Acknowledgement

Listed below are the names of the people who provided us with significant help in developing our graduation project in addition to technical Sponsorship by Valeo. To all we extend our sincere thanks.

Dr. Ahmed Hussein
Eng. Abdelrahman Aboutaleb
Eng. Mohamed Nagy, SW Manager at Valeo

A special thanks to Dr. Ahmed Hussein, it has been a great pleasure and honor being our supervisor. You were continuously encouraging us, even before we decided to work on this project.
Finally, though only our names appear on the cover of this thesis, but many have - knowingly or unknowingly – contributed to its production, and for that we are extremely thankful,
Ali, Areeg, Amany, Gehad, Mahmoud

Success is the result of perfection, hard work, learning from failure, loyalty and persistence.

By Colin Powell

So, We Proudly Share Our Project Results With You.

Scan QR Code!

Table of Contents

Table of Figures

CHAPTER 1: INTRODUCTION

Abstract

Most accidents occur because the driver can only see, with the sensors and the current electronic driver aids, as far as the vehicles directly in front of him/her, behind him/her, or on either side. A competent driver might notice more than one car ahead or behind, notice the signal lights and act preemptively to prevent any sudden actions or accidents. However, sometimes this isn't enough. If any sudden action was taken faster than the driver's reaction such as a vehicle coming in a very high speed next to him/her or realizing there's a huge obstacle when the car is too near to take the needed precautions, this will lead to dangerous consequences. As a result, there has to be another solution that car itself will notice the sudden changes to take precautions if the driver couldn't. Also, there has to be a solution to make the driver able to see more than 2 vehicles ahead or behind and to alert the driver of the changes that happen a little further than his/her sight so that the driver can act smoothly and preemptively. Car accidents have risen to 14500 accident in 2015. A total of 63.3 percent of car accidents were caused by humans. A total of 6203 were killed and 19325 were injured due to such accidents in 2015.
One of the technological advances that could solve this problem is vehicle to vehicle communication. This report will include more information about V2V communication, its benefits and its market nowadays. Then, the actual implementation of our project along with the testing methods and results are furtherly explained. Finally, the lessons learned while working on our project as well as the next phases are discussed.

1.1 ABOUT V2V

Vehicle-to-Vehicle Communications: Summary of what have been done in this area in the last 10 years

Using vehicle-to-vehicle (V2V) communication, a vehicle can detect the position and movement of other vehicles up to a quarter of a kilometer away. In a real world where vehicles are equipped with a simple antenna, a computer chip and GPS (Global Positioning System) Technology, your car will know where the other vehicles are, additionally other vehicles will Know where you are too whether it is in blind spots, stopped ahead on the highway but hidden from view, around a blind corner or blocked by other vehicles. The vehicles can anticipate and react to changing driving situations and then instantly warn the drivers with emergency warning messages. If the driver doesn't respond to the alerts message, the vehicle can bring itself to a safe stop, avoiding a collision.

History of V2V communication

V2V communications research initially began under the Vehicle Infrastructure Integration Initiative in 2003, but its origins date back to the Automated Highway System (AHS) research of the 1990s, to show the evolution of v2v communication we choose some papers from different years, we will summarize them with their topics chronologically

Lane Change Warning System

In 2014 a paper was published that discussed:

This paper proposes a lane change warning system based on V2V communication which can be used for the real road driving. Simulation model is built to test the performance of the system. Two meaningful conclusions are got as below:

1- The lane change warning system can coordinate both lane change safety and driving comfort to realize effective warning. The lane change warning system can make a comprehensive analysis of the collision risk between the changing-lane vehicle and its four surrounding vehicles. Furthermore, driving style index is introduced to modify the warning timing to satisfy various drivers. 2- The analysis method of the minimum safe distance can precisely estimate the initial distance needed to safely complete lane change process. To make the minimum safe distance more practical and specific, minimum safe distance of rear vehicle in target lane is analyzed according to the goal of both collision avoidance and vehicle following safety, the counterpart of front vehicle in target lane
is analyzed to avoid emergency collision and counterparts of vehicles in original lane are analyzed to ensure vehicle following safety

Communication Channel Modeling

In 2011 a paper was published that discussed:

In this paper we describe the importance of accurately modelling communication channel for ITS applications. After a review of important channel parameters, we show how the V2V channel differs from the channel in other settings, and can exhibit severe fading and statistical non-stationarity. Both these features should be taken into account when modeling the channel. In this work, we explored the efficiency of the existing radio access networks which can be used for V2V communication. On comparing the performance of three networks, we can conclude that WLAN supports less Doppler in comparison to 3G and WiMAX. If we compare 3G and WiMAX we see that it supports Doppler with speed in similar range but then also 3G is supporting more velocity of receiver, Thus by changing the radio network in proper instant (vertical handoff), it is possible achieve ubiquitous communication using the existing network infrastructure.

In 2006 a paper was published that discussed:

In this article we described results from recent wideband channel measurement campaigns in cities and on highways, undertaken to characterize the wireless channel for V2V applications in the 5.2 GHz band. We first described the motivation behind this work, and the five types of settings that we used to portray the channels likely to be encountered in ITS applications. Steps used in creating channel models were discussed. Implementation issues regarding model complexity and fidelity trade-offs were addressed. A detailed tapped-delay line channel model for an urban worst case was provided, with amplitude statistics and example tap correlation values. As expected, the dispersive nature of the V2V channel model varies from one V2V region to another. There are some significant differences with terrestrial models, including the presence of worse than Rayleigh fading in some taps and correlation among taps. In addition, we also noted the necessity of implementing a multipath persistence process to realistically account for the dynamic behaviour of the V2V channel where both transmitter and receiver platforms can be mobile.

Collision Pre-Warning Algorithm

In 2008 a paper was published that discussed:

In this paper, we have discussed the importance and challenges of using v2v wireless communication for vehicle safety applications. A stringent EWM delay constraint is identified as the key metric for protocol design. An integrated rear-end avoidance
protocol is presented, which is based on 802.11 MAC and multihop broadcast. Simulation results from both single lane and multiple lane scenarios demonstrate that the EWM propagation delay in the proposed protocol satisfies the stringent delay requirements. With appropriate EWM broadcast power, more than 99% of vehicles are free of rear-end collisions, even in the dense multiple lane scenario plus the worst case visibility assumption.

