Faculty of Engineering and Applied Science

ENGR 4941U Capstone Systems Design for ECSE II

Design and Development of CARLA Hardware in the loop

R5: Final Engineering Report

Team Members

Savan Patel, 100583384

Justin Kaipada, 100590167

Amalnnath Parameswaran, 100585138

Georgy Zakharov, 100588814

Faculty Advisor: Dr. Akramul Azim

Capstone Coordinator (Fall 2019): Dr. Q. Mahmoud

Table of Contents

List of Tables

List of Figures

1. Executive Summary

CARLA hardware in the loop(CHIL) is a project that aims to make a modularized test bench system and a simulator based test system for autonomous driving features. We found out that the Automotive industry lacks open source documentation and test systems when it comes to car hardware. Existing systems are either too proprietary or too expensive. For students to learn or for professionals to test the safety technologies that are developed for modern cars requires vast amounts of simulator based testing. With CHIL we tried to make the bridge between hardware and a simulator easier and cheaper with open source technologies. After a few issues with the hardware and software integration part for which we found a solution, the team was able to connect a BMW's digital cluster with the open source simulator called CARLA. Now we were able to add several features into the mix to show how the system can be useful. Features include a basic LiDAR based obstacle detection, Lane departure mitigation system based on RGB cameras, fully working digital cluster, a steering wheel and pedals to control the simulation and the digital cluster. Conclusively making this modularized hardware and software loop taught us a lot of things and once again reminded us how open source technologies and good documentation could lead to the development of good technologies and software faster with better quality.

2. R1

Problem Identification

With the new generation of cars, analog clusters are getting switched to digital clusters which are cheaper to manufacture and customize and implement in all models of cars to streamline the user experience. However, with the digital clusters, it is hard to test all the core features that are needed. For example, to test the safety or self-driving feature you would have to add the cluster to a real car and take it out for testing. But if you were to test in a simulated or emulated environment that is cost-effective, then it will help speed up the development and testing of the software. To address this issue, we will be designing and developing a car instrument cluster that will be used in the simulated and emulated driving environment to test different software safety features that get implemented in cars. This system will also act as a means of users getting a driving experience without the need to be behind a car.

Background and Research Review

The automotive components are continuously evolving with more and more software. An example of such a component is the instrument cluster. Buying an off-the-shelf instrument

cluster is expensive and also gives us inflexibility in adding the functionalities of interest. It adds costs to testing out new features and if it happens to be damaged the entire car needs to be replaced. In this project, we aim to design a system using an instrument cluster, steering wheel, and a car simulator to get a simulated and emulated driving experience on different types of roads in different environments to test various software features. This will enable users to try out new sensors and driving capabilities being implemented in cars.

One of the research we did was figuring out how the cluster works in the cars. We learned that it works slightly differently for each car model as the OEMs have used their own pin layout design for the clusters. Some pins are used to receive PWM signals where some are meant to receive CAN BUS data while others just output a PWM signal and CAN BUS data. For the cluster is connected to all the sensors in the car such as speed, oil, temperature, fuel, etc. These all provide data to the clusters so it can display it to the user. Hacking a cluster is difficult as it requires you to know what each pin is used for on the back of the cluster and what type of CAN BUS data can be sent. Each OEM has a seperate code that they use to identify what type of data is coming in and what pin to direct the signal too. In this project getting the cluster to work with that data being sent from Carla is that hardest part.

Design Process

a) In terms of the design process, we will be using an iterative methodology because it will ensure we can build on top of what we have. Since we have two different systems that need to be built and the second system will leverage the foundation of the first system. This methodology seemed like the fit for what we needed to do. One of the reasons we are choosing this approach is because it will allow us to make errors and make fixes and iterate to new versions. Another benefit is that this model also allows us to shrink time frames for areas where not needed to have a rapid turnaround time. Since we only have 2 months till demo we needed a model that would allow for a quick turnaround time while being agile. As opposed to waterfall model we are able to quickly go through the different phases faster because some of the phases are closely related. Iterative model also allows us to meet the changing demands of the clients.

b) With this model we will firstly gather the requirements and then do some planning on how we will be executing it. After some analysis we will then meet with stakeholders to gather clarification and talk about the design that we have come up with. Once what is required is understood we are able to move onto the implementation part where we will be spending most of our time. We will be splitting up into teams to tackle the implementation of the system. Savan and George will work on the implementation of the cluster to Carla while Justin and Amal will be working on the software side adding self-driving features to Carla. We will have to do testing while we develop to ensure our parts work, however when we integrate the two parts we will

need to perform major testing to ensure the entire system works in uniform. At the end we would like to demo this system to the clients.

In the below table we have identified the BOM that we will be needing for this project

Product	Quantity	Price ($CDN)
USB steering wheel/pedals	1	160
Cluster (BMW e36)	1	200
Arduino Uno	1	30
Jumper cables	40	5
DC jack socket plug	1	5
AC power supply	1	10
RC Car	1	500

Table 2.1: BOM

The total price for hardware comes to ~$910.

c) Quality assurance activities will be applied by testing the hardware for usability. Does the hardware purchased work as expected. When modified does the hardware work as intended. These tests can be recorded and logged. In terms of the software portion, we will be doing software testing where we test each new piece of code that is added and also make sure that previous code works as well too. We will need to test various test scenarios that the system will encounter. Since there is a possibility to be many scenarios present and having limited resources to test the various scenarios there is a probability that a scenario might be skipped. Once the hardware is integrated with Carla, there will have to be a communication test between both mediums to ensure that everything is working as designed. We will have a checklist that gets signed off for each working criteria we set. This will help us track what works and what has not been tested yet. It is also a good traceability matrix to keep when presenting to stakeholders.

A sample checklist can look like the following example:

Test Case	Tested By (signature)	Tested Date	Fail/Pass

Table 2.2: Checklist

d) We will be tracking the project with the use of a website called Trello. It is a site that helps us pin activities/cards to the board. This helps us see what we are working on, assign members to the activity, give deadlines to the activities. Put up graphs, comments and other useful information to communicate with the team. We will be actively watching and updating the Trello board to help us stay on track and log our progress. The main reason behind using Trello is that it is free to use and easy to get started. These two are most important as we are on a budget and members do not need a tutorial on how to get started. Furthermore, we will also have weekly update meetings with the supervisor to make sure we are making progress and any issues are being addressed proactively. This will ensure that if any risk arises it will be taken care of and having weekly meetings and group discussions help minimize the possible risks involved in the implementation. Weekly group meetings where we discussion about our parts and progress helps identify early risks and solutions to risks. We also perform review and approval on actions that are requested as it is important to review actions for whether we need to do it or is there a better solution. These actions can include things like purchasing material, adding features or switching simulators. Each action needs to be carefully examined and its pros and cons need to be viewed as a group to ensure the best path is taken for the team.

e) The customer and other stakeholders are involved in this to a certain degree in which they can provide the requirements and suggestions only. The customer has only specified the basics that it is looking for and has kept the design and implementation of the system open to us developers. Additional features we would like to add is also upon us and how we go about adding them is purely our say in the project and design. In our weekly meetings, the stakeholders will be present and will be able to take part in the discussion of the product, its progress and risks encountered so far and the solutions taken to address them. Here they will be able to specify any additional requirements or comments. With the use of Trello the stakeholders will also be able to comment on the cards there and monitor the progress. Using Trello is a quick way for stakeholders to communicate with the team as everyone as visibility of the board. Another method is email if the stakeholder would like to communicate something important and urgently.

f) The team is organized in a non structured manner. We have identified one member as the team leader who is in charge of the communications and team organizablity. The team lead is also a developer too just like the rest of the members. In terms of splitting the work between the members, we all will be pitching into all aspects of the project but during certain times where work tasks are many, we will divide the parts up by volunteering ourselves to take on the tasks presented. The parts will be assigned on Trello and each member will update their own card on Trello. By doing this we can all be working on a part for the project and help build the system

rapidly by completing the separate parts needed to build the entire system. The parts that have been assigned for the first system are as below:
Savan & George - Cluster integration with Carla
Amal & Justin - Self driving feature development in Carla

Scenarios and/or Use Cases

The infotainment cluster we plan to develop will serve as a system that can be used to replicate autonomous driving and navigation technology. It can be used for first-level testing of new updates to autonomous transportation. It can also be used to give a user or customer a first level experience of what it's like to use new technology like it.

Use Case: Classic instrument cluster that comes with our system should show previous auto users how accurate and applicable a new computer-based driving technology can be.

Scenario: John is an auto dealership manager focusing on selling Audi cars. New cars have been getting more and more digital every year . Since John sells cars more on the expensive side of the market, the majority of his customers are above the age of 30. One day Marques a 35 year old businessman walks into the dealership looking for the latest and greatest cars Audi has to offer. Marquess looks at some top of the line executive sedans and says to John, all of these cars are filled with small displays with quickly flashing numbers, is there any new cars Audi sell which have the latest safety features but old user interface like an analog cluster? John sighs and says no. He said “All new Audi cars have fully digital cluster”. Marques said I will think about the cars later and leaves.

Use Case: Self-parking, lane detection and following capabilities of the system can be used for customer experience without an actual automobile. This system can also be used to test autonomous procedures without requiring an automobile.

Scenario: Mark is a new backend engineer,working in autonomous driving team for tesla. Current development and testing of each new feature or updating old features require him to write the code for his feature on his laptop and then find a testable car where he can upload his new patch and ask the automotive technicians and other safety staff to test his new feature for him. Since Mark is new to the team he has been making a lot of small changes and he is bothering the test team often. Because of the scheduled updates and tests that also needed to be done by other departments and other team members sometimes scheduling for a hardware test for a small change might take weeks. Mark wished there was an easy method to test his soft changes.

Stakeholder Requirements and Traceability Matrix

The primary stakeholders are the researcher groups. This is because they will be the main ones using this system to test various features and explore the driving experience without having to be inside a car.

The secondary stakeholders will be automotive companies, Dr. Azim, and the Faculty because these users are concerned with the input and output of the system as they will be using the system for adding and testing new features for autonomous cars.

The tertiary stakeholders will be the general public. However, the use of our system is limited for these stakeholders as the impact of success of the system is not a high priority to them but they do care about its success/failure status. The researchers would benefit the most if this system was a success.

The requirements from the primary stakeholders are:

- Create a system that will allow researchers to get a real-world driving experience without the need to get inside a car.
- The system created must be module such that if someone wants to do research on autonomous car it would allow for this task.
- The system must also allow researchers to implement sensors and perform various test cases with them.

The requirements from the secondary stakeholders are:

- Must have an embedded portion to the system
- Must be within budget of $1000
- There must be a realistic feeling provided
- The system should have a simulated environment to display the autonomous features such as lane assist and self parking
- The system must be able to be implemented in a scaled down car so that the experience it enhanced and the autonomous features can be tested in the real world with real world objects
- The speed of the car should be displayed on a digital cluster or an instrument cluster

The requirements for tertiary stakeholders are:

- System should work for all car models
- All dealerships should have this testing system

The traceability matrix table below shows the priority of the requirements mentioned above and how each requirement will be validated or verified. It will also help with figuring out the progress of the project so far based on completed requirements vs remaining requirements.

Requirement ID	Requirement	Priority	Verification/Validation
1.1	Connect the embedded system to Carla simulator to enable the cluster to display the speed of the car from the simulator	High	Will be verified by having the cluster display various speeds based on the cars speed in the simulator
1.2	Write the code to enable the car to drive by itself within the lane in the simulator	High	When the button is pressed, the car should be staying in the lane and driving forward
1.3	Write the code to enable the car to park itself in the simulator	High	When the button is pressed the car should park in the empty spot
1.4	Connect steering wheel to simulator	Low	The car should move in the direction the steering wheel is turned
2.1	Control the car through the external steering wheel.	High	If the car reacts in the direction of the steering wheel
2.2	Enable car to provide speed data to cluster	Medium	If the cluster moves according to the data coming in
2.3	Small scaled car setup has a camera attached to the car that does object detection and provides real-time footage to display	High	The car will stop if a human comes in the way or another car

2.4	Small scaled car will self drive itself within a lane	Medium	The car will not cross over the yellow/white lanes drawn
2.5	Small scaled car will park itself in the empty spot	High	Will park in the empty parking spot by itself

Table 2.3: Traceability Matrix

The requirement ID will act as a unique identifier so that the particular requirement can be tracked throughout the project life cycle. The Requirement column will be the description of the requirement. Priority column tells us what is nice to have, mandatory, or high, low, medium. Verification/Validation column will help us mark what has been completed and verified to be working by completing a user acceptance testing (UAT) with the client.

Definition of Acceptance Test

Acceptance Testing also called User Acceptance Testing will take the form of a usage scenario, an example, or requirements at separate milestones comprising of a pass or fail tests. The purpose of this stage is to verify product compatibility with business requirements and assess if the system is acceptable for delivery through meeting the user requirements.

Contract Acceptance Testing - The project team identifies and agrees upon relevant criteria and specifications at project inception. The acceptance testing is then performed on these criteria once the software is developed.

Alpha Testing - Performed on the first prototype produced and will be done internally by the members of the development team to assess issues and improve usability.

Operational Acceptance Testing - Verifying the processes for start-up and usage of the software/system, backup plans, user training and rebooting as required for the project.

Beta Testing - Takes place in the user environment and allows users outside of the development team the opportunity to test and provide feedback on the product, this will be done before the final stages of the development cycle to fix any outstanding issues prior to completion of product and preparation for production.

Project Plan

Capstone I

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)	TEAM MEMBER	PERCENT COMPLETE
Automation						
Automation Research	9/4	9/15	0	12	Amal, Justin, George	100%
Lane Detection	9/15	11/14	11	61	Amal	40%
Lane Assist	9/15	11/14	11	61	Amal	20%
Object Detection	9/26	11/14	22	50	Justin, George	10%
Parking Assist	9/26	11/14	22	50	Justin, George	10%
Simulator						
Simulator Research	9/4	9/17	0	14	All	90%
Ordering/Waiting for Parts	9/18	9/28	14	11	All	90%
Steering Wheel Kit Integration	9/29	10/10	25	12	Amal	100%
Detect in-game speed and submit to Arduino	10/11	10/20	37	10	Savan	75%
Instrument Cluster/Arduino Integration	10/21	11/1	47	12	Savan	10%
General						
Integration	11/15	11/21	72	7	All	0%
Optimization	11/22	11/28	79	7	All	0%
Reports						
Report 1	10/14	10/24	40	11	All	100%
Report 2	11/4	11/14	61	11	All	0%

Table 2.4: Capstone 1 Project Plan

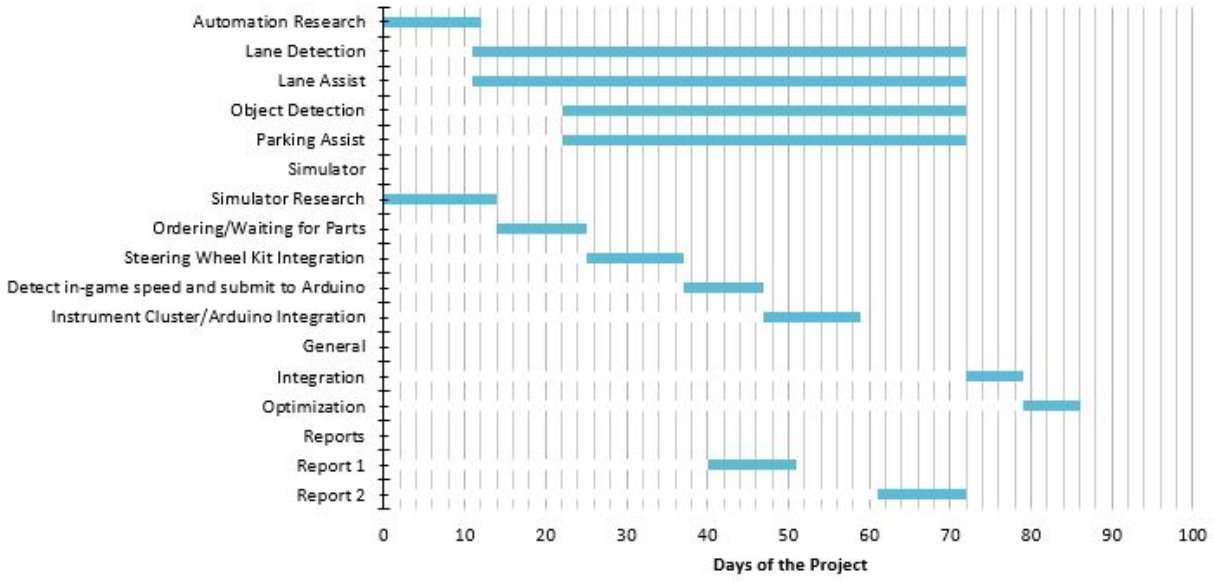
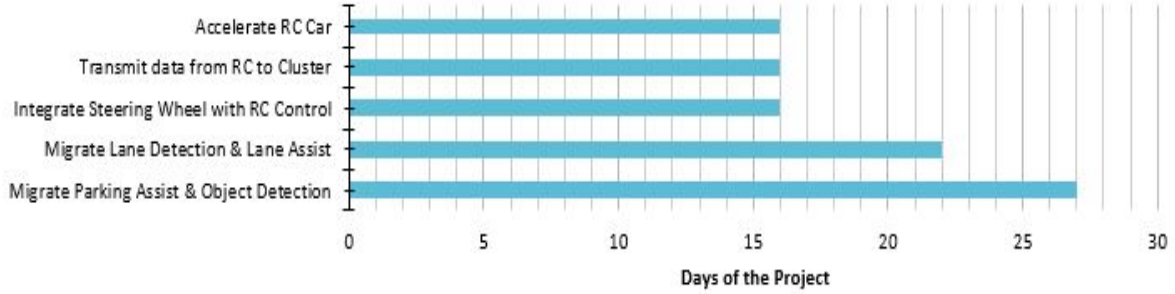


Fig 2.1: Capstone 1 Timeline

The tasks within this handbook can be broken down into two separate components that can be integrated at the end. These components are the 'Automation' component and the 'Simulator' component. A majority of the group will be working on the automation aspects due to its high level of difficulty. The vehicle simulator is quite hardware-intensive, therefore we have decided it is most efficient if we assign all those components to one member to make the most out of it.

Capstone II

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)	TEAM MEMBER	PERCENT COMPLETE
Remote Control						
Accelerate RC Car	1/5	1/20	0	16	George	0%
Transmit data from RC to Cluster	1/5	1/20	0	16	Savan	0%
Integrate Steering Wheel with RC Control	1/5	1/20	0	16	Amal	0%
Migrate Lane Detection & Lane Assist	1/20	2/10	15	22	Amal	0%
Migrate Parking Assist & Object Detection	1/20	2/15	15	27	Justin	0%



We roughly predicted the tasks that we will have for the second semester, though with very low accuracy. We expect this chart to change significantly when approaching the second semester to allow for more detailed tasks and more accurate time allocation. Though, for now, it does provide the team with a rough idea of what needs to be done in 2020.

Contribution Matrix

	George	Savan	Justin	Amal
Problem Identification		50%	50%	
Background and Research Review	25%	25%	25%	25%
Design Process		100%		
Scenarios and Use Cases			100%	
Stakeholder Requirements and Traceability Matrix		75%		25%
Definition of Acceptance Tests	100%			
Project Plan				100%
Contribution Matrix	100%			

Table 2.5: R1 Contribution Matrix

3. **R2**

1. Concept Generation and Analysis

1.1 Concept Analysis

For this project we have decided to go with a hardware and software approach as opposed to only hardware or software. The reason we want to use a hardware and software approach is so that we can have a simulator that is given the input from hardware and have the simulator provide an output to a piece of hardware. This is how some parts of the car work today, a user gives input to the car through a piece of hardware and then there is some software computation that provides an output. The hardware only approach would have not been feasible as its difficult for us to implement. The software only approach was feasible but it did not provide a good user experience. Due to these constraints with hardware and software only approaches we decided to go with a mix of software and hardware. We have decided to go with a Arduino, instrument cluster and steering wheel kit as the hardware components and CARLA as the simulator which will be the software part. With this as the core design we can build more features onto this design as it allows for modularity.

