Automated Enrichment of
Global World View Information
based on Car2X

Master Thesis

for
the fulfillment of the academic degree
M.Sc. in Automotive Software Engineering

Faculty of Computer Science
Department of Computer Engineering

Submitted by: Thanaset Phothithiraphong, date of birth 09.07.1986

Supervisor: Prof. Dr. W. Hardt
Aquib Rashid, M.Sc.
Abstract

The purpose of this thesis is to develop the architecture to use the Car2X for observation the local traffic sign and displays it on the OpenStreetMap to provide more information of the road side to the driver. The proposed architecture of this thesis is to convert the traffic sign into the barcode and to be scanned by the barcode scanner and then wirelessly transfers the data to the web server to store the data and displays the traffic sign on the OpenStreetMap in the web browser. It uses two Raspberry Pi boards with CAN-Bus shields for transmitting the data on the CAN-Bus system in the car, a barcode scanner to scan the barcode, a GPS module to get its location, and a WiFi dongle to wirelessly send the data. The thesis also includes the camera to detect the traffic light using OpenCV and sends the GO or STOP command to the car. The results provide the OpenStreetMap with the traffic sign which helps the driver to realize the traffic sign on the road of the desired destination. However, the accuracy of GPS is not satisfied as well as the distance of the barcode scanning, therefore, this thesis suggests that includes the gps position in the barcode and uses the camera to detect the barcode for the improvement in the future.

Keywords – openstreetmap, car2x communication, wireless standards, manet, vanet, opencv, can-bus, traffic light detection, traffic sign recognition, raspberry pi, python
Acknowledgements

First of all, I would like to thank my thesis advisor Aquib Rashid who provided me with the required hardware and arranged me the laboratory for doing this thesis. I really appreciate all the things you have done to help me.

I would like to express my sincere gratitude to Prof. Dr. Wolfram Hardt who accepted me into this master course and thank you for your support, guidance and encouragement not only on this thesis but also on the entire of my study at Technische Universität Chemnitz.

I would also like to thank Owes Khan. Without his helps on the YellowCar, the testing part could not have been successfully conducted. And many thanks to all department staffs who organized and guided me on this thesis, especially to Dr. Ariane Heller and Daniel Reißner for providing the weekly consultation at the beginning. Thank you to all my friends for sharing all the knowledge, experiences and my life here in Chemnitz.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support, continuous encouragement and always allow me to take my own decisions throughout my entire life. This accomplishment would not have been possible without them. Thank you.

Author

Thanaset Phothithiraphong
# Table of Contents

1. Introduction ..................................................................................................................... 1
   1.1 Motivation .................................................................................................................. 1
   1.2 Problem Definition .................................................................................................... 2
   1.3 Structure of this Thesis ............................................................................................. 4

2. State of the Art .................................................................................................................. 5
   2.2 Related Work ............................................................................................................. 5
   2.2 Proposed Work ......................................................................................................... 6

3. Basic Background ............................................................................................................ 8
   3.1 Mapping Technology ............................................................................................... 8
   3.2 Car2X Communication ........................................................................................... 11
      3.2.1 MANET VS VANET Characteristics ................................................................. 12
      3.2.2 Wireless Standards in Automotive Communication .......................................... 13
      3.2.3 Short-Range Communication Technology ......................................................... 14
      3.2.4 Medium-Range Communication Technology ..................................................... 17
      3.2.5 Long-Range Communication Technology ......................................................... 19
      3.2.6 Discussion .......................................................................................................... 20
   3.3 CAN-Bus System ....................................................................................................... 22
   3.4 Computer Vision ...................................................................................................... 29
   3.5 Microcontroller ........................................................................................................ 34

4. Concept ............................................................................................................................ 37
   4.1 Architecture ............................................................................................................. 37
   4.2 Selected Hardware ..................................................................................................... 44
      4.2.1 Microcontroller ................................................................................................. 44
4.2.2 USB GPS .................................................................44
4.2.3 USB WiFi Dongle ..........................................................45
4.2.4 USB Barcode Scanner ......................................................45
4.2.5 Raspberry Pi Camera .......................................................45

5. Implementation ......................................................................48
5.1 Scanning Barcode ..................................................................48
5.2 Getting GPS Location ...........................................................50
5.3 Transmitting Data through CAN-Bus .......................................51
5.4 Receiving Data from CAN-Bus .................................................53
5.5 Sending Data to Web Server ..................................................55
5.6 Receiving Data on Web Server ...............................................56
5.7 Storing Data on the Database ...............................................57
5.8 Reading Data for Embedding the Map .....................................58
5.9 Enrichment the OpenStreetMap ............................................60
5.10 Detecting the Traffic Sign ..................................................62

6. Result ....................................................................................66
6.1 Scanning Traffic Sign Barcode and Enrichment OpenStreetMap ......................................................66
6.2 Detecting Traffic Light ..........................................................71

7. Conclusion .............................................................................73

Bibliography ..............................................................................75
List of Figures

1.1 Accidents in Germany 2008 statistics [3] ................................................................. 2
1.2 Intelligent Transport Systems at a glance [4] ............................................................ 3
2.1 Proposed Architecture ............................................................................................... 6
2.2 The traffic sign on the OpenStreetMap ...................................................................... 7
3.1 The overview of the OpenStreetMap [10] ................................................................. 9
3.2 Node [12] .................................................................................................................. 9
3.3 Open way [13].......................................................................................................... 9
3.4 Closed way [13] ....................................................................................................... 9
3.5 Area [13] ................................................................................................................ 9
3.6 Relation [14]............................................................................................................ 10
3.7 Tag [15]................................................................................................................... 10
3.8 The example of the OpenStreetMap with Traffic Sign ............................................. 10
3.9 The type of VNET [16]............................................................................................ 12
3.10 (a) Single-hop, (b) Multi-hop IVC Systems [17]..................................................... 13
3.11 The 802 wireless space [19].................................................................................... 14
3.12 IEEE 802.15.4 characteristics [20]........................................................................ 15
3.13 Comparison parameters of the short range technologies [22]................................. 16
3.14 Standards for DSRC [23]......................................................................................... 18
3.15 DSRC standards and communication stack [23]..................................................... 18
3.16 Comparison parameters of the medium range technologies [22]............................ 19
3.17 Comparison parameters of the long range technologies [22].................................. 20
3.18 Comparison Throughput WiFi and WiMax in Laboratory [24]................................. 21
3.19 Comparison Throughput of WiMax in vary frame duration [24]............................... 21
3.20 Comparison Throughput WiFi and WiMax [24]....................................................... 21
3.21 Comparison Delay WiFi and WiMax [24] ................................................................. 22
3.23 Low Speed Fault Tolerant CAN Network. ISO 11898-3 [25] ..................................... 23
3.25 CAN-Frame in base format with electrical levels without stuffbits [25] ................. 25
3.26 CAN error frame [26] ..................................................................................................... 26
3.27 CAN overload frame [26] ............................................................................................. 27
3.28 CAN-Frame before and after the addition of stuffbits (in purple) [25] ............... 28
3.29 ECU with and without CAN-Bus [27] ........................................................................ 29
3.30 To a computer, the car’s side mirror is just a grid of numbers [29] ................... 30
3.31 An example of an unmanned land-based vehicle [28] .............................................. 32
3.32 "Arduino Uno" SMD Revision 3 [33] ........................................................................ 34
3.33 Raspberry Pi 2 model B [34] ....................................................................................... 34
4.1 The overview and data flow of the architecture ......................................................... 37
4.2 The block diagram of the architecture ...................................................................... 38
4.3 Transforming the traffic sing to the barcode .............................................................. 38
4.4 The selected building path on the official OpenStreetMap block diagram [38] .... 39
4.5 The database for the traffic sign .................................................................................. 39
4.6 The database for the point of traffic sign ................................................................... 39
4.7 The query of the traffic sign from the database in the JSON data format ............. 40
4.8 The example of traffic signs on the OpenStreetMap ................................................. 41
4.9 Extracting the traffic light image ............................................................................... 41
4.10 The example of positive images .............................................................................. 41
4.11 The example for the training of OpenCV ................................................................. 42
4.12 The Raspberry Pi 2 B [34] .......................................................................................... 43
4.13 PICAN-Bus shield [39] .............................................................................................. 43
4.14 The GPS module [40] ............................................................................................... 43
4.15 The USB barcode scanner [41] .................................................................43
4.16 The USB WiFi dongle [42] .................................................................43
4.17 The Raspberry PI camera [43] .................................................................43
4.18 The overview of the hardware for this architecture ........................................44
4.19 The overview of block diagram for this architecture .........................................45
6.1 The barcode of the traffic signs that are used in this thesis ........................................65
6.2 The traffic sign information that is stored in the mySQL database ..............................66
6.3 The point of traffic signs that are stored in the mySQL database .................................67
6.4 The example of JSON Data of two traffic signs ..............................................68
6.5 The example of OpenStreetMap with traffic signs on the web browser .......................68
6.6 The PI camera is attached to the Raspberry Pi board ........................................69
6.7 The Raspberry Pi board is connected to the YellowCar .....................................70
List of Tables

3.1 Comparison of MANET and VANET [18] ................................................................. 13
3.2 Overview of short-range communications [21, 22] ................................................. 16
3.3 Overview of medium range communications [22] .................................................... 19
3.4 Overview of long range communications [22] ......................................................... 20
3.5 Standard data frame [25] ...................................................................................... 25
3.6 Shows the node priority works [25] ........................................................................ 28
3.7 The main differences between Arduino and Raspberry Pi [37] ................................. 35
6.1 The available testing traffic sign on the database in the thesis ................................. 64
Listings

3.1 The example of the Leaflet ................................................................. 11
4.1 The XML result from the training of OpenCV ...................................... 42
5.1 Importing required libraries for barcode scanner .................................. 46
5.2 ASCII codes represent for the barcode scanner ....................................... 47
5.3 Setting the input path of barcode scanner ............................................. 47
5.4 Reading the data from barcode scanner ................................................ 47
5.5 Importing a library and setting port for gps module ............................... 48
5.6 Reading the data from gps module ....................................................... 48
5.7 Importing required libraries, setting bus type and channel for the 1st CAN-Bus shield .. 49
5.8 Formatting the data for sending the CAN-Bus data frame ........................ 50
5.9 Verifying the data before sending the data through CAN-Bus .................... 50
5.10 Transmitting the data through the CAN-Bus shield ............................... 51
5.11 Importing required libraries, setting bus type and channel for the 2nd CAN-Bus shield ............................................................................................... 51
5.12 Setting the variable for receiving the CAN-Bus data frame ...................... 52
5.13 Receiving the CAN-Bus data frame ..................................................... 52
5.14 Importing required libraries for sending the data over URL ..................... 53
5.15 Sending the barcode and gps data over URL ......................................... 54
5.16 Receiving the barcode and gps data over HTTP GET method .................. 54
5.17 Setting connection to connect the mySQL database ............................... 55
5.18 Inserting the barcode and gps data to the mySQL database ..................... 56
5.19 Receiving the latitude and longitude for querying the database ............... 56
5.20 SQL syntax for querying the database ................................................. 57
5.21 Querying the traffic signs from the database ......................................... 57
5.22 Returning the data as the JSON data format ..........................................................58
5.23 Plotting the OpenStreetMap on the web browser ..................................................58
5.24 Requesting the plotting data from the HTTP GET method .....................................59
5.25 Showing the OpenStreetMap on the web browser ..................................................59
5.26 Importing the required libraries and setting the bus type and channel for the CAN-Bus for detecting the traffic light ..................................................................................60
5.27 Importing the required libraries for PI Camera ..........................................................61
5.28 Reading the video from the PI Camera ......................................................................61
5.29 Importing required libraries and setting path for the OpenCV ..................................61
5.30 Detecting the traffic light using the OpenCV own classifier ......................................62
5.31 Calculating the distance of the traffic light ...............................................................62
5.32 Sending the GO and STOP signals to the car .........................................................63
Chapter 1

Introduction

This chapter presents how the thesis is motivated, definition of the problem to be solved and an overview of the structure.

1.1 Motivation

During the past 25 years, when the first car (Mazda Eunos Comos) with built-in GPS (Global Positioning System) navigation system was introduced in 1990 by Mazda [1] although the first GPS Satellite (TRANSIT) was built in 1960 by NASA [2], it has been helping to navigate and guide car driver to safely reach at the desired destination. However, GPS alone cannot work without having map and it is needed to be displayed to the car driver. The more information it has, the more accurate and reliable information it can provide to the car driver. Some of these services are provided by Google Map, Apple Map and OpenStreetMap. The OpenStreetMap is free to use and can be embedded with more information on it using various layers, this comes to the idea to develop more automated approach to embed the map by observing the local data such as traffic signs to enrich them into global world view information such as on the OpenStreetMap which can be accessible everywhere around the globe.

Car2X communication system means the car communicates to X which is the network of the car, internet infrastructure or road side units including traffic lights, traffic signs to communicate to each other in order to exchange the data for better safely driving conditions. It sends the data wirelessly by using the technologies such as WiFi, 3G or WiMax. This can be illustrated by a scenario where the car driver informs about an accident by sending the signal to warn the car going in towards the accident opposite direction. The other driver with this knowledge drives more carefully and slowly or changes the route to avoid traffic jam. The other example is an ambulance which, when in an emergency situation needs to have a
clear way to reach the desired destination in time. The ambulance driver sends an emergency wireless signal to the cars so that they can give it higher priority and make way for it.

As an example of the Car2X communication system, we can adapt the car to observe the local data of the traffic signs to add more information on the map to help the other drivers know the traffic sign beforehand to prevent the accident or improve the driving experience by providing additional information. The idea is to use the car which is already driving on the road and is scanning the traffic signs. Initial realization was done by attaching a barcode scanner to a model car, with each barcode representing a specific traffic sign. After the data of the barcode is received, then it is sent wirelessly using the WiFi, to the web server and then the web server process this data to add or update the data on the database. Finally, the other drivers can use this data to get the local traffic signs that will display on the OpenStreetMap of the web browser.