In 2009 a paper was published that discussed:
Driving safety is one of the most important issues in car industry. There are some researches trying to provide collision warning systems to achieve the goal of more safety. However, some of them need to build infrastructure to reach their purposes. It may be expensive. In this paper, we have proposed a Collision Pre-Warning Algorithm
In 2009 another paper was published that focused on collision avoidance on highways with stringent propagation delay:
A stringent EWM delay constraint is identified as the key metric for protocol design. An integrated rear-end avoidance protocol is presented, which is based on 802.11 MAC and multihop broadcast. Simulation results from both single lane and multiple lane scenarios demonstrate that the EWM propagation delay in the proposed protocol satisfies the stringent delay requirements. With appropriate EWM size, more than 99% of vehicles are free of rear-end collisions, even in the dense multiple lane scenario plus the worst case visibility assumption.
In 2011 a paper was published that discussed:
Many researches have focused on active Driving Safety. Most of them proposed schemes or algorithms to warn drivers before the accident comes. In the ECWA, we can filter some vehicles that are far away from the host vehicle in order to save computing resource. Besides, we can set suitable warning distance by users themselves. If a user does not want to receive too many warning messages, the warning level can be set to low. ECWA is a low-cost collision warning scheme because it does not need infrastructure. It is because the collision calculation does not depend on the distance of a vehicle to an intersection. Driving safety is one of the most important issues in car industry. There are some researches trying to provide collision warning systems to achieve the goal of more safety. However, considering the cost, applicability, omni-directional protection, an ECWA is required. In this paper, we have proposed an ECWA based on V2V communication that is low cost and more easily implemented. ECWA utilizes RI packets that transmitted and received through the WAVE communication protocol. After the RI packet was received, ECWA utilizes location and vector
information to calculate whether the collision will happen or not. Besides, warning level can be set by drivers themselves, and the warning distance also can be set depending on the length of a car and GPS error. In our simulation, ECWA has high accurate rates at most scenarios, and some miss rates occur when the number of vehicles is increased. On the other hand, GPS error was considered and simulated in our experiments. According to the result, ECWA has the warning message rate almost higher than 80% when GPS error is less than 6 m. The warning message rate goes down to 40% when GPS error is not greater than 15 m because the warning distance (DCPA) is only set to 3 m. Otherwise, the warning message rate can be higher if we add GPS error in the warning distance calculation. Some future work is considered as follows. The proposed ECWA can be used for collision warning not only for two vehicles but also for other objects. Some radar systems that are equipped on vehicles provide the azimuth and acceleration information of target objects. If the related information of target objects can be obtained from the radar system, the collision calculation in ECWA is applied for avoiding colliding with these objects. That is, ECWA will warn to drivers correctly whether the objects are moving or not Therefore, collision warning in ECWA for any fixed or non-vehicular objects such as human, motorcycle, or obstacle is achievable
Monitoring Neighboring Vehicles for Safety
In 2011 a paper was published that discussed:

In this paper we introduced a new type of query calledCRNN queries which enable drivers to observe vehicles in the neighborhood and to avoid collisions. We developed a method for a vehicle to maintain the locations of neighboring vehicles via V2V communication. The method dynamically adjusts the location broadcast frequency using Greenshield's traffic flow model. Simulation and road test results showed that our method achieves a superior performance over its rival in terms of the bandwidth usage. Then we analyzed the computational cost of processing a CRNN query. In the on-line case it takes O(log2n) time to process a motion update; in the off-line case it takes O(nlogn) time. Thus on-line processing is more efficient than off-line processing. Finally we demonstrated that CRNN queries are applicable to several safety-related application scenarios of great interest. In the future, we plan to take the location uncertainty into account. Because the motion update is discrete, there is uncertainty when a vehicle predicts another vehicle's location. In this case, a CRNN query may ask for vehicles that are possibly the k nearest neighbors, or the vehicles that are definitely the k nearest neighbors. We will study CRNN queries under various query semantics and various location uncertainty models. Another research direction is how to make the processing more efficient by using appropriate spatio-temporal indexing structures.

Protocol for Cooperative Collision Warning

In 2004 a paper was published that discussed:

This paper proposes a Vehicular Collision Warning Communication (VCWC) protocol to improve road safety. In particular, it defines congestion control policies for emergency warning messages so that a low emergency warning message delivery delay can be achieved and large number of co-existing abnormal vehicles can be supported. It also introduces a method to eliminate redundant emergency warning messages, exploiting the natural chain effect of emergency events.

Short Range Communications

In 2005 a paper was published that discussed:

In this paper, we have presented our experiences with short range communications between vehicles and between vehicles and roadside stations for a realistic highway scenario. Similar studies should be conducted for a broad range of settings (e.g., varying packet size, different locations, etc.) to give a more comprehensive understanding. Another important area of work is to characterize and identify other factors (e.g., weather and vehicle traffic) that may impact the communication.

IP and MAC Protocols for Dedicated Short Range Communications (DSRC)

In 2004 a paper was published that discussed:

This paper presents four categories of DSRC applications: unicast R2V, unicast V2V, broadcast R2V, and broadcast V2V.

Each category of applications has unique IP and MAC requirements as illustrated by a message flow diagram. A major contribution of this paper is to identify a new feature, bridging with layer-3 forwarding, to extend the communication distance in a wireless environment. This feature is available in the Windows XP operating system and can be easily ported to OBD to support the V2V applications. In this paper, we consider each OBD operates in a single RF channel. In theory, an OBD could operate in multiple frequency channels and support more than one application simultaneously. The use of multiple RF channels in DSRC is another topic of ourfurther study and research.

CAN Gateway

In 2015 a paper was published that discussed:

In this paper, we propose a new CAN Gateway method for fast V2V Communication. Since the proposed method can reduce the number of CAN frames which have to be transmitted to request the information to

ECUs by using the partial network table, it can decrease the time to be required for the CAN frame transmission. For this
reason, it can be considered that the proposed method is suitable for V2V communications which are applied for delay-sensitive services.
Taking Advantage of V2V Communications for Traffic Management
In 2011 a paper was published that discussed:
With reference to a scenario where vehicles are equipped with devices collecting measurements and sending them to a remote control center through cellular networks, in this paper we discussed the advantage that vehicle to vehicle communications could provide. Numerical results, obtained through a simple analytical model, highlighted the significant reduction in data transmissions over the cellular networks that can be obtained even with a reduced number of equipped vehicles.

1.2 V2V MARKET

V2X communication ‘s market is growing every year due to the enhancement of technology use in vehicles. A lot of investment is done in this field nowadays. Middle Eastern countries are considered a great potential for this technology due to the increase in population as well as the focus of many automobile companies on regions such as the Middle East and Africa. The development of this technology by automotive manufacturers, chip manufacturers as well as technology and solution providers is accelerating.