In this project we can divide our system into 2 major components. One is the Embedded systems component which deals with the digital cluster, the remote control (RC) car which will represent a real car, the controls and meta data we sent to the cluster which we get from the CARLA simulator. The other one is the CARLA simulator itself in which we plan to simulate all our semi-autonomous features such as lane and obstacle detection. The simulator is also how we are planning to train and test our semi-autonomous features before applying it to the hardware we plan to choose.

Some of the features being added are software based and use internal sensors built into the simulator such as lidar [5] for object detection, lane assist using the onboard RGB camera[5]. We are working on training a model using the simulator and then apply that on an RC car to see how it can be translated into the real world. Another approach that one can take with this system is to use some of the sensors from the simulator to provide information to the RC car.

1.2 Design 1 Idea

Below you will find some high-level design diagrams that showcase what can be built with the hardware and software approach, and how each component interfaces with each other.

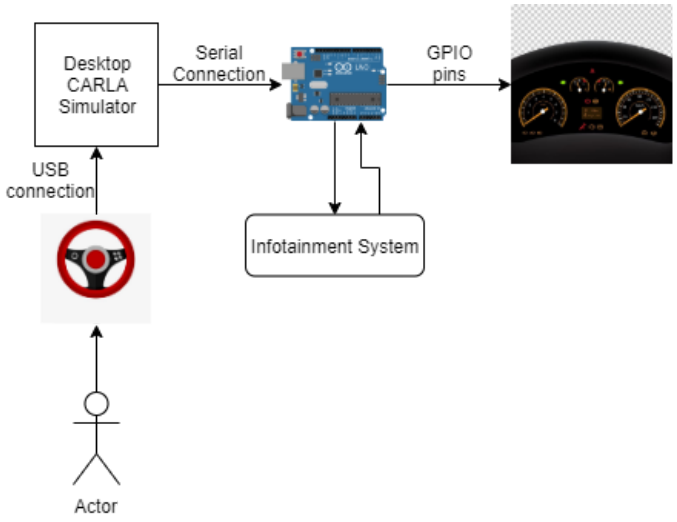


Fig. 3.1: Design 1

This first design depicted by Fig1. shows a control (steering wheel with pedals) which the user will control to give input to the simulator. The simulator will either be running on a PC. The simulator thus then will send the appropriate information to the instrument cluster via Arduino through a serial communication port. The system design above also shows an infotainment system connected to the simulator as this is supposed to be a modular system where you can add hardware or remove hardware. With some real hardware and a simulator, we will be able to give a realistic driving simulation experience.

1.3 Design 2 Idea

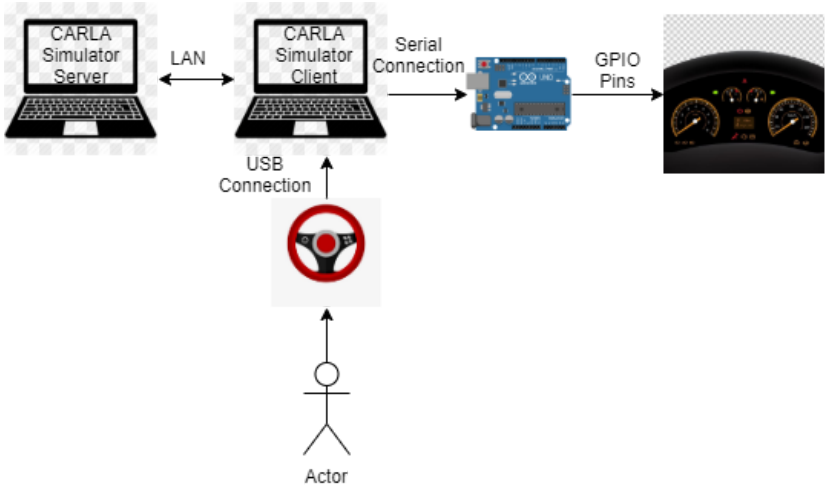


Fig 3.2: Design 2 outlining overall system communication

The system above shows the Simulator removed and replaced with a client and server. The simulator is split into two components a server-side and a client-side. The system in Fig.1 had the simulator server and client running on one machine whereas, this now separates it out. The benefit of distributing the client and server to different machines whether it be a laptop or desktop is to free up CPU and memory resources on a single machine. We predict this might help with the lag that is between the client and the Arduino which controls the instrument cluster. We also think there might be a latency in the cable that connects the Arduino and the Client. However, we would need to test this by checking the frames per second before and after and test drive in the environment and see if there is any lag or not.

1.4 Design 3 Idea

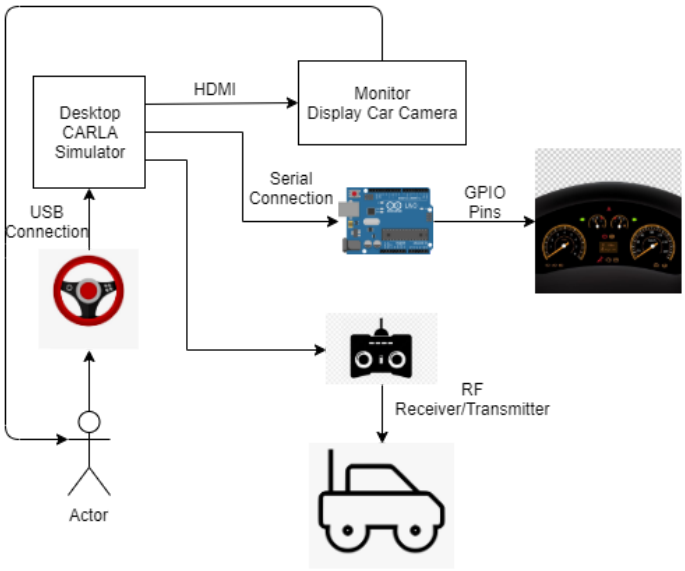


Fig. 3.3: Design 3 illustrating overall system communication

The diagram above shows a system where there is no simulator turned on, it's been toggled off to deliver a more realistic driving experience without being inside a car. This is done by having a monitor now that showcases what the RC car sees. If the simulator is toggled on then the RC car part would toggle off.

The steering wheel is the controller which will give input to the laptop that gets mapped to an RC controller which then would send a signal to manipulate the RC car. We plan on using the RC controller as the transmitter and the built in antenna will be the receiver on the car end.

The RC car has a camera and some sensors like an accelerometer to determine the speed. The camera is to provide the user with footage of what the car is seeing so that the user can steer the car accordingly. The video will be streamed via a wireless medium to the computer which will relay it to the monitor attached. The sensor will be communicating with the computer via Bluetooth or some other wireless medium. The speed data will then be displayed on the cluster. The cluster will be connected to an Arduino which is connected to the computer.

1.5 Design 4 Idea

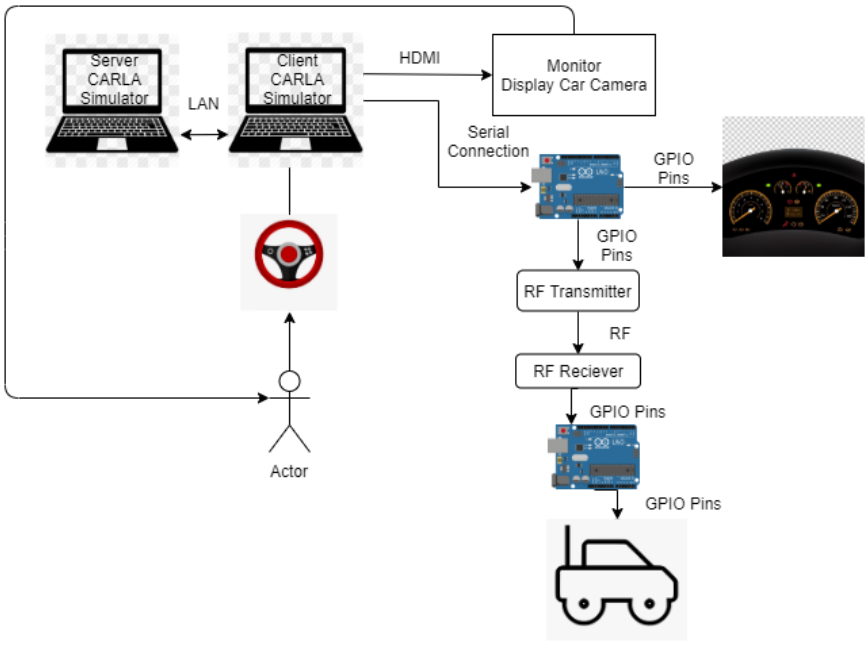


Fig. 3.4: Design 4 illustrating the overall system communication

The above design is very identical to Fig3. Design 3 with the only difference being that the RC controller is removed and replaced with an RF device and the simulator is distributed into client and server machines. The distribution is needed in case the user would like to toggle the simulator and not use the RC car. We do not know as of yet how we will integrate the RF device, but its goal will be to help us communicate between the computer and the RC car. This will be the most difficult part of this system. The benefit of this system is that it allows researchers to scale this up to a bigger car like a golf car or an actual car with more sensors to add to enable autonomous driving. The laptop will contain the script that gets input from the controller and then sends that data to the RF device which would relay the command to the RC car so it can act accordingly.

1.6 Communication Design

To communicate between CARLA and the Arduino we plan to use Python to send the data from CARLA to the Arduino. It will be done through a serial communication port that gets created on both ends will connect to it. The Arduino will start listening on COM 3 port 3000 and the Python script that is connected to CARALA will connect to that COM and port. Once the connection is established Python will be able to send bytes to the Arduino and the Arduino will be able to read that data. The below flow chart shows how the Python script will be trying to communicate with the Arduino.

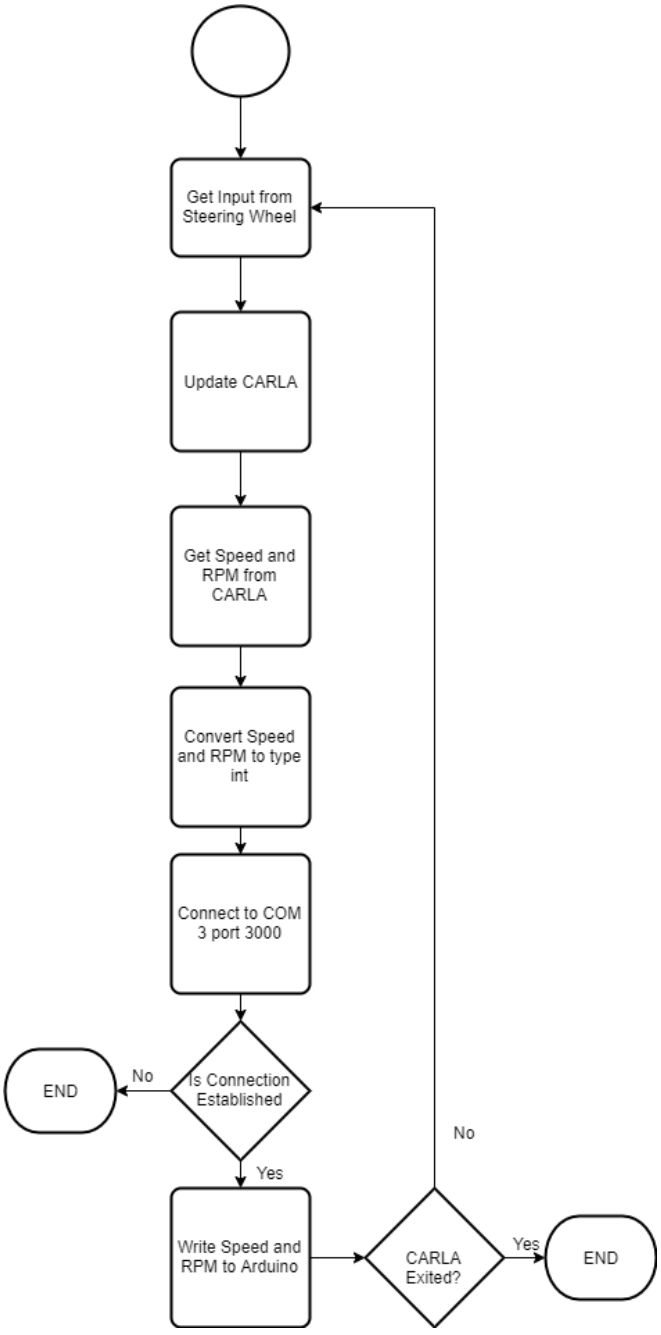


Fig. 3.5: A flow chart illustrating how Python code works

As you can see from above that the script loops till CARLA has exited because it constantly needs to be getting data. This is because we want to show the speed value of 0 when the car is stable and not moving. Also the script needs to be running so that we can get other sensor data too for our semi-autonomous driving capability that we have added.

In the below flow chart you can see how the Arduino will read the data given from the Python script and then relay that information to the instrument cluster. The speed to manipulate the cluster is very fast, however the communication gap between the Python and Arduino is what generates a little lag.

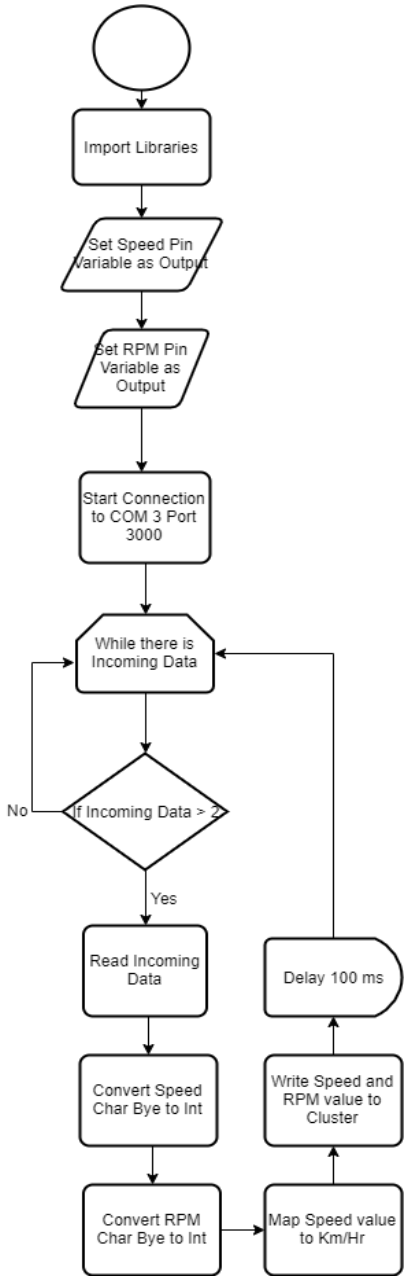


Fig. 3.6: A flow chart illustrating how the Arduino code works

As you can see in the flow chart above that there is a 100 ms delay after writing the values to the instrument cluster whereas there was no delay when the Python script was sending

the data to the Arduino. This is because the serial data being received takes time and the Arduino can process instructions much faster so when dealing with serial data we needed to add a small delay. We have noticed that serial communication is one of the slowest methods and so to make the delay short we have tested with different delay times and have found that the lowest delay time we can use for our scenario is 20 ms.

2. Conceptual System Design

2.1 Summary

After looking through all our design ideas, we have decided to choose design 3 as our final design. The final two designs came up to being between design 3 and design 4. The primary reason for choosing design 3 over design 4 was network latency. Though design 4 is theoretically feasible, in practice, we found that the network latency between the client and server was far higher than originally anticipated. We will touch on this in more detail.

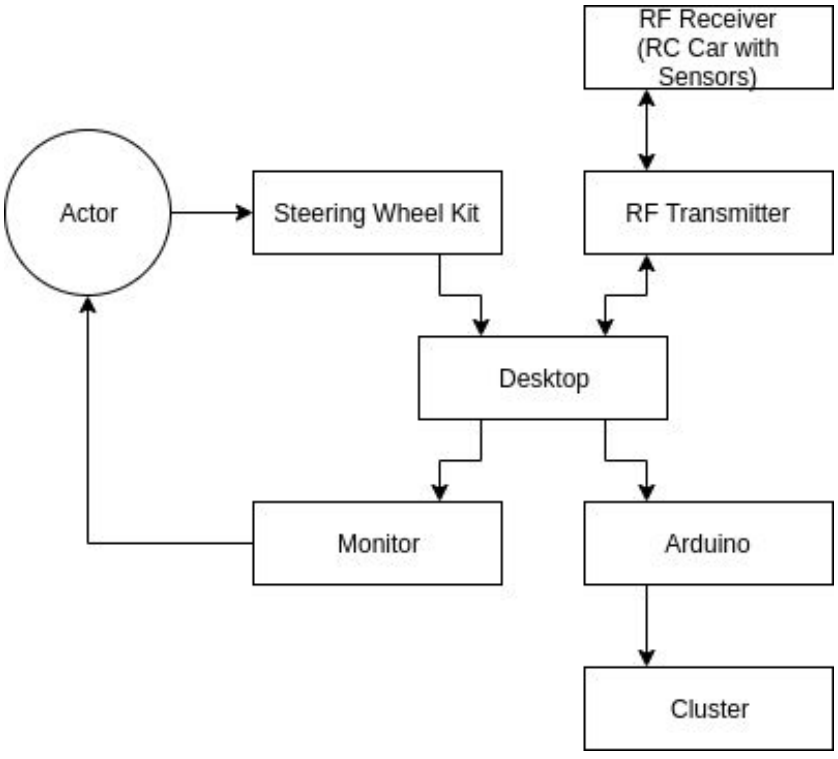


Fig. 3.7: Design 3A illustrating the overall system communication

2.2. Explaining Design 3A

Design 3A is simply design 3 as mentioned earlier. Some things that we need to refine include the communication between the desktop and the Arduino. There is currently a notable lag between the cluster and the Arduino. It is unclear where the latency is due to the bottleneck provided by the cable or other variables.

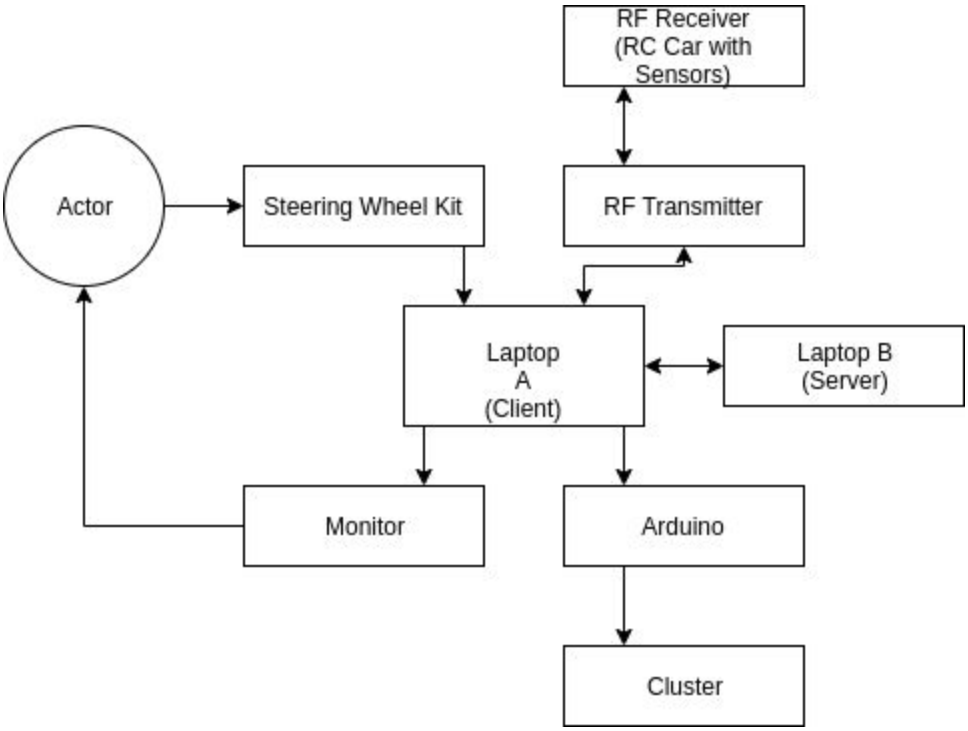


Fig. 3.8: Design 4A illustrating the overall system communication

2.3 Explaining Design 4A

Design 4A is inspired by design 4 and was the runner-up. It distributes the client and server into two different devices. We recognize that the laptop is able to handle either one or the other. The problem arises when the device has to run both simultaneously. 4A attempts to solve this problem by having one laptop run the client and one run the server and have them communicate via an Ethernet cable. This can help keep costs low while doubling the amount of resources available, though this may introduce network latency as a new problem.

2.4 Differences between 3A and 4A

Both designs function similarly, however of one key difference. In design 4A, we replace the desktop with two laptops that communicate with each other on a network. During development, we have tested our system with the laptops provided at Ontario Tech University as well as a mid-range gaming desktop. The gaming desktop does perform better, as expected, though mobility would be an issue, and lag does not completely disappear.

Design 4A's approach is to simply build a more powerful desktop to handle this simulator. This may work, however it may be cost-inefficient. Design 3A theoretically would work, however a concern that our group has is network latency. Our original experiments on a closed wire network proves that the latency is higher than expected, therefore, we decided to simply use a much more powerful system.

2.5 Connections & Interfaces

The steering wheel kit and Arduino are connected to the PC using USB connections. A monitor is connected to the PC in order to visualize the client via HDMI. The Arduino will connect to the instrument cluster via GPIO pins. [8]

Below is a diagram of the GPIO pin connection between the Arduino and the instrument cluster. [8]

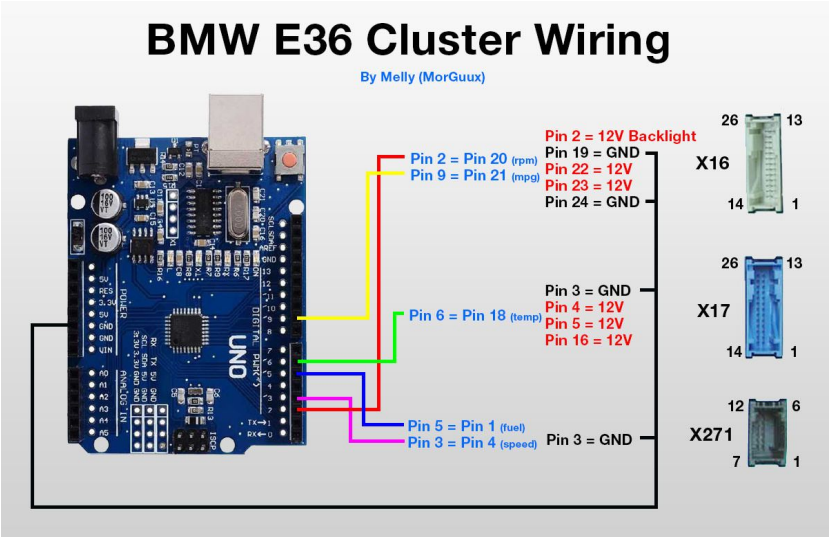


Fig. 3.9: GPIO pin layout of how Arduino should be connected to BMW cluster for SimHub

https://github.com/SHWotever/SimHub/wiki/BMW-E36-Cluster-Setup

The 2005 BMW E36 cluster was selected due to its huge community. It means there are plenty of resources intended on modifying this cluster for alternate uses. SimHub is a software tool that is used to allow games to utilize a connected BMW E36 cluster. Though we did not use SimHub, we utilized the setup that was necessary for the software to recognize the cluster. We then opened the serial port using the Python script that the client was running on and sent the Arduino similar messages as it would have expected if it were SimHub.

This was not the first cluster that we had tried. We were able to obtain a 2006 Audi A4 cluster but were unable to get it wired up and working simply due to its lack of support. We attempted a trial-and-error methodology when it came to that cluster and when we were unable to get it to work, we decided to research on well-documented and well-established moddable instrument clusters. This is how we found the E36 cluster and decided to invest in it.

The radio frequency (RF) transmitter and receiver have yet to be implemented, therefore we left the diagram vague when it comes to these components. For the RF transmitter, it is possible that we might have to use the same Arduino to send and receive messages to and from the RF transmitter. We may look into other solutions such as a separate Raspberry Pi or similar devices to carry out the task. Similarly with the RF receiver located on the RC car itself. Radio frequency will be used to communicate between the receiver and the transmitter.

2.6 Behaviour

In design 3A, we illustrate how the actor (user) will be able to control the steering wheel and pedals which will control the CARLA client simulator. The client simulator will act in the CARLA server which is hosted locally on the same device. The server will communicate back to the client on how it is interacting with the environment. The client then writes a serial signal to the Arduino, which then writes a square wave in the form of an integer for the cluster to understand. The client then also outputs the video feed to the user via the monitor. Simultaneously, the client would also be able to control the RC car's acceleration and steering.

CARLA's client code calculates the speed using the following formula:

$$speed = 3.6 \sqrt{x_{velocity}{}^2 + y_{velocity}{}^2 + z_{velocity}{}^2}$$

The 3.6 is used to convert the m/s that the velocity is utilizing into km/h which is used in the speed that will be displayed on the cluster. Everything under the root helps us determine the velocity of the diagonal in 3D space. Once the in-simulation speed is determined, we divide the number by 10 and write this to the Arduino. We divide the number by 10 because of how the

Arduino is mapped to the cluster. For instance, sending the Arduino 4 would point the instrument cluster to 40 km/h. This is why the speed is then divided by 10.

Currently, the RPM is mapped according to the speed. However, this is most likely to change in the future so we can get true RPM readings from the CARLA client.

Once we are able to get the RC car integrated, we want to assure that the RC car is able to use local automation when there is no signal detected. For instance, if the RC car travels out of the RF transmitter's line of sight, we want the car to be able to safely continue driving using local automation or safely pull over and come to a stop.

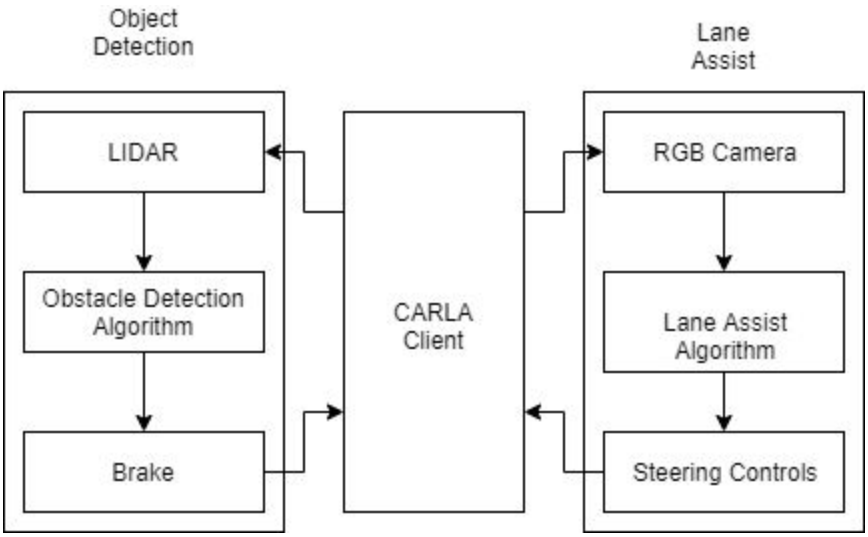


Fig. 3.10: Illustration of how CARLA client communicates with Lane Assist and Object Detection

We also enabled either buttons for the user to control on the steering wheel to control a lane assist feature. When enabled, the steering wheel will be downgraded to a second priority, allowing the vehicle to take control of the wheel. It will then attempt to keep the vehicle within the lane. This is a continuously improving feature that we plan to further improve with machine learning. Obstacle detection is another passive software feature added to the background. It does not need to be toggled and would passively observe the environment to see if there are any obstacles in the way. It will then automatically brake if it sees something in the way.

3. Definition of Integration Tests

Before beginning integration testing we will be doing unit testing on each component of our system; steering wheel kit, Arduino, dashboard cluster, simulator, RF transmitter & RC car.

3.1 Unit Testing

Steering Wheel Kit: Connect to a laptop or PC and test functionality using any racing simulator that is known to interface successfully with our kit (checking manufacturer website will yield possible candidates). Additionally, we can also utilize driver calibration tool in order to verify that controls are recognized by the computer.

Arduino: Connect to a laptop or PC and perform rudimentary calculations and/or send/receive sample signals between Arduino and laptop/PC to confirm functionality. Additionally, a test.ino file can be executed to test if predefined fixed values can be recognized and displayed on the cluster.

Dashboard Cluster: Connect to a power source to test that all lights/meters work.

Simulator (CARLA): Initiate a simulation on a laptop or PC and test whether inputs from keyboard/steering wheel and pedals will have desired output in simulator or not.

Semi-autonomy: The vehicle should stop if any obstacle is detected in front of the car. The vehicle should detect lanes and try to stay in it.

RF Transmitter: Transmitter functionality can be confirmed by transmitting something and simultaneously testing with an oscilloscope or building a simple analog ammeter circuit to test the activity of the transmitter.[9] Similar actions can be used to test the RF Receiver if required.

RC Car: Power on and use included remote control to test sending/receiving of signals between RC car and a remote device.

3.2 Integration Testing

For the integration testing to confirm the successful interfacing between all components we will employ a mix of Big Bang and Bottom-up testing approaches. [6]

Since we have verified individual functionality through unit testing we can interface the Arduino, simulator and steering wheel kit together and perform integration testing on this set of items by looking to see if actions taken using the steering wheel kit will reflect in the simulator

and whether the metrics of the simulated vehicle (e.g. speed, RPMs, etc) will be transmitted and successfully displayed on the dashboard cluster.

The next set of integration testing we can do is by modifying the Arduino to take on the role of the RF Transmitter, and modifying the RC car to take on the role of the RF Receiver, we will then test whether or not the transmitter & receiver are able to send signals back and forth. Finally, we can integrate the dashboard cluster, steering wheel kit & RC car as a whole system and test the complete integration functionality of the system, that is we will be using the steering wheel inputs as outputs (actions) for the RC car to perform, and the metrics of the RC car operation should be shown on the dashboard cluster.

4. Estimated Cost

Estimated manpower needed:

Number of days you plan to work on the project: 169 days, excluding holidays, weekends and days we need to focus on other academic commitments.

Number of workers: 4

Number of hours for each worker per day: 1.5

Total: 169 x 4 x 1.5 = 1014 hours of work

Percent of work time spent on each of the project tasks:

- Design - 30%
- Implementation - 30%
- Testing - 30%
- Documentation - 10%

Estimated cost of materials:

Product	Quantity	Price ($CDN)
USB steering wheel/pedals	1	105
Cluster (BMW e36)	1	135
Arduino Uno	1	30

Jumper cables	40	5
DC jack socket plug	1	5
AC power supply	1	10
RC Car	1	500

Table 3.1 : Cost Estimation table

The total price for hardware comes to ~$850. Additional costs that overflow our given budget will be taken care of by the Project Advisor. We estimate little to no overflow.

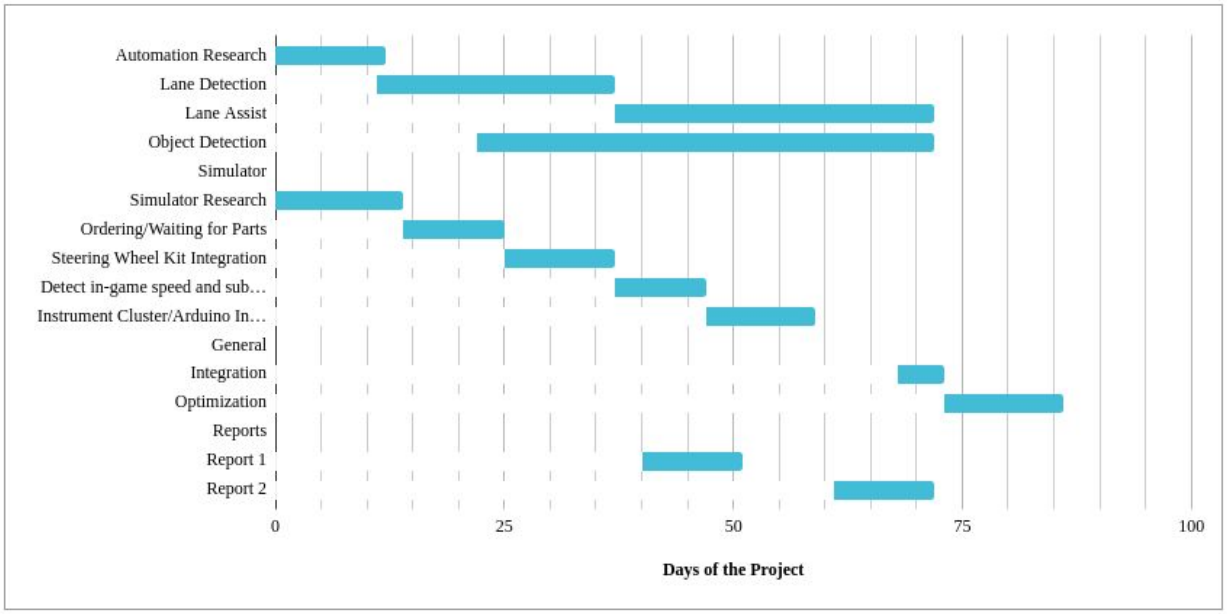
5. Updated Project Plan

5.1 Capstone I

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)	TEAM MEMBER	PERCENT COMPLETE
Automation						
Automation Research	9/4	9/15	0	12	Amal, Justin, George	100%
Lane Detection	9/15	10/10	11	26	Amal	100%
Lane Assist	10/11	11/14	37	35	Amal	100%
Object Detection	9/26	11/14	22	50	Justin, George	90%
Simulator						
Simulator Research	9/4	9/17	0	14	All	100%
Ordering/Waiting for Parts	9/18	9/28	14	11	All	100%
Steering Wheel Kit Integration	9/29	10/10	25	12	Savan	100%
Detect in-game speed and submit to Arduino	10/11	10/20	37	10	Savan	100%
Instrument Cluster/Arduino Integration	10/21	11/1	47	12	Savan	100%
General						
Integration	11/11	11/15	68	5	All	70%
Optimization	11/16	11/28	73	13	All	20%
Reports						
Report 1	10/14	10/24	40	11	All	100%
Report 2	11/4	11/14	61	11	All	100%

Many of our tasks have been completed according to the plan outlined earlier. A couple deadlines had to be shifted. Due to the complexity of the object/obstacle detection task, we decided as a collective to remove parking assist from the scope of the project.

Integration and optimization is also appearing to be a challenge. This is because the integration has been successful, though due to an unexpected latency between all the devices, we may have to consider re-integrating. Optimization is definitely helping reduce the latency, however we do have to investigate further into the issue.



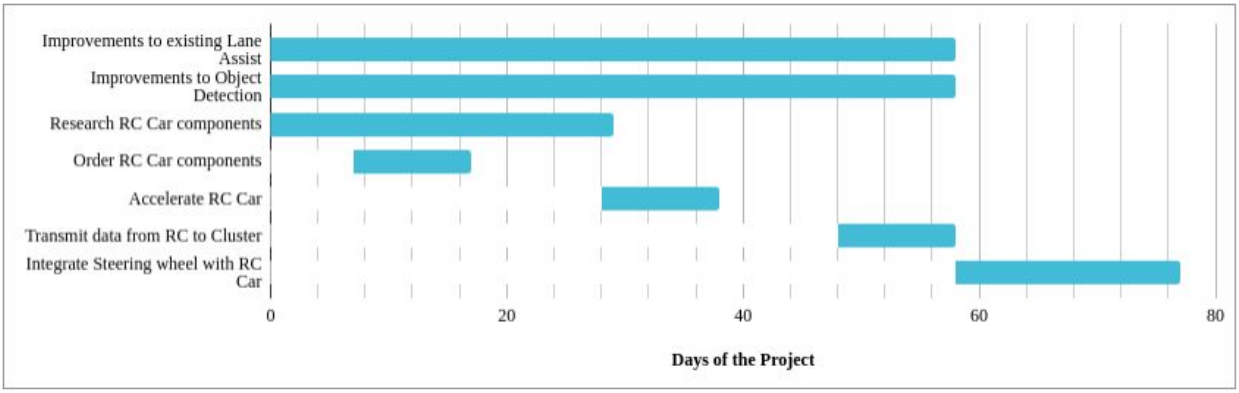
5.2 Capstone II

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)	TEAM MEMBER	PERCENT COMPLETE
Automation						
Improvements to existing Lane Assist	1/4	3/1	0	58	Amal	0%
Improvements to Object Detection	1/4	3/1	0	58	Justin, George	0%
RC Car						
Research RC Car components	1/4	2/1	0	29	Savan, George	0%
Order RC Car components	1/11	1/20	7	10	George	0%
Accelerate RC Car	2/1	2/10	28	10	Savan, Justin	0%
Transmit data from RC to Cluster	2/21	3/1	48	10	Savan, Amal	0%
Integrate Steering wheel with RC Car	3/2	3/20	58	19	Savan, Amal	0%

Table 3.2: Capstone 2 Project Plan

The tasks listed above are what our team has planned for the second term of this project. As expected, this is subject to change as additional tasks may be added to the scope and some may even be removed. However, to the best of our knowledge, this is the most accurate representation of what we plan on doing for the second term.

As performed in the current term, we plan on dividing the teams to the “Automation” and “Software” team. Though there are members working on both teams, we view these as separate instances. This is because during the development stages, automation is rather independent and the same with RC car. Since automation tasks are already integrated to CARLA, integrating the RC car with CARLA is essentially the only tasks that involve the RC car.



6. Contribution Matrix

The list of tasks and contributors are shown in Table 3.3.

Table 3.3: Contribution matrix

	People			
Task	**George**	**Amal**	**Justin**	**Savan**
Concept Generation & Analysis				X (100%)
Conceptual System Design		X (100%)		
Definition of Integration Tests	X (90%)		X (10%)	
Estimated Project Cost			X (100%)	
Updated Project Plan		X (100%)		
Contribution Matrix	X (100%)			
Formatting	X (25%)	X (25%)	X (25%)	X (25%)
Table of Contents				X (100%)
List of Contents			X (100%)	
List of Figures			X (100%)	
References	X (25%)	X (25%)	X (25%)	X (25%)

4. Revised design report

4.1 Overview

Since the last system detail diagram, we have added and removed some functions to the overall system. With the new changes to the system function, the below diagram depicts the changes made. The diagram below also shows the connection loop between the different hardware and the actor and the software. You can see how one input generates multiple outputs which lead to inputs to other parts of the system which leads back to the starting point as input. With the newly added features, we can add more functions to them. From the budget we have, we decided to purchase additional hardware to enhance the system.

In this new system, we have gotten rid of the RC car portion that was initially planned, we have replaced that with a head unit and Attention Monitor System. With the head unit that runs Android, we are able to root the device and manipulate it to act as a real head unit. We can use the Navigation Maps in the head unit and get the coordinates from CARLA to feed into the Navigation Maps. We can also add custom android apps that can be triggered by CARLA. With the addition of the Attention Monitor system, we are able to add a safety feature along with the lane assist. The Attention Monitor system helps track the driver's attention and when the driver is not paying attention to the road, the system plays a warning sound. The Attention Monitor system uses a webcam that is facing the driver, with the webcam we can also use this for other functions, such as capturing and logging the driver's emotions. With this additional information, we can see if there is any correlation between an accident and the driver's emotion. Maybe the accident occurred because the driver was in an angry mood. If this is the case then this data would back up the claim and a solution can be created to address this issue to decrease accidents caused by negative emotions.

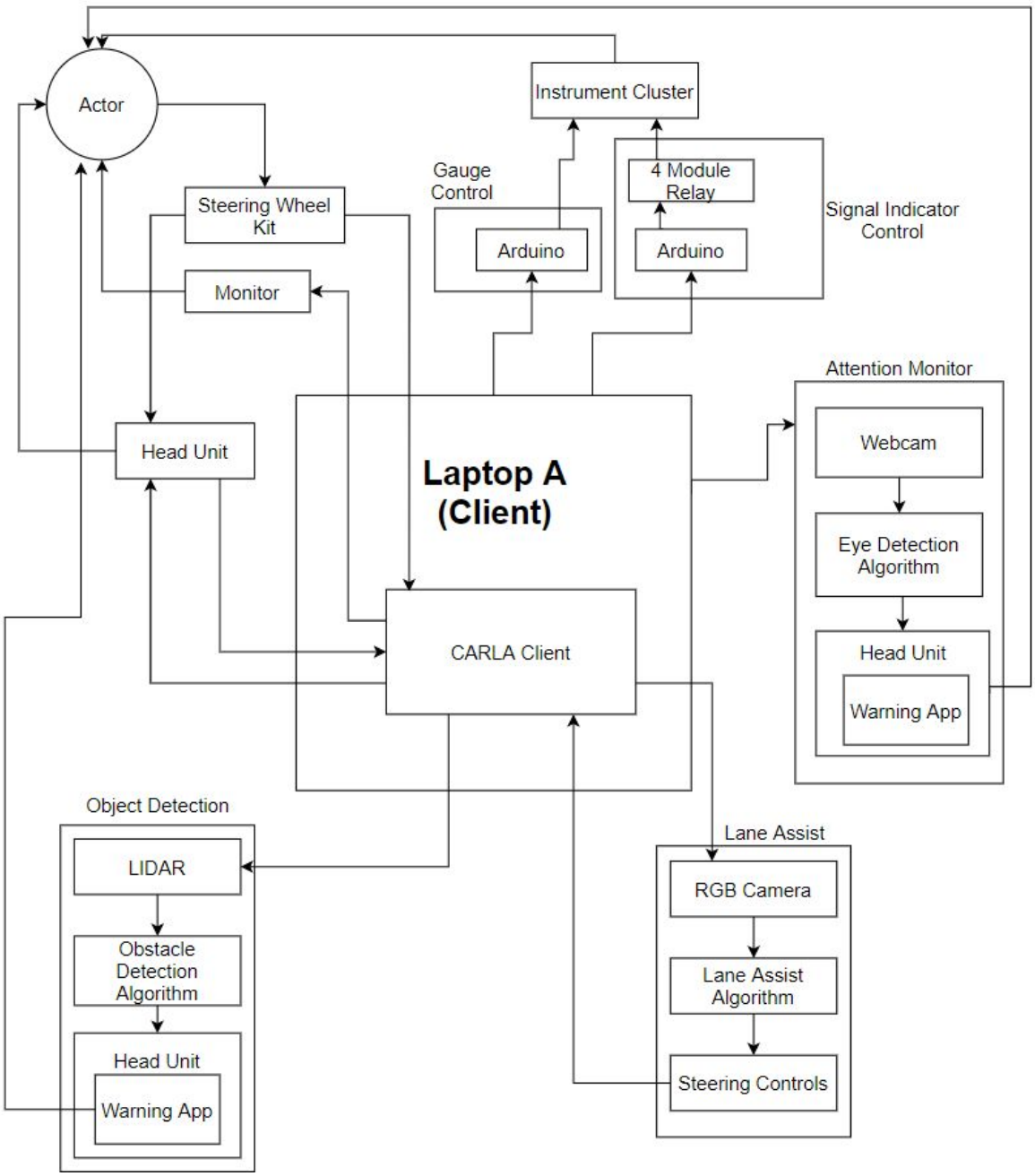


Fig. 4.1: Detail System Diagram

4.2 Head Unit

Head unit interaction with CARLA is another feature that we have decided to add to the existing solution. The head unit opens up multiple possibilities to explore and one of them that we have chosen to do is navigation. The below diagram explains how we plan on integrating the head unit with CARLA that will be running the prebuilt map in CARLA.

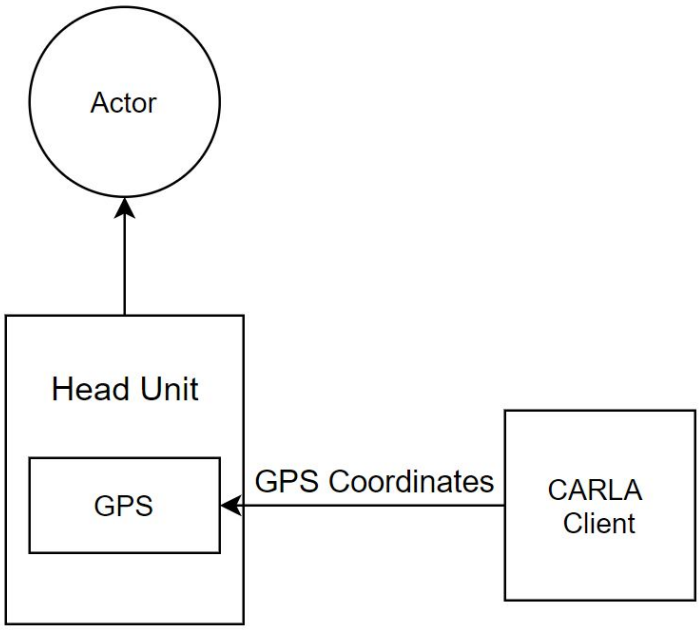


Fig. 4.2: Head Unit Map Design Diagram

As you can see from the diagram above that we will be using CARLA Client Map as our testing map for controlling the navigation on the head unit. Additionally, we will need to modify the CARLA API code so that it can send GPS coordinates to the head unit which will be running Waze, a navigation application. We will be sending GPS coordinates based on the coordinates that Google uses in the real map so that when the car drives around the map in CARLA, the Waze app in the head unit will display the direction movement accordingly.

4.3 Attention Monitoring System

The attention monitoring system uses an algorithm based on an eye blink detection algorithm[12], with some modifications to meet our needs. We decided to use an existing algorithm because it would be more accurate, robust and time-efficient. There are already algorithms out there for attention monitoring and they all have a similar foundation code, so why reinvent the wheel. Below is a pseudocode that explains how the attention monitoring system works.

```
Algorithm Awarness Detection is:
    Input: Alarm sound file,
           face landmarks model file
    Output: Alarm sound on loop till eyes are detected within frame

    (Note that EYE_AR_THRESH and EYE_AR_CONSEC_FRAMES values need to be adjusted to your enviornment and
    case)

    EYE_AR_THRESH <- 0.3
    EYE_AR_CONSEC_FRAMES <- 50

    COUNTER <- 0
    ALARM_ON <- False

    detector <- dlib facial detector
    predictor <- dlib facial detection file

    (lStart, lEnd) <- left eye facial landmark points
    (rStart, rEnd) <- right eye facial landmark points

    vs <- Webcam VideoStream source
    time.sleep(1.0)
    EyeCounter <- 0

    while True:
        frame <- read webcam frames
        gray <- convert frame to grayscale
        rects <- detect faces within frame, detector(gray,0)

        faceFound(rects)

        for rect in rects:
            shape <- predictor(gray, rect)
            shape <- face_utils.shape_to_np(shape)

            leftEye <- left eye coordinates
            rightEye <- right eye coordinates

            leftEAR <- calculate_eye_aspect_ratio(leftEye)
            rightEAR <- calculate_eye_aspect_ratio(rightEye)

            EyeAspectRatio <- (leftEAR + rightEAR) / 2.0

            leftEyeHull <- cv2.convexHull(leftEye)
            rightEyeHull <- cv2.convexHull(rightEye)

            Draw leftEyeHull on frame
            Draw rightEyeHull on frame

            eyesClosed(EyeAspectRatio,EYE_AR_THRESH,EYE_AR_CONSEC_FRAMES)

        Show Webcam Frame

        if key pressed = "q":
            break

    vs.stop()
    Stop webcam stream and shut down
```

Fig. 4.3: Attention Monitoring Algorithm

```
    function calculate_eye_aspect_ratio(eye):
        A = dist.euclidean(eye[1], eye[5])
        B = dist.euclidean(eye[2], eye[4])
        C = dist.euclidean(eye[0], eye[3])
        EAR = (A + B) / (2.0 * C)
        return EAR

    function faceFound(rects):
        if not rects:
            EyeCounter += 1
            if (EyeCounter = 70):
                if not ALARM_ON:
                    ALARM_ON <- True
                    Play Alarm Sound
                Display Alert Message
                EyeCounter <- 0
                ALARM_ON <- False

    function eyesClosed(EyeAspectRatio,EYE_AR_THRESH,EYE_AR_CONSEC_FRAMES):
        if EyeAspectRatio < EYE_AR_THRESH:
                COUNTER += 1

                if COUNTER >= EYE_AR_CONSEC_FRAMES:

                    if not ALARM_ON:
                        ALARM_ON <- True
                        Play Alarm Sound

                    Display Alert Message on frame

                    ALARM_ON <- False
            ELSE:
                COUNTER <- 0
                ALARM_ON <- False
```

Fig. 4.3: Attention Monitoring Algorithm Continued

For this algorithm, we are using an already built facial landmarks file that contains the coordinates of facial landmarks such as eyes, nose, chin, face, and eyebrows. With this we can easily detect a face within a frame, then once we have detected the face we use the eye landmarks to identify the eyes on the face within the frame. We then determine the euclidean distance of both the eyes and see if its value is below a certain threshold that we decide. If it falls below a certain threshold, say less than 0 then it would indicate the eyes are closed. If it's greater then they are open. If the eyes are closed then increment the counter to determine how long the eyes were closed. If they were closed for more than the threshold set, sound the alarm. If no faces are detected in the frame then increment the counter. If the counter is greater than equal to

the threshold then sound the alarm. This logic is encapsulated within a while loop that does not end till the driver shuts the feature off by pressing ‘q’ on the keyboard.

4.4 LiDAR based obstacle warning

Previously we were able to configure the LiDAR system in CARLA to be activated while driving and then provide us with real time 3D data points. We were also able to render the LiDAR data in a 2d format using a PyGame window, alongside the 3D world view of the simulator running on the Unreal engine. This semester we looked into making the LiDAR data useful for an obstacle detection and warning system, even possibly adding a collision mitigation system to apply breaks when a direct obstacle is detected. We quickly found out that to fully make use of the LiDAR data we need neural networks based deep learning systems that need to be trained to detect any anomalies on the road. We knew this would be a big overhead on the project as a whole and this one feature alone could take months to collect data for, train and implement. So we decided to go with a simple bounding box based algorithm that is already used in RADAR based safety systems. Similar algorithms[13] are used by most of the manufacturer's to implement collision mitigation using RADAR.

The bounding box algorithm[14] is very simple in theory. We pick a region in the 3D space that we monitor all the time, we check if anything enters this box. If it does, a collision is bound to happen. Translating this to CARLA we analyzed the 3D point cloud and found the points that could create a bounding box in front of the car. Then we checked the LiDAR data for every tick to see if any of the points are detected inside this region. If so, warn the user about an obstacle. We could also apply breaks to mitigate collision and alert the user using the same system we implemented for the attention monitoring system.

5. **Ethical Considerations**

Our research project didn’t require any 3rd party participants to be present during the design or development phase of the project. All research was done online and with the help of previous and older research papers and algorithms. Only 3rd party help we acquired was when we were doing the finishing touches which involved getting the help of some engineers from the company Vector Zero when trying to resolve issues with re-building the CARLA executable. With the help of the Faculty Advisor we were able to make sure the help we got from these engineers was permitted within the project scope. We ensured that we documented the help provided in the design report. In the end the feature we were asking for help with was not even completed due to unexpected circumstances.

- We made sure any communication between all the stakeholders involved was done with honesty and full transparency.
- We made sure we addressed all the problems, failures and struggles when it came to any of the features we designed or implemented and was brought to light immediately to mitigate the risk on all people involved and move the project forward smoothly.
- We made sure we don't mislead or misrepresent any of the details involved in our project in any capacity. All previous reports showed exactly what we are doing with what products and how much work was done on each project feature.
- While handling the licensed product from Vector Zero we made sure we maintained an adequate level of security and confidentiality when dealing with the software product. We made sure all the software we were given stayed on the one lab computer that was supposed to be licensed for.

6. **Safety Considerations**

When we were building this project, we knew that the hardware portions of the project would require safety procedures to be followed. That is why when testing the instrument cluster for the pin layout we used the correct and indicated power supply voltage of 12V. We also used proper wire handling techniques such as grounding the power and using jumper cables. When soldering the wires together we used a well ventilated area and made sure to wrap the exposed wires in a shrink wrap so that they are not exposed anymore. The hardware we had did not require us to build anything from scratch so the tools we used were kept to a minimal set of basic tools such as heat gun, soldering iron, and screwdriver. Some of the other safety considerations we took into account were not overloading the Arduino or instrument cluster, making sure we did not leave the power running overnight on the system. Overall we made sure to follow the safety protocols we normally follow in a lab. We had kept our bench clean of liquids and floors free of backpacks.

In terms of the software safety considerations, we made sure to make the system such that the user can experiment with various features of CARLA and our system without affecting each other or having any crashes. We made our lane assist, attention monitoring and lidar obstacle detection systems optional. This will allow the user to enable or disable the systems based on their preferences. However, when the system starts, the attention monitoring and lidar obstacle detection systems are enabled by default. Also the emotion logger in the background is enabled when the attention monitoring system starts and we used that to add a bit of stress to the system to see if it would have any effect on the performance, however it did not which means that the system is robust and can handle multiple running processes. This simulates an actual car that runs many safety systems and logs it all in the background.

7. **Conclusions**

To conclude, CHIL is a platform that was necessary to create in the current fast-evolving automotive generation. Not only does it act as a testing bench where many modules can be tested independently of the system, but it also fuels innovation into the automotive industry. By introducing a simulated environment that is both cost efficient and can be scaled up, we are able to get CHIL into the hands of many developers. This allows for much more developers to develop custom autonomous algorithms or compare two different braking algorithms to see which one is most efficient.

Our team believes that CHIL is the future of automotive development and would love to continue contributing to it past this stage. We have opened the repo to allow it to continue being an open-source tool that other developers can contribute to and modify as necessary. This does not only stem from the fact that CARLA itself is open-sourced, but that for a small group like us, it is the best way to scale CHIL up and get it into the hands of the developers that can make the most impact.

CHIL will help bring automotive technology into the current generation. Though recent technologies have been impressive, they are still lagging behind when it comes to innovation in other fields such as computer technology. By merging the two industries together, we are able to fade the lines between the two industries and bring innovation back into the automotive industry. We have seen what the presence of software industries has done to the automotive industries, such as Google's Android Auto, Apple's CarPlay, Blackberry QNX & more. We hope to be one of those software industry players, fueling the automotive industry to the next generation.

8. **Acknowledgements**

We would like to thank Dr. Akramul Azim for providing guidance on this project, providing the use of a lab room for development and testing purposes, and working with us to acquire the necessary software licenses.

We would like to thank Dr. Qusay Mahmoud for helping clarify and resolve any issues with the Capstone reports, working around scheduling issues, reserving rooms and time slots for Capstone development work and communicating any and all needed information to the students in a timely manner.

We would like to thank Vector Zero for working with us and providing an academic license to their software RoadRunner so that we could explore the option of integrating custom map importing functionality into our CARLA workbench.

We would like to thank the Vector Zero staff member Tim Wang for assisting us in attempting to resolve the issues we have with re-building the CARLA executable to integrate map importing functionality.

We would also like to thank our fellow Ontario Tech students engaged in their own Capstone projects for sharing spare hardware, resources and working together to bounce ideas off of each other.

9. **References**

[1] A. Dosovitskiy, A. Lopez, F. Codevilla, G. Ros, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” *arXiv.* [Online]. Available: https://arxiv.org/pdf/1711.03938.pdf. [Accessed: 07-Oct-2019].

[2] A. Nu, “An Introduction to the CAN Bus: How to Programmatically Control a Car,” *Medium*, 06-Jun-2017. [Online]. Available: https://news.voyage.auto/an-introduction-to-the-can-bus-how-to-programmatically-control-a-car-f1b18be4f377. [Accessed: 28-Sept-2019].

[3] CARLA Team, “CARLA Documentation,” *CARLA Simulator*. [Online]. Available: http://carla.org/. [Accessed: 02-Oct-2019].

[4] TechnicalEngineer, “Speedometer Using Arduino and CAN Protocol,” *Hackster.io*, 01-Oct-2018. [Online]. Available: https://www.hackster.io/TechnicalEngineer/speedometer-using-arduino-and-can-protocol-9f72e5#code. [Accessed: 29-Sept-2019].

[5] - zeldarulez and Phil Frost, “How can I tell if an RF transmitter is transmitting?,” *Electrical Engineering Stack Exchange*, 09-Aug-2013. [Online]. Available: https://electronics.stackexchange.com/questions/78530/how-can-i-tell-if-an-rf-transmitter-is-transmitting. [Accessed: 10-Nov-2019].

[6] - “Integration Testing,” *Software Testing Fundamentals*, 03-Mar-2018. [Online]. Available: http://softwaretestingfundamentals.com/integration-testing/. [Accessed: 10-Nov-2019].

[7] - Subirón, N. (2019). *Cameras and sensors - CARLA Simulator*. [online] Carla.readthedocs.io. Available at: https://carla.readthedocs.io/en/stable/cameras_and_sensors/#ray-cast-based-lidar [Accessed 1 Nov. 2019].

[8] - fast90, “fast90/E36Clock,” *GitHub*, 23-Aug-2015. [Online]. Available: https://github.com/fast90/E36Clock. [Accessed: 13-Nov-2019].

[9] - oceanservice.noaa.gov. (2019). *What is LIDAR?*. [online] Available at: https://oceanservice.noaa.gov/facts/lidar.html [Accessed 2 Nov. 2019].

[10] - *https://www.youtube.com/playlist?list=PLQVvvaa0QuDeI12McNQdnTlWz9XlCa0uo*. (2019). [video] Texas: Youtube.

[11] - SHWotever, “SHWotever/SimHub,” *GitHub*, 24-Jun-2019. [Online]. Available: https://github.com/SHWotever/SimHub/wiki/BMW-E36-Cluster-Setup. [Accessed: 13-Nov-2019].

[12] A. Rosebrock, “Eye blink detection with OpenCV, Python, and dlib,” *PyImageSearch*, 24-Apr-2017. [Online]. Available: https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/. [Accessed: 28-Jan-2020].

[13]"How does Honda's Collision Mitigation Braking System™ work", *Matt Castrucci Honda*, 2017. [Online]. Available: http://www.mattcastruccihonda.com/blog/how-does-hondas-collision-mitigation-braking-system-work/. [Accessed: 19- Dec- 2019].

[14]"3D collision detection", *MDN Web Docs*, 2020. [Online]. Available: https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection. [Accessed: 15- Dec- 2019].

[15] “Hardware Integration Test -,” *Pi Innovo*, 06-Jul-2018. [Online]. Available: https://www.pi-innovo.com/hardware-integration-test/. [Accessed: 10-Feb-2020].

10. Appendices

- https://github.com/aphrx/carla-capstone
- https://www.vectorzero.io/ Vector Zero is the developer of the Road Runner software that is (according to CARLA) compatible with the CARLA simulator for creating and importing custom maps.
- http://carla.org/
- https://github.com/carla-simulator/carla

11. **Contribution Matrix**

Table 11.1: Contribution matrix

Tasks	**People**			
	George	**Amal**	**Justin**	**Savan**
Report Formatting	X(25%)	X(25%)	X(25%)	X(25%)
Revised Design Report	X(25%)	X(25%)	X(25%)	X(25%)
Executive summary			X(100%)	
Ethical Considerations			X(100%)	
Safety Considerations				X(100%)
Conclusion		X(100%)		
Acknowledgements	X(100%)			
References	X(50%)			X(50%)
Appendices	X(50%)	X(10%)	X(30%)	X(10%)
Contribution Matrix	X(25%)	X(25%)	X(25%)	X(25%)