1.2 Problem Definition

Car accidents can be reduced if the warning and the relevant information are provided to the drivers or even much lower if cars can communicate with each other. As shown in the Figure 1.1 below, the half of the accidents caused by traffic sign which including, “wrong use of lane”, “entry, exit, turning”, “right of way, traffic regulation”, “wrong distance”, “wrong speed” and “road conditions”, this can be prevented or reduced this kind of accidents by giving the information about traffic signs on the road or the condition of the road to the driver.

![Accidents (Germany 2008) with personal injuries and deaths caused by drivers and road conditions](image)

Figure 1.1: Accidents in Germany 2008 statistics [3]
However, the traffic sign or the road condition might not always be the same and can be changed at some time. The information that the driver has on hand before may not be the present data or out of date. The other information such as the road construction work or the accident might occur for not very long time. Some local traffic signs may not be on the physical map.

This leads to this thesis topic “Automated Enrichment of Global World View Information based on Car2X”, to automatically update the traffic sign on the road by using the barcode scanner to scan the traffic sign barcode and send the data to the server over WiFi. It helps to provide more information about the local traffic sign and updates it on the open source map which is the OpenStreetMap and to improve the better driving environment systems in the near future and become the intelligent transport system as shown in the Figure 1.2.

![Intelligent Transport Systems at a glance]

Figure 1.2: Intelligent Transport Systems at a glance [4]
1.3 Structure of this Thesis

This thesis is structured as follows;

Chapter 1 describes the motivation of this thesis and defines the problem and goal.

Chapter 2 reviews the related previous works about the detecting traffic sign on Car2X, the comparison of the traffic sign, the usage of the OpenStreetMap. Then it presents the proposed work on this thesis.

Chapter 3 gives the basic background of the knowledge and theory that are needed for this thesis. It explains the Mapping Technology, Car2X Communication, CAN-Bus System, Computer Vision and Microcontroller.

Chapter 4 presents the concept of architecture and how the selected hardware that are used in this thesis work.

Chapter 5 is the implementation detail that explains the how each step, scanning barcode, transmitting data through CAN-Bus and sending data to web server over WiFi work.

Chapter 6 shows the result of the implementation when scanning the barcode of traffic sign and then enriches the OpenStreetMap and the other one is detecting the traffic light.

Chapter 7 concludes the work of this thesis and proposes the future work to be developed.

In conclusion, this chapter provides:

- The Motivation of the thesis
- The problem that is needed to be solved
- The structure of the thesis

The next chapter gives the detail of state of the art of the thesis.
Chapter 2

State of the Art

This chapter reviews the related work about the traffic sign detection on Car2X, the comparison of the traffic sign, the reading of the additional text on the traffic sign and the usage of the OpenStreetMap for the autonomous robot. It also gives the problems of the previous works and then proposes the idea of this thesis.

2.1 Related Work

The traffic signing recognition on automotive field has been studied by many research papers in the past to improve the algorithm for the real-time processing on the detection. However, the only small sample of the traffic signs have been taken for the example in the research since there are many types of different signs in the reality. In order to detect it automatically, pre-processed images of each traffic signs have to be done such as using the OpenCV cascade classifier to train for detecting it, but this training process takes very long time to get the result. It shows in the paper, “Real Time Recognition System for Traffic Sign Detection and Classification” [5], “Real-Time Traffic Sign Detection and Recognition System Based on Friendly ARM Tiny4412 Board” [6] and “Traffic Sign Detection and Recognition Using OpenCV” [7], that the researchers have to select the sample of the traffic signs and do the pre-processed image before it can be detected this would be impossible for the reality in the real world since there are many different types of the traffic signs in each country. The study in “Traffic Sign Detection for U.S. Roads” [8] proves that less than 50% of speed limit signs of the European signs can be detected on the US signs and the authors also conclude that there are wide range of the traffic signs that are used in each state of the US and there is no system can be able to process at that moment.

Moreover, not only the regularly traffic sign is on the road side but there are also the additional text on the sign as well. This leads to the study of the “Additional Traffic Sign Detection” [9], the authors conclude that the high-resolution images give the detection quite
well, above 85% detection rate, however the lower-resolution or the low sign boundary are reached the limit of their approach. Further studying on the OpenStreetMap, the paper “Autonomous Robot Navigation Based on OpenStreetMap Geodata” uses geodata of the OpenStreetMap to navigate the robot to the desired destination. However, the authors have not inserted any data into the OpenStreetMap and conclude to be their future work.

As the studying on the related works, it is hard and almost impossible to collect all the traffic signs and prepare the pre-processed images for the automatically detection and there is still lack of the integration between the real-time detection of the traffic sign and the inserting the local data on map. It will be better to provide the other way of detecting the traffic signs and then add them on the map.

### 2.2 Proposed Work

This thesis proposes the idea from the previous works that the lack of the integration of the traffic signs and map. Therefore, integrating the information of the traffic sign and embedding it on map provides the driver to have more information about the road side which reduces the accidents and finally gets more safety driving environment system.

To implement the idea, the traffic sign should be converted to the barcode to make it easier to be detected. The barcode is a one-dimension and uses to store the traffic sign identification number which this number is linked to the online database of the traffic sign information. A barcode scanner is attached to the car and connected to the bus system of the car, and then it scans the barcode and gets the gps position of its location. After the car gets the data, it sends
the data to the online server to store the data on the database via the WiFi dongle. The OpenStreetMap queries the data about the traffic sign from the online database to display map on the web browser when the user needs information about the desired destination as the example shown on the Figure 2.2 below.

![Figure 2.2: The traffic sign on the OpenStreetMap](image)

In conclusion, this thesis provides the driver to have more information on the road side in the desired area in order to reduce the accidents and have the better and safer driving environment system by using the online OpenStreetMap that displays the traffic sign and its information. This chapter provides:

- The related works from the previous papers
  - The proposed work that is needed to develop

The next chapter gives the detail of the basic knowledge background for the proposed work.
Chapter 3

Basic Background

This chapter gives the basic knowledge background about the mapping technology, Car2X communication, CAN-Bus system, computer vision and the microcontroller that are used in this thesis.

3.1 Mapping Technology

The physical map has been widely used in the past to help us know the place on this planet. The detail about the place that is on the map mostly done by the publisher who printed map and information that is shown on the paper is very limited due to space on it. Moreover, the map might be printed very long time ago and some places maybe not existed on that place anymore.

The later generation of the map is the digital map as the image is shown on the screen. It is easy and widely to use around the world because it does not require to carry the physical map around. But the place and detail on the map cannot be changed easily due to the publisher who must be the one that put the data on the map. Some places and details are still limited due to space on the map even it has the more larger space on map, as well as the up to date of the current information on map.

The current generation of map is the online one that is displayed on the screen of the mobile phone everywhere around the world that can connect to the internet. The detail and the information can have many more details due to the link on the map that can be clicked to show more data. Most of the places and details are up to date because the online user can update their information such as the opening and closing of their shop. Some of them are free to use such as Google Map, Apple Map and OpenStreetMap but some of map need to pay for it such as TomTom Map and Garmin Map.
In this thesis, the OpenStreetMap is chosen to display the traffic sing of the road on the web browser because it is free and an open source as shown in the Figure 3.1 below.

**OpenStreetMap**

It is created by Steve Coast in the UK in 2004, it is a free editable map of the world and was inspired by the success of the Wikipedia which allow the users to add and edit the information on the map. OpenStreetMap represents the feature of the physical on the ground such as road, building which it uses the tags that are attached to the primitive data structure of the OpenStreetMap consists of four main core elements [11]:

- **Node**: It is consist of the geographic location including latitude, longitude and the node id.

- **Way**: It is the list of the nodes to form the way such as the “Open way”, “Closed way” and “Area”.

![Figure 3.1: The overview of the OpenStreetMap [10]](image)

![Figure 3.2: Node [12]](image)

![Figure 3.3: Open way [13]](image)  ![Figure 3.4: Closed way [13]](image)  ![Figure 3.5: Area [13]](image)
• **Relation**: It is the list of the nodes, ways and tags as the member.

![Relation](image)

Figure 3.6: Relation [14]

• **Tag**: It is the ‘key’ and ‘value’ which pairs together to describe the specific data of the element (nodes, ways, relations)

![Tag](image)

Figure 3.7: Tag [15]

The setting up of the OpenStreetMap consists of three main parts: the raw data of the OpenStreetMap, The Tile Server, and the Web browser as shown in the Figure 3.8. In this thesis, we choose to use the OpenStreetMap Tile Server which already has the raw data on it and ready to use, however, the data of the barcode and the location of the traffic sign are stored on dedicated online database which is the mySQL database.

![Map](image)

Figure 3.8: The example of the OpenStreetMap with Traffic Sign
Leaflet

It is free to use which is an open source JavaScript library for quickly embedding the map on the web browser. The traffic sign will be embedded as the Icon Markup on the Leaflet which is placed on the point of the latitude and longitude. Below is the example how to use the Leaflet to embed the traffic sign which can be modified the Icon to point to the URL of the image of the traffic sign as shown on line 69 of Listing 3.1. It also gets the latitude and longitude from the JSON data format and then adds them to the point of this location as shown in Line 73 of Listing 3.1. After that it adds this layer to display the embedded traffic sign on the OpenStreetMap.

In summary, the OpenStreetMap Tile Server has been used to display the embedded traffic sign which the data is provided on our online MySQL database by using the Leaflet JavaScript library.

```javascript
if (ajaxRequest.status==200) {
    plotlist=eval("\" + ajaxRequest.responseText + \"\")
    removeMarkers();
    for (i=0;i<plotlist.length;i++) {
        var TFSIcon = L.Icon.Default.extend({
            options: {
                iconUrl: 'tfs_images/*+plotlist[i].iconurl

            }
        })
        var tfsIcon = new TFSIcon();
        var plotl1 = new L.LatLng(plotlist[i].latitude,plotlist[i].longitude, true);
        var plotmark = new L.Marker(plotl1, {icon: tfsIcon});
        plotmark.data=plotlist[i];
        map.addLayer(plotmark);
        plotmark.bindPopup("<h3>"+plotlist[i].name+"</h3>"+plotlist[i].desc);
        plotlayers.push(plotmark);
    }
}
```
Listing 3.1: The example of the Leaflet

3.2 Car2X Communication

Statistic shows that the majority of the causes for the car accidents are related to the drivers and the road conditions, therefore, one of reducing and preventing techniques for those accidents have been studied by using the Car to X Communications to send the warning signal and giving many information as soon as possible to the driver decision to act in the automotive driving environments systems. Many researchers focus on the different
techniques using in vehicular communications up until now the IEEE 802.11p/WAVE has been assigned and specified in the wireless standard. Because of the natural behavior of the fast and mobility of the moving nodes (cars), the needs of the wireless communications are best selected way of communications. This topic shows the boards of wireless standards from short to long ranges such as Bluetooth, Ultra-WideBand, ZigBee, WiFi, DSRC, WAVE, WiMax and Cellular Technologies.

![Diagram of VNET](image)

Figure 3.9: The type of VNET [16]

In the Vehicular communication systems are divided into 3 main networks in the forms of vehicle to vehicle communication (V2V) and vehicle to the road-side-unit (RSU) or the infrastructure communication (V2I).

### 3.2.1 MANET vs VANET Characteristics

MANET or Mobile Ad Hoc Network is a self-configuring and no infrastructure network consists of mobile devices that are connected by the wireless links. Each device in the MANET freely moves between the networks which the topology of the networks is changed rapidly. However, each network may be self-operated or connected to the larger Internet.

VANET or Vehicular Ad Hoc Network is the sub set of MANET, Mobile Ad Hoc Network, which is the decentralized network topology and highly mobile nodes (vehicles). VANET consists of the road-side-unit that is connected into the infrastructures.
The communication between each node is using single-hop or multi-hop as shown in the Figure 3.10 and the comparison of the MANET and VANET is shown in Table 3.1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>MANET</th>
<th>VANET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost of production</td>
<td>Cheap</td>
<td>Expensive</td>
</tr>
<tr>
<td>Mobility</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Node density</td>
<td>Sparse</td>
<td>Dense and frequently variable</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>Hundred kbps</td>
<td>Thousand kbps</td>
</tr>
<tr>
<td>Range</td>
<td>Up to 100 m</td>
<td>Up to 500 m</td>
</tr>
<tr>
<td>Node lifetime</td>
<td>Depends on power resource</td>
<td>Depends on lifetime of vehicle</td>
</tr>
<tr>
<td>Multihop routing</td>
<td>Available</td>
<td>Weakly available</td>
</tr>
<tr>
<td>Reliability</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Moving pattern of nodes</td>
<td>Random</td>
<td>Regular</td>
</tr>
<tr>
<td>Addressing scheme</td>
<td>Attribute based</td>
<td>Location based</td>
</tr>
<tr>
<td>Position acquisition</td>
<td>Using ultrasonic</td>
<td>Using GPS, Radar</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of MANET and VANET [18]

### 3.2.2 Wireless Standards in Automotive Communication

In order to categorize the different wireless standards, the range is used to separate those standards into 3 main categories: short-range, medium-range and long-range communications and the overview of the IEEE 802 space is shown in the Figure 3.11.

#### A. Short-Range Communication

These standards range is from 1 meter up to 100 meters which consist of Bluetooth, Ultra-WideBand and ZigBee. Most of them are used for the infotainments in the automotive purpose such as remote control, mobile phone, laptop computer, video game console, voice application and multimedia application
B. Medium-Range Communication

The longer ranges are up to 1 km which is used in the inter-vehicle communication and to communicate between road-side-unit and infrastructures. WiFi, DSRC and WAVE are the wireless standards categorized into this range.

C. Long-Range Communication

Main purpose in this range is to connect the networks into the infrastructures to extend the longer range. Applications that are usually used in this range are internet access, e-mail, VoIP and high speed vehicle communication. Cellular technologies such as 3G or 4G and WiMax are in this category.