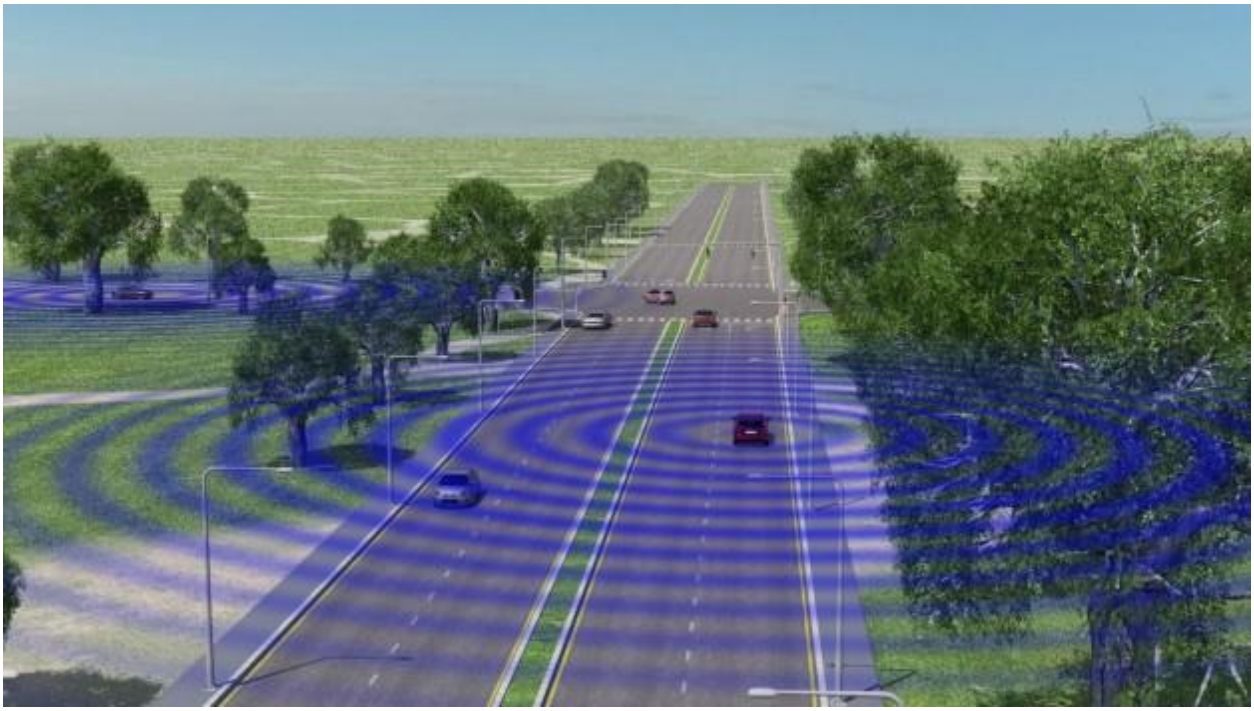


Figure 1 The range of signals sent by the car

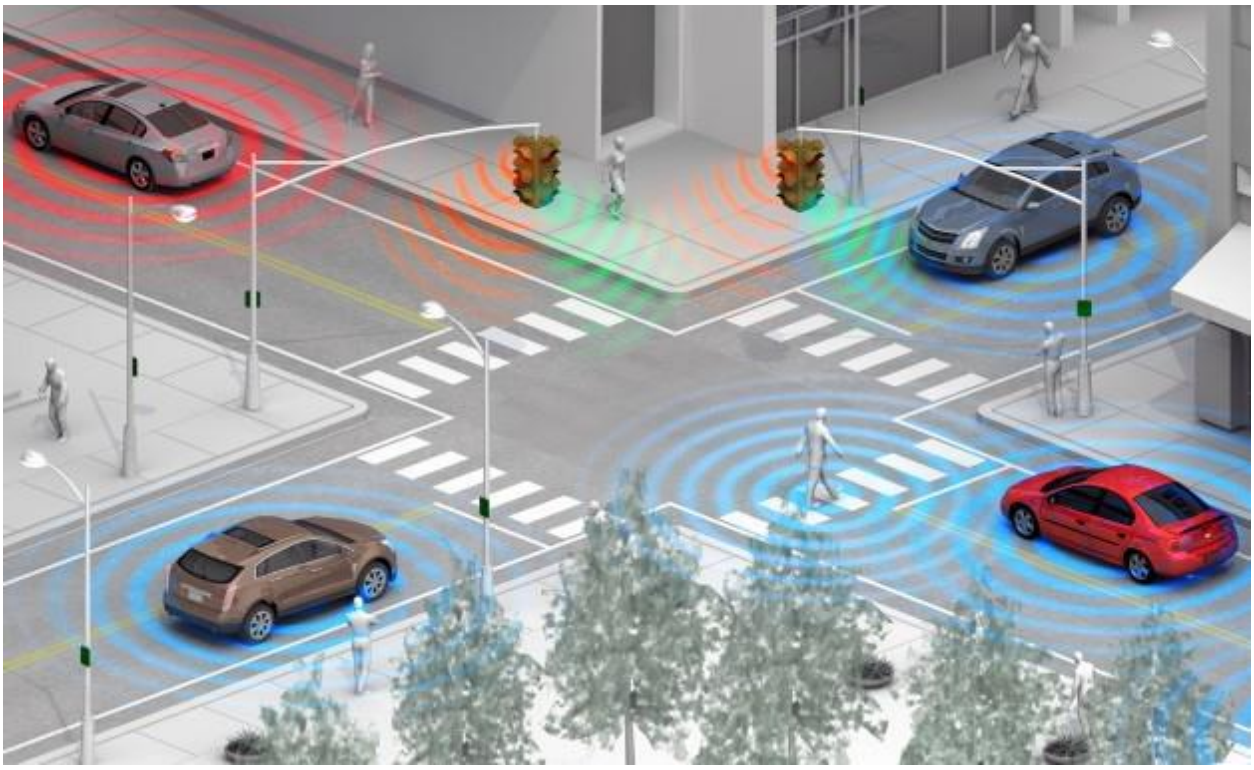


Figure 2 the future of V2X systems

1.3 V2V COMPETITION LANDSCAPE

There are many companies interested in this technology such as BMW, Audi, Daimler, Volvo, and Ford-Applink. Among the solution providers Etrans Systems, Qualcomm Technologies Inc., Cisco Systems Inc., Delphi Automotive PLC, Autotalks Ltd., Denso, Arada Systems, Kapsch Group and Savari Inc., are included in the vehicle to vehicle communication market.

1.4 STANDARDIZED V2V PROTOCOLS

Since, V2X requires devices and vehicles of different manufacturers communicate with each other, there has to be a standard that all companies and manufacturers will follow. That's why IEEE developed the 802.11p standard which explains the physical and mac layers of vehicular transceivers. That way, any other European or American standards developed, will have to be based on the lower-level IEEE 802.11p standard, to ensure the compatibility of different devices communicating with each other.

1.5 PROJECT OBJECTIVES

We wanted our project to fulfill the user's needs and expectations, thus we had some vision of these needs and planned its scope and our approach to satisfy it.

- Regulation of cars motion so that no car can not detect the other cars motion information in its prespecified range .
- Guaranteeing safety to people on the road.

1.6 PROJECT DESCRIPTION

Using vehicle-to-vehicle (V2V) communication, a vehicle can detect the position and movement of other vehicles up to a quarter of a kilometer away. In area world where vehicles are equipped with a simple antenna, a computer chip and GPS (Global Positioning System) Technology, your car will know where the other vehicles are, additionally other vehicles will Know where you are too whether it is in blind spots, stopped ahead on the highway but hidden from view, around a blind corner or blocked by other vehicles. The vehicles can anticipate and react to changing driving situations and then instantly warn the drivers with emergency warning messages. If the driver doesn't respond to the alerts message, the vehicle can bring itself to a safe stop, avoiding a collision.

