	20 km/h	5.5m/s
Obstacle (B) speed	40 km/h	11 m/s
Relative speed b/w A and B	20 km/h	5.5 m/s
Distance to be travelled by obstacle (range of detection)	3 m	

Table 6: Speed and Distance Values to be Used for RC Vehicle

Using numerical values in Table above, we are able to calculate critical time:

$$t = \frac{d}{v} = \frac{3\ m}{5.5\ {}^{m}/_{sec}} = 0.6\ sec$$

The microcontroller will acquire the distance (d) from the sensors. The velocity (v) will be calculated with from the trigger rate of the sensors, and the rate of change in distance (d) values. Using these, the critical time (t) will be calculated as demonstrated above. Once the critical time (t) is found, it is to be compared against the look-up table defined in Appendix H.

Appendix H: Classification of Danger Levels (Tanzim Mokammel)

The following table is to be used by the microcontroller to compare calculated values and determine danger levels. The table also provides corresponding outputs, and cases where counter steer is required.

LED Frequency (Hz)	Critical Time (s)	Classification	Description	Enable Counter Steer
0	>15	Safe	No obstacles present	N
5	11-15	Caution needed	Obstacle approaching	N
20	6-11	Unsafe	Obstacle approaching	N
50	3-6	Very unsafe	Obstacle approaching and near	N
50	1-3	Extremely unsafe	Obstacle very near	Y
100	0	Extremely dangerous	Obstacle already in blind spot	Y

Table 7: Classification of Danger Levels

Appendix I: code (All Members)

Arduino (Master) Code

```
#include <LiquidCrystal.h>
#include <Wire.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=20;//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=5; //maximum allowed distance delta b/w two intervals to steady
speed calculations [TEST CASE VARIABLE]
const int SLAVE_ADDRESS=4; //address of the slave ardunio



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

//for danger level classification
const float LEAST_ALLOWABLE_DISTANCE=7.0;
```

```
//boolean lock_steering = false;
int danger_level=0;
int led_value=0;
String message = "";

//secondary sensor activation/calculation parameters
float sharp_distance=0.0; //distance on the secondary sensor
const float LANE_WIDTH = 23.0; //for SHARP IR sensor


//STEADY STATE ALGORITHM variables
int ss_index=0;
const int SS_CHECKER_SIZE=9; //AMOUNT OF LAST SPEEDS TO CHECK, MUST BE LESS THAN
NUM_SPEED_READINGS
const float ALLOWABLE_DEVIATION=10.0;
boolean steady_state_reached = false;
int steady_state_amount=0;
float average_speed_to_use =0.0;
float fake_total=0.0;
float fake_average_speed =0.0;

//for reading keyboard data
byte incomingByte = 0; // variable to store serial data

int current_pwm=0;




//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 sharp_in=A0; //secondary sensor for blind spot

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;
  }

  Wire.begin(); // join i2c bus (address optional for master)
  Serial.begin(9600); //start serial monitor
}



//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];


  average_speed=total_speed/float(NUM_SPEED_READINGS);



/*****************************************************************************
**************************/
  //is steady_state reached

   //set array position

  for (int k = 0; k < SS_CHECKER_SIZE; k ++) {
```

```
  ss_index=s_index-k;

  if (ss_index== 0) { //for cycle effect
    ss_index = NUM_SPEED_READINGS-1;
  }


  //Serial.print ("comparing : ");
 // Serial.print (speeds [ss_index]);
 // Serial.print ("to: ");
  //Serial.println (average_speed);



  if ( abs (speeds [ss_index] - average_speed) <=  ALLOWABLE_DEVIATION ) {
    steady_state_amount ++;

    //Serial.print ("STEADY STATE COUNT: ");
    //Serial.println (steady_state_amount);
  }

}

if (steady_state_amount >= SS_CHECKER_SIZE) {
  //Serial.println ("STEADY STATE REACHED ");
  steady_state_reached =true;

  for (int k = 0; k < SS_CHECKER_SIZE; k ++) {
    fake_total = fake_total + speeds [k];

  }
  fake_average_speed = fake_total/SS_CHECKER_SIZE;
  fake_total=0;
}
else {
  //Serial.println ("NOT IN STEADY STATE");
  steady_state_reached =false;

}

steady_state_amount=0;

s_index +=1;
```

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


  if (steady_state_reached){
    average_speed_to_use=fake_average_speed;
  }
  else {
    average_speed_to_use=average_speed;
  }

  //TIME LEFT CALCULATIONS
  //factor in secondary sensor
  sharp_distance = 12343.85 * pow(analogRead(sharp_in),-1.15);

  if (sharp_distance <= LANE_WIDTH) {
    time_left =-1.0;
  }
  else
  {


      if (average_speed < 0) {
        if (average_distance < LEAST_ALLOWABLE_DISTANCE) {
          time_left=0.0;
        }
        else {
          time_left = float(average_distance)/abs(average_speed_to_use);
        }

        if (time_left >= 30.0){
          time_left=30.0;
        }
        //Serial.println(time_left);
        //printFloat(time_left, 2);
      }
      else {
        time_left=30.0; //default max value
      }
  }
}
```

```
//danger level calculations
if (time_left > 15.0) {
  danger_level=0;
  message = "No Danger";
}
else if (time_left >11.0 && time_left < 15.0) {
  danger_level=1;
  message = "Approaching";
  //led_delay=100;
}
else if (time_left >6.0 && time_left<11.0) {
  danger_level=2;
  message = "Approaching";
  //led_delay=50;
}
else if (time_left >3.0 && time_left<6.0) {
  danger_level=3;
  message = "Approaching";
  //led_delay=25;
}
else if (time_left >= 0.0 && time_left<3.0 ) {
  danger_level=4;
  message = "Caution";
  //led_delay=10;
  //lock_steering=true;
}
else if (time_left < 0.0 ) {
  danger_level=5;
  message = "Danger";
  //led_delay=10;
  //lock_steering=true;
}
```

```
  //SERIAL DEBUG

//  Serial.print("a_index: ");
//  Serial.println(a_index, DEC);
  //Serial.print("d: ");
  //Serial.print(current_distance, DEC);
  //Serial.println();
//  Serial.print("old_d: ");
//  Serial.println(old_distance, DEC);
  //Serial.print("delta_t: ");
  //Serial.println(elapsedTime);
  //Serial.println();
  //Serial.print("delta_d: ");
  //Serial.println(delta_d, DEC);
  //Serial.print("inst_speed: ");
  //Serial.println();
  //if (average_speed != 0) {
    //Serial.println(current_distance, DEC);
    //Serial.print("    ");
   //printFloat(average_speed, 7);

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

  //Serial.print("average_speed: ");
  Serial.print(average_distance);
  Serial.print(",");
  Serial.print(average_speed_to_use);
  Serial.print(",");
  Serial.print(time_left);
  Serial.print(",");
  Serial.print(current_pwm);
  Serial.print(",");
  Serial.print(danger_level);
  Serial.print(",");
  Serial.print(sharp_distance);
  //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);

 if (danger_level != 5){
  // print distance from obstacle

  lcd.print(average_distance);
  //lcd.print("");
  lcd.setCursor(5, 0);

  if (average_speed_to_use < 0.0) {
    lcd.print(abs(average_speed_to_use));
  }

 }
 else{
  lcd.print ("Check Blind Spot");
 }

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


 //led implementation

 if (danger_level == 0){
  led_value=0;
 }
 else if (danger_level >=1 && danger_level <=3){
  led_value=200-average_distance;
 }
 else{
  led_value=254;
 }
//
// if (average_distance < 254) {
//    led_value = 254 - average_distance;       // this means the smaller the distance the
brighterthe LED.
//  }
//  else{
//    led_value=0;
//  }
//
```

```
 analogWrite(pin_led_trigger, led_value);



//KEYBOARD MONITOR AND TRANSMIT CODE


 if (Serial.available() > 0) { // check for serial data
  incomingByte = Serial.read(); // read the incoming byte


  Wire.beginTransmission(SLAVE_ADDRESS); // transmit to device #4

  //Serial.print("I received: "); // say what you got
  //Serial.println(incomingByte); // incomingByte takes value from the serial read

  //Wire.send("I received: ");        // sends five bytes
  Wire.send(incomingByte);        // sends one byte
  Wire.endTransmission();    // stop transmitting

  //get PWM data

    Wire.requestFrom(SLAVE_ADDRESS,1);    // request 6 bytes from slave device #2

    while(Wire.available())    // slave may send less than requested
    {
      current_pwm = Wire.receive(); // receive a byte as character
      //Serial.print(current_pwm);        // print the character
    }



 }


 delay (TIME_DELAY);


}
```

Arduino (Slave) Code

```
#include <Wire.h>

const int SLAVE_ADDRESS=4;
const int DELAY=0;

int M1_PWM = 11; // PWM H-bridge enable pin for speed control
int M1_A = 12; // H-bridge leg 1 Motor 1
int M1_B = 10; // H-bridge leg 2 Motor 1


int M2_PWM = 3; // PWM H-bridge enable pin for speed control
int M2_A = 2; // H-bridge leg 1 Motor 2
int M2_B = 4; // H-bridge leg 2 Motor 2

int LED = 13; // LED pin attached to Arduino pin 13
int LED_PWM=6;
int incomingByte = 0; // variable to store serial data
int speed_max = 254;
int speed_val = speed_max; // variable to store speed value
int speed_min = 254*0.2;
int increment = 254.*0.1;

////////////////////////////////////////////////////////////////////////////////////


void setup(){

  Wire.begin(SLAVE_ADDRESS);              // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
  Wire.onRequest(requestEvent); // register request event

//Serial.begin(115200);

// set digital i/o pins as outputs:
//
  pinMode(LED, OUTPUT);
  pinMode(LED_PWM, OUTPUT);
```

```
  pinMode(M1_PWM, OUTPUT);
  pinMode(M1_A, OUTPUT);
  pinMode(M1_B, OUTPUT);

  pinMode(M2_PWM, OUTPUT);
  pinMode(M2_A, OUTPUT);
  pinMode(M2_B, OUTPUT);

}

///////////////////////////////////

void loop(){
  delay (DELAY);

}



/////////// motor functions ///////////////




void M1_stop(){


  digitalWrite(M1_B, LOW);
  digitalWrite(M1_A, LOW);
  digitalWrite(M1_PWM, LOW);


  //debug w/ LED
  //digitalWrite (LED_PWM, HIGH);
  //digitalWrite (LED_PWM, LOW);

}

void M2_stop(){

  digitalWrite(M2_B, LOW);
  digitalWrite(M2_A, LOW);
```

```
  digitalWrite(M2_PWM, LOW);


  //digitalWrite (LED_PWM, LOW);

  //digitalWrite (LED_PWM, HIGH);
  //digitalWrite (LED_PWM, LOW);

}



void M1_forward(int x){
  //analogWrite(LED_PWM, 255);

  digitalWrite(M1_B, LOW);
  digitalWrite(M1_A, LOW);


  digitalWrite(M1_A, LOW);
  digitalWrite(M1_B, HIGH);
  analogWrite(M1_PWM, x);

  //debug w/ LED
  //analogWrite(LED_PWM, 255);

}


void M1_reverse(int z){

  digitalWrite(M1_B, LOW);
  digitalWrite(M1_A, LOW);


  digitalWrite(M1_A, HIGH);
  digitalWrite(M1_B, LOW);
  analogWrite(M1_PWM, z);


   //debug w/ LED
  //analogWrite(LED_PWM, 200);

}
```

```
void M2_right(int y){

  digitalWrite(M2_A, LOW);
  digitalWrite(M2_B, LOW);

  digitalWrite(M2_A, HIGH);
  digitalWrite(M2_B, LOW);
  analogWrite(M2_PWM, y);

   //debug w/ LED
  //analogWrite(LED_PWM, 225);
}

void M2_left(int y){

  digitalWrite(M2_A, LOW);
  digitalWrite(M2_B, LOW);

  digitalWrite(M2_A, LOW);
  digitalWrite(M2_B, HIGH);
  analogWrite(M2_PWM, y);

   //debug w/ LED
  //analogWrite(LED_PWM, 150);
}


void M1_change_speed(int z){

  analogWrite(M1_PWM, z);


   //digitalWrite (LED_PWM, LOW);
   //delay (150);
   //digitalWrite (LED_PWM, HIGH);
   //delay (150);
   //digitalWrite (LED_PWM, LOW);
  //debug with LED

}
```

```
void receiveEvent(int howMany)
{
//  while(1 < Wire.available()) // loop through all but the last
//  {
//    char c = Wire.receive(); // receive byte as a character
//    //Serial.print(c);         // print the character
//  }
  int incomingByte = Wire.receive();    // receive byte as an integer
  //Serial.println(x);         // print the integer


    // if byte is equal to "105" or "i", go forward
    // if byte is equal to "105" or "i", go forward
    if (incomingByte == 105){
      M1_forward(speed_val);
      delay(DELAY);

    }
    else if (incomingByte == 107){

      M1_reverse(speed_val);
      delay(DELAY);
    }
    // check incoming byte for direction
    //go left, j
    else if (incomingByte == 106){

      M2_left(254);
      delay(DELAY);
      //M2_stop();
    }

    //go righ, l
    else if (incomingByte == 108){

      M2_right(254);
      delay(DELAY);
      //M2_stop();
    }
```

```
    // if byte is equal to "46" or "," - raise speed
    else if (incomingByte == 97){
      speed_val = speed_val + increment;

      if (speed_val > speed_max) {
        speed_val=speed_max;
      }
      M1_change_speed (speed_val);
      delay(DELAY);
      //Serial.println(speed_val);
    }
    else if (incomingByte == 115){
      speed_val = speed_val - increment;

      if (speed_val < speed_min) {
        speed_val=speed_min;
      }
      M1_change_speed (speed_val);
      delay(DELAY);
      //Serial.println(speed_val);
    }
    else if (incomingByte == 32) {
      M2_stop();
      delay(DELAY);
    }
    else {
      M1_stop();
      M2_stop();
      delay(DELAY);
    }


}

void requestEvent()
{
  Wire.send(speed_val); // respond with message of 6 bytes
                        // as expected by master
}
```

GUI Code

```
import processing.serial.*;
import org.gwoptics.graphics.graph2D.Graph2D;
import org.gwoptics.graphics.graph2D.traces.ILine2DEquation;
import org.gwoptics.graphics.graph2D.traces.RollingLine2DTrace;

PFont fontA;

String COM_PORT="COM8";


int lf = 10;    // Linefeed in ASCII
int comma = 44;
String myString = null;
float distance = 0;
float speed = 0;
float time= 0;

int s_level_MAX=8; //change this accroding to slave's pwm levels
int s_level=s_level_MAX;
float current_pwm=0.0;
float danger_level=0.0;
int danger_box_color=0;
float sharp_distance=0.0;

Boolean first_up=true;
Boolean first_down=true;
Boolean first_left=true;
Boolean first_right=true;



int fill_off=0;
int fill_on=255;

color fillVal1 = color(fill_off);
color fillVal2 = color(fill_off);
color fillVal3 = color(fill_off);
color fillVal4 = color(fill_off);

color fillVal_a = color(fill_off);
color fillVal_s = color(fill_off);
```

```
int EDGE = 770;
int h_offset=340;

//graph 1 (speed) parameters
int GRAPH_Y=350;
int GRAPH_X= 450;
int GRAPH_YST=50;
int GRAPH_XST=75;

//graph 2 (distance) parameters
int GRAPH_Y2=350;
int GRAPH_X2= 450;
int GRAPH_YST2=50;
int GRAPH_XST2=610;

//graph 3 (time) parameters
int GRAPH_Y3=200;
int GRAPH_X3= 450;
int GRAPH_YST3=450;
int GRAPH_XST3=75;

//information box
int INFO_BOX_SIZE_X=450;
int INFO_BOX_SIZE_Y=200;
int INFO_BOX_Y=450;
int INFO_BOX_X=610;
//int INFO_BOX_X=EDGE-50,
//int INFO_BOX_Y=EDGE-50;



int BOX_SIZE = 50;

// The serial port:
Serial myPort;


//graph class definition; DEFINE THE DATA HERE
class eq implements ILine2DEquation{
        public double computePoint(double x,int pos) {
                return speed;
```

```
        }
}

class eq2 implements ILine2DEquation{
        public double computePoint(double x,int pos) {
                return distance;


        }
}

class eq3 implements ILine2DEquation{
        public double computePoint(double x,int pos) {
                return time;


        }
}

//declare graph classes
RollingLine2DTrace r,r2,r3;
Graph2D g,g2,g3;

void setup () {


  size(EDGE+h_offset,EDGE);


  //font setup
  fontA = loadFont("Arial-BoldMT-48.vlw");
  textFont(fontA, 12);
  //COM Port Setup
  myPort = new Serial(this, COM_PORT, 9600);
  myPort.clear();
  myString = myPort.readStringUntil(lf);
  myString = null;


  //graph setup
  r  = new RollingLine2DTrace(new eq() ,100,0.1f);
  r.setTraceColour(255, 0, 0);
```

```
r2=new RollingLine2DTrace(new eq2() ,100,0.1f);
r2.setTraceColour(0, 255, 0);

r3=new RollingLine2DTrace(new eq3() ,100,0.1f);
r3.setTraceColour(0, 0, 255);

//speed graph
g = new Graph2D(this, GRAPH_X, GRAPH_Y, false);
//distance grpah
g2 = new Graph2D(this, GRAPH_X2, GRAPH_Y2, false);
//time graph
g3 = new Graph2D(this, GRAPH_X3, GRAPH_Y3, false);

//trace data
g.addTrace(r);
//g.addTrace(r2);
g.setYAxisMax(100);
g.setYAxisMin(-150);
g.setXAxisLabel("Sample Point");
g.setYAxisLabel("Speed [cm/s]");
g.position.y = GRAPH_YST;
g.position.x = GRAPH_XST;
g.setYAxisTickSpacing(10);
g.setXAxisMax(8f);


//definitions for distance graph
g2.addTrace(r2);
//g.addTrace(r2);
g2.setYAxisMax(350);
g2.setYAxisMin(0);
g2.setXAxisLabel("Sample Point");
g2.setYAxisLabel("Distance [cm]");
g2.position.y = GRAPH_YST2;
g2.position.x = GRAPH_XST2;
g2.setYAxisTickSpacing(10);
g2.setXAxisMax(8f);


//definitions for time graph
g3.addTrace(r3);
//g.addTrace(r2);
g3.setYAxisMax(10);
```

```
  g3.setYAxisMin(0);
  g3.setXAxisLabel("Sample Point");
  g3.setYAxisLabel("Time Remaining [s]");
  g3.position.y = GRAPH_YST3;
  g3.position.x = GRAPH_XST3;
  g3.setYAxisTickSpacing(2.5);
  g3.setXAxisMax(8f);

}

void draw() {

  background(255);
  //keypress debug
  get_serial_data();
  //println(speed);



  //up box
  fill(fillVal1);
  rect(h_offset+EDGE-2*BOX_SIZE, EDGE-2*BOX_SIZE, BOX_SIZE, BOX_SIZE);

  //down box
  fill(fillVal2);
  rect(h_offset+EDGE-2*BOX_SIZE, EDGE-BOX_SIZE, BOX_SIZE, BOX_SIZE);

  //left box

  if (danger_level < 4.0){

    fill(fillVal3);
    rect(h_offset+EDGE-3*BOX_SIZE, EDGE-BOX_SIZE, BOX_SIZE, BOX_SIZE);
  }
  else
  {
    fill(255,0,0);
    rect(h_offset+EDGE-3*BOX_SIZE, EDGE-BOX_SIZE, BOX_SIZE, BOX_SIZE);
  }
```

```
  //right box
  fill(fillVal4);
  rect(h_offset+EDGE-BOX_SIZE, EDGE-BOX_SIZE, BOX_SIZE, BOX_SIZE);

  //a box
  fill(fillVal_a);
  rect(0, EDGE-BOX_SIZE, BOX_SIZE, BOX_SIZE);

  //s box
  fill(fillVal_s);
  rect(BOX_SIZE, EDGE-BOX_SIZE, BOX_SIZE, BOX_SIZE);


  //information box
  fill (255);
  rect (INFO_BOX_X, INFO_BOX_Y,INFO_BOX_SIZE_X, INFO_BOX_SIZE_Y);






  //danger display box

  int text_offset_x=20;
  int text_offset_y=18;
  int box_offset_x=20;
  int box_offset_y=25;

  if (danger_level == 0.0){fill(0,254,0);text ("No Danger", INFO_BOX_X+text_offset_x,
INFO_BOX_Y+text_offset_y);}
  else if (danger_level == 1.0){fill(125,125,0);text ("Obstacle Approaching",
INFO_BOX_X+text_offset_x, INFO_BOX_Y+text_offset_y);}
  else if (danger_level == 2.0){fill(200,200,0);text ("Obstacle Approaching",
INFO_BOX_X+text_offset_x, INFO_BOX_Y+text_offset_y);}
  else if (danger_level == 3.0){fill(254,254,0);text ("Obstacle Approaching",
INFO_BOX_X+text_offset_x, INFO_BOX_Y+text_offset_y);}
  else if (danger_level == 4.0){fill(254,0,0);text ("Caution, Steering Locked",
INFO_BOX_X+text_offset_x, INFO_BOX_Y+text_offset_y);}
  else if (danger_level == 5.0){fill(254,0,0);text ("Danger, Steering Locked",
INFO_BOX_X+text_offset_x, INFO_BOX_Y+text_offset_y);}
```

```
  rect (INFO_BOX_X+box_offset_x, INFO_BOX_Y+box_offset_y,BOX_SIZE, BOX_SIZE);



  //reset fill back to black
  fill(fill_off);

  int info_offset_x=20;
  int info_offset_y=90;
  int info_value_offset=85;
  int vert_spacing = 10;

  text ("Distance: ", INFO_BOX_X+info_offset_x, INFO_BOX_Y+info_offset_y);
  text (distance,INFO_BOX_X+info_offset_x+info_value_offset, INFO_BOX_Y+info_offset_y);

  text ("Speed:", INFO_BOX_X+info_offset_x, INFO_BOX_Y+info_offset_y+vert_spacing);
  text (speed, INFO_BOX_X+info_offset_x+info_value_offset,
INFO_BOX_Y+info_offset_y+vert_spacing);

  text ("Time:", INFO_BOX_X+info_offset_x, INFO_BOX_Y+info_offset_y+2*vert_spacing);
  text (time, INFO_BOX_X+info_offset_x+info_value_offset,
INFO_BOX_Y+info_offset_y+2*vert_spacing);

  text ("Danger Level:", INFO_BOX_X+info_offset_x,
INFO_BOX_Y+info_offset_y+3*vert_spacing);
  text (danger_level, INFO_BOX_X+info_offset_x+info_value_offset,
INFO_BOX_Y+info_offset_y+3*vert_spacing);

  text ("PWM LEVEL: ", 5, EDGE - 65);
  //text (s_level, 5+75, EDGE - 65);
  text (current_pwm/254, 5+75, EDGE - 65);



  g.draw();
```

```
  g2.draw();
  g3.draw();

}

//serial data function
void get_serial_data () {


        while (myPort.available() > 0) {
          myString = myPort.readStringUntil(lf);
          //myString = myPort.readString();
          if (myString != null) {
            //text (myString,EDGE - 500 ,25);
            //speed = float(myString);

            //String[] p = splitTokens("a,b,c,d,e\n", ",");



            String [] p = splitTokens(myString, ",");




            //for (int i=0; i<3; i++) {
              //if (p[i] != null) {
                if (p.length == 6) {
                  distance = float(p[0]);
                  speed = float(p[1]);
                  time= float (p[2]);
                  current_pwm= float (p[3]);
                  danger_level= float (p[4]);
                  sharp_distance= float (p[5]);
                }

                //text (distance, EDGE - 500 ,25+10); // Prints "a"
                //text (speed, EDGE - 500 ,25+20); // Prints "b"
                //text (time, EDGE - 500 ,25+30); // Prints "c"
              //}
            //}

            //println(p[2]); // Prints "c"
```

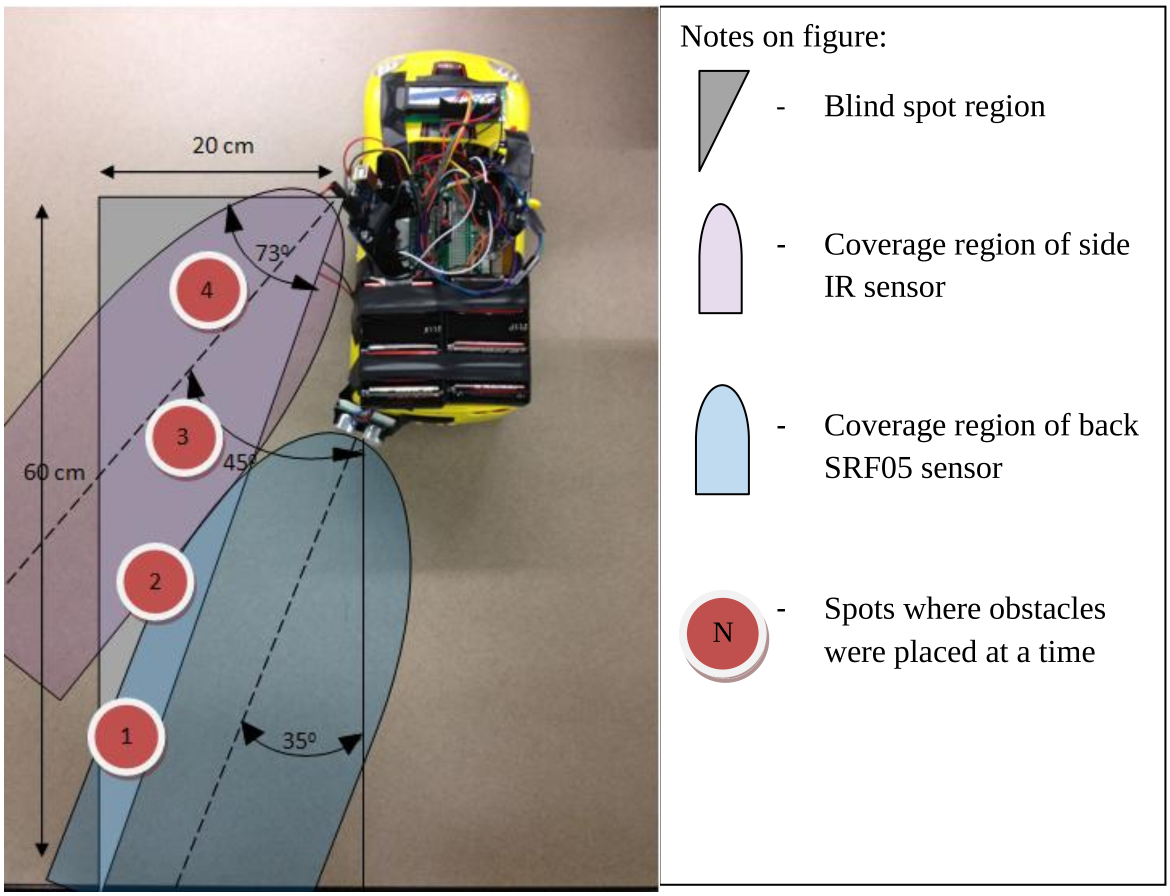
```
        }
      }

      //return speed;
}



//keyboard functions

void keyPressed() {
  if (key == CODED) {
    if (keyCode == UP) {
      fillVal1 = fill_on;


      //need to only write once
      if (first_up) {
        myPort.write(105);
        first_up=false;
      }

    }
    else if (keyCode == DOWN) {
      fillVal2 = fill_on;

      //only write once
      if (first_down) {
        myPort.write(107);
        first_down=false;
      }



    }
    else if (keyCode == LEFT) {
      fillVal3 = fill_on;

      //only write once
      if (first_left) {
        //don't allow collision
        if (danger_level <4.0){
          myPort.write(106);
          first_left=false;
```

```
      }

     }

    }
    else if (keyCode == RIGHT) {
      fillVal4 = fill_on;

      //only write once
      if (first_right) {
        myPort.write(108);
        first_right=false;
      }
    }
    else {
    //fillVal1 = 0;
    //fillVal2 = 0;
    //myPort.write(120);
    }
  }
}

void keyReleased() {
  if (key == CODED) {
    if (keyCode == UP) {
      fillVal1=fill_off;


      myPort.write(107);
      myPort.write(120);
      first_up=true; //reset first_x

      //delay (100);
      //myPort.write(120);
    }
    else if (keyCode == DOWN) {
      fillVal2=fill_off;
      myPort.write(105);
      myPort.write(120);
      first_down=true; //reset first_x

      //delay (100);
      //myPort.write(120);
    }
```

```
    else if (keyCode == LEFT) {
      fillVal3=fill_off;
      //sends SPACE to reset position
      myPort.write(32);
      first_left=true; //reset first_x
    }
    else if (keyCode == RIGHT) {
      fillVal4=fill_off;
      //sends SPACE to reset position
      myPort.write(32);
      first_right=true; //reset first_x
    }

  }
}


void keyTyped() {
  if (key == 'a') {

      if (fillVal_a == fill_off) { fillVal_a = fill_on;}
      else {fillVal_a = fill_off;}

      myPort.write(97);
      if (s_level != s_level_MAX) {
        s_level ++;
      }
      //fillVal_a = 0;
  }
  if (key == 's') {
      if (fillVal_s == fill_off) { fillVal_s = fill_on;}
      else {fillVal_s = fill_off;}

      myPort.write(115);

      if (s_level != 1) {
        s_level --;
      }
```

Appendix J: Blind Spot Coverage (Valikhan Kuparov)

In the figure below, parameters of the blind spot region were calculated based on the blind spot region of a real car (Appendix F) with 1:16 scale down factor. Sensors' coverage regions were drawn based on the parameters found in datasheets.



An obstacle was placed at different spots as indicated by the red circles. In all cases, the sensors were able to properly detect the obstacle. Thus, this test verifies the coverage of the entire blind spot region.

Appendix K: Distance Measurements (Hani Hadidi)

In this appendix, test results for both the side IR sensor, and the back SRF05 sensor are given. The obstacle car was placed at different distances, and distance readings of each sensor were captured. Both sensors have a high level of accuracy.

Back SRF05 sensor

Actual distance	Test result	Error, %
5 cm	Distance:5.000 Speed: 0.040 Time: 30.000	0
1 m	Distance:101.000 Speed: 0.000 Time: -0.000	1
2 m	Distance:201.000 Speed: 0.000 Time: -0.000	0.5
3 m	Distance:300.000 Speed: 1.610 Time: 30.000	0

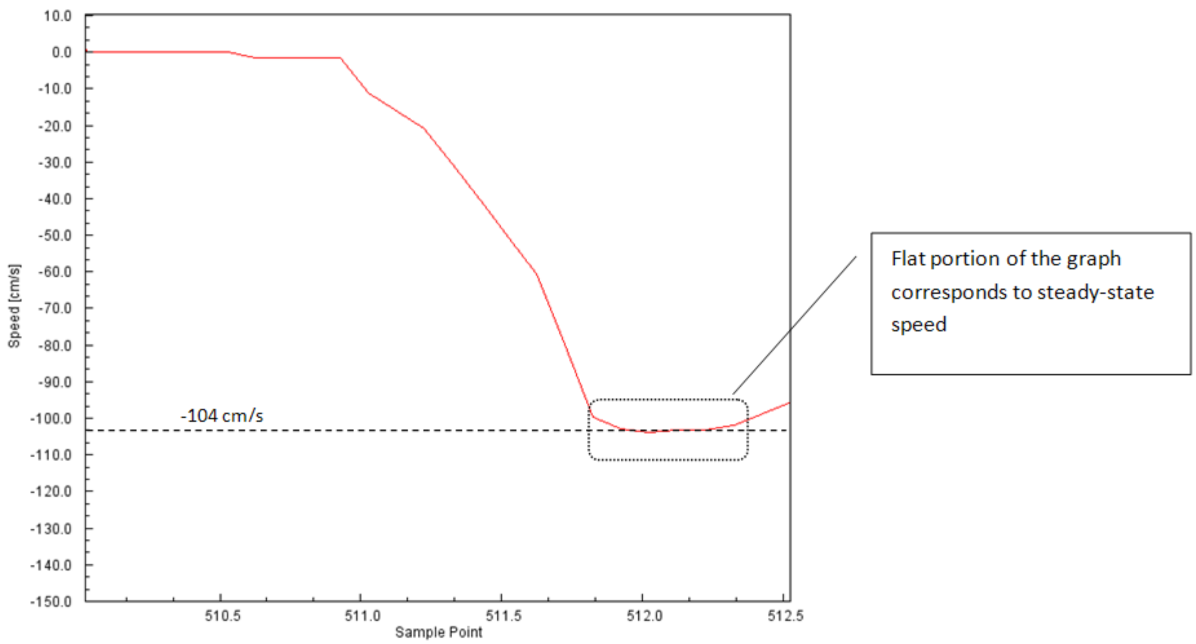
Side IR sensor

Actual distance	Test result	Error, %
7 cm	Speed: -3.600 Time: -1.000 Danger Level: 5.000 IR Reading: 7.200	2.5
12 cm	Speed: 0.000 Time: -1.000 Danger Level: 5.000 IR Reading: 12.080	0.6
20 cm	Speed: 12.150 Time: -1.000 Danger Level: 5.000 IR Reading: 20.260	1.3

Appendix L: Relative Speed and Remaining Time Calculations (Valikhan Kuparov, Hani Hadidi)

Case 1: Stationary user car and approaching obstacle vehicle

The following graph shows a plot of relative speed of obstacle while approaching user vehicle. Experimentally acquired speed is 106 cm/s. Average value on plot is 104 m/s. Corresponding error in value is 2%. Since all tests were performed using a prototype with the 1:16 scale down factor, measurement of the remaining time was problematic. To resolve this issue, our team decided to use the formula: time=distance/speed, to calculate remaining time. All calculations were performed by the microcontroller, and displayed on the GUI. By choosing appropriate values of speed and distance, the time is calculated and then compared to the one displayed on the GUI. On the graphs below, one can observe a plot of speed and general trend of decreasing time while obstacle car is approaching the user car.



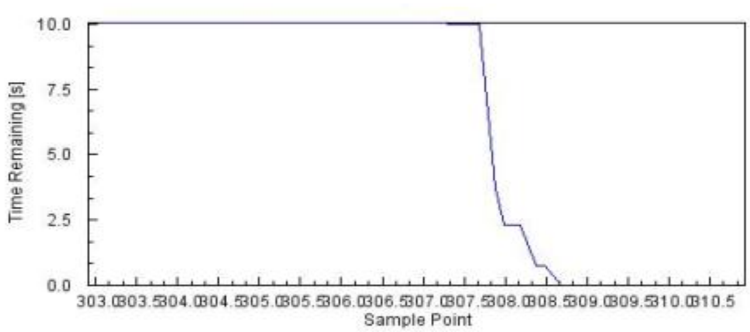


Figure 7: Plot of obstacle speed and remained time while approaching stationary user vehicle

Case 2: Both user car and obstacle in motion

In order to conduct the experiment, a 6 m long track was constructed. Using a stopwatch, the time needed for the vehicle to pass 6 m while in constant speed was measured. The experiment was run 5 times for 3 different PWM values (corresponding to varying user vehicle speeds), and then using the formula: speed= distance/time, the speed of the vehicle was calculated for each round. Finally, the relative speed between the user vehicle and the obstacle car was computed. The table below summarizes results obtained during testing:

	Speed measurement 1, cm/s	Speed measurement 2, cm/s	Speed measurement 3, cm/s	Speed measurement 4, cm/s	Speed measurement 5, cm/s	**Average measured speed (v_{avg}), cm/s**
Obstacle vehicle	105.24	105.9	106.1	105.36	106.2	**105.86**
User vehicle PWM = 41%	29.96	29.59	30.62	30.31	29.91	**30.08**
User vehicle PWM=51%	82.53	81.18	81.32	81.72	81.86	**81.72**
User vehicle PWM=61%	105.65	106.52	106.66	105.81	106.33	**106.2**

Table 8: Data acquired from speed measurements

	Relative speed, ($v_{obstacle} - v_{user}$), cm/s	Speed acquired from test, cm/s	Error, %
User vehicle Pwm = 41%	105.86 – 30.08 = 75.78	72	5.2
User vehicle Pwm=51%	105.86 – 81.72 = 24.14	23	4.9
User vehicle	105.86 – 106.2 = -0.6	-0.6	≈ 0

Pwm=61%			

Table 9: Calculation of relative speeds by microcontroller and % error

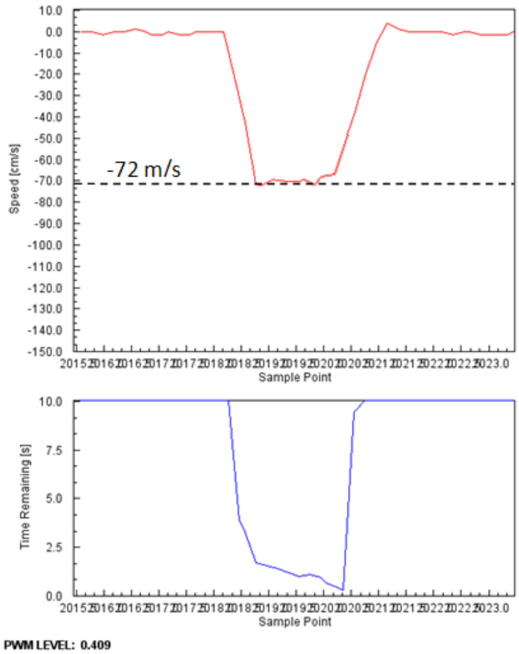


Figure 8: Relative speed between obstacle and user vehicle (running on pwm 41%)

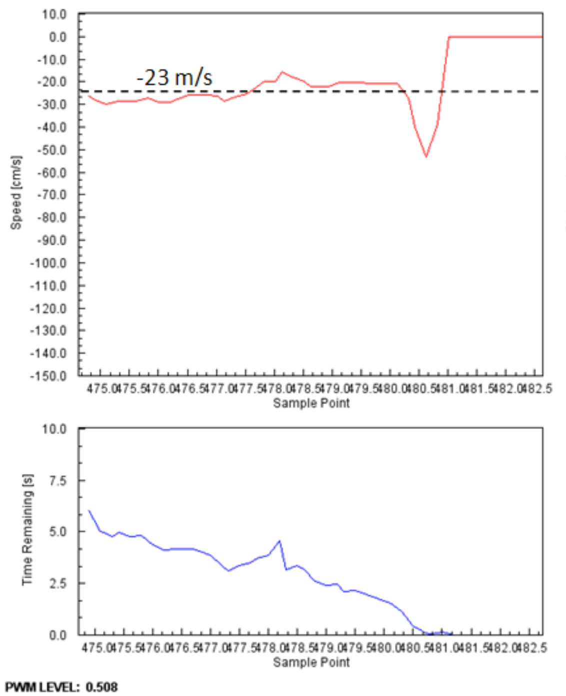


Figure 9: Relative speed between obstacle and user vehicle (running on pwm 51%)

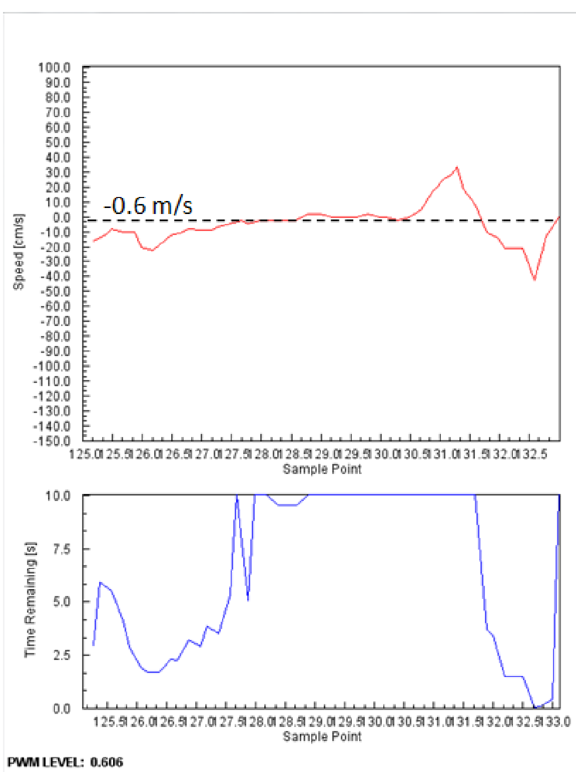


Figure 10: Relative speed between obstacle and user vehicle (running on pwm 61%)

Note:

The averaging factor determines the size of the array containing the subsequent instantaneous speed values. Increasing this smoothes out the sensor data, but slows down response time.

The time delay refers to the amount of delay added at the end of each microcontroller computation cycle. In conjunction with the averaging factor, it can be altered to smooth out sensor data.

To improve the appearance of the relative speed graphs, the following algorithm is used:

- Detect if the relative speed has reached steady state (by checking if the last x values are within a certain range)
- If steady state has been reached, continuously check if there is any acceleration. If not, hold the current average speed value. Do this for as long as the instantaneous speed readings fall within a range. Once the readings start going outside the specified bound, release the hold, and display the actual speed readings again until steady state has once again been reached.

Appendix M: Output Module test (Valikhan Kuparov)

The output module consists of a LCD display and a LED light. The LCD display consists of three values: distance to obstacle, relative speed and warning message. The pictures below shows the test results for the output module. Because of lighting conditions in a room where the photos were taken, brightness levels of LED are not distinguishable in the pictures below:

Case 1: Obstacle present in blind spot

Location of LED light

Figure 11: Output module feedback for case 1. LED is solid

Case 2: There is not sufficient time left to change lane

Figure 12: Output module feedback for case 2. LED light is dimmer.

Case 3: Obstacle is approaching user vehicle but not reached danger distances

Figure 13: Output module for case 3

Case 4: No obstacle present in blind spot or approaching user car.

Figure 14: Output module for case 4. LED light is turned off.

Test results conclude that the output module operates as expected. For different cases, the LCD display consists of relevant values and the brightness of the LED changes with respect to the distance to the obstacle.

Appendix N: Automation (Steering Lock) Test Results (Hani Hadidi)

Automation (Steering Lock) is enabled for danger levels 4 and 5. In order to test the steering lock, two simulation cases were implemented.

Case 1: Simulation of danger level 4 – no sufficient time left to change lane

Distance value of 10 cm and relative speed of 100 cm/s were sent to microcontroller, which computes remaining time and based on this data decides whether to enable lock steering or not.

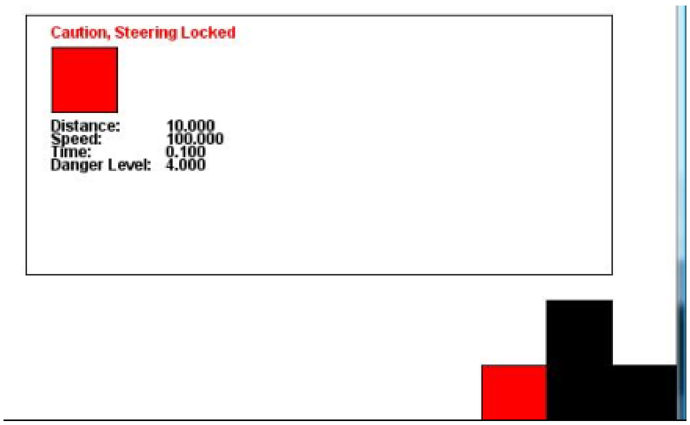


Figure 15: Lock steering enabled due to insufficient time remaining to change lanes

Case 2: Simulation of danger level 5 – obstacle present in blind spot

In this case obstacle was placed within the blind spot and user car attempted to turn towards obstacle. However microcontroller enabled lock steering.

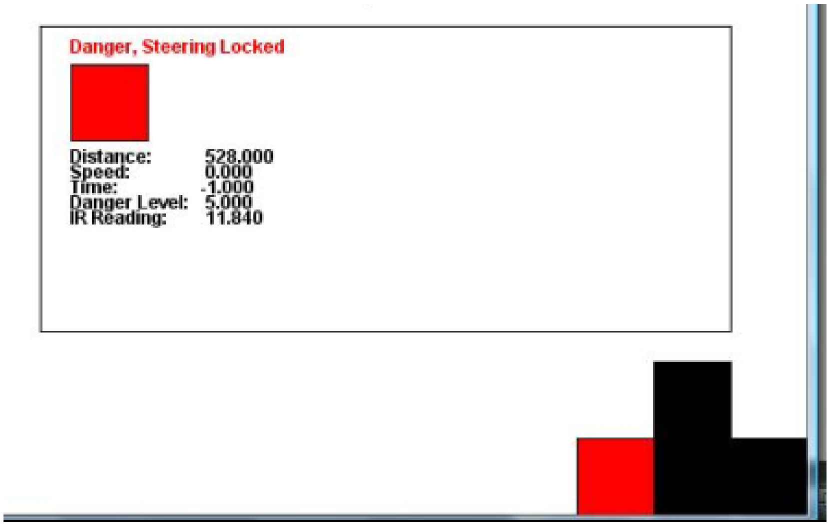
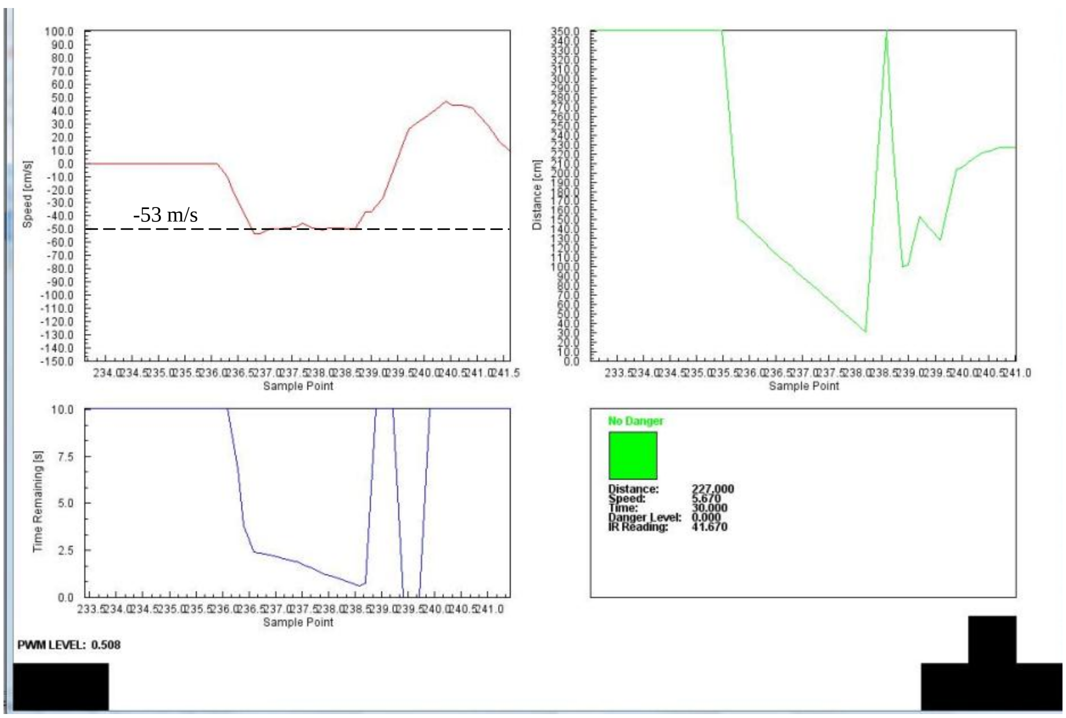


Figure 16: Lock steering enables due to presence of obstacle in blind spot region

Appendix O: System Level Test Results (All Members)

This appendix includes the results of testing for the integrated system.

Test 1: Obstacle car and user car were run at a known relative speed of 56 m/s. The microcontroller calculated the relative speed to be 53 cm/s. Corresponding error is 5.6%, which stays within requirement for relative speed calculations. Graphs of distance and remaining time accurately reflects the dynamics of the approaching obstacle car (i.e. as the obstacle comes closer to user car there is less time remaining to change lane). Since the screenshot was made at a time later than the instance where there was danger, the box at the bottom right block is green, and also indicates no danger.



Test 2. Main objective of this test was to verify correct performance of the steering lock feature. The test was conducted the under same conditions as Test 1, except the user car was kept stationary.. On graphs below, one can observe that as obstacle car approaches the blind spot of user car, time remaining to change lane decreases. When danger level 4 is reached, automation lock steering towards left is enabled. Red square and warn message in bottom-right box indicates it. As obstacle car enters blind spot region (danger level 5) lock steering is still activated. This test was executed several times, and in all rounds the integrated system operation satisfied the requirements and results were as expected.