3.2.3 Short-Range Communication Technology

A. Bluetooth

Bluetooth is in the IEEE 802.15.1 standard using the 2.4 GHz frequency band or 2.5 GHz in version 1.2. Its specification allows the transmission rate is from 1Mbps to 12Mbps in version 2.0 and the transmission range is up to 100 meters which is still smaller than the need of
Inter-vehicle communication system. However, Bluetooth is the most used automotive wireless technology that many vehicles have it already set.

B. Ultra-Wide Band

Ultra-Wide Band is in the IEEE 802.15.3a standard using the frequency range is from 3.1 up to 10.6 GHz depends on the specifications. The transmission range is very small around 20 meters but this is the trade-off to have the high bandwidth from 50Mbps up to 100Mbps and because of its high data rate, the most common uses in the automotive domain are multimedia and healthcare applications. Thus, it is still not consider to be used in the Inter-vehicle communication system due to the fact that it has very low transmission range.

C. ZigBee

ZigBee is in the IEEE 802.15.4 standard and depends on the area to determine that the frequency will be used which is shown in the Figure 3.12. Although, it is a recently wireless technology built on PHY and MAC layers that is used in many application due to its open standard together with low cost and low power characteristics but it is not well suite in the Inter-vehicle communication system because of its very low date rate is around 250kbps. However, ZigBee is expected to be used in many automotive fields such as monitoring and control application, related to temperature and humidity measurement, ventilation, air-condition, heating and lighting control. [22]

<table>
<thead>
<tr>
<th>PHY (MHz)</th>
<th>Frequency Band (MHz)</th>
<th>Coverage</th>
<th>Data Rate (Kbps)</th>
<th>Number of Channels</th>
<th>RX Sensitivity</th>
<th>Modulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>868/915</td>
<td>868-868.6, 902-928</td>
<td>ISM Europe, ISM America</td>
<td>20, 40</td>
<td>1, 10</td>
<td>- 92 dBm, - 92 dBm</td>
<td>BPSK</td>
</tr>
<tr>
<td>868/915</td>
<td>868-868.6, 902-928</td>
<td>ISM Europe, ISM America</td>
<td>250, 250</td>
<td>1, 10</td>
<td>- 85 dBm, - 85 dBm</td>
<td>ASK</td>
</tr>
<tr>
<td>2450</td>
<td>2400-2483.5</td>
<td>ISM Worldwide</td>
<td>250</td>
<td>16</td>
<td>- 85 dBm</td>
<td>O-QPSK</td>
</tr>
</tbody>
</table>

Figure 3.12: IEEE 802.15.4 characteristics [20]
In summary, these standard range is from 1 meter up to 100 meters that are used for the infotainments in the automotive purpose such as remote control, mobile phone, laptop computer, video game console, voice application and multimedia application but not in the Inter-vehicle communication system due to its specification and the overview is shown in the Table 3.2 and graph of the bit and coverage rate is shown in the Figure 3.13.

<table>
<thead>
<tr>
<th>Features</th>
<th>Bluetooth</th>
<th>UWB</th>
<th>ZigBee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>IEEE 802.15.1</td>
<td>IEEE 802.15.3a</td>
<td>IEEE 802.15.4</td>
</tr>
<tr>
<td>Frequency band</td>
<td>2.4 GHz &amp; 2.5 GHz(V1.2)</td>
<td>3.1-10.6 GHz</td>
<td>2.4 GHz</td>
</tr>
<tr>
<td>Range</td>
<td>Up to 100 m</td>
<td>Up to 20 m</td>
<td>Up to 70 m</td>
</tr>
<tr>
<td>Network</td>
<td>Point-to-point</td>
<td>Point-to-point</td>
<td>Mesh</td>
</tr>
<tr>
<td>Modulation</td>
<td>FHSS</td>
<td>OFDM, DS-UWB</td>
<td>DSSS</td>
</tr>
<tr>
<td>Bit rate</td>
<td>1 Mbps(V1.0), 3 Mbps(V1.2), 12 Mbps(V2.0)</td>
<td>50-100 Mbps</td>
<td>250 kbps</td>
</tr>
<tr>
<td>Automotive usage</td>
<td>Portable devices, Device connectivity</td>
<td>Multimedia applications, Healthcare applications</td>
<td>Sensor/control applications</td>
</tr>
<tr>
<td>Strong point</td>
<td>Dominating PAN technology In vehicle today Easy synchronization of mobile device Frequency hopping tolerant to harsh environments</td>
<td>Easy and cheap to build Consume very little power Provides high bandwidth Broad spectrum of frequencies</td>
<td>Static network Control/sensor Many devices/nodes Small data packets Low duty cycle Low power</td>
</tr>
<tr>
<td>Weak point</td>
<td>Interference with Wi-Fi Consume medium power</td>
<td>Short range</td>
<td>Interference</td>
</tr>
</tbody>
</table>

Figure 3.13: Comparison parameters of the short range technologies [22]

Table 3.2: Overview of short-range communications [21, 22]
3.2.4 Medium-Range Communication Technology

A. WiFi

WiFi or Wireless Fidelity is in the IEEE 802.11 a/b/g/n standards for wireless local area networks (WLAN) which it is the most widely used to connected each computer to the access point to open connection to the Internet. It is using the 2.4 GHz frequency band for the b/g/n standards and in 5 GHz for a/n standards which allow the transmission range is from 100 meters up to 1 km and the transmission rate is from 54 Mbps and up to 600 Mbps in the n standard. WiFi was used in the Inter-vehicle communication system due to the limitations in the degree of coverage, capacity and the interference of the channel that conflict with the topical users, the high mobility of the cars which result in the frequently changed topology and network fragmentation. The different routing and MAC layers have to be modified in order to use in Inter-vehicle communication system.

B. DSRC

DSRC or Dedicated Short-Range Communication is in the IEEE 802.11p standard which its physical layer based on IEEE 802.11a and the modified on MAC layer. The frequency bands that are used either 5.8 GHz or 5.9 GHz, the transmission range is up to 1 km and the channel bandwidth is up to 10MHz that gives the data transmission rate is up to 27Mbps. This is one of the wireless standards that are property designed for the Inter-vehicle communication system.

DSRC is currently considered the most promising wireless standard in vehicular networks that adopting WiFi standards facilitates operations in infrastructures and ad hoc modes which map to V2I and V2V communications, respectively. [22]

The different standards for DSRC specifications are divided into each country such as Japan, Europe and America that are shown in the Figure 3.14.
C. WAVE

WAVE or Wireless Access in Vehicular Environments is the combination of IEEE 802.11p and the IEEE 1609 protocol suite as shown in the Figure 3.15. The frequency band that is used is 5.8 GHz, the transmission range is up to 1 km and the transmission rate is up to 27Mbps.

The IEEE 1609 protocol suite for VANETs are: IEEE P1609.1 (Wireless Access for Vehicular Environments -WAVE), P1609.2 (standard for vehicular network security), P1609.3 (routing and transport services) and P1609.4 (multiple channels in the DSRC standard).

This standard is also one of the VANETs wireless standards that are used in the Inter-vehicle communication system.
In summary, these standards are range up to 1 km that are used for the Inter-vehicle communication system and its specification and the overview is shown in the Table 3.3 and graph of the bit and coverage rate is shown in the Figure 3.16.

![Graph showing bit rate vs. coverage rate for WAVE, DSRC, and Wi-Fi](image)

**Table 3.3: Overview of medium range communications [22]**

<table>
<thead>
<tr>
<th>Features</th>
<th>WiFi</th>
<th>DSRC</th>
<th>WAVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>IEEE 802.11 (a/b/g/n)</td>
<td>IEEE 802.11p</td>
<td>IEEE 1609 based on MAC and Network Layers</td>
</tr>
<tr>
<td>Frequency band</td>
<td>2.4 GHz (b/g/n) &amp; 5 GHz (a/n)</td>
<td>5.9 GHz</td>
<td>5.9 GHz</td>
</tr>
<tr>
<td>Range</td>
<td>100 m to 1 km</td>
<td>1 km</td>
<td>1 km</td>
</tr>
<tr>
<td>Network</td>
<td>Point-to-point</td>
<td>Point-to-point</td>
<td>Point-to-multipoint</td>
</tr>
<tr>
<td>Modulation</td>
<td>OFDM or DSSS with CCK, BPSK, QPSK, 16 or 64 QAM</td>
<td>BPSK, QPSK, 16 or 64 QAM</td>
<td>BPSK, QPSK, 16 or 64 QAM</td>
</tr>
<tr>
<td>Bit rate</td>
<td>54 Mbps (a/g) &amp; 11 Mbps (b) 600 Mbps (n)</td>
<td>3 to 27 Mbps</td>
<td>3 to 27 Mbps</td>
</tr>
<tr>
<td>Automotive usage</td>
<td>V2V, V2I</td>
<td>V2V, V2I</td>
<td>V2V, V2I</td>
</tr>
</tbody>
</table>

**3.2.5 Long-Range Communication Technology**

**A. WiMax**

WiMax or Worldwide Interoperability for Microwave Access is in IEEE 802.16 standard and is part of 4G telecommunication technologies using frequency band from 2.3, 2.5, 3.5 GHz depends on the specifications. The transmission rate is up to 75 Mbps cover the range is up to 5 km.
B. Cellular Technology

This includes 3G, LTE, UMTS, HSPA and GSM which the standards and frequency bands depend on each cellular technology. The coverage range is up to 15km and the data transmission bandwidth is up to 100 Mbps.

In summary, there main standards purpose in this range is to connect the networks into the infrastructures to extend the longer range. The specification and the overview are shown in the Table 3.4 and graph of the bit and coverage rate is shown in the Figure 3.17.

<table>
<thead>
<tr>
<th>Features</th>
<th>WiMax</th>
<th>Cellular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>IEEE 802.16</td>
<td>Based on 3G/4G cellular technologies standard</td>
</tr>
<tr>
<td>Frequency band</td>
<td>2.3, 2.5 and 3.5 GHz bands</td>
<td></td>
</tr>
<tr>
<td>Range</td>
<td>Up to 50 km</td>
<td>Up to 15 km</td>
</tr>
<tr>
<td>Network</td>
<td>Point-to-multipoint, Full mobile</td>
<td>Full mobile</td>
</tr>
<tr>
<td>Modulation</td>
<td>OFDMA, 16 or 64 QAM (BPSK ½, QPSK ½)</td>
<td>FDD, TDD, CDMA</td>
</tr>
<tr>
<td>Bit rate</td>
<td>75 Mbps</td>
<td>2 to 100 Mbps</td>
</tr>
<tr>
<td>Automotive usage</td>
<td>V2I</td>
<td>V2I</td>
</tr>
</tbody>
</table>

Table 3.4: Overview of long range communications [22]

![Figure 3.17: Comparison parameters of the long range technologies][19]

3.2.6 Discussion

In the paper “A Feasibility Study on Vehicle-to-Infrastructure Communication WiFi vs. WiMax”, the WiFi and WiMax were studied to determine the performance of each technology. The study shows that the throughput of WiFi is significantly larger than WiMax due to it uses 20MHz channel bandwidth but 3.5MHz is set to WiMax in the experiment that is shown in the Figure 3.18.
In WiMax, on the other hand, the data can still be transmitted from the distance around 1 km which gives the throughput up to 1 Mbps using the 64QAM modulation technique which is shown in the Figure 3.20.

The throughput of the WiMax increases as the frame size is set to a higher value on the base station and the frame size also can affect the RTT which the study shows that the RTT of WiMax is significantly larger than the WiFi when the transmission range is shorter as shown in the Figure 3.21 and the measurement of the throughput for WiMax is also significantly lower than the theoretical estimate which is shown in the Figure 3.19.

In summary, the authors claim that one might be able to get the more throughputs and a shorter latency from WiFi at a short distance, assuming that there are much interference in the 2.4 GHz band and the frame duration which can have the critical effect on the performance when using the WiMax. [24]
In conclusion, most of the important wireless standards that are used in the automotive domain are shown in this topic which broad range from the short to long ranges and low to high bandwidths including Bluetooth, Ultra-Wideband, ZigBee, WiFi, DSRC, WAVE, WiMax and Cellular Technologies. Different applications used in the automotive domain require the variant of the specifications to fulfill the functionality. In the Car2X Communication or Inter-Vehicular Networks (VANET) to communicate between the vehicles and infrastructures, then the most suitable to acquire the most use of the wireless standards are using the IEEE 802.11p or WAVE which it is well defined for the requirements of the Inter-vehicle communication system. However, this thesis chooses the WiFi, IEEE 802.11 a/b/g/n, to send the data to the web server in the InDoor Lab because it is cheaper and suitable to demonstrate the architecture that only need to send the data wirelessly to the web server.

### 3.3 CAN-Bus System

CAN stands for Controller Area Network was first developed by Robert Bosch GmbH in 1983 but was official released in 1986. It is the most popular bus standard in the car system which is the multi-master serial bus communication standard used in car for the data exchanged of the controllers so that they can share the information on the bus system. It is very reliable, robust and inexpensive. [25]
There are two different frame formats. The first one is the standard or the base format which is described in CAN 2.0A and CAN 2.0B. The second one is the extended frame format which is only described in the CAN 2.0B. The only main difference between this two frames format is that the extended frame format has the extend frame identifier which has the 29-Bit instead of the standard one which has 11-Bit frame identifier.

There are two International Standards ISO 11898 which the first one is “ISO 11898-2” called as the high speed CAN that supports the speed applications up to 1 Mbps and the second one is “ISO 11898-3” called as the low speed CAN that supports the speed application is up to 125 Kbps.
The node of the CAN-Bus consists of main parts as shown in the Figure 3.24:

- Central processing unit decides whether to receive the messages from the CAN-Bus and the messages that are going to send.

- CAN Controller is often connected or integrated to the microcontroller and responsible for the receiving the message by waiting the completed serial bits then sends to the processor, and sending the message from the processor to the CAN-Bus when the bus is free.

- Transceiver is the medium access unit standard and defined by ISO 11898-2/3 and it converts data stream from the CAN-Bus for receiving and transmitting data to the CAN Controller.

![Figure 3.24: CANbus Node](image)

However, the CAN-Bus message consists of four different frames that help the node know which type of message that being sent.

- DATA FRAME: The frame that includes the actual data in the message.

- REMOTE FRAME: The frame that requests another node to send the data frame.

- ERROR FRAME: The frame that is sent by any node that detects the error on the bus.
- OVERLOAD FRAME: The frame that is used for providing more delay before or after the other frame of the data frame or remote frame.

The INTERFRAME SPACE is the frame that separates the DATA FRAME and REMOTE FRAME from the other frame.

CAN Data Frame

There are two types of the data frame which is the standard or base data frame and the extended data frame. The only difference is the identifier that the standard frame has 11-Bit and the extended frame has 29-Bit. This data frame is the frame that actually sends the data to the CAN-Bus system.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Length (bits)</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-of-frame</td>
<td>1</td>
<td>Denotes the start of frame transmission</td>
</tr>
<tr>
<td>Identifier (green)</td>
<td>11</td>
<td>A (unique) identifier which also represents the message priority</td>
</tr>
<tr>
<td>Remote transmission request (RTR) (blue)</td>
<td>1</td>
<td>Must be dominant (0) for data frames and recessive (1) for remote request frames</td>
</tr>
<tr>
<td>Identifier extension bit (IDE)</td>
<td>1</td>
<td>Must be dominant (0) for base frame format with 11-bit identifiers</td>
</tr>
<tr>
<td>Reserved bit (r0)</td>
<td>1</td>
<td>Reserved bit. Must be dominant (0), but accepted as either dominant or recessive.</td>
</tr>
<tr>
<td>Data length code (DLC) (yellow)</td>
<td>4</td>
<td>Number of bytes of data (0–8 bytes)</td>
</tr>
<tr>
<td>Data field (red)</td>
<td>0–64 (0–8 bytes)</td>
<td>Data to be transmitted (length in bytes dictated by DLC field)</td>
</tr>
<tr>
<td>CRC</td>
<td>15</td>
<td>Cyclic redundancy check</td>
</tr>
<tr>
<td>CRC delimiter</td>
<td>1</td>
<td>Must be recessive (1)</td>
</tr>
<tr>
<td>ACK slot</td>
<td>1</td>
<td>Transmitter sends recessive (1) and any receiver can assert a dominant (0)</td>
</tr>
<tr>
<td>ACK delimiter</td>
<td>1</td>
<td>Must be recessive (1)</td>
</tr>
<tr>
<td>End-of-frame (EOF)</td>
<td>7</td>
<td>Must be recessive (1)</td>
</tr>
</tbody>
</table>

Table 3.5: Standard data frame [25]
CAN Remote Frame

This remote frame is used to send to the CAN-Bus system when the node needs to request the data from another node to transmit the data to the CAN-Bus. The CAN remote frame is almost identical with the data frame. But there are only two different things between the CAN data frame and the CAN remote frame. The first one is the RTR-bit is sent as a dominant (RTR = 0) in data frame but recessive (RTR = 1) in the remote frame. The second is that there is no data field for the remote frame.

CAN Error Frame

This error frame is sent when the node detects an error. It consists of two different fields which are the ERROR FLAG and the ERROR DELIMITER. There are two types of error frame

- ACTIVE ERROR FLAG which is transmitted by a node that detects an error on the bus network with sending the “six dominant bits”

- PASSIVE ERROR FLAG which is transmitted by a node that detects the active error frame on the bus network with sending the “six recessive bits”

- ERROR ELIMITER will be transmitted after the ERROR FLAG by sending the “eight recessive bits”

Figure 3.26: CAN error frame [26]
CAN Overload Frame

This overload frame is mostly the request frame for the node that needs the delay. It consists of two different fields which are the OVERLOAD FLAG and the OVERLOAD DELIMITER. However, there are two conditions of the overload.

1. The node needs the delay before the start of the next DATA FRAME or REMOTE FRAME

2. When the node detects a “dominant bit” during the INTERMISSION.

The condition for the first one is that it is allowed to be started only at the first bit time of the expected INTERMISSION and the second one it that it is allowed to be started after one bit of detecting the “dominant bit”.

- OVERLOAD FLAG consists of the “six dominant bits” as the same form as the active error flag.

- OVERLOAD DELIMITER consists of the “eight recessive bits” and it is the same form as the error delimiter.

![Figure 3.27: CAN overload frame](image)

Bit Stuffing

The bit stuffing is the method to maintain synchronization by adding a opposite polarity bit after the five consecutive bits of the same polarity which is non information bit and the data will be destuffed by the receiver. However, bit stuffing is not including in the CRC delimiter.
ACK field and end of frame fields because these fields have the fixed sized. Whenever the node receives the six consecutive bits of the same polarity, it will consider it as an error. The Figure 3.28 shows the difference between the normal frame and the frame with the bit stuffing.

**Figure 3.28:** CAN-Frame before and after the addition of stuffbits (in purple) [25]

### How the CAN-Bus works

When the node needs to send the data to the CAN-Bus, it checks the bus whether it is free to be transmitted the data or not. When the bus is free, the node sends the data into the bus system this data message contains the frame identifier or arbitration id and the required fields depend on the type of the frame. The frame message is then transmitted into the bus system, this message can be received on all nodes but each node decides whether this message should be received or not by using the frame identifier. However, when multiple nodes are trying to send the message at the same time, a node with the highest priority or the one that has the lower frame identifier, has the more zeros in the front, wins and sends the data and the others have to wait and try to send again.

<table>
<thead>
<tr>
<th>Start Bit</th>
<th>ID Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node 15</strong></td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td><strong>Node 16</strong></td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td><strong>CAN Data</strong></td>
<td>0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

**Table 3.6:** Shows the node priority works [25]
In summary, the CAN-Bus provides the cheap and durable network with the multiple devices can communicate to each other by using the single bus system. It also reduces the cost of wiring to connect every node together by using only single wire to connect to the CAN-Bus system as shown in the Figure 3.29. The message that is sent by one node can be received by all nodes. However, each node decides whether the message should be received by using the frame identifier. Thus, the frame identifier is also used to decide when multiple nodes are trying to send the message at the same time. A node with the highest priority which is the node that has the lower frame identifier wins and sends the data to the CAN-Bus system.

3.4 Computer Vision

Computer vision is a field that uses the computer to understand the meaning of the image as the abilities of human vision by processing and analyzing the image data using models constructed with the aid of geometry, physics, statistics, and learning theory into the descriptions of world so that the appropriate actions and other thought processes can be interfaced with. [28]

The human eyes can understand the meaning of the image and can tell whether it comes from the same object even the image is taken in different angles. Because of humans are the intelligent creatures that can recognize the image through the very fast of processing of their brain. We, as humans, can tell the image is a car and can divide parts of a car such as windows, mirror and doors. However, the computer sees this image as the digital numbers
The number represents the view of 3D world from the 2-dimensional view. It is very hard to determine the 2D images that taken from the change of the view point of the same image as the numbers are different. Not only the different angles that causes this problem, it also includes the data may corrupted by noise and distortions such as the weather, lighting, reflections, dust and the movement. These come to the challenges in the processing of the image in the computer vision.

In the practical system design, the additional contextual knowledge can often be used to determine the object of interest. For the example of the mobile robot that needs to find the object in the building such as staplers. The knowledge that the fact most staplers are usually on the desk which gives the robot to realize a staple must be fit on the desk. The robot then uses the data to process the image and looks for the staple. The contextual information also includes the machine learning techniques by using the hidden variables such as size and orientation to the gravity to correlate with their values in the label training set. The other factor that is the problem for the computer vision is noise. The typically way to deal with noise or the distortion is by using the statistical method to build the explicit models learned from the available data such as using the parameters for a simple polynomial model to describe the lens distortions. [29]
The application for the computer vision is often in the machine vision systems which the computer pre-programmed on the robot to solve the particular task and this method uses the computer vision to automated analysis the image. The examples for the computer vision system such as:

- Controlling processes in an industrial robot
- Navigation by an autonomous vehicle or mobile robot
- Detecting event for counting people or the movement
- Modeling object for medical image analysis
- Computer-human interaction that uses the input from the device
- Automatic inspection in the manufacturing applications

Medical computer processing or medical image processing is one of the most prominent application fields which the extracted information from the image data is characterized on the specific area for the purpose of making a medical diagnose of the patient. So that can determine for the detection of tumors, arteriosclerosis or other malign changes. The second area where the computer vision is used in the industrial area by using the information to support the manufacturing process such as in the quality control process the finally products are automatically inspected to find the defects. The robot arms pick up the object by using the measurement of the position and orientation details. It is also used in the agricultural process for detecting and removing undesirable food stuff from the bulk material which this process called optical sorting. [28]

One of the largest areas of the computer vision is in the military fields that the applications are detecting of the enemy soldiers, vehicles and missile guidance. Instead of sending the missile to the specific target at the launch time, the missile is sent to the area, where the possible target is, and then the target is based on the locally acquired data when the missile reaches the area. It also provides the rich set of information to support the decision making about the combat scene by using various sensors including image sensors. [28]

Autonomous vehicle is the latest version of the example application areas such as the unmanned aerial vehicles (UAV) that uses the computer vision for navigation. The pilot also uses the computer vision to support in various situations. Moreover, it is also used in the
space exploration by using the autonomous vehicles such as NASA's Mars Exploration Rover and ESA's ExoMars Rover. [28]

![An example of an unmanned land-based vehicle](image)

The typical tasks of the computer vision can be presented as the recognition, motion analysis, scene reconstruction and image restoration which each application employs range of the computer vision tasks to the problems by using the various methods. The different varieties techniques are used to recognition the problem such as object recognition, identification, detection, content based image retrieval, pose estimation, optical character recognition, 2D code reading and facial recognition.

The following are the typical functions of the computer vision systems.

**Image acquisition** is the digital image which in an ordinary 2D image, a 3D volume or the sequence of images.

**Pre-processing** is the process to assure the data is satisfied by using the re-sampling, noise reduction, contrast enhancement and scale space methods.

**Feature extraction** is the extracting the image features from the image data.

**Detection/segmentation** is the process of selection of the relevant points or regions from the image.
**High-level processing** is the remaining processes to deal with the image that assumed it contains the specific object such as verification, estimation, image recognition and image registration processes.

**Decision making** is the final decision of the image whether fits the application by automatic inspection, recognition or flag to require the human review.

There is a tool for helping the above processes called OpenCV that provides basic tools to solve the computer vision problems.

**OpenCV**

OpenCV is an open source library of programming function intended for real-time applications which can be downloaded on http://www.opencv.org. It is originally developed by Intel and officially launched in 1999. OpenCV library plays role in the computer vision which reduces the cost of preparation time for research environment needed by researchers. It provides simple to use functions to get work done in a effective manner which the library has more than 2500 optimized algorithms to use in the computer vision and machine learning algorithms such as the detecting and recognizing face, identifying object, classifying human actions in video, tracking camera movement, tracking moving object and etc. The OpenCV has more than 47 thousand people of user communities which the library has extensively used in the companies, research groups and the government bodies. [30]

The main OpenCV modules are listed below [31]:

- **core**: This is the basic standard module of OpenCV which provides the basic data structure and image processing functions.
- **highgui**: This module provides simple image and video capturing, user interfaces capabilities.
- **imgproc**: This module gives the basic image processing that includes linear and non-linear image filtering, geometrical image transformations
- **video**: This is a module for a video analysis includes motion estimation, background subtraction, and object tracking algorithms
- **objdetect**: This module provides the object detection, recognition algorithms and instances of the predefined classes for standard objects.
3.5 Microcontroller

A microcontroller is a small computer, system on chip, on a single integrated circuit that contains the processor core, memory, programmable I/O peripherals. It is intended for the embedded applications such as automobile engine control systems, remote controls, appliances, power tools, toy and other embedded systems. The size is reduced and cost is cheaper than the design that uses a separate microprocessor, memory, and I/O peripherals because of the minimal requirements of memory, program length and low software complexity unlike the personal computer that needs keyboard, screen, disks, printer and other peripheral devices to connect with. [32]

The microcontroller provides the real-time interrupt to respond to the event that needs to be controlled. The interrupt system sends the signal to the processor to perform the processing that responds to the event based on the source of the interrupt event occurs; this is called interrupt service routine or ISR. The programs must fit in the memory of the microcontroller which it is converted to the machine codes by using the compliers and assembles. This memory can be permanent which can only be programmed at the vendor factory or programmable which can be re-programmed later. The general purpose input/output pins (GPIOs), Universal Asynchronous Receiver/Transmitter (UART), Inter-Integrated Circuit (I²C), Serial Peripheral Interface (SPI), Universal Serial Bus (USB), and Ethernet are usually attached on most microcontroller for programming the software to communicate with other peripherals. [32]

The following boards are the example of microcontrollers that low-cost, smaller size and widely used in the embedded application areas today, Arduino and Raspberry Pi which are shown in the Figure 3.32 and 3.33, respectively:
**Arduino**

Arduino is an open-source prototyping platform based on easy-to-use hardware and software that designs and manufactures microcontroller based kits for building the digital devices. The Arduino board reads the input such as from lights on the sensor, a finger on a button and changes it to the output such as activating a device, blinking the LED, posting something online. The board can be interfaced to the other expansion boards, which is called shields. It has the serial communications interfaces, including USB that used to connect to the personal computer for uploading the program by using the integrated development environment (IDE) for Arduino that supports C and C++ programming languages. [33][35]

**Raspberry Pi**

Raspberry Pi is a credit-card sized, low-cost, computer originally designed for education of teaching basic computer science in school which was developed in England, United Kingdom by the Raspberry Pi Foundation. It provides more functionality than the basic microcontroller such as Arduino, it is a small complete Linux computer but it is slower than the personal computer. [34][36]

However, choosing between Arduino and Raspberry Pi depends on the task and the purpose of using them. Some applications do not require the complete computer system such as the sensor for turning on the light when it detects the movement which this requirement should use the Arduino instead of using Raspberry Pi because of Arduino can provides this simple functionality by using the sensor and some programming but if the Raspberry Pi is chosen, it will be required to setup the complete Linux system and do the programming for it. Both boards have about the same price and size, however, the Raspberry Pi provides more memory and faster clock speed, it also includes with Ethernet port in the B model. But Arduino does not require the operation systems which it is pure real-time and analog capability and the Arduino IDE is much easier than using the Linux. The main major differences between Arduino and Raspberry are shown in the Table 3.7.

<table>
<thead>
<tr>
<th></th>
<th>Arduino Uno</th>
<th>Raspberry Pi Model B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Price</strong></td>
<td>$30</td>
<td>$35</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>7.6 x 1.9 x 6.4 cm</td>
<td>8.6 x 5.4 x 1.7 cm</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>0.002 MB</td>
<td>512 MB</td>
</tr>
<tr>
<td><strong>Clock Speed</strong></td>
<td>16 MHz</td>
<td>700 MHz</td>
</tr>
<tr>
<td><strong>On Board Network</strong></td>
<td>None</td>
<td>10/100 wired Ethernet RJ45</td>
</tr>
<tr>
<td><strong>Multitasking</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Input Voltage</strong></td>
<td>7 to 12 V</td>
<td>5 V</td>
</tr>
<tr>
<td><strong>Flash</strong></td>
<td>32 KB</td>
<td>SD Card (2 to 16 GB)</td>
</tr>
<tr>
<td><strong>USB</strong></td>
<td>None, input only</td>
<td>Two, peripherals OK</td>
</tr>
<tr>
<td><strong>Operation System</strong></td>
<td>None</td>
<td>Linux Distributions</td>
</tr>
<tr>
<td><strong>Integrared Development Environment</strong></td>
<td>Arduino</td>
<td>Scratch, IDE, anything with Linux support</td>
</tr>
</tbody>
</table>

Table 3.7: The main differences between Arduino and Raspberry Pi [37]
In conclusion for this chapter:

- It provides the detail about the mapping technologies which required for this thesis that is using the OpenStreetMap.
- It gives the research about the wireless communication for Car2X in different wireless standards and concludes that the IEEE 802.11 a/b/g/n is suitable to send the wireless signal of the data to the web server.
- It explains the basic knowledge of CAN-Bus system for transmitting the data in the car.
- It introduces the overview of computer vision including OpenCV.
- Finally, it explains the microcontroller that used in embedded system and shows the brief differences between Arduino and Raspberry Pi.

The next chapter gives the concept detail for the proposed work.
Chapter 4

Concept

This chapter gives the concept for the proposed work. It explains how the barcodes are interpreted as traffic signs, how to scan the barcode, how to transmit the data on the CAN protocol, how to send the data to the web server and how to embedded the traffic signs on the OpenStreetMap.

4.1 Architecture

The Figure 4.1 illustrates the overview of the architecture. Each barcode represents a traffic sign and is stored in a database. The barcode scanner scans the barcode together with the GPS location from the scanning point. This data is transmitted through CAN-Bus communication in the car. After that, the car sends the data to the web server over WiFi communication. The web server looks up the meaning of the barcode from the online database and returns a respective traffic sign which then together with other data e.g., the GPS location is used to embed traffic sign on OpenStreetMap.

Figure 4.1: The overview and data flow of the architecture
The data flow of the architecture is shown in the Figure 4.1, the barcode identity number and GPS location is provided by USB Barcode Scanner and USB GPS, respectively, to Raspberry Pi board. Then the data is converted into the CAN-Bus data frame format and is sent by the PICAN-Shield over the CAN-Bus. After that, the data is received by the other Raspberry Pi via the PICAN-Shield over the CAN-Bus. Finally, the data is transmitted to the web server and the online database over the WiFi for enriching the OpenStreetMap. The block diagram is also represented in the Figure 4.2 below.

![Figure 4.2: The block diagram of the architecture](image)

In order to display the OpenStreetMap on the web browser, the OpenStreetMap Tiles Server has to be used either using the Local Tiles Server or the OpenStreetMap Tiles Server. In this thesis, the local tiles server is used together with the OpenStreetMap Tiles server. The Local Tiles Server, the CentOS Linux is the operation system for running the web server and the database. After the Local Tiles Server has been installed, the LeafLet is also needed to embed the traffic signs for the OpenStreetMap which generated from the data that has been stored in the database previously. The block diagram of the selected Local Tiles Server is drawn in the official OpenStreetMap block diagram of the red color in the Figure 4.4.

![Figure 4.3: Transforming the traffic sing to the barcode](image)
However, the traffic sign has to be transformed into the barcode which is the one dimensional (1D) barcode that represents the meaning of each traffic sign as shown in the Figure 4.3. To do this, the barcode identity number is assigned to each traffic sign and this barcode identity number together with the meaning are stored in the online database to serve as the database for the querying of the data that is used during the observation of the local traffic sign and the embedding of the OpenStreetMap. The database for the traffic sign has 5 entities: “id” is for the barcode identity number, “name” is for the name of the traffic sign, “type” is for the type of the traffic sign, “desc” is for the meaning of the traffic sign, and “iconurl” is for the name of the image of the traffic sign. The example of the traffic sign database is shown in the Figure 4.5 below.

![Figure 4.5: The database for the traffic sign](image)

### Table: Traffic Sign Database

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>type</th>
<th>desc</th>
<th>iconurl</th>
</tr>
</thead>
<tbody>
<tr>
<td>100001</td>
<td>Speed Limit 30km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 30km per hour</td>
<td>670/0.png</td>
</tr>
<tr>
<td>100002</td>
<td>Speed Limit 40km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 40km per hour</td>
<td>570.png</td>
</tr>
<tr>
<td>100003</td>
<td>Speed Limit 20km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 20km per hour</td>
<td>670/20.png</td>
</tr>
<tr>
<td>100004</td>
<td>Speed Limit 60km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 60km per hour</td>
<td>670/06.png</td>
</tr>
<tr>
<td>100005</td>
<td>Speed Limit 50km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 50km per hour</td>
<td>670/50.png</td>
</tr>
<tr>
<td>100006</td>
<td>Give way</td>
<td>Regulatory signs</td>
<td>Give way to traffic on the major road</td>
<td>602.png</td>
</tr>
<tr>
<td>100007</td>
<td>No entry</td>
<td>Regulatory signs</td>
<td>All vehicles prohibited except pedal cycles being ...</td>
<td>617.png</td>
</tr>
<tr>
<td>100008</td>
<td>Road narrows on left ahead</td>
<td>Warning signs</td>
<td>Road narrows on left ahead</td>
<td>511.png</td>
</tr>
<tr>
<td>100009</td>
<td>Zebra crossing ahead</td>
<td>Warning signs</td>
<td>Zebra crossing ahead</td>
<td>544.png</td>
</tr>
<tr>
<td>100010</td>
<td>Road hump on or ahead</td>
<td>Warning signs</td>
<td>Road hump on or ahead</td>
<td>5571.png</td>
</tr>
</tbody>
</table>

![Figure 4.6: The database for the point of traffic sign](image)
In addition to the traffic sign database, the database for the point of the traffic sign is used to determine the location of them. Therefore, this database has 4 entities: “id” is to identify each point, “barcode_id” is used to query the information of the traffic sign, “latitude” is for the latitude of its location, and “longitude” is also refer to the longitude of its location. The overview of the point database is show in the Figure 4.6.

To embed the traffic sign on OpenStreetMap, the Leaflet is selected JavaScript Library which is small and simple for the standard embedding map in the web browser. The JSON data is needed for the Leaflet to embed the map and this is done by passing the HTML GET method to query the databases, the traffic sign and the point, to return the data in the JSON data format. However, the boundaries of the latitude and the longitude have to be given as the parameter for the querying. The querying URL has 4 parameters: “minla” is for the minimum latitude, “maxla” is for the maximum latitude, “minlo” is for the minimum longitude, and “maxlo” is for the maximum longitude. The example of the querying data as the JSON data format is shown in the figure 4.7 below.

![Figure 4.7: The query of the traffic sign from the database in the JSON data format](image)

Finally, by using the HTML GET method, it gets data as the JSON data format which has 5 entities in each array: “name” is for the name of the traffic sign, “desc” is for the meaning of traffic sign, “latitude” is for the latitude of the traffic sign, “longitude” is for the longitude to the traffic sign, and “iconurl” is for the file name of the traffic sign. Then Leaflet uses this data to embed the OpenStreetMap to display the traffic signs on the web browser as shown in the Figure 4.8.
Moreover, the PI Camera is also used in this thesis to improve the proposed work by detecting the traffic light in the motion. The car gets the command for the GO or STOP signal on the CAN-Bus from the Raspberry Pi board. It uses the OpenCV to process the data from the PI Camera that streams the video of traffic light and sends the command to the car. The example can be given such as when the car is going on the road to the junction and it sees the green light and suddenly changes to red light, this scenario can be explained on the following steps: firstly the Raspberry Pi board gets the video streaming data from the PI Camera, the it uses OpenCV to detect the traffic light whether it is green, yellow or red light and then sends the command signal corresponding to the detected light. However, there is a requirement for the STOP signal that when the yellow or red light is detected, it calculates the distance between the traffic light and the car whether it is in the distance that the car should be stopped. This prevents the car stops immediately after it detects the red or yellow light in a very far distance or any fault detection.

To do this, the Training of the OpenCV cascade classifier for the traffic light has to be trained to detect the traffic light first which the detail and the usage are available at [46]. Thus, the positive and negative sample images have to be prepared. The images are cropped into the smaller images with region of interest as set positive images which are be detected, as shown in the Figure 4.9. In total, 69 positive images of the traffic light are prepared and 4000
negative sample images, from the online image data set at [47], which this image do not include any traffic light in the image. After that, the positive sample images are generated by combining the negative sample images together with the positive images in the gray scale. This can be done by using `opencv_createsamples` function of OpenCV or using more easily method on Perl script, `createsamples.pl`, created by Naotoshi Seo [47]. The script places a positive image on a negative sample image in random rotation to create a new positive sample image. In total, 1380 positive sample images are generated from 69 positive images and 4000 negative sample images as shown in the Figure 4.10.

After all the images sets are prepared, OpenCV is used to generated a cascade classifier by running the `opencv_traincascade` function of the OpenCV by passing the prepared images as the parameters argument [46]. All the OpenCV functions can be used directly in the terminal shell after OpenCV has been successfully installed. This step can be run on any computer for the faster processing time because the Raspberry Pi board has very low speed and low memory and takes longer time. The XML result file can be copied to the Raspberry Pi board to use it for the cascade classifier after it has been created. It took about 4 days to complete the running on the Mac book Pro on the following specification:

- 2.4GHz Intel Core 2 Duo
- 8GB 1067MHz DDR3 RAM
- NVIDIA GeForce 320M 256MB

The example shows in the Figure 4.11 and the XML file has been generated as shown in the Listing 4.1.
Listing 4.1: The XML result from the training of OpenCV

```
<?xml version="1.0" ?>
<opencv_storage>
  <cascade>
    <stageType>BOOST</stageType>
    <featureType>HAAR</featureType>
    <height>60</height>
    <width>40</width>
    <stageParams>
      <boostType>GAB</boostType>
      <minHitRate>9.9500000476837158e-01</minHitRate>
      <maxFalseAlarm>5.000000000000000e-01</maxFalseAlarm>
      <weightedTrimRate>0.4900000000000006e-01</weightedTrimRate>
      <maxDepth>1</maxDepth>
      <maxWeakCount>100</maxWeakCount>
    </stageParams>
    <featureParams>
      <maxCatCount>0</maxCatCount>
      <featSize>1</featSize>
      <mode>BASIC</mode>
    </featureParams>
    <stageNum>14</stageNum>
    <stages>
      <stage 0 -->
        <maxWeakCount>3</maxWeakCount>
        <stageThreshold>1.5565726757049561e+00</stageThreshold>
        <weakClassifiers>
          <classifier>
            <internalNodes>0 - 1 19.4.2758192667040634e-02</internalNodes>
            <leafValues>6.8153983354568481e-01 - 9.797703809738159e-01</leafValues>
          </classifier>
          <classifier>
            <internalNodes>0 - 1 564.4.086438062191910e-02</internalNodes>
            <leafValues>-9.5947057008743286e-01 5.97790355682373e-01</leafValues>
          </classifier>
        </weakClassifiers>
      </stage 0 -->
    </stages>
  </cascade>
</opencv_storage>
```

Figure 4.11: The example for the training of OpenCV
4.2 Selected Hardware

4.2.1 Microcontroller

Raspberry Pi 2 B is used as the microcontroller board. It is the second generation of the Raspberry Pi. It has a 900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM, 4 USB ports, 40 GPIO pins, full HDMI port, Ethernet port, combined 3.5mm audio jack and composite video, camera interface, display interface, micro SD card slot and video core IV 3D graphic core [34]. The Raspbian is used as the operating system and is installed on the Raspberry Pi. The thesis uses two Raspberry Pi 2 B boards to implement the architecture.

PICAN-Bus Shield is the additional board connected on the 40-GPIO pins to provide the capability for the Raspberry Pi to be able to communicate in CAN-Bus protocol. It uses Microchip MCP2515 CAN controller with MCP2551 CAN transceiver [39]. Two PICAN-Bus shields are also needed for this thesis.

4.2.2 USB GPS Module

The GPS module, as shown in the Figure 4.14, which is connected to the USB port, is used to determine the location of the traffic sign. It is connected to the first Raspberry Pi 2 B, thus, only one is needed in this thesis.
4.2.3 USB WiFi Dongle

The USB WiFi dongle, as shown in the Figure 4.16, is used to send the data to the web server wirelessly. It is the IEEE 802.11b/g/n standard and has the speed up to 150Mbps. We use the IEEE 802.11b/g/n standard instead the IEEE 802.11p because it is cheaper and we focus more on the data that transfer in the CAN-Bus system. It is connected to the second Raspberry Pi 2 B.

4.2.4 USB Barcode Scanner

The USB Barcode Scanner, as shown in the Figure 4.15, is used to scan the traffic sign barcode and send the data to the first Raspberry Pi 2 B board. It has HID USB interface and the baud rate is 9600.

4.2.5 Raspberry Pi Camera

The Raspberry Pi Camera, as shown in the Figure 4.17, is used to stream the video and send the data to the first Raspberry Pi 2 B board to determine the traffic light whether is it green, yellow or red.

In summary, two Raspberry Pi 2 B boards together with two PICAN-Bus shields are used as the microcontrollers. These two boards are communicated via the CAN-Bus protocol. The USB GPS Module, the USB Barcode Scanner, and the Raspberry Pi Camera are connected to the first Raspberry Pi 2 B board that is used to send and read the required data from the local observation. The USB WiFi dongle is connected to the second Raspberry Pi 2 B board to send the data to the web server. The overview of the Hardware Architecture is illustrated in the Figure 4.18 and the overview the block diagram for this thesis is also illustrated in the Figure 4.19.

Figure 4.18: The overview of the hardware for this architecture
Figure 4.19: The overview of block diagram for this architecture
In conclusion for this chapter:

- It shows the architecture for the proposed work.
- It also provides the improvement architecture by using PI Camera.
- It explains basic specifications of the hardware for this thesis.

The next chapter gives the implementation part for the proposed work.
Chapter 5

Implementation

To implement the architecture, some programming languages are needed to be done to make the hardware runs the way as planned. The programming code in the first Raspberry Pi board is to receive and send the data from the USB Barcode scanner, the USB GPS dongle and the Raspberry Pi Camera, and send the data to the CAN-Bus. And in the second Raspberry Pi board is to receive the data from the CAN-Bus and send the data to the web. The main programming languages that are used in the thesis include Python, PHP and JavaScript.

It also explains how to write the programming of each step in the following sections from scanning the barcode, getting the GPS location, transmitting the data through the CAN-Bus, receiving the data from the CAN-Bus, sending the data to the web server, receiving the data on the web server, storing the data on the database, reading the data for embedding the OpenStreetMap, enrichment the OpenStreetMp and detecting the traffic light.

5.1 Scanning barcode

The barcode of the traffic sign is scanned by the USB Barcode Scanner and then it sends the data to the Raspberry Pi board. The programming code that is used for getting the barcode data is in the filename, get_send_osm.py, it is written in the Python programming language. The following required libraries are needed to be imported as shown on line 9 and 10 of the Listing 5.1.

```python
7. #Import required library for Barcode scanner
8. from evdev import InputDevice, ecodes, list_devices
9. from select import select
10.
Listing 5.1: Importing required libraries for barcode scanner
Then the function `getBarcode()` is the one that gets the barcode data from the Barcode Scanner. Inside the function, the definition the ASCII code to read the data from the Barcode Scanner has to be defined as the array name “keys” from line 23 to 31 of the Listing 5.2.

Listing 5.2: ASCII codes represent for the barcode scanner

```python
23. keys = {
24.     # Scancode: ASCIICode
25.     0: None, 1: u'ESC', 2: u'\0', 3: u'2', 4: u'3', 5: u'4', 6: u'5', 7: u'6', 8: u'7', 9: u'8',
30.     50: u'M', 51: u'\0', 52: u'\0', 53: u'\0', 54: u'RSHIFT', 56: u'LALT', 100: u'RALT'
31. }
32.
```

The path of the event from the Barcode Scanner has to be set. First, the “event” number for that is needed, by using the “lsinput” command in the linux terminal. As in this thesis the “event0” is refer to the event from the Barcode Scanner. The default value of the barcode should be empty as well. The example of the defined valued as shown on line 33 and 34 of the Listing 5.3.

Listing 5.3: Setting the input path of barcode scanner

```python
33. dev = InputDevice('/dev/input/event0')
34. barcode = ""
35.
```

After that, it uses “while” loop to check each time of the event from the Barcode Scanner that needs to be converted into the defined key as in the “keys” array then it checks the length of the barcode if it equals to six and then the function returns the barcode value to the caller function.

Listing 5.4: Reading the data from barcode scanner

```python
36. while True:
37.     #Using the select() call to wait until there are events on dev
38.     r,w,x = select([dev], [], [])
39.     #Loop for the each event for the barcode reader
40.     for event in dev.read():
41.         if event.type == 1 and event.value == 1:
42.             barcode += (keys[event.code])
43.     #Check the length of the barcode then return the value to the caller
44.     if (len(barcode) == 6):
45.         return barcode
46.
```
In summary, the Barcode Scanner scans the barcode of the traffic sign and then the value of the barcode data is read from the input device event of the Raspberry Pi board. After that, the python program is created to detect and get the data from the event of the Barcode Scanner and send the data to the caller function.

5.2 Getting GPS location

To do this, the USB GPS Module is plugged into the Raspberry Pi board and then it is shown as the “ttyUSBXX” in the “/dev/” directory. It can be checked whether the device is on the Raspberry Pi board by using the “lsusb” command in the linux terminal. Then the GPS Daemon, gpsd, has to be installed into the Raspberry Pi board by using the apt-get command, “sudo apt-get install gpsd gpsd-clients python-gps”. After the successfully installed, the gps daemon is pointed to the right path of the USB GPS module by running this command in the linux terminal “sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock”.

The next thing to do is to write the programming to get the latitude and longitude from the gpsd by using the Python programming language. The required libraries are also needed to be imported first as shown on line 13 of the Listing 5.5.

```
11. #Import required library for the GPS
12. import gps
13.
14.
15. #Listen on port 2947 (gpsd) of localhost
16. session = gps.gps("localhost", "2947")
17. session.stream(gps.WATCH_ENABLE | gps.WATCH_NEWSTYLE)
18.
```

Listing 5.5: Importing a library and setting port for gps module

Then the session is set to stream the data from the gps daemon on the localhost port, 2947 as on line 16 and 17 of the Listing 5.5. After that, the function getGPS() is used to get the GPS data from the USB GPS Module. It uses the session to read the data from the device and check whether it has the latitude and longitude as shown on line 47 to 59 of the Listing 5.6. Finally, the function returns the value of the latitude and longitude to the caller function.

```
47. def getGPS():
48.     while True:
49.         try:
50.             report = session.next()
51.             #Wait for a 'TPV' report and display the current time
52.             if report['class'] == 'TPV':
53.                 #Check for the latitude and store it as lat
54.                     if hasattr(report, 'latitude'):
55.                         lat = report.latitude
56.                 #Check for the longitude and store it as lon
57.                     if hasattr(report, 'longitude'):
58.                         lon = report.longitude
59.             return lat,lon
```

Listing 5.6: Reading the data from gps module
In summary, the USB GPS module is read by the GPS Daemon, gpsd, and the python program is needed to read the session of the localhost port to get the latitude and longitude out of the raw gps data, NMEA format. Then it returns the value of the latitude and longitude to the caller function.

### 5.3 Transmitting data through CAN-Bus

The CAN-Bus is not included in the Raspberry Pi 2 B board by default and the PICAN-Bus shield is needed to be installed first. The PICAN-Bus is from the SK Pang Electronics Ltd. So the proper installations of all the hardware and the software for the PICAN-Bus shield have to be done by following the instruction as shown in the official site at [39]. To be able to use the CAN-Bus in the Python, the Python-CAN library is also needed to be installed because it is not included in the Python by default. To install this library, follow the instruction in the official site at [44].

After everything has done and installed, the next thing is to write the python programming to send the data to the CAN-Bus. The following required libraries are needed to be imported as on line 2 to 6 of the Listing 5.7 and the defined bus type and channel as shown on line 19 and 20 of the Listing 5.7.

```python
1. #Import required library for the CAN-Bus
2. import can
3. import re
4. from random import randint
5. from __future__ import print_function
6. import time
7.
19. bustype = 'socketcan_ctypes'
20. channel = 'can0'
```

Listing 5.7: Importing required libraries, setting bus type and channel for the 1<sup>st</sup> CAN-Bus shield

After that, the `getData()` function is used to format the barcode identity number into the hexadecimal number and add one zero in the front if the length is odd and then converts it into the array string as shown on line 68 to 79 of the Listing 5.8.
The `sendCAN()` function is used for sending the data through CAN-Bus. The timestamp has to be included in each data frame in order to determine the data that comes in the different data frame. Three message data frames are needed for the traffic sign data; the first one is for the barcode identity number which has the frame identity number is 0x7c7 in hexadecimal or 1991 in decimal, the second one is for the latitude which has the frame identity number is 0x7c8 in hexadecimal or 1992 in decimal, and the third one is for the longitude which has the frame identity number is 0x7c9 in hexadecimal or 1993 in decimal. These three data frames have to be the same timestamp.

```
def get_data(value):
    # Convert to the integer format
    value = int(value)
    # Convert to the Hexadecimal format, 10001 -> 186A1
    encoded = format(value, 'X')
    # Get the length of the data
    length = len(encoded)
    # Add a '0', zero, in front of the data if the length is odd, 186A1 -> 0186A1
    encoded = encoded.zfill(length + length % 2)
    # Use Regular Expression to substring into 2 characters, 0186A1 -> ['01', '86', 'A1']
    get_encoded = re.findall(r'[1-9][0-9]|0[1-8]', encoded, re.DOTALL)
    return get_encoded
```

Listing 5.8: Formatting the data for sending the CAN-Bus data frame

```
def sendCAN(barcodeId, lat, lon):
    # Define the sequence number by using the timestamp
    timestamp1 = int(time.time())
    # Format the barcode value from the get_data function
    barcodeHex = get_data(barcodeId)
    # Convert to the Hex string format to the dec int format
    barcodeHex = [int(x, 16) for x in barcodeHex]
    # Convert to the bytearray
    barcodeHex = bytearray(barcodeHex)
```

Listing 5.9: Verifying the data before sending the data through CAN-Bus

The “`can.interface.Bus`” uses the defined bus type and channel and the “`can.Message`” includes the timestamp, arbitration_id or frame identity number, data, and extended_id as shown on line 109 to 123 of the Listing 5.10. Finally, calling the function “`bus.send(msg1)`” by passing the message to send the data through CAN-Bus. To check the return value if it returns “true” then it successfully sent the data and “false” means it unsuccessfully sent the data as shown on line 126 of the Listing 5.10.
In summary, the Barcode ID, Latitude and Longitude are sent to the `sendCAN()` function as the parameters and after that it converts the data into the sending format of the Python-CAN library format. Then it sends the data as the CAN-Bus data frame into the CAN-Bus system of the car.

### 5.4 Receiving data from CAN-Bus

The data that has been sent from the first Raspberry Pi board is on the CAN-Bus system. The second Raspberry Pi board then uses the PICAN Shield to receive the data from the CAN-Bus system as same as in the first one. So, the instruction how to install PICAN Shield and Python-CAN are needed by following on the official sites at [39] and [44], respectively. After that a program to get this data is required. The programming code that is used for getting the data is in the filename, `receive_send_osm.py`, it is written in the python programming language. As the sending data, the receiving data also needs following required libraries to be imported as shown on line 2 to 6 of the Listing 5.11 and the defined the bus type and channel as shown on line 12 and 13 of the Listing 5.11.

```python
#Import required library for the CAN-Bus
import can
import re
from random import randint
from _future__import print_function
import time

bustype = 'socketcan_ctypes'
channel = 'can0'
```

Listing 5.11: Importing required libraries, setting bus type and channel for the 2nd CAN-Bus shield
The `receiveCAN()` function is the function to receive the data from the CAN-Bus. It has to define the channel and the bustype into the “`can.interface.Bus`”. The default values have to be set for the data as the data array as the variables: “`barcodeData`” is for the Barcode ID, “`latData`” is for the latitude for the GPS location and “`lonData`” is for the longitude of the GPS location as shown on line 15 to 21 of the Listing 5.12.

```
15. def receiveCAN():
16.     bus = can.interface.Bus(channel=channel, bustype=bustype)
17.     # Define the data array for barcode, latitude, longitude
18.     # the first array index for the timestamp and the second one for the data
19.     barcodeData = ["","]
20.     latData  = [","]
21.     lonData = [","]
```

Listing 5.12: Setting the variable for receiving the CAN-Bus data frame

The data from the CAN-Bus is received by “`bus.recv()`” of the Python-CAN library. It stores this data into the “`msg`” variable. After the data is received, it checks the data frame id or the arbitration id because each data frame has the different data frame identity. The 0x7c7 in hexadecimal or 1991 in decimal represents the data frame identity for the Barcode ID data frame. The 0x7c8 in hexadecimal or 1992 in decimal represents the data frame identity for the latitude data frame. The 0x7c9 in hexadecimal or 1993 in decimal represents the data frame identity for the longitude data frame. To check this, it uses the simple “`if`” function to check as shown on line 27 of the Listing 5.13.

```
24.    # Receive the data from the CAN
25.    msg = bus.recv()
26.    # Check for the Barcode ID
27.    if (msg.arbitration_id==1991):
28.        # Check whether it is on the same data, meaning same timestamp
29.        if (barcodeData[0]==int(msg.timestamp)):
30.            # If not, store the new timestamp in the first index array
31.            barcodeData[0] = int(msg.timestamp)
32.            # Then empty the data in the second index array
33.            barcodeData[1] = ""
34.        # Get the data from the CAN-Bus
35.        for i in range(len(msg.data)):
36.            # Convert the data to the Hex format
37.            decoded = format(msg.data[i],"x")
38.            # Check the length
39.            length = len(decoded)
40.            # Add a '0', zero, in front of the data if the length is odd, 186A1 -> 0186A1
41.            decoded = decoded.zfill(length+length%2)
42.            # Store the value in the second index array
43.            barcodeData[1] = barcodeData[1]+decoded
```

Listing 5.13: Receiving the CAN-Bus data frame

After that, it checks the timestamp of the data frame whether it is the same as previously stored on the “`barcodeData[0]`” variable or not. If it is not the same, that means it gets the new data and it has to set the data of the Barcode ID to be empty in the “`barcodeData[1]`”
variable as shown in Line 29 to 33. Then it checks the length of the data frame message and uses the “for” loop function to get all the data and stores into the “barcodeData[1]” variable as shown in Line 35 to 43. The same method applies the latitude and longitude data frame as well but the data frame identity is the 0x7c8 in hexadecimal or 1992 in decimal and the 0x7c9 in hexadecimal or 1993 in decimal, respectively.

In summary, it gets the Barcode ID, Latitude and Longitude data on the CAN-Bus from the receiveCAN() function by looking at the data frame identity or the attribution identity on the CAN-Bus system. Then it checks whether the data frame is the same data set or not by comparing the timestamp on the data frame.

5.5 Sending data to web server over WiFi

The USB WiFi dongle is connected into the second Raspberry Pi 2 B board and connected to the internet via WiFi hotspot. To send the data to the web server, it needs to import the required library into the filename, receive_send_osm.py, as shown in Line 9 and 10. It is the same file as the receiving the data from the CAN-Bus. But this part explains more details about how to send the data to the web server after the required data has been received.

```
 8. # Import the required library for sending the data over HTTP
 9. import urllib
10. import urllib2
11. 
```

Listing 5.14: Importing required libraries for sending the data over URL

So, it gets all the data, Barcode ID, latitude and longitude from the previous step. However, it has to check that these data have the same timestamp by using the “if” function as shown on line 68 of the Listing 5.15. Then it checks again that the timestamp is not empty as shown on line 70 of the Listing 5.15. After that, it uses the “data” as the dictionary variable to store the data into the parameters of the HTTP GET method:

- ‘barcode_id’ is for the Barcode ID
- ‘latitude’ is for the latitude
- ‘longitude’ is for the longitude
- ‘isSubmit’ is represent that this call is the sending data to the web server

As shown on line 72 to 76 of the Listing 5.15 and encode the data by using the “urllib.urlencode(data)” function as shown on line 77 of the Listing 5.15. In this thesis, this domain, http://car2x.thanaset.work, is created and it is hosted on the outside web server. In order to send the data to the web server, this URL: http://car2x.thanaset.work/insertdata.php, has to be set on the path as shown on line 78 of the Listing 5.15 and it explains how to write the programming code for receiving the data in the later step.
In summary, after it gets the data from the CAN-Bus system then it uses the HTTP GET method to send the data as the parameters to the specific URL that has been prepared for receiving the data on the web server side.

### 5.6 Receiving data on web server

On the web server side, to receive the data from the HTTP GET method, this file, `insertData.php`, is created and it is written in the PHP programming language. Firstly, it checks the required parameters, `barcode_id`, `latitude` and `longitude` that are not the empty value and checks whether the `isSubmit` parameter is equal to `yes` as shown on line 14 to 17 of the Listing 5.16. After that, it trims the string value by using the `pg_escape_string()` function and then it stores the value to the variables: "$barcode_id" is for the Barcode ID, "$latitude" is for the latitude and "$longitude" is for the longitude as shown on line 19 to 21 of the Listing 5.16.

```php
if($_GET['isSubmit']=='yes')
    && ($_GET['barcode_id'] != '')
    && ($_GET['latitude'] != '')
    && ($_GET['longitude'] != '')
{
    $barcode_id = pg_escape_string($_GET['barcode_id']);
    $latitude = pg_escape_string($_GET['latitude']);
    $longitude = pg_escape_string($_GET['longitude']);
}
```

Listing 5.16: Receiving the barcode and gps data over HTTP GET method

In summary, it uses the HTTP GET method to receive the data and temporary stores each data into the variables: "$barcode_id", "$latitude" and "$longitude".
5.7 Storing data on the database

The online database that is used in this thesis is mysql and it is connected to the database by using the php mysql which this is also written in the filename, insertData.php, the same as the receiving the data from the HTTP GET method but it does more after it gets the data by inserting the data to the online database. The most common server name for the mysql connection is the localhost as shown on line 2 of the Listing 5.17. Then the username and the password are needed to provide the access to online database as shown on line 3 and 4 of the Listing 5.17.

After that, it needs the name of the databases: “vfe00217_tfs_barcode” is the database for storing the Barcode Information which includes the Barcode ID, Name, Type, Description and the Icon Image and “vfe00217_tfs_point” is the database for storing the point of the traffic signs that has the information of the Barcode ID, Latitude and Longitude, as shown on line 5 and 6 of the Listing 5.17. Then the connection to the database using the mysqli() php function is needed to be set as shown on line 7 and 8 of the Listing 5.17.

Before it inserts the new traffic sign into the database, it has to check whether the same location already has the traffic sign information or not. To do this, it queries the database by passing the new latitude and longitude on the “point” database name. And it checks the count result of the query whether it has the data or not as shown on line 23 to 26 of the Listing 5.18.

If the data is not on the database yet, then it adds the new information about the traffic sign which includes the Barcode ID, Latitude and Longitude into the “point” database name as shown on line 28 and 29 of the Listing 5.18. And if the data has been previously stored on the database, it shows an error message as shown in Line 32 and 39, respectively.

---

Listing 5.17: Setting connection to connect the mysql database

```
1. <?php
2. $servername = "localhost";
3. $username = "xxxxx";
4. $password = "xxxxx";
5. $dbname = "vfe00217_tfs_barcode";
6. $dbname2 = "vfe00217_tfs_point";
7. $conn = new mysqli($servername, $username, $password, $dbname);
8. $conn2 = new mysqli($servername, $username, $password, $dbname2);
```
In summary, after it gets the data from the HTTP GET method, it has to check whether this data has already been stored in the database or not by using the latitude and longitude. Then it inserts the new data into the database, otherwise it shows an error.

### 5.8 Reading data for embedding the map

Embedding the traffic signs on the OpenStreetMap, it has to set the boundaries of the data set that it wants to show on the map. In this case, it uses the minimum and maximum of the latitude and longitude for the boundaries. To do this, it passes the values by using the same method, HTTP GET, as the previous step. So, it sets the parameters: ‘minla’ is for the minimum latitude, ‘maxla’ is for the maximum latitude, ‘minlo’ is for the minimum longitude and ‘maxlo’ is for the maximum longitude.

Listing 5.19: Receiving the latitude and longitude for querying the database

```
<?php
$min_la = $_GET['minla'];
$max_la = $_GET['maxla'];
$min_lo = $_GET['minlo'];
$max_lo = $_GET['maxlo'];
```

The next things that are needed to be set are servername, username, password, databases and the connection which are the same as the previous step as shown on line 7 to 14 of the Listing 5.19. The main thing is it has to query string for the data on the ‘point’ database name first.
by passing the minimum and maximum of latitude and longitude as the parameter as shown on line 15 of the Listing 5.20.

```
7. $servername = "localhost";
8. $username = "xxxxx";
9. $password = "xxxxx";
10. $dbname = "vfe00217_tfs_barcode";
11. $dbname2 = "vfe00217_tfs_point";
12. $conn = new mysqli($servername, $username, $password, $dbname);
13. $conn2 = new mysqli($servername, $username, $password, $dbname2);
14. $query = "SELECT * FROM point WHERE (latitude >= $min_lat AND latitude <= $max_lat) AND (longitude >= $min_lng AND longitude <= $max_lng)";
```

Listing 5.20: SQL syntax for querying the database

Then it queries and gets the results which it uses the “while” loop to get the data of each row. After it gets the data which includes the Barcode ID, Latitude and Longitude, it uses the Barcode ID to query the data of the barcode for the traffic sign on the “barcode” database name as shown on line 30 of the Listing 5.21. Then it gets the result of Barcode ID from the database and stores into the variables: “$name” is for the name of the traffic sign, “desc” is for the description of the traffic sign, “$latitude” is for the latitude of the traffic sign, “$longitude” is for the longitude of the traffic sign and “$iconurl” is for the name of the image of the traffic sign as shown on line 31 to 36 of the Listing 5.21. After that, it converts these data into the array name “$posts[]” as shown on line 37 of the Listing 5.21.

```
21. $result2 = $conn2->query($query);
22. $response = array();
23. $posts = array();
24. 
25. while($row2=$result2->fetch_assoc())
26. {
27.     if ($conn->connect_error) {
28.         die("Connection failed: " . $conn->connect_error);
29.     }
30.     $result = $conn->query("SELECT * FROM barcode WHERE id='$row2[barcode_id]'";
31.     $row=$result->fetch_assoc();
32.     $name = $row['name'];
33.     $desc = $row['desc'];
34.     $latitude = $row2['latitude'];
35.     $longitude = $row2['longitude'];
36.     $iconurl = $row['iconurl'];
37.     $posts[] = array('name' => $name, 'desc' => $desc, 'latitude' => $latitude, 'longitude' => $longitude, 'iconurl' => $iconurl);
```

Listing 5.21: Querying the traffic signs from the database
Finally, it uses the "json_encode()" function to convert the data into the JSON data format and returns the data to the caller function as shown on line 40 of the Listing 5.22.

```
39. $response = $posts;
40. echo json_encode($response);
41. ?>
```

Listing 5.22: Returning the data as the JSON data format

In summary, it uses the minimum and maximum of the latitude and longitude to be the boundaries of the query data set and then it queries the database of the point. Then it gets the Barcode ID which it uses this data to query the traffic sign information on the barcode database. After it gets all the results, it converts the data into the JSON data format and returns the results to the caller function.

### 5.9 Enrichment the OpenStreetMap

The data that has been returned from the `querymap.php` is in the JSON data format. After it gets this data, this JavaScript together with HTML programming languages file is needed to display the embedded OpenStreetMap and it is a filename, `viewmap.html`. The `initmap()` function is written in JavaScript, it uses the Tiles Server from the `tile.osm.org` site which is the OpenStreetMap site as the map server as shown on line 26 of the Listing 5.23, it sets the initial map view location to display it on the web browser by giving the latitude and longitude as shown on line 30, 32 and 34 of the Listing 5.23. Then this function calls the `askForPlots()` function to get the data of the traffic signs as shown in Line 36.

```
22. function initmap() {
23.   // set up the map
24.   map = new L.Map('map');
25.   // create the tile layer with correct attribution
26.   var osmUrl='http://{s}.tile.osm.org/{z}/{x}/{y}.png';
27.   var osmAttrib='Map data © <a href="http://openstreetmap.org">OpenStreetMap</a> contributors';
28.   var osm = new L.TileLayer(osmUrl, {minZoom: 8, maxZoom: 18, attribution: osmAttrib});
29.   //TU Chemnitz, Campus 1
30.   map.setView(new L.LatLng(50.839189, 12.928469),18);
31.   //TU Chemnitz, Campus 2
32.   //map.setView(new L.LatLng(50.815373, 12.930055),17);
33.   //Lautbusch
34.   //map.setView(new L.LatLng(51.467493, 14.134017),16);
35.   map.addLayer(osm);
36.   askForPlots();
37.   map.on('moveend', onMapMove);
38. }
```

Listing 5.23: Plotting the OpenStreetMap on the web browser
The `askForPlots()` function is the function that gets the data from `querymap.php` in the JSON data format. It requests the HTTP GET method to the `querymap.php` by passing the required parameters which are ‘minla’ is for the minimum latitude, ‘maxla’ is for the maximum latitude, ‘minlo’ is for the minimum longitude and ‘maxlo’ is for the maximum longitude as shown on line 53 of the Listing 5.24. The values of the latitude and longitude change every time that the mouse moves.

```javascript
48. function askForPlots() {
49.     // request the marker info with AJAX for the current bounds
50.     var bounds = map.getBounds();
51.     var minll = bounds.getSouthWest();
52.     var maxll = bounds.getNorthEast();
53.     var msg = querymap.php?minlo='+minll.lng+'&minla='+minll.lat+'&maxlo='+maxll.lng+'&maxla='+maxll.lat;
54.     ajaxRequest.onreadystatechange = stateChanged;
55.     ajaxRequest.open('GET', msg, true);
56.     ajaxRequest.send(null);
57. }
```

Listing 5.24: Requesting the plotting data from the HTTP GET method

This HTML file first calls the JavaScript to initialize the OpenStreetMap as shown on line 92 of the Listing 5.25. After that it displays the traffic signs on the map in web browser as shown on line 98 of the Listing 5.25.

```html
92. <body onLoad="javascript:initmap();"> 
93.     <center><div style="margin:50 auto;"/>
94.     <text style="font-size:30px;">Welcome to Car2X InDoor Lab</text>
95.     </br></br><a href="index.html">Home</a>/</br></br>
96.     <a href="insertdata.php">Insert New Traffic Sign</a> | <a href="viewmap.html">View Sachsen O
97.     SM Map</a>
98. </div></center>
99. <div id="map">
```

Listing 5.25: Showing the OpenStreetMap on the web browser

In summary, a webpage is created that is written in HTML and JavaScript to request the traffic signs location and its information by sending the boundaries of the area as minimum and maximum latitude and longitude as the HTML GET request. Then it returns the JSON data format and uses this data to plot the traffic sign on the OpenStreetMap.
5.10 Detecting the Traffic Light

In addition to the Barcode Scanner, the PI Camera is installed into the Raspberry Pi 2 B board to detect the traffic light on the camera whether it is safety to cross the traffic light or not. The idea is to use the camera to stream the video in the real-time and then uses the OpenCV to detect the traffic light and then sends the command of specific data frame identity to the CAN-Bus of the car. Then the car gets the information of the start and stop commands from the CAN-Bus.

To implement this, it uses the OpenCV to detect the traffic light by creating its own cascade classifier to detect the example of the traffic light. First, the video of the traffic lights are recorded and then it is extracted as the frame image. After that, the image is cropped into the small region of interest which is the lamp of the traffic light as the positive images. Then the positive sample images are generated by running `opencv_createsamples` function of OpenCV which it combines the positive images with the negative images. After that the XML file for the cascade classifier is created by using `opencv_traincascade` function of the OpenCV. This is done on the computer for faster processing and save time.

Next the python program is needed to run this OpenCV cascade classifier. This filename, `detect_traffic_light.py`, includes the three main parts: streaming the video from the camera by PI Camera, detecting the traffic light by OpenCV and sending the data by CAN-Bus. First, it has to import the required libraries and the defined bustype and channel for the CAN-Bus as shown on line 2 to 5 and 18 to 19 of the Listing 5.26.

```
1. #For CAN
2. import can
3. import re
4. from random import randint
5. from __future__ import print_function

17. #CANBus setup
18. bustype = 'socketcan_ctypes'
19. channel2 = 'can0'
```

Listing 5.26: Importing the required libraries and setting the bus type and channel for the CAN-Bus for detecting the traffic light
The next thing is it has to import the required libraries for the PI Camera as shown on line 14 and 15 of the Listing 5.27 in order to send the streaming video to the OpenCV. It sets the resolution to 640x480 pixels and the frame rate is 32 frames per second. Then it sends frame by frame as the image to detectTFL() function as the gray color as shown on line 110 to 122 of the Listing 5.28.

Listing 5.27: Importing the required libraries for PI Camera

```python
13. # For PI Camera
14. from picamera.array import PiRGBArray
15. from picamera import PiCamera
```

Listing 5.28: Reading the video from the PI Camera

```python
10. camera = PiCamera()
11. camera.resolution = (640, 480)
12. camera.framerate = 32
13. rawCapture = PiRGBArray(camera, size=(640, 480))
14. # Allow the camera to warmup
15. time.sleep(0.1)
16. for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
17.     frame = frame.array
18.     # Resize the frame, convert it to grayscale, and blur it
19.     frame = imutils.resize(frame, width=500)
20.     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
21.     # Detect the traffic light
22.     detectTFL(frame, gray)
```

After it gets the image from the PI Camera, it uses this image as the input for the OpenCV to detect the traffic light in this image whether it has the one as the defined object, traffic light, in the XML file. But it needs to import the required libraries for the OpenCV first as shown on line 8 to 11 of the Listing 5.29. Then the XML path for the cascade classifier has to be defined for detecting the traffic light that has been created before as shown on line 27 and 29 of the Listing 5.29.

Listing 5.29: Importing required libraries and setting path for the OpenCV

```python
7. # OpenCV
8. import cv2
9. import cv2.cv as cv
10. import sys
11. import numpy as np
26. # Cascade classifier path
27. cascPath = "detectTrafficLight.xml"
28. # Create the haar cascade
29. TFCascade = cv2.CascadeClassifier(cascPath)
```
The detectTFL() function is the main function for detecting the traffic light. It receives the image from the caller function and uses this image as the input for the OpenCV HAAR cascade classifier function to detect the traffic light as shown on line 46 to 54 of the Listing 5.30.

Before continue to further steps, the reference for the image of the fixed distance of the traffic light to the car is needed. It uses this data as the reference to calculate the distance of the traffic light while streaming the video from the camera. A image of the traffic light is taken in the distance of 36 inches, Actual_D, and it is measured the width of the traffic light as 4 inches, Actual_Width, and the width of the image as 33 pixels, Actual_PixelW. So, the focal length, sF, can be calculated by this equation below:

$$ sF = \frac{(Actual\_PixelW * Actual\_D)}{Actual\_Width} $$

As shown on line 32 to 35 of the Listing 5.31, this focal length equals $[(33 * 36)/4] = 297$. So, if it gets the new image of the traffic light of the pixel width of 40. The distance can be calculated by this equation below:

$$ Actual\_D = \frac{(Actual\_Width * sF)}{Actual\_PixelW} $$

Then the distance is $[(4*297)/40] = 29.7$ inches.
The car should not be stopped immediately after the camera detects the red light to prevent the accident or the error. So, the amount of the distance from the car to the traffic length that the car should be stopped in the InDoor Lab to 50 inches is set to test on the YellowCar. The data frames identity for the CAN-Bus to send the GO or STOP to the YellowCar is 0x32 in hexadecimal number. To send the GO signal, it sends the ‘01’ to the CAN-Bus, or sends ‘00’ for the STOP signal as shown on line 81, 82 and 94, 95 of the Listing 5.32.

```
81. bus = can.interface.Bus(channel=channel2, bustype=bustype)
82. msg1 = can.Message(arbitration_id=0x32, data=[01], extended_id=False)
94. bus = can.interface.Bus(channel=channel2, bustype=bustype)
95. msg1 = can.Message(arbitration_id=0x32, data=[00], extended_id=False)
```

Listing 5.32: Sending the GO and STOP signals to the car

In summary, the car receives the image of the traffic light from the streaming video of the PI camera which is connected to the Raspberry Pi board. Then the Raspberry Pi board uses the OpenCV in the Python to process this image to detect the traffic light. After that, it checks whether the traffic light is for STOP signal (red or yellow light) or GO signal (green light) then it sends the data frame command to the CAN-Bus and the car stops or goes at the specific distance.

To conclude this chapter:

- It explains the implementation of the architecture mostly in the coding which focus on the main things in the code that it is used during the thesis.
- It describes how to scan and send the traffic sign barcode by using the USB Barcode Scanner, how to get the GPS location from the USB GPS dongle, and then how to send these data to the CAN-Bus system.
- It also explains how to receive the data from the CAN-Bus system and send the data to the web server by using WiFi dongle.
- It gives more details about how to receive the data and store the data on the web server.
- It shows how to read the data from the database and enrichment the OpenStreetMap on the web browser.
- It also includes how to detect the traffic light by using PI camera.

The next chapter shows the result part of the proposed work.
Chapter 6

Result

This chapter shows the result from the implementation chapter. It explains and displays some of the screenshots from this architecture. It is divided into two parts, the enrichment of OpenStreetMap and the traffic sign detection.

6.1 Scanning Traffic Sign Barcode and Enrichment OpenStreetMap

First of all, the information of the traffic sign needs to be stored in the database. The main important key is the barcode identification which refers to the traffic sign. In the Table 6.1 shows the example of the traffic sign that available for the testing in this thesis.

<table>
<thead>
<tr>
<th>Barcode ID</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Image Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>100001</td>
<td>Speed Limit 30km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 30km per hour</td>
<td>30</td>
</tr>
<tr>
<td>100002</td>
<td>Speed limit 40km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 40km per hour</td>
<td>40</td>
</tr>
<tr>
<td>100003</td>
<td>Speed limit 20km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 20km per hour</td>
<td>20</td>
</tr>
<tr>
<td>100004</td>
<td>Speed limit 60km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 60km per hour</td>
<td>60</td>
</tr>
<tr>
<td>100005</td>
<td>Speed limit 50km</td>
<td>Speed limit signs</td>
<td>The maximum speed is 50km per hour</td>
<td>50</td>
</tr>
<tr>
<td>100006</td>
<td>Give way</td>
<td>Regulatory signs</td>
<td>Give way to traffic on the major road</td>
<td>12</td>
</tr>
<tr>
<td>100007</td>
<td>No entry</td>
<td>Regulatory signs</td>
<td>All vehicles prohibited except pedal cycles being pushed by pedestrians</td>
<td>12</td>
</tr>
<tr>
<td>100008</td>
<td>Road narrows on left ahead</td>
<td>Warning signs</td>
<td>Road narrows on left ahead</td>
<td>12</td>
</tr>
<tr>
<td>100009</td>
<td>Zebra crossing ahead</td>
<td>Warning signs</td>
<td>Zebra crossing ahead</td>
<td>12</td>
</tr>
<tr>
<td>100010</td>
<td>Road hump</td>
<td>Warning signs</td>
<td>Road hump or series of road humps ahead</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 6.1: The available testing traffic sign on the database in the thesis
After that, the barcode that represents for each traffic sign is generated by using the one-dimensional barcode. The barcode that are used in this thesis is shown in the Figure 6.1 below.

![Barcode of Traffic Signs](image)

**Figure 6.1:** The barcode of the traffic signs that are used in this thesis

However, this information must be stored in the database which is the online mySQL database for this thesis. It is available for the querying where internet is connected to, thus, the access account must be granted first. The testing database includes the major information for the traffic sign such as identification number, name, type, description and image fields. But it can be extended to have more information such as country, date and more fields in the future if it is required.

The easy way to access to the mySQL database is using phpMyadmin which is the tools for administration of mySQL on the web browser. Inserting the new information of the traffic sign can be done manually by using phpMyadmin as same as deleting, editing and browsing the traffic sign. The Figure 6.1 shows the information of the traffic signs that are used in this thesis for testing.
After the database for the traffic sign is created and stored all the required information that is needed for the testing, the next database for the point of the traffic sign on the road is created. It represents the traffic sign by using the latitude, longitude and barcode id which is the traffic sign identification number. Thus, the data that is sent from the car contains this information and is stored in this database.

When the barcode is scanned from the traffic sign on the road, it sends the data through the CAN-Bus of the car system and the car sends the data which includes the latitude and longitude to the online database by using the WiFi dongle and the web server receives the data via HTTP GET method from the car. Then it stores this data into the database. The data of this traffic sign is linked to its information by using the barcode identification to query on the traffic sign information database which is the one before. The screenshot for the point of traffic sign is shown in the Figure 6.3.

Figure 6.2: The traffic sign information that is stored in the mySQL database
The all required information is stored in the online database and the next thing to do is using this information to enrichment the traffic sign on the OpenStreetMap. The data that queried from the database is displayed in the JSON data format and it contains the name, description, latitude, longitude and the image name of the traffic signs. This data is from the two databases, “traffic sign information” and “point of traffic sign” databases, and combines it together.

However, the boundaries of the area have to be defined and sent to the web browser such as the latitude between 49 and 54 and the longitude between 9 and 14, so the query request URL looks like this:


The Figure 6.4 displays the example of two traffic signs information in the nested JSON data format.
Finally, the traffic signs are embedded on the OpenStreetMap in the example area as displayed in the Figure 6.5 below.

![Figure 6.5: The example of OpenStreetMap with traffic signs on the web browser](image-url)
6.2 Detecting Traffic Light

Moreover in this thesis, the PI camera is attached to the Raspberry Pi board and used to stream the video of the traffic light to detect the red, green, yellow lights and sends the GO or STOP signal to the YellowCar. The Figure 6.6 shows the PI camera is attached to the Raspberry Pi board.

![Figure 6.6: The PI camera is attached to the Raspberry Pi board](image)

When the camera streams the video of traffic light, it sends the data to the Raspberry Pi board and then it uses the OpenCV to detect the traffic whether it contains the traffic light in the image of the video. Then it sends the command data to the YellowCar through the CAN-Bus to whether GO or STOP.

Since the YellowCar uses the CAN-Bus data frame identification number “0x32” to receive the GO or STOP command of the running wheels. If the green light is detected, the Raspberry Pi board sends the GO command contains the number one, ‘1’, to the CAN-Bus, and otherwise it sends the STOP command contains the number zero, ‘0’, when it detects the yellow or red lights and the specific distance.

Finally, this detecting traffic light camera is working as designed and the illustrated of the example how the Raspberry Pi and the YellowCar are connected is shown the Figure 6.7.
In conclusion:

- It shows the example of the traffic sign information that is stored on the database.
- It gives the example of the barcode of traffic signs that are used in the thesis.
- It illustrates the screenshot of the databases for traffic sign and point.
- It presents the JSON data formation when querying the data.
- It displays the example of the OpenStreetMap with the traffic signs.
- It also gives the result and detail about traffic light detection.

However, some improvement should be done in the future and it includes in the conclusion chapter.
Chapter 7

Conclusion

In this thesis, two scenario approaches have been shown to provide more information to the driver. The first one is using the OpenStreetMap to display the traffic sign from the local observation of the Car2X. It uses the barcode to represent the traffic sign instead of using the image of the traffic sign itself because the different kind of the traffic signs and hard to detect. In this thesis, the barcode scanner is chosen for the testing since it is easy and provides more realizable reading. The barcode can be scanned really fine in the distance of the barcode scanner. In contrast, the GPS module cannot provide the realizable and more accurate location for the traffic sign because it has to be in the open clear skies. Thus, the manual location is used for the testing period. The transmitting data in the CAN-Bus works perfectly fine and it can be sent and received from the Raspberry Pi boards.

The WiFi that is used in the thesis is IEEE 802.11 a/b/g/n standard instead of using the IEEE 802.11p standard which is in the early stage of development for the car communication and because of this thesis requires only the transmitting data to the web server that is connected to the internet and not for the car to car communications. Actually, this architecture works well whether the car sends the data to the web server using any kind of wireless technologies such as 2G, 3G, 4G or WiMax. The web server is hosted on the commercial one which is ready to use such as mySQL database, PHP programming language are already installed. However, the creating of the PHP web pages need to be done manually as well as the database tables. This process is working and running very well which it can show the OpenStreetMap of the traffic sign on the web browser perfectly without any problem.

In conclusion of the first scenario approach, the OpenStreetMap can be displayed in the web browser with the traffic sign as the example for this thesis is on this URL, http://car2x.thanaset.wok, the information of the traffic sign location can help to provide more information to the driver better than no information displays on it. This information can also help to reduce the accident in the future.

The second scenario approach, the PI camera is chosen to use for the detecting of the traffic light and then sends the command signal to the car itself. It streams the video of the traffic light and uses the OpenCV to detect whether it is green, red or yellow light and then sends the command to the CAN-Bus. YellowCar is chosen to demonstrate of this architecture in the testing. The result from this architecture is the YellowCar can be stopped and started depends whether which light is detected.
To improve the better of the architecture of this thesis, one of the ways is to include the gps position on the barcode to get more accurate location of the traffic sign. And instead of using the barcode scanner, it can be replaced by using the PI camera and then uses the OpenCV to detect the barcode from the image of the video frame. To combine it together, when there is camera module, it populates information on the shared maps which provides information to others and it provides more flexible method to read the traffic sign from the road side in the reality no matter the traffic sign is too far or too high from the car by using the high resolution camera. On the other hand, it requires more effort and cost to put the gps position on every traffic signs.

To summary the work of this thesis, it provides that local observation of the traffic sign on the road side and enriches the traffic sign to display on the OpenStreetMap. It helps the driver to lookup the desired area of travel to understand the local traffic sign and this of course helps to reduce the accident on the road in the future and provides the better online map that many user can access it.
Bibliography


[40] “GPS Module”, Available at: https://www.ghielectronics.com/catalog/product/325