So, our project elaborates Implementation of a real time vehicle to vehicle communication system over a local WIFI network between cars where we broadcast car data and state to avoid potential collisions. Specifically handling 4 main applications

1- EMERGENCY ELECTRONIC BRAKE LIGHTS (EEBL)

when there is a hard braking vehicle in the pass ahead , three vehicles are traveling in the same lane you are driving the last vehicle and you can't see the first vehicle because it's been blocked by the car in front of you ,unexpectedly the vehicle slams on its brakes ,duo to v2v communication the car can produce a waning of the braking car ahead

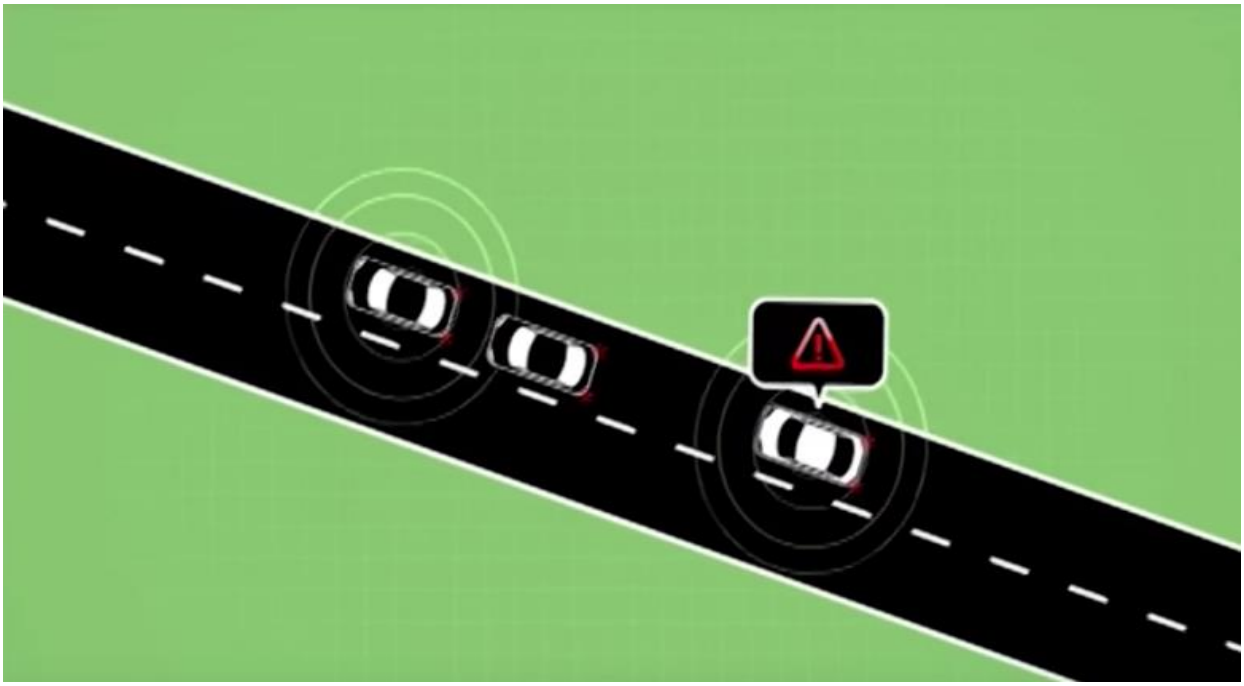


Figure 3 EEBL use case

2- BLIND SPOT WARNING(BSW)

when there is a car that may not be visible to the car driver, because of v2v communication a warning message is issued to make you aware of the presence of this vehicle , should you attempt to lane change this warning message will tell you that it's not safe to lane change

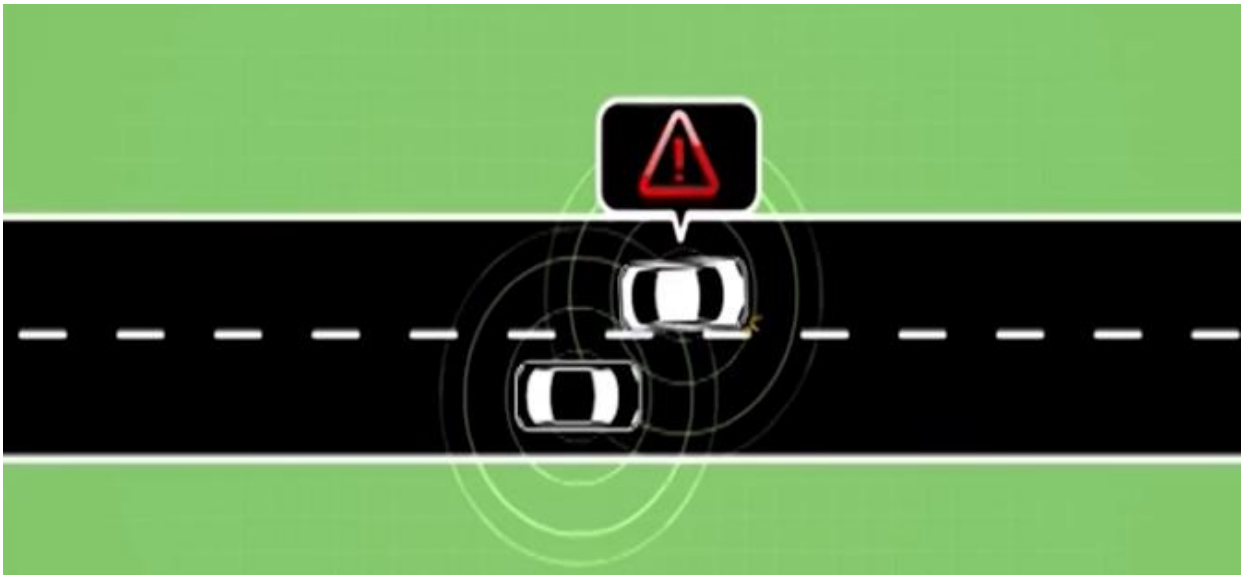


Figure 4 BSW use case

3- DO NOT PASS WARNING(DNPW)

a safety application is intended to let the driver know that it's not safe to attempt to pass a slower moving vehicle because of incoming traffic in the passing zone, if a vehicle is detected in the passing done a warning message is provided letting you know that the passing situation is potentially unsafe

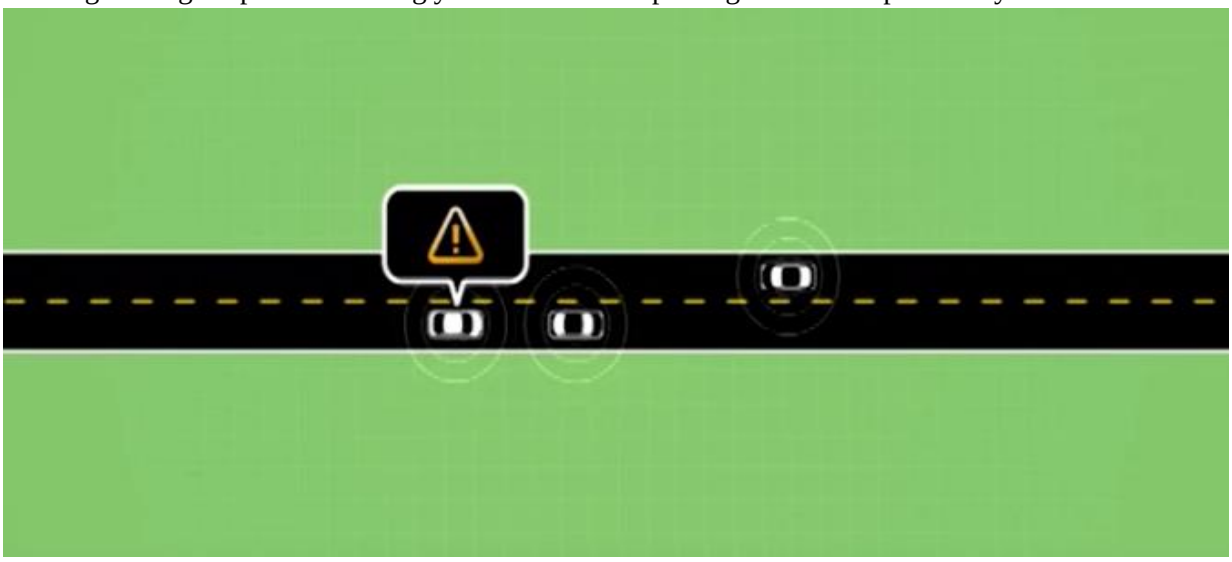


Figure 5 DNPW use case

1.7 DEVELOPMENT PLAN

At the begging of any project we need to consider a development plan contains well defined phases and deadlines to assure utilization of time during the project as well as Global design document which includes deep technical details about the whole project.

1.7.1 Project Time line

This part includes our project deliverables, durations, deadlines, milestones of our project.

Project timeline was developed on Microsoft project, a tool used by professional project managers to develop timelines and other matters for moderate size projects. Full project management plan attached in references.

In this project we follow agile methodology in our project development process to assure best results in the shortest time.

The timeline is divided into 4 major phases each phase is divided into number of sprints and each sprint is divided into smaller and smaller tasks.

1. Phase 1 includes submission of a laptop communicating with stationary car

Task Name	Duration	Start	Finish	Resource Names	Notes
Graduation Project	**150 days?**	**Mon 10/8/18**	**Sun 5/5/19**		
Phase 1	**33 days**	**Mon 10/8/18**	**Wed 11/21/1**		**Submission of Laptop communicating with a stationary car**
Sprint 1	**18 days**	**Mon 10/8/18**	**Thu 11/1/18**		
ARM Revision	0 days	Mon 10/8/18	Mon 10/8/18	shared	
Wi Fi Algorithm Optimization	6 days	Mon 10/8/18	Fri 10/12/18	Aly,Amany,Gehad	
GPS Configuration	6 days	Mon 10/8/18	Fri 10/12/18	Atrby,Areeg	on labtob
Coding WiFi	14 days	Mon 10/15/18	Wed 10/31/18	Aly,Amany,Gehad	on arm, building all needed layer from mcal to service layer
Coding GPS	14 days	Mon 10/15/18	Wed 10/31/18	Atrby,Areeg	on arm, building all needed layer from mcal to service layer
Complete application Layer section in GDD	0 days	Thu 11/1/18	Thu 11/1/18	shared	https://docs.google.com/document/d/1BiYsrhvc-48nnKMkmJD
Midterm Exams	5 days	Sat 11/10/18	Wed 11/14/18		
Sprint 2	**3 days**	**Sun 11/18/18**	**Wed 11/21/1**		
LCD interfacing	5 days	Sun 11/18/18	Wed 11/21/18	Aly	building up the interface between arm and LCD
Integration and development of simple Algorithm	5 days	Sun 11/18/18	Wed 11/21/18	Atrby	

2. Phase 2 includes submission of a laptop communicating with moving car

Phase 2	**64 days**	**Sun 11/25/18**	**Thu 2/21/19**		**Submission of Laptop communicating with a moving car**
Sprint 3	**8 days**	**Sun 11/25/18**	**Wed 12/5/18**		
Implementation of Timers , PWM , External interrupts	6 days	Sun 11/25/18	Thu 11/29/18	Atrby,Areeg	
Adding Motors	4 days	Fri 11/30/18	Tue 12/4/18	Areeg	
Controlling car motion using Bluetooth application	5 days	Fri 11/30/18	Wed 12/5/18	Gehad,Amany	
Measure speed using Speed Sensor	5 days	Fri 11/30/18	Wed 12/5/18		
Sprint 4	**19 days**	**Sun 1/27/19**	**Thu 2/21/19**		**Ultrasonic**
Integration, Testing	11 days	Sun 2/10/19	Thu 2/21/19	Atrby,Aly	
Code finalization	11 days	Sun 2/10/19	Thu 2/21/19	Atrby,Aly	
update GDD	2 days	Sun 1/27/19	Mon 1/28/19	Areeg,Gehad,Amar	to be done 14/2
Building functionality of single ultrasonic	5 days	Sun 1/27/19	Thu 1/31/19	Areeg,Gehad,Amar	
unit testing of a single ultrasonic	3 days	Thu 1/31/19	Sat 2/2/19	Areeg,Gehad,Amar	
Integration & implementation of complex driver for 6 ultrasonics	5 days	Sun 2/3/19	Thu 2/7/19	Areeg,Gehad, Amany	
Integration testing	5 days	Sun 2/17/19	Thu 2/21/19	shared	
graduation book	12 days	Sun 1/27/19	Sat 2/9/19	shared	
Camera study(Research, idea development and replanning)	12 days	Sun 1/27/19	Sat 2/9/19	shared	

3. Phase 3: submission of 2 cars communicating together over WIFI

Phase 3	59 days	Sun 2/10/19	Thu 5/2/19	
Sprint 5	**24 days**	**Sun 2/17/19**	**Thu 3/21/19**	
Implementation of the 3 Applications	10 days	Sun 2/17/19	Thu 2/28/19	
Testing of the 3 Applications	5 days	Sun 3/3/19	Thu 3/7/19	
Adding new car	5 days	Sun 3/10/19	Thu 3/14/19	
integration between 2 cars and Final test	5 days	Sun 3/17/19	Thu 3/21/19	
Sprint 6 (Parallel to sprint 5)	60 days	Sun 2/10/19	Sun 5/5/19	
Camera topics learning	15 days	Sun 2/10/19	Thu 2/28/19	
implementation	21 days	Thu 2/28/19	Thu 3/28/19	
Midterm Exams	10 days	Sun 3/31/19	Thu 4/11/19	
Integration between new system and main Microcontroller	11 days	Sun 4/14/19	Fri 4/26/19	
Final Test after integration with camera	5 days	Sun 4/28/19	Thu 5/2/19	
Submission of a full testable car with all functionalitis	0 days	Sun 5/5/19	Sun 5/5/19	

4. Phase 4: PCB Design and printing

Phase 4	1 day?	Fri 5/3/19	Fri 5/3/19	
Circuit design	11 days	Sun 4/14/19	Fri 4/26/19	
Components purchasing	5 days	Sun 4/28/19	Thu 5/2/19	
Printing	0 days	Sun 5/5/19	Sun 5/5/19	

1.7.2 Global design Document (GDD)

This document is the High-Level Design Document for our graduation project identified by project team members.

The software high-level/global design activity defines for a software project:

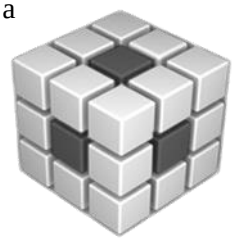
- The real time constraints on the project (issued from upstream requirements or defined during the global design),
- The software hypothesis and choices taken to respect these constraints,
- The shared data identification,
- The software architecture.

This document contains separate parts :

- The static design:
  - The software static architecture,
  - The treatments allocation in components,
  - The dataflow diagrams,
  - The components description.
- Software architecture
  - Modular software architecture
  - Functional description of each module
  - Interfaces between layers
  - Applications identifications

1.8 Development Platform

1.8.1 Code Composer Studio (CCS) Integrated Development Environment (IDE)



Code Composer Studio is an integrated development environment (IDE) that supports TI's Microcontroller and Embedded Processors portfolio. Code Composer Studio comprises a suite of tools used to develop and debug embedded applications. It includes an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler, and many other features. The intuitive IDE provides a single user interface taking you through each step of the application development flow. Familiar tools and interfaces allow users to get started faster than ever before. Code Composer Studio combines the advantages of the Eclipse software framework with advanced embedded debug capabilities from TI resulting in a compelling feature-rich development environment for embedded developers.

Code Composer Studio is primarily designed as for embedded project design and low-level (baremetal) JTAG based debugging. However, the latest releases are based on unmodified versions of the Eclipse open source IDE, which can be easily extended to include support for OS level application debug (Linux, Android, Windows Embedded) and open source compiler suites such as GCC.

Early versions included a real time kernel called DSP/BIOS and its later inception SYS/BIOS. Currently, the successor to these tools, the TI-RTOS embedded tools ecosystem, is available for downloading as a free plugin to Code Composer Studio.

1.8.2 ARM® Cortex®-M4F Based MCU TM4C123G Launchpad™ Evaluation Kit

Description

The TM4C123G Launchpad Evaluation Kit is a low-cost evaluation platform for ARM Cortex-M4F based microcontrollers from Texas Instruments. The design of the TM4C123G Launchpad highlights the TM4C123GH6PM microcontroller with a USB 2.0 device interface and hibernation module.
The EK-TM4C123GXL also features programmable user buttons and an RGB LED for custom applications. The stackable headers of the TM4C123G Launchpad Booster Pack™ XL Interface make it easy and simple to expand the functionality of the TM4C123G Launchpad when interfacing to other peripherals with Texas Instruments MCU Booster Pack.

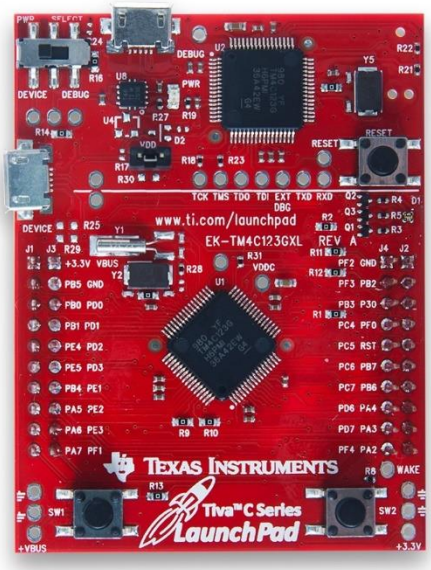


Figure 6 TM4C123G Launchpad

Features

The ARM Cortex-M4F Based MCU TM4C123G LaunchPad Evaluation Kit (EK-TM4C123GXL) offers these features:

- High Performance TM4C123GH6PM MCU:
  - 80MHz 32-bit ARM Cortex-M4-based microcontrollers CPU
  - 256KB Flash, 32KB SRAM, 2KB EEPROM
  - Two Controller Area Network (CAN) modules
  - USB 2.0 Host/Device/OTG + PHY
  - Dual 12-bit 2MSPS ADCs, motion control PWMs
  - 8 UART, 6 I2C, 4 SPI
- On-board In-Circuit Debug Interface (ICDI)
- USB Micro-B plug to USB-A plug cable

1.9 System layout

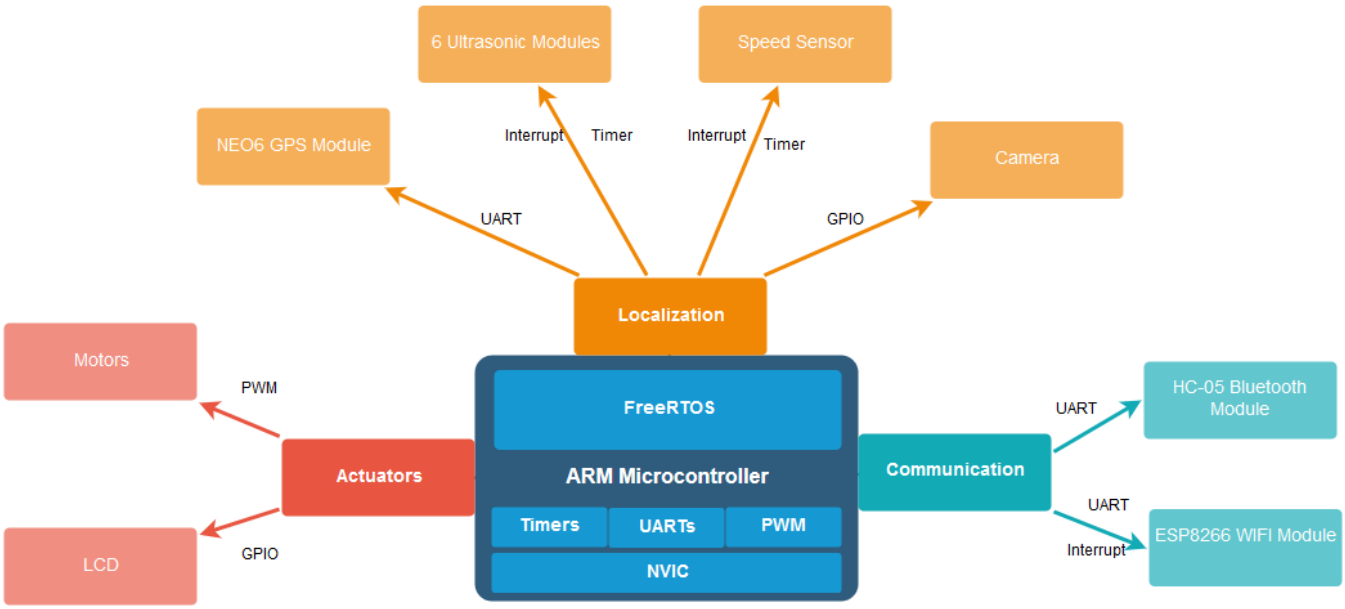


Figure 7 Project Layout

In this point we have finished thesis introduction chapter and we are going to deeply explain each part of the project in the upcoming chapters.

CHAPTER 2: CAR LOCALIZATION

Localization is how a car figures out where it is in the world. Determining the vehicle's precise position on the map is called "localization". By localizing itself, the vehicle can determine its precise relationship to all of the elements of the environment. Is the vehicle in the middle lane or the right lane? How far away is the curb? Where is the nearest car? And more questions can be answered through localization of the Car.

In our project we used different modules to achieve the objective of localization before sending the required data over communication protocol.

In this chapter we will discuss the modules we used in localization, how they work and our contribution to make them fit the objective of localization.

- Firstly, we use GPS to locate the car in the global map relative to latitude, longitude and Hight from the sea level. Using GPS to localize the vehicle seems like an easy win. Why not use GPS for localization only? The answer is that GPS is only accurate to within about 10s of meters. Think about how big 1-2 meters and then imagine what happens if a self-driving car is wrong about where it is by 1-2 meters. That curb off to the right? If the vehicle is wrong about its own location, it might wind up driving on the curb, which would be horrible. Or imagine what would happen if a vehicle nosed into an intersection because its localization was off by a meter or two.
- In order to increase the accuracy of localization and for better determination of relationship to all of the elements of the environment, we use Ultrasonic modules to get 360-degree vision of surrounding environment of the car with accuracy of some Centimeters which is optimum for near objects detection up to 4 meters around the car.
- Currently we can make accurate localization but we still can’t classify which object we are going to avoid, however our decision may depend on the type of object. For example if the object was a human we need to stop immediately but if we face a speedbump we may decrease the speed of the car only so the classification of the detected object is important. To be able to classify the object we use a camera module to capture the front view of the car and process it using a CNN model to classify is it a speedbump or not.
- In order to have more accurate localization we need to consider the factor of car speed, so we also use a speed sensor to get car data and use it in our algorithm.

2.1 Global Positioning System (GPS)

2.1.1 GPS Theory of work

There are several different methods for obtaining a position using GPS. The method used depends on the accuracy required by the user and the type of GPS receiver available. Broadly speaking, the techniques can be broken down into three basic classes:

Autonomous Navigation using a single stand-alone receiver. Position Accuracy is better than 100m for civilian users and about 20m for military users.

Differential Phase position Gives an accuracy of 0.5-20mm. Used for many surveying tasks, machine control etc.

Differentially corrected positioning. More commonly known as DGPS, this gives an accuracy of between 0.5-5m. Used for inshore marine navigation, GIS data acquisition, precision farming

in the project we will use the first class.

Autonomous Navigation.

This is the simplest technique employed by GPS receivers to instantaneously give a position and height and/ or accurate time to a user.

The accuracy obtained is better than 100m (usually around the 30-50m mark) for civilian users and 5-15m for military users. Receivers used for this type of operation are typically small, highly portable handheld units with a low cost.

The GPS concept is based on time and the known position of GPS specialized satellites. The satellites carry very stable atomic clocks that are synchronized with one another and with the ground clocks. Any drift from true time maintained on the ground is corrected daily. In the same manner, the satellite locations are known with great precision. GPS receivers have clocks as well, but they are less stable and less precise.

Each GPS satellite continuously transmits a radio signal containing the current time and data about its position. Since the speed of radio waves is constant and independent of the satellite speed, the time delay between when the satellite transmits a signal and the receiver receives it is proportional to the distance from the satellite to the receiver. As presented in one of Isaac Newton's laws of motion is

Distance = Velocity x Time

A GPS receiver monitors multiple satellites and solves equations to determine the precise position of the receiver and its deviation from true time. At a minimum, four satellites must be in view of the receiver for it to compute four unknown quantities (three position coordinates (x, y, z) and clock deviation from satellite time).

Detailed explanation of locating process:

Each GPS satellite continually broadcasts a signal (carrier wave with modulation) that includes:

1. A pseudorandom code (sequence of ones and zeros) that is known to the receiver. By time-aligning a receiver-generated version and the receiver-measured version of the code, the time of arrival (TOA) of a defined point in the code sequence, called an epoch, can be found in the receiver clock time scale

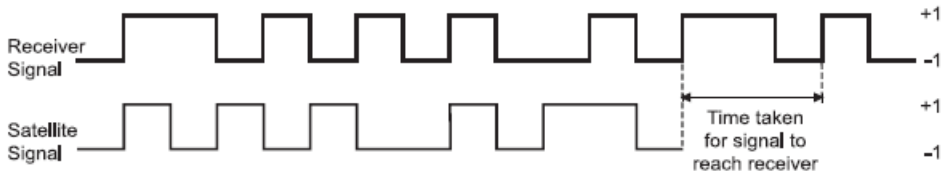


Figure 8 GPS Time of flight Resolve

2. A message that includes the time of transmission (TOT) of the code epoch (in GPS time scale) and the satellite position at that time.

Conceptually, the receiver measures the TOAs (according to its own clock) of four satellite signals. From the TOAs and the TOTs, the receiver forms four time of flight (TOF) values, which are (given the speed of light) approximately equivalent to receiver-satellite ranges. The receiver then computes its three-dimensional position and clock deviation from the four TOFs.

TOF=TOT-TOA

Distance = TOF*C

In practice the receiver position (in three dimensional Cartesian coordinates with origin at the Earth's center) and the offset of the receiver clock relative to the GPS time are computed simultaneously, using the navigation equations to process the TOFs.

The receiver's Earth-centered solution location is usually converted to latitude, longitude and height relative to an ellipsoidal Earth model. The height may then be further converted to height relative to the geoid, which is essentially mean sea level.

It is sometimes incorrectly said that the user location is at the intersection of three spheres. While simpler to visualize, this is the case only if the receiver has a clock synchronized with the satellite clocks (i.e., the receiver measures true ranges to the satellites rather than range differences). So we need more than 3 satellites to resolve receiver location.

Receiver in continuous operation

The description above is representative of a receiver start-up situation. Most receivers have a track algorithm, that combines sets of satellite measurements collected at different times—in effect, taking advantage of the fact that successive receiver positions are usually close to each other. After a set of measurements are processed, the tracker predicts the receiver location corresponding to the next set of satellite measurements. When the new measurements are collected, the receiver uses a weighting scheme to combine the new measurements with the tracker prediction.

In general, a tracker can

(a) improve receiver position and time accuracy,

(b) reject bad measurements, and

(c) estimate receiver speed and direction.

The disadvantage of a tracker is that changes in speed or direction can be computed only with a delay, and that derived direction becomes inaccurate when the distance traveled between two position measurements drops below or near the random error of position measurement. GPS units can use measurements of the Doppler shift of the signals received to compute velocity accurately. More advanced navigation systems use additional sensors like a compass or an inertial navigation system to complement GPS.

Problem description

The receiver uses messages received from satellites to determine the satellite positions and time sent. The x, y, and z components of satellite position and the time sent are designated as [xi, yi, zi, si] where the subscript i denotes the satellite and has the value 1, 2, ..., n, where $n \geq 4$. When the time of message reception indicated by the on-board receiver clock is $\tilde{t}i$, the true reception time is $ti = \tilde{t}i - b$, where b is the receiver's clock bias from the much more accurate GPS clocks employed by the satellites. The receiver clock bias is the same for all received satellite signals (assuming the satellite clocks are all perfectly

synchronized). The message's transit time is t̃i − b − si, where si is the satellite time. Assuming the message traveled at the speed of light, c, the distance traveled is (t̃i − b − si) c.

$$d_i = \left(\tilde{t}_i - b - s_i\right) c,\ i = 1, 2, \ldots, n$$

For n satellites, the equations to satisfy are:

where di is the geometric distance or range between receiver and satellite i (the values without subscripts are the x, y, and z components of receiver position):

$$d_i = \sqrt{(x - x_i)^2} \quad p_i = (\tilde{t}_i - s_i)\, c + (z - z_i)^2$$

Defining pseudo ranges as , we see they are biased versions of the true range:

$$p_i = d_i + bc,\ i = 1, 2, \ldots, n$$

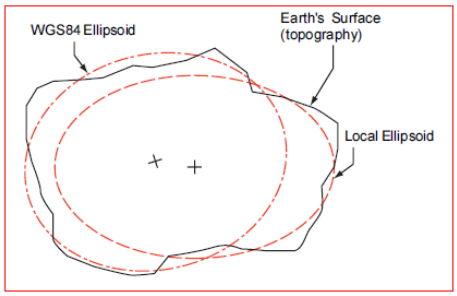
Since the equations have four unknowns [x, y, z, b]—the three components of GPS receiver position and the clock bias—signals from at least four satellites are necessary to attempt solving these equations. They can be solved by algebraic or numerical methods. When n is greater than 4 this system is overdetermined and a fitting method must be used.

The receiver location is expressed in a specific coordinate system, such as latitude and longitude using the WGS 84(World Geodetic System) geodetic datum or a country-specific system.

A geodetic datum or geodetic system is a coordinate system, and a set of reference points, used to locate places on the Earth. Each starts with an ellipsoid (stretched sphere), and then defines latitude, longitude and altitude coordinates. One or more locations on the Earth's surface are chosen as anchor "base-points". There can be different datums and difference in co-ordinates between datums is commonly referred to as *datum shift*. Different datums use different interpolations for the precise shape and size of the Earth Because the Earth is an imperfect ellipsoid, localized datums can give a more accurate representation of the area of coverage.

Horizontal datums are used for describing a point on the Earth's surface, in latitude and longitude or another coordinate system. Vertical datums measure elevations or depths.

By getting the value of latitude, longitude and altitude we can define any point on the earth with these coordinates.



The relationship between ellipsoids and the earth's surface

2.1.2 NEO-6 u-blox 6 GPS Modules

Overview

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high-performance u-blox 6 positioning engine. These flexible and cost-effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints.
The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppress jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challenging environments.

GPS performance

This table illustrates performance measures for all The NEO-6 module series but will focus on NEO-6 M.

Parameter		**Specification**	
Time-To-First-Fix[1]	NEO-6G/Q/T	NEO-6M/V	NEO-6P
Cold Start2	26 s	27 s	32 s
Warm Start[2]	26 s	27 s	32 s
Hot Start[2]	1 s	1 s	1 s
Sensitivity	NEO-6G/Q/T	NEO-6M/V	NEO-6P
Tracking & Navigation	-162 dBm	-161 dBm	-160 dBm
Maximum Navigation update rate			
NEO-6G/Q/M/T		NEO-6P/V	
5Hz		1 Hz	
Horizontal position accuracy			
GPS		2.5 m	
Velocity accuracy		0.1m/s	
Operational Limits			
Dynamics		4g	
Altitude		50,000 m	
Velocity		500 m/s	

Block diagram

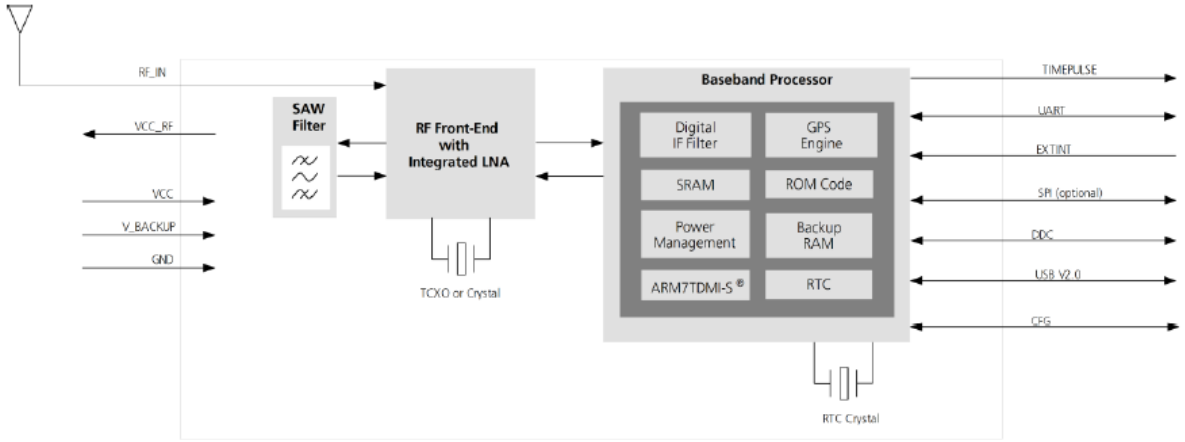


Figure 9 GPS Module Block Diagram

Assisted GPS (A-GPS)

Supply of aiding information like ephemeris, almanac, rough last position and time and satellite status and an optional time synchronization signal will reduce time to first fix significantly and improve the acquisition sensitivity. All NEO-6 modules support the u-blox AssistNow Online and AssistNow Offline A-GPS services[11] and are OMA SUPL compliant.

Assist Now Autonomous

AssistNow Autonomous provides functionality similar to Assisted-GPS without the need for a host or external network connection. Based on previously broadcast satellite ephemeris data downloaded to and stored by the GPS receiver, AssistNow Autonