Department of Electrical
& Computer Engineering
Faculty of Engineering & Architectural Science

Vulnerable Road Users Safety for Intelligent Transportation System (MJ06)

COE 70A/B

Final Report

By

Nirav Patel (niravkp97@gmail.com)

Computer Engineering Capstone Design Project

Bachelor of Computer Engineering (BEng)

Ryerson University - 2022

Table of Contents

Acknowledgements

I would like to give a huge thanks to our **FLC Prof. Muhammad Jaseemuddin** for taking the time each week to guide us through this project.

I would also like to thank all the Open-Source software developers that made Xamarin Forms (Framework used to create the Smartphone App), NodeJS and ExpressJS (Used for Web App and Web Server), Arduino (Used for Model Car's microcontroller programming), React (Used for Web App), and many more.

A Huge thanks to all the family members and friends that supported us on this project. This project would not be possible without the help and support from many individuals.

Certification of Authorship

I hereby certify that I am the author of this document and that any assistance I have received in its preparation is fully acknowledged and disclosed in the document. I have also cited all sources from which I obtained data, ideas, or words that are copied directly or paraphrased in the document.

Project Member	Signature
Nirav Patel (niravkp97@gmail.com)	

*By signing above, you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Abstract

The objective of this project is ensuring safety for vulnerable road users. While many other existing technologies such as the smart navigation systems used by popular corporations like Tesla already covers the scope of making vehicles and road systems safer to a certain extent, our project takes a unique approach to it by utilizing the ubiquity of roadside infrastructures by converting them into smart RSU alongside the implementation of V2X which is responsible for communication between vehicle and vehicle (V2V), vehicle and infrastructure (V2I) and vehicle and pedestrian(V2P). It will be implemented using an IEEE 802.11p design because that is the design that is used in the majority of V2X communication systems.

The functionality of our model is based on the interaction between the smart RSU which detects any VRU on a crosswalk or any approaching vehicle, and the smartphone applications that were developed for the drivers of the vehicles and vulnerable road users (VRUs) on the crosswalk to send out warning signals. A comprehensive design for the intended purpose has been successfully modelled and the general idea behind the use cases it covers has been outlined in further details below.

Introduction

Road safety is a very important part of mobility. This is important not only to an individual but also the policy makers. The number of fatalities that come from the road has been decreasing over the years. But that's still not enough, the number of fatalities should be zero. That's the whole purpose behind this Vulnerable Road Users Safety for Intelligent Transportation System. Its whole purpose is to reduce the number of collisions and reduce the number of fatalities on the road as much as possible. This is done by increasing the safety in vehicles, but that's not enough. More improvements must be made to the VRUs because they account for most of the accidents. Essentially everyone in the world is considered a VRU because at some point in their lives they will most likely be crossing a street whether on foot or on a cycle. So, in order to fix the problem of road fatalities the VRUs must be given more attention. In 2020 5GAA Automotive Association released a white paper on Vulnerable Road User Protection with the following scope and use cases:

Scenario Category	Scope	Use Case (UC) Categories
S1: VRU high risk zones	Drivers or AVs are delivered warnings when they enter the area	**UC-1.1:** Pedestrian in crosswalks – send out alerts to motorists when pedestrian in a mid-block crossing. (Can be infrastructure-based messages to motorists). **UC-1.2:** Pedestrians in crosswalks at intersections. Provide alerts to motorist when pedestrians are crossing a side street on right or left of vehicle. Also provide alerts when pedestrian in crosswalk and signal is green for vehicle. (Can be infrastructure-based messages to motorist). **UC-1.3:** School Zone Warning – Send out alerts to motorist when they are entering a school zone area that is active. High concentration of pedestrians around the area at specific times, such as arriving and leaving times. **UC-1.4:** School bus warning. Similar to 1.3, but a dynamic high risk area (e.g. school bus, ice cream truck).
S2: VRU communicating directly with vehicles	Negotiation between VRU's device and approaching vehicle	**UC-2.1:** Car doors opening in the path of a cyclist **UC-2.2:** Interactive VRU crossing.
S3: VRU safety messages and AI	PC5 enabled vehicles and smartphone VRU devices send PSM, VAM Vehicles send out CAM/DENM, BSM	**UC-3.1:** Pedestrian/Motorists Collision alert – Alert provided to motorist (and pedestrian) if vehicle is driving over a certain speed and has a predicted collision path with a pedestrian. **UC-3.2:** Cyclist/Motorists Collision alert – Alert provided to motorist (and cyclist) if vehicle is driving over a certain speed and has a predicted collision path with a Cyclist. **UC-3.3:** Send alerts to smartphone mounted on eBikes. **UC-3.4:** Cyclist/pedestrian collision alert. **UC-3.5:** High density VRUs in urban crosswalk.

Figure 1: Use Cases outlined in the 5GAA White paper

While some of our scopes coincide with scopes and use cases of the whitepaper, this project however takes a different approach at achieving these goals. This project aims to focus on 3 scenarios that will involve different VRUs. The first one will be a VRU crossing a street. The second one will be a vehicle approaching a highly dense area with a lot of VRUs. The third one will be a vehicle approaching a VRU from a blind spot. This project is unique compared to others because it will be using raspberry pi devices to tackle the issue of road fatalities. The 3 use cases chosen also to set it apart from other projects because each use case deals with a VRU and as mentioned above, the VRUs need more attention in road safety to prevent road fatalities. However, only the pedestrian is the VRU that is considered in this project. That's the whole idea behind this project and why it's important. These 3 use cases in question are outlined and discussed in further details below.

Use Cases	
1) VRU Crossing	A pedestrian is crossing the road with a vehicle approaching it as the pedestrian is crossing
2) Vehicle approaching high populated area such as school	A vehicle approaches a high populated area such as a school with a lot of pedestrians at a specific time of day. Vehicles can be approaching at arriving times or leaving times of the school.
3) Vehicle approaching without the view of VRU	A vehicle is approaching an intersection with the view of a pedestrian who is about the cross the street.

Figure 2: Use Cases for the Design

Objective

While safety systems in many 4-wheel-vehicles have been improved over the last few years, the safety of many other road users such as pedestrians and bikers are still being overlooked. The vulnerable road user's safety system looks forward to solving such issues in more developed countries where infrastructures are provided, while giving room for further improvement as the system has big potential to grow bigger when required technology is distributed worldwide. As for now, the system is divided into 2 main parts, Roadside Unit and phone application. The Roadside Unit acts as an index server in a star topology communication network, while it also acts as real-time control signal when it detects possible collision (using camera and sensor). The phone application shows precautions and warnings about possible collisions to users, in addition, it automatically communicates to the RSU for real-time collision prevention.

With the combination of the mentioned main parts, the system is looking forward to reducing the number of collisions between vehicles and pedestrians at intersections to almost 0, reduce collision between cars and motorcycles/bicycles to at least 50% and reduce incidents due to jaywalking (To see detailed use-case, please refers to figure 0 for reference). Moreover, while providing a working model for the present, the system also lays infrastructure for future improvements, hence contributing to road safety for the near future.

Theory

1. Design overview and basic calculation

Like every safety system, time is a crucial factor that determines usability of the system. In order to create liability for this road safety system, response time must be calculated in milliseconds. However, due to technological limitations, the safety system will have both a real-time warning system and a non-real-time warning system.

- *Roadside Unit RSU*:
    - Responsible for providing local area network (LAN) for fast peer-to-peer communication. This way collisions can be prevented within an acceptable time interval.
    - Responsible for calculating, detecting and sending signals about possible collisions by using GPS location provided by users' devices.
    - Responsible for detecting pedestrians and incoming vehicles, then provides warnings using both traffic signals and phone application notification. This is the fastest and most reliable form of collision prevention.
- *Smartphone application*:
    - Responsible for sending precautions to users when they enter dangerous regions. This can be done by using heatmap (shows density of pedestrians in a region) and point map (shows distance and direction in which nearby motorcycles and bicycles are heading). All this information is processed by centralize cloud server in the region (first test region: GTA)
    - When users enter regions protected by the RSU, the phone application is responsible for automatically connecting to the LAN provided by RSU and streaming users' location in real time.

2. Mathematical calculation for collision detection

- *Camera-based collision detection system*:
    - Elegoo HC-SR04 motion sensor

      The ultrasonic sensor is built near crosswalks; therefore, it will only be triggered when someone with a tendency to cross the road gets close to the crosswalk. To prevent false alarms (which will significantly reduce credibility of the system), the sensor will only be triggered when it detects objects within a specified range.

- *GPS + Cloud Server -based collision avoidance system*:
    - How distance can be calculated:
        - To reduce unnecessary time complexity for calculation (algorithm to calculate distance between two points is polynomial), These GPS addresses are divided into regions. GPS coordinate's structure: $degree\ minute\ second\ N/S\ degree\ minute\ second\ E/W$, for example $40^o44'45''N\ \ 73^o59'11''W$.
        - This then will be converted to decimal degree where West/South indicates negative direction, for example: (40.7486, −73.9864). From this *Haversine* formula is applied as follow:
        - $haversine = sin^2(\Delta latitude) + cos(latitude_1) \times cos(latitude_2) \times sin^2(\Delta longitude/2)$

          $distance = earth\ radius \times (2 \times atan2(\sqrt{haversine}, \sqrt{1 - haversine}))$

          $earth\ radius = 6371\ km$
    - Algorithm in use:
        - Closest Pair Algorithm: since all devices will stream its GPS coordinates to the server every 0.5 seconds (server will have to process a lot of data), an optimal solution to detect road users is crucial. Time complexity of the algorithm: O(n)

    - When should users receive precautions:
        - Cloud computing can be unpredictable when it comes to latency, even in this situation where a low latency cloud server is in use. Therefore, the best course of action is to send multiple precaution signals (range to trigger signals 500m, 250m, and 50m.)
        - Latency for REST API: 50ms in average case. While using REST API means using TCP/IP protocol, which is slower compared to UDP, however, packet loss will not happen in this situation.

- *GPS + RSU as Wireless Access Point (LAN approach)*:
    - Similar to the GPS + Cloud Server. However, instead of using REST API, P2P communication here will be text file based (UDP/IP protocol, latency is almost negligible).
    - Raspberry Pi has a 1.7MHz microprocessor, so it can compute roughly about 1700 million instructions per seconds. Basic UDP P2P file transfer protocol used in this program is about 2000 instructions. With 100 users connected to RSU, latency can be as low as 0.1 milliseconds.

Additionally, sample calculations with assumed speed of vehicles and typical lengths of crosswalks have been performed for each of the use cases in the following manner:

VRU Crossing:

Worst Case Scenario:

- Alert signal doesn't get sent out in time and vehicle collides

Best Case Scenario:

- Alert gets sent out in time and collision is avoided

Example:

Vehicle Speed: 26.4 m/s (fastest speed)
Pedestrian Speed: 1.2 m/s
Length of Crosswalk: 9 m
Length of road: 100 m
Distance Between Vehicle and Passenger: 70 m

Time for Pedestrian to cross the road: $\frac{9\,m}{1.2\,m/s} = 7.5\,seconds$

Time for Vehicle to reach Pedestrian(Estimated time for a collision to occur):
$\frac{70\,m}{26.4\,m/s} = 2.65\,seconds$

Driver's reaction time: Studies have shown that it takes the average driver from one-half to three-quarters of a second to perceive a need to hit the brakes, and *another* three-quarters of a second to move your foot from the gas to the brake pedal. Hence the average stopping time for a driver is approximately $2.3\,s$.

Since 2.3 < 2.65, *2.3 seconds* is our margin for error.

The alert signal must be sent within this time frame. Also must consider the time it would take the vehicle to come to a complete stop.

Vehicle approaching high populated area:

Worst Case Scenario:

- Alert signal doesn't get sent out in time and vehicle reaches high populated area

Best Case Scenario:

- Alert gets sent out in time and vehicle stops before reaching high populated area

Example:

Vehicle Speed: 26.4 m/s (fastest speed)
Length of road: 200 m
Distance Between Vehicle and High Populated Area: 90 m

Time for Vehicle to reach Pedestrian(Estimated time for a collision to occur):

$$\frac{90\ m}{26.4\ m/s} = 3.41\ seconds$$

Driver's reaction time: Studies have shown that it takes the average driver from one-half to three-quarters of a second to perceive a need to hit the brakes, and *another* three-quarters of a second to move your foot from the gas to the brake pedal. Hence the average stopping time for a driver is approximately $2.3\ s$.

Since 2.3 <3.41, *2.3 seconds* is our margin for error.

The alert signal must be sent within this time frame. Also must consider the time it would take the vehicle to come to a complete stop.

Vehicle approaching without the view of VRU:

Worst Case Scenario:

- Alert signal doesn't get sent out in time and vehicle collides

Best Case Scenario:

- Alert gets sent out in time and vehicle stops before collision

Example:

Vehicle Speed: 26.4 m/s (fastest speed)
Length of road: 200 m
Distance Between Vehicle and VRU blindside: 90 m

Time for Vehicle to reach Pedestrian(Estimated time for a collision to occur):

$$\frac{90\ m}{26.4\ m/s} = 3.41\ seconds$$

Driver's reaction time: Studies have shown that it takes the average driver from one-half to three-quarters of a second to perceive a need to hit the brakes, and *another* three-quarters of a second to move your foot from the gas to the brake pedal. Hence the average stopping time for a driver is approximately $2.3\ s$.

Since 2.3 <3.41, *2.3 seconds* is our margin for error.

The alert signal must be sent within this time frame. Also must consider the time it would take the vehicle to come to a complete stop.

Preliminary Design

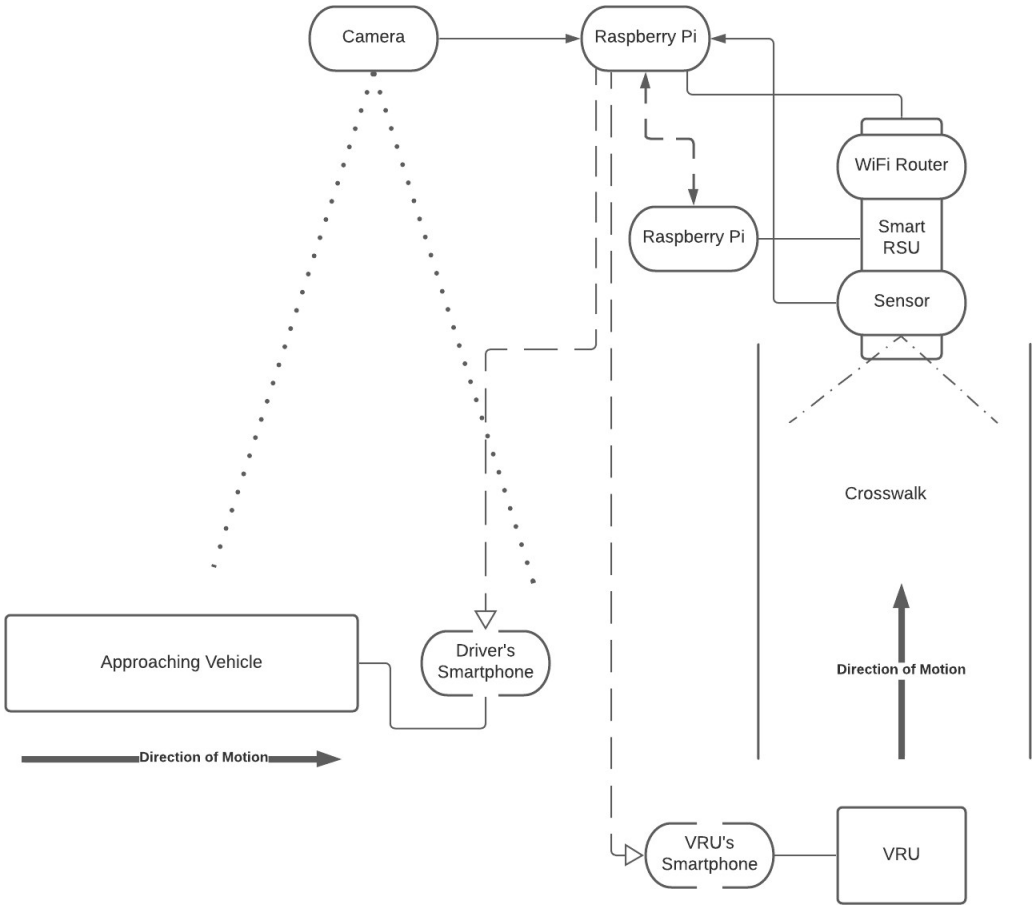


Figure 3: Block Diagram of the Design

The block diagram above portrays the essential components of the design. The way in which each component interacts with each other is listed below:

1) **Raspberry Pi**: The Raspberry Pi devices are our primary platform for converting the RSU into a smart RSU. Each of the Raspberry Pi works independently. The purpose of the first Raspberry Pi is to carry out the functionalities of the traffic signal control by taking data inputs from the camera and the sensor, while the other one mediates a peer-to-peer communication and performs all relevant calculations for collision avoidance by utilizing GPS signals.

2) **Camera**: The Camera is mounted on the Raspberry Pi for a specific type of data acquisition. It acts as an input device that always remains active and can detect a moving vehicle instantaneously. The following model of camera is to be used for the purpose of this project:

Raspberry Pi Camera, KEYESTUDIO Fisheye Wide Angle Lens 5MP 1080p OV5647 Sensor Module for Raspberry Pi
https://www.amazon.ca/KeyestudioCameraFishEyeWideAngleRaspberry/dp/B076MPL9P1/ref=asc_df_B076MPL9P1/?tag=googleshopc0c20&linkCode=df0&hvadid=292991886665&hvpos=&hvnetw=g&hvrand=2712712422753855864&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9000991&hvtargid=pla-521018536779&th=1

3) **Sensor**: The sensor is the other primary input device alongside the camera for data acquisition. It is mounted on the RSU and connected to the Raspberry Pi, triggering whenever a VRU is using the crosswalk. The following model of sensor is to be used for the implementation of this design:

Sainsmart HC-SR04 Ranging Detector Mod Distance Sensor –

https://www.amazon.ca/SainsmartHCSR04RangingDetectorDistance/dp/B004U8TOE6/ref=sr_1_5?gclid=Cj0KCQiAqGNBhD3ARIsAO_o7ykwJy5HBZ1UEH2JWTnZYHejOr4f_mzJXXjW24nXI3ss_XqQVh6FLEaAlPKEALw_wcB&hvadid=508153319919&hvdev=c&hvlocphy=9000991&hvnetw=g&hvqmt=b&hvrand=16306861266876354838&hvtargid=kwd59062832485&hydadcr=3821_9969054&keywords=hcsro4&qid=1638517076&sr=8-5

4) **Smart RSU**: The RSU is the roadside infrastructure that integrates all the devices mentioned above. Therefore, its function is to carry out all the functionalities performed by the devices mounted on it. The two Raspberry Pi are the hardware that make it a smart RSU with a functional Operating system. In short, the main features of the smart RSU are:

- Sending out a warning signal to the vehicle drivers and VRU whenever both the camera detects a moving vehicle, and the sensor detects a VRU crossing simultaneously.
- Mediate a peer-to-peer communication between all the devices mounted on the roadside infrastructure.
- Perform necessary calculations using GPS signals for ensuring collision avoidance

5) **VRU**: The VRU is our targeted vulnerable road user which mainly focuses on pedestrians on the crosswalk adjacent to the RSU. The objective of this design is to send out a warning signal to the VRU (alongside the driver) to minimize the chances of a collision.

6) **Approaching Vehicle**: Like the VRU, drivers of the vehicle must also receive appropriate signals on their smartphones to react appropriately and avoid a collision.

Final Design

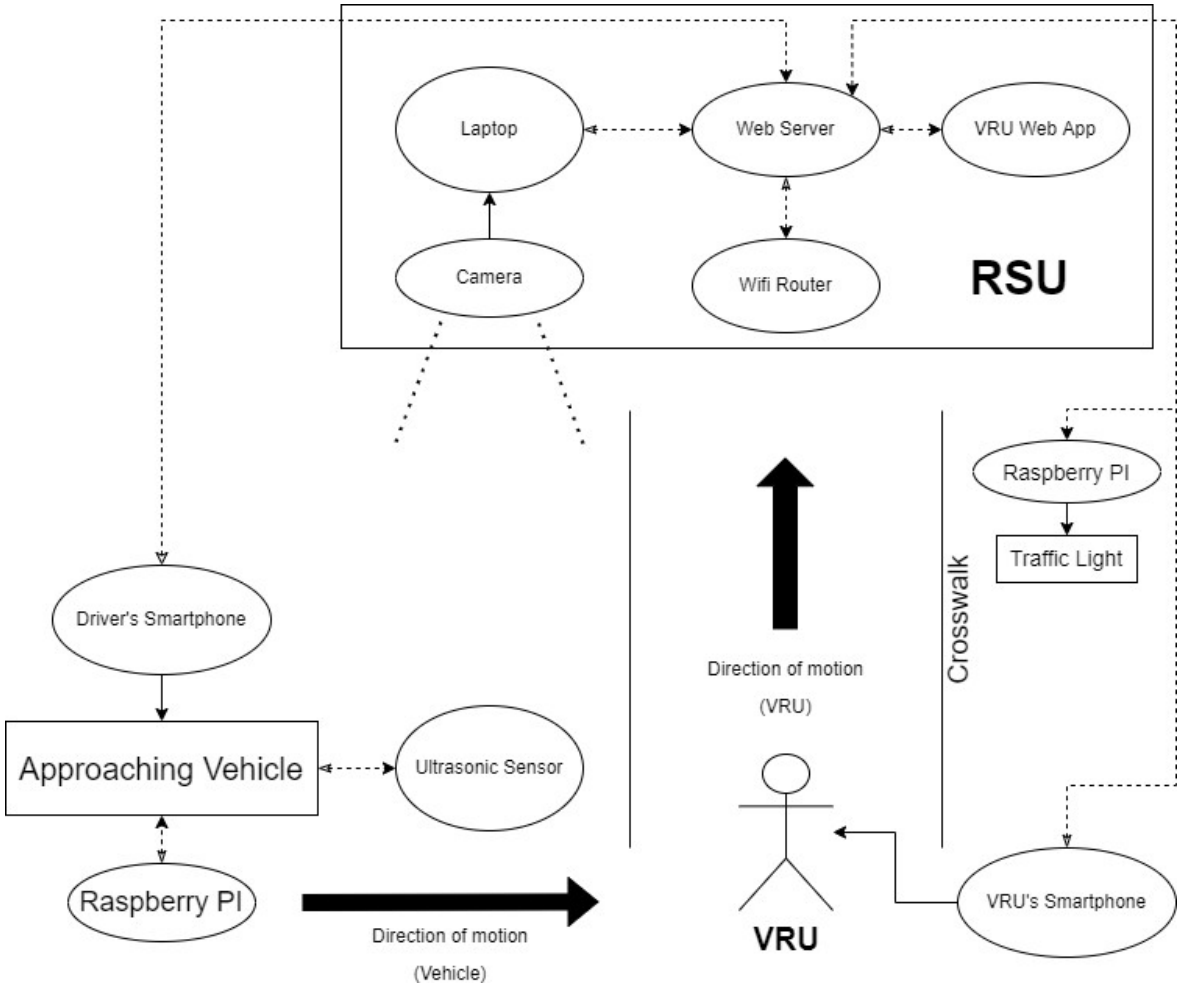


Figure 4: Block Diagram of the Final Design

Functions of Components

Raspberry Pi: Used to control the remote car and interface with the ultrasonic sensor, connect to the VRU server and get/send data.

Ultrasonic Sensor: Used to detect motion in the sensor path. Sends motion data to the Raspberry Pi that is connected to the remote car.

Laptop + Camera: Used for creating the object detection system which will recognize license plate numbers of vehicles on the road

Wi-Fi Router: Responsible for mediating communication among the components.

Web Server: Used to store and process data and send it to VRU's smartphones. Used to host the VRU Web App.

VRU Web App: Used to display data received by the Web Server, such as the location of all the VRUs and vehicles. The Web Server and the VRU Web App are tightly connected, one cannot function without the other.

VRU/Vehicle Smartphone App: Used to register a user with their name, license number, and vehicle type. Also used to communicate with the web app to send data of users and send out collision warnings.

The Flow Diagram

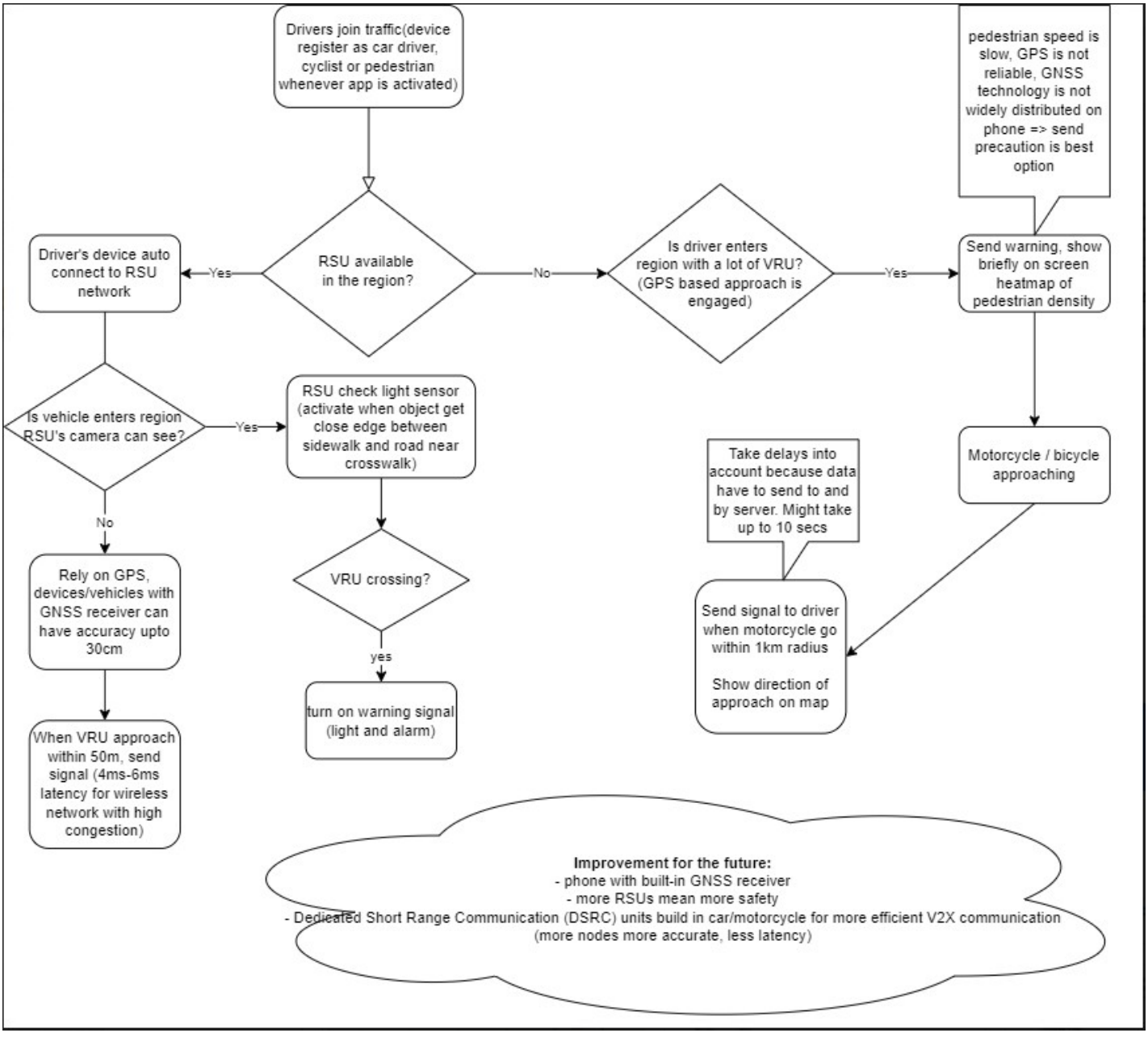


Figure 5: Flow diagram of the system

Software Design

The VRU project is highly dependent on software design as the primary goal of this project is to notify the users of any possible accidents that they may encounter near any roads and highways. The best way to do this is through a Smartphone Application. It will be installed on the users' Smartphone and will notify them of any possible warnings/dangers. The software design is split into two parts. The first component is the Web Application, this will be the portal to the VRU Project's status and will allow authorized users to see and analyze data collected via the VRU Network. The Web Application will run on a VRU Server, this server will be responsible for collecting data from various VRU network nodes and process it and store it in a database.

Web Server

The Web Server's purpose will be to store and retrieve data relating to the user's location, speed, and possible collision with another user. Data will be transferred in JSON string format. Rest APIs will be used to transfer data using GET, POST methods.

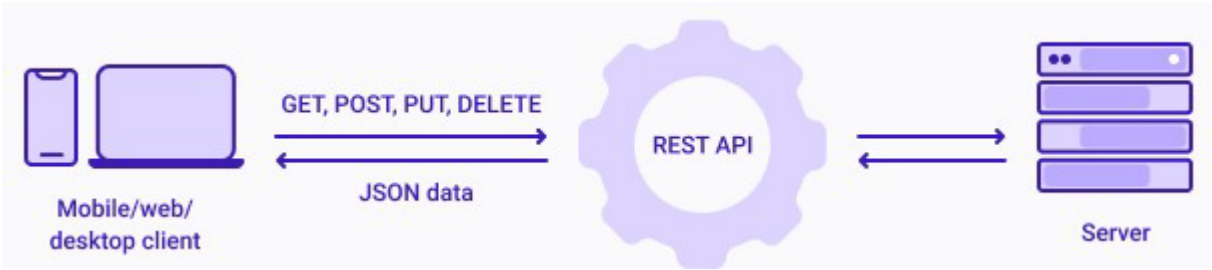


Figure 6: REST APIs will be used to send and receive data. The data will be in JSON string format.

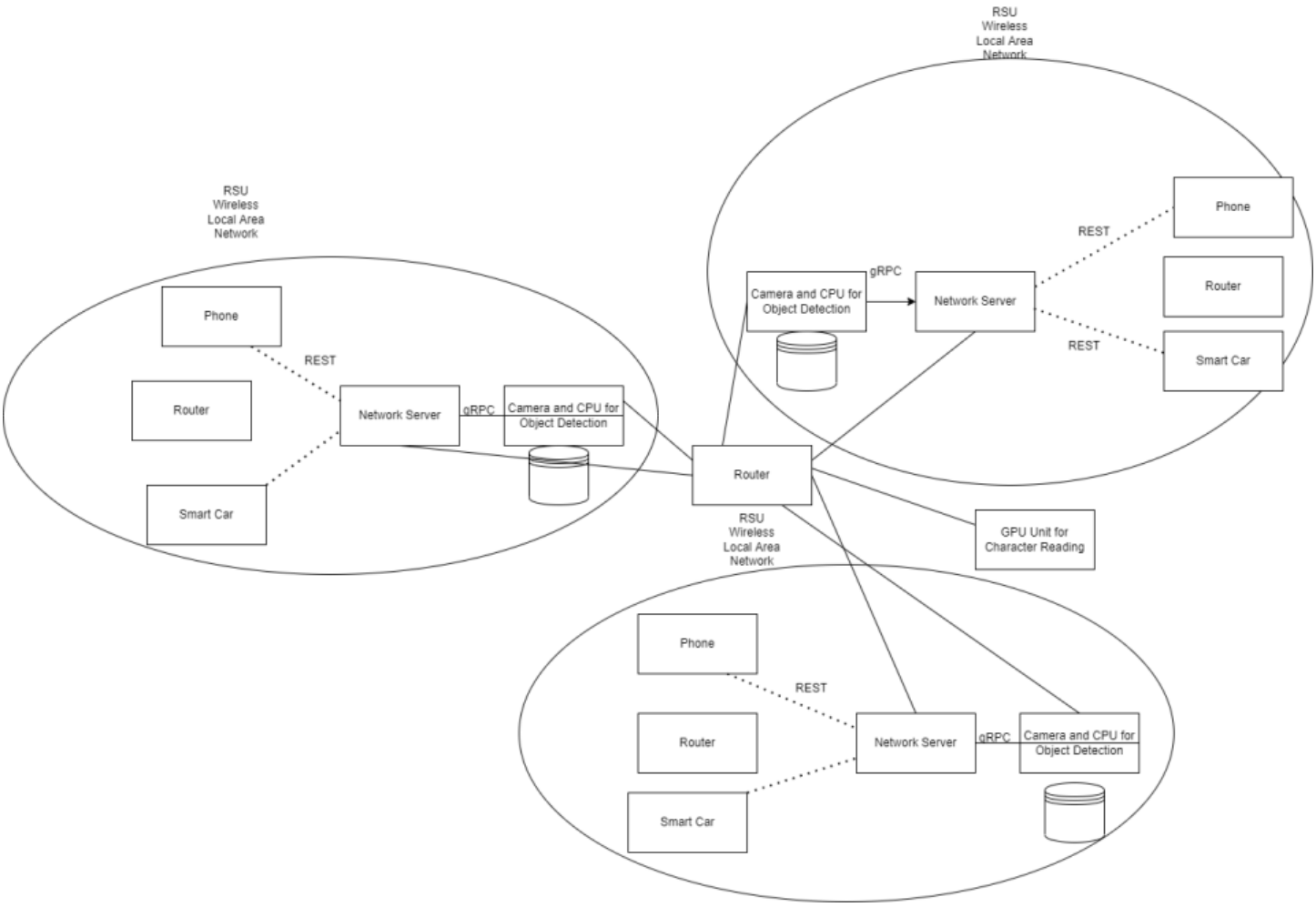


Figure 7: The RSU Network Diagram.

Web Application

The web application is responsible for displaying all the data collected by the Server. All the GPS coordinates of user vehicles will be displayed on a large interactive map. The map will update in real-time with minimal lag. There will also be a panel showing possible collisions and other useful information. If time allows, an additional area will be added that will display data on various charts. Like possible collisions on the y-axis and the speed of the car on the x-axis.

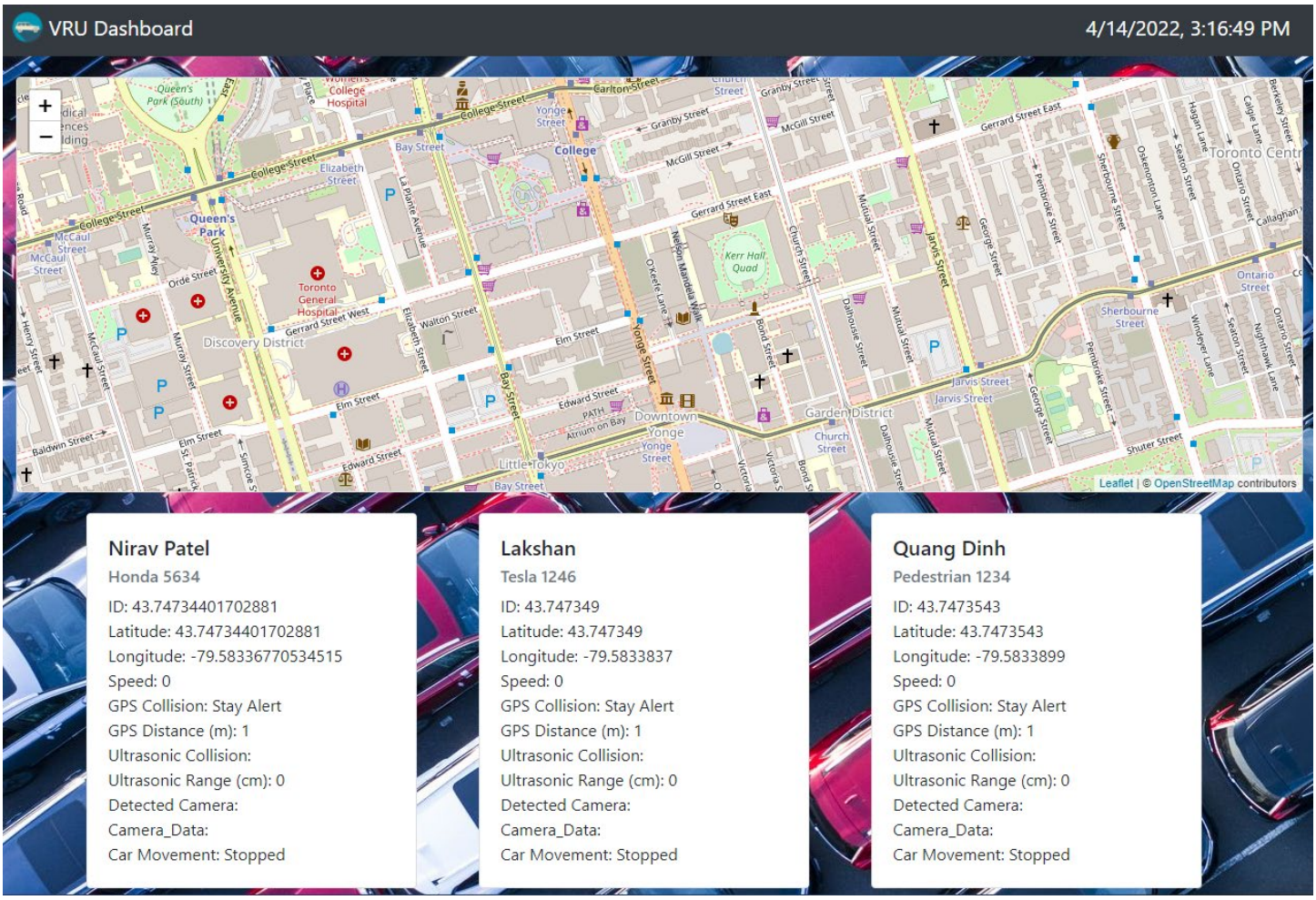


Figure 8: The Web Application. Displays User Information as well as their location on a real-time map. The map is updated every second and is interactive. Users can move the map around and click on a user icon to show information about them.

Smartphone Application

The smartphone application is very important to the VRU project. Most users will receive collision alerts through their smartphones. The application will show the location and location of other nearby vehicles on a map. The App will receive data from the Web Server.

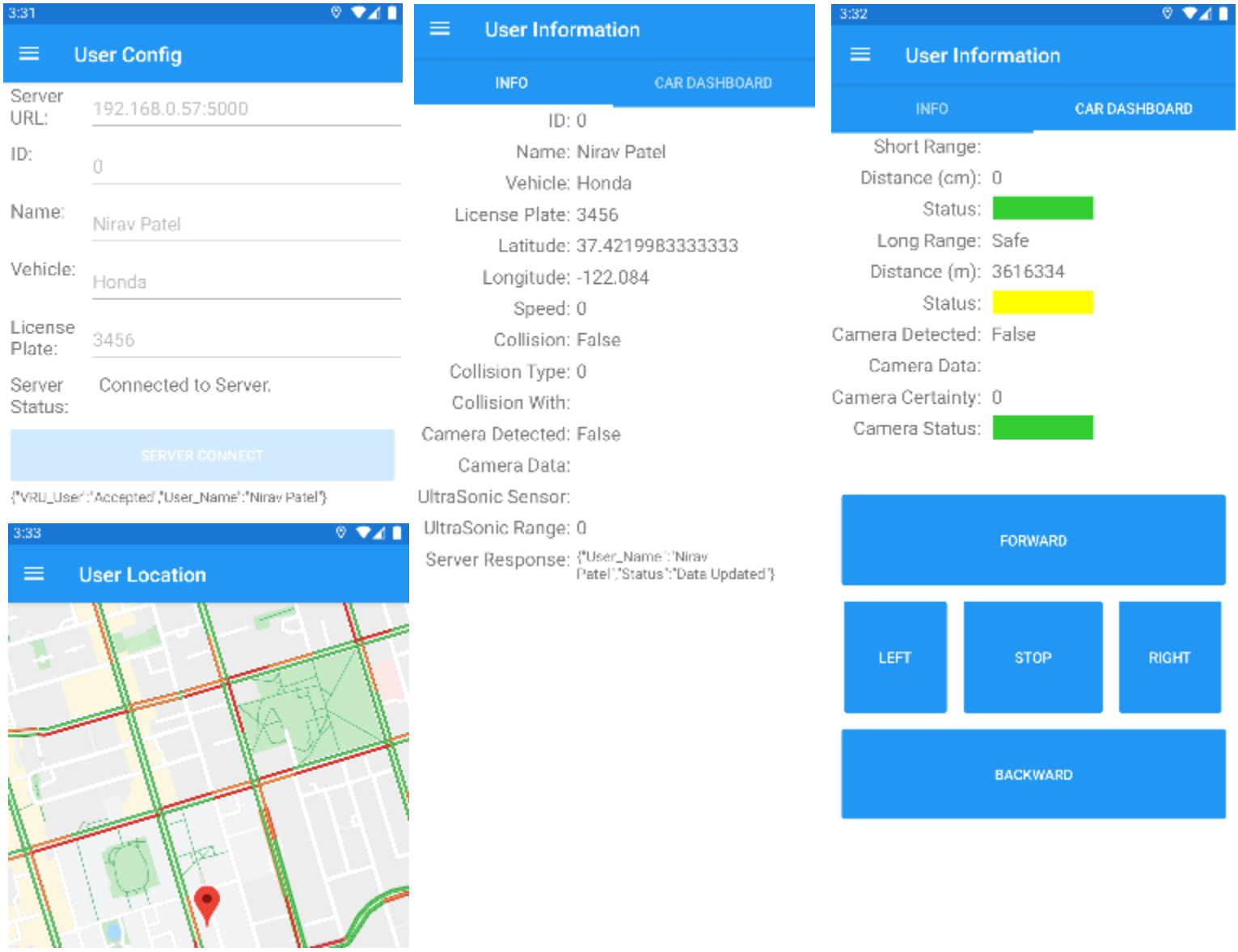
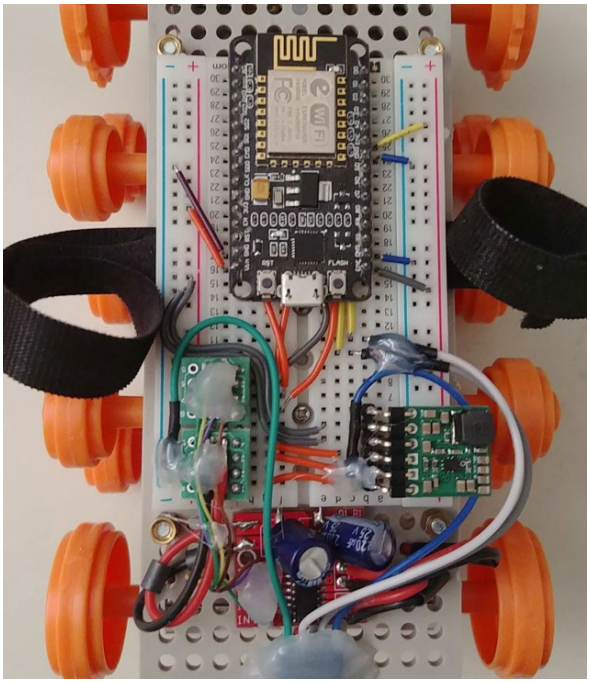


Figure 9: The Smartphone Application. Allows Users to receive collision warnings and sends GPS coordinates to the Web Server. Also allows Vehicle Users to control a model car.

Hardware Design

The Model Cars allowed us to demonstrate the three Use Cases of this project. The car's design initially used a raspberry PI however due to its power consumption, the ESP8266 Microcontroller was chosen instead. The ESP8266 connects to the Web Server and retrieves the Car Movement data and sends Ultrasonic Sensor readings back. The car is build using a DC Motor Driver circuit, a Gearbox, a 2x Li-Ion battery pack, Ultrasonic sensor, a microcontroller, and some passive components.



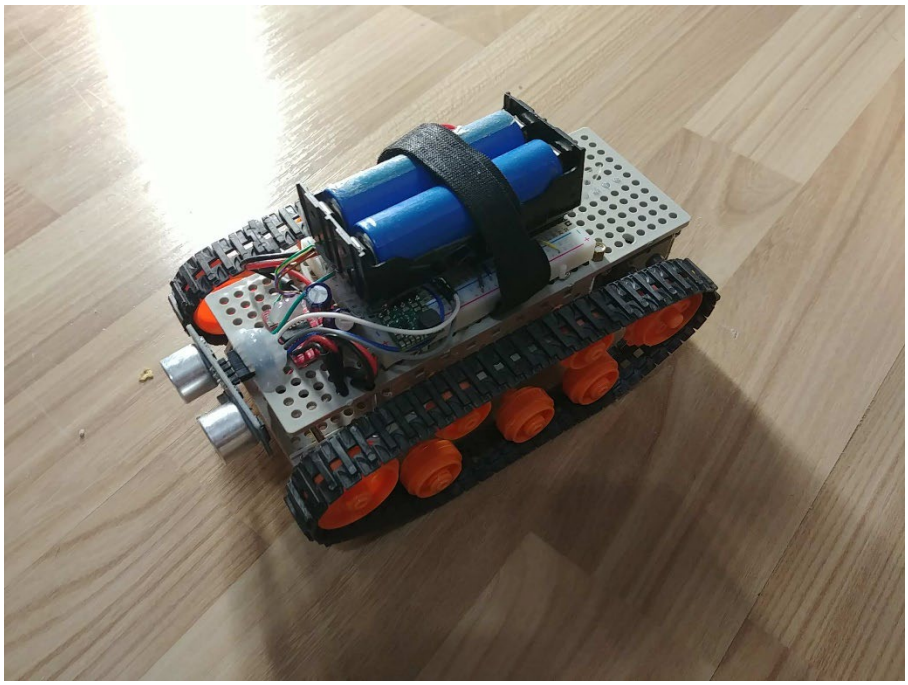


Figure 10: The Model Car, used to demonstrate the three Use Cases.

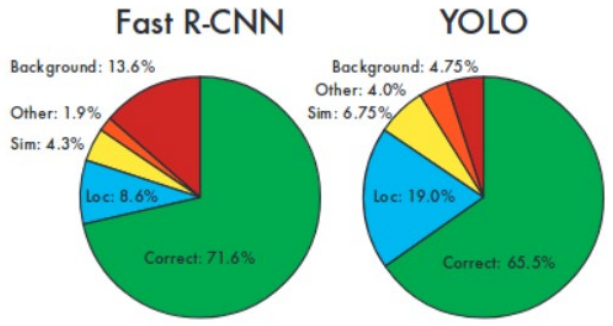
Camera: Object Detection System Optimization

The object detection system for this project has undergone several changes throughout the different phases of this project to reach its current state. Numerous libraries deep-learning libraries were utilized and tested. Some of the essentials that were used throughout all the phases are:

• TensorFlow: TensorFlow is Google's open-source AI framework for machine learning and high-performance numerical computation. TensorFlow is a Python library that invokes C++ to construct and execute dataflow graphs. It supports many classification and regression algorithms, and more generally, deep learning and neural networks.

• OpenCV: OpenCV is a computer vision framework that helped us do all sorts of processing on images and videos, alongside other python libraries.

During the earlier phases of the implementation the open-source algorithm YOLO (You Only Look Once) was used. You Only Look once is a modern fast framework. YOLOv3 was used in the implementation of this project. While it was extremely fast, unlike other Fast R-CNN, YOLO also has its limitations:



Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Figure 11: YOLO vs Fast R-CNN.

From the pie charts above, it can clearly be observed YOLO falls behind in terms of errors.

During the second phase of implementation, the object detection system was further enhanced to detect License Plate Numbers. For this implementation easyOCR was used. EasyOCR optical character recognition library reads short texts (such as serial numbers, part numbers and dates).

It uses font files (pre-defined OCR-A, OCR-B and Semi standard fonts, or other learned fonts) with a template matching algorithm that can recognize even badly printed, broken or connected characters of any size. The issue with this implementation however was latency. Although the library can utilize the GPU, due to lack of funding this project did not have a proper GPU. The only GPU that was available to us was Nvidia GTX 960m. Despite the GPU being decent, the drivers for it are outdated. The CUDA versions used by the license plate detection application, CUDA 11.3 is incompatible, making it impossible for us to harness the GPU.

During Phase 3 alternatives to easyOCR were implemented to solve the latency problem. During this phase a different approach was utilization of Tesseract. Unlike easyOCR which is a lightweight model, Tesseract performs well for high resolution images. However, the latency issue remained unsolved since Tesseract does not have GPU support either and only uses the CPU. After some more trial and errors with other frameworks like Google vision, it was concluded that the License Detection plate system's latency issue was not fixable due to hardware limitations, hence the focus of the object detection was solely shifted towards VRU detection.

During Phase 4, the object detection system that was developed during the first phase, created with YOLO, OpenCV and TensorFlow was further optimized by removing all unnecessarily trained models since its prime objective is to detect VRU like pedestrians. Training our own model made the objection system much faster and more solved the latency issue, making it a fully viable for this project.

Conclusion

Road safety is a big concern today. Many accidents occur and some of these accidents result in fatalities which is very unfortunate. Many vulnerable road users should feel safe when crossing the street or walking on the street. Our project helps give VRUs more safety on the road to prevent these unforeseen accidents. Our project if deployed on a bigger scale will help lower road accidents by a large margin. As displayed on a smaller scale, the vehicles and the passengers receive a collision warning on their smartphones which alerted them when they are in danger of a collision to help prevent the collision. I had an object detection system for VRUs to know when they are in danger of a collision. I used ultrasonic sensors for vehicles to know when they are in danger of a collision. Finally, I used the distance between the smartphone users as another precaution to know whether the user of that smartphone is in danger of a collision. These three precautions were in place to prevent a collision. All in all, the project was successful, and I believe that with the right equipment (RSU, smartphones, cameras) VRUs will be more protected and road safety would be better with our intelligent transportation system.

References

1. Bhattacharya, J. (2013) “Intelligent Devices: The Open-Source Hardware Movement Lets Anyone Build Intelligent Electronic Devices.” Economic and Political Weekly 48, no. 44: 81–82. http://www.jstor.org/stable/23528817.

2. Gibson, Peter. (2021) “Internet of Things Sensing Infrastructures and Urban Big Data Analytics in Smart Sustainable City Governance and Management.” Geopolitics, History, and International Relations, vol. 13, no. 1, Addleton Academic Publishers, pp. 42–52, https://www.jstor.org/stable/27031366.

3. Google Maps Platform Documentation |. (n.d.). Google Developers. Retrieved December 9, 2021, from https://developers.google.com/maps/documentation

4. Guo, J., & Baugh, J. P. (2006). Security and Privacy in Vehicle Safety Communication Applications. SAE Transactions, 115, 721–727. http://www.jstor.org/stable/44700103

5. McLellan, Charles. , (2019) “*What Is V2X Communication? Creating Connectivity for the Autonomous Car Era”*, ZDNet: https://www.zdnet.com/article/what-is-v2x-communication- creating-connectivity-for-the-autonomous-car-era/

6. Prinz, Anna C. B., Vikas K. Taank, Vincent Voegeli, and Eric L. Walters. (2016) “A Novel Nest- Monitoring Camera System Using a Raspberry Pi Micro-Computer.” Journal of Field Ornithology 87, no. 4: 427–35. http://www.jstor.org/stable/44994020.

7. React Native • Learn once, write anywhere. (n.d.). React Documents. Retrieved December 9, 2021, from https://reactnative.dev/

8. SQL Database – Managed Cloud Database Service. (n.d.). Microsoft Azure. Retrieved December 9, 2021, from https://azure.microsoft.com/en-ca/products/azure-sql/database/

9. Strycker, Jesse. (2015) “The Raspberry Pi: Not a Poor Man’s Computer, but an Interesting Possibility.” Educational Technology 55, no. 4: 30–34. http://www.jstor.org/stable/44430387

10. Thomas L. (2020) “Vulnerable Road User Protection”, 5GAA Automotive Association: https://5gaa.org/wp-content/uploads/2020/08/5GAA_XW3200034_White_Paper_Vulnerable-Road-User-Protection.pdf

11. Vincent, John C. (1992) “Smart Systems for Smart Roads.” The Military Engineer 84, no. 553: 43–45. http://www.jstor.org/stable/44552405

12. VSC-Service-Account. (n.d.). Xamarin.Forms - Working with Maps - Code Samples. Microsoft Docs. Retrieved December 9, 2021, from https://docs.microsoft.com/en-us/samples/xamarin/xamarin-forms-samples/workingwithmaps/

13. Wang, Pangwei, Juan Zhang, Hui Deng, and Mingfang Zhang. (2020) “Real-Time Urban Regional Route Planning Model for Connected Vehicles Based on V2X Communication.” Journal of Transport and Land Use 13, no. 1: 517–38. https://www.jstor.org/stable/26967257

Appendices

```
let VRU_Users = [
    {
        ID: 0,
        isConnected: false,
        Name: '',
        Vehicle:'',
        License_Plate: 0,
        Latitude: 0,
        Longitude: 0,
        Speed: 0,
        isCollision: false,
        Collision_Type: 0,
        Collision_With: '',
        isCamera_Detected: false,
        Camera_Certainty: 0,
        Camera_Data: '',
        UltraSonic_Data: '',
        UltraSonic_Range: 0,
        GPS_Data: '',
        GPS_Range: 0,
        Car_Movement: 0,
        Auto_Stop: false
    },
    {
        ID: 1,
        isConnected: false,
        Name: '',
        Vehicle:'',
        License_Plate: 0,
        Latitude: 0,
        Longitude: 0,
        Speed: 0,
        isCollision: false,
        Collision_Type: 0,
        Collision_With: '',
        isCamera_Detected: false,
        Camera_Certainty: 0,
        Camera_Data: '',
        UltraSonic_Data: '',
        UltraSonic_Range: 0,
        GPS_Data: '',
        GPS_Range: 0,
        Car_Movement: 0,
        Auto_Stop: false
    },
    {
        ID: 2,
        isConnected: false,
        Name: '',
        Vehicle:'',
        License_Plate: 0,
        Latitude: 0,
        Longitude: 0,
        Speed: 0,
        isCollision: false,
        Collision_Type: 0,
        Collision_With: '',
        isCamera_Detected: false,
        Camera_Certainty: 0,
        Camera_Data: '',
        UltraSonic_Data: '',
        UltraSonic_Range: 0,
        GPS_Data: '',
        GPS_Range: 0,
        Car_Movement: 0,
        Auto_Stop: false
    }
```

Figure 12: Web Server User Data Structure

```
// HTTP Connection
app.listen(3443, () => console.log(`Listening on port 3443`));

// HTTPS Connection
const sslserver = https.createServer({
    key: fs.readFileSync(path.join(__dirname, 'certificates', 'key.pem')),
    cert: fs.readFileSync(path.join(__dirname, 'certificates', 'cert.pem'))
}, app);

sslserver.listen(port, () => console.log(`Listening on port ${port}`))
```

Figure 13: Web Server HTTP and HTTPS Connections

```
1 reference
private void Check_Server_Connected(object sender, System.Timers.ElapsedEventArgs e)
{
    if (Global_Variables.isConnected)
    {
        Server_Communication.Enabled = true;
        Check_Server_Connected_Timer.Stop();
        Check_Server_Connected_Timer.Dispose();
        Device.BeginInvokeOnMainThread(async () =>
        {
            try
            {
                var response = await Server_Connect.GetAsync("https://" + Global_Variables.Server_IP + "/VRU_User_GET/" + Global_Variables.ID);
                if (response.IsSuccessStatusCode)
                {
                    var content = await response.Content.ReadAsStringAsync();
                    GET_Full_Data_JSON Full_Data = JsonConvert.DeserializeObject<GET_Full_Data_JSON>(content);

                    Global_Variables.Name = Full_Data.Name;
                    Global_Variables.Vehicle = Full_Data.Vehicle;
                    Global_Variables.License_Plate = Full_Data.License_Plate;

                    Info_VM.Name = Full_Data.Name;
                    Info_VM.Vehicle = Full_Data.Vehicle;
                    Info_VM.License_Plate = Full_Data.License_Plate;
                }
            }
            catch (Exception)
            {

            }
        });
    }
}
```

Figure 14: Smartphone App uses HTTPS Get Request to get data from Web Server.

```
const char* http_Post_Get_Request_serverName = "http://192.168.0.57:3443/VRU_User_POST/Update_UltraSonic/1";
void loop() {
  Update_UltraSonic_Sensor_Reading();
  //Check WiFi connection status
  http_Post_Get_Request(http_Post_Get_Request_serverName);
}

void http_Post_Get_Request(const char* serverName) {
  http.begin(client, serverName);
  http.addHeader("Content-Type", "application/json");

  JSONVar UltraSonic_Range_Object;
  UltraSonic_Range_Object["UltraSonic_Range"] = UltraSonic_Sensor_distance_CM;
  String JSON_Data = JSON.stringify(UltraSonic_Range_Object);
  int httpResponseCode = http.POST(JSON_Data);

  if (httpResponseCode > 0) {
    String payload = http.getString();
    Parse_Car_Movement_JSON(payload);
  }
  http.end();
}
```

Figure 15: Model Car connects to the Web Server to get and share data.

```
Server_IP = "192.168.0.57:3443"
Server_Begin_URL = "http://"
Server_End_URL = "/VRU_User_POST/Update_Camera/"

User_ID = 0

if classes[idx] == 'person':
  Camera_JSON_Data = {'isCamera_Detected': 'true', 'Camera_Certainty': round(score * 100, 2)}
  try:
    x = requests.post((Server_Begin_URL + Server_IP + Server_End_URL + str(User_ID)), data = Camera_JSON_Data)
  except:
    print("An exception occurred")
```

Figure 16: Object Detection Script connects to Web Server and Sends Data when a person is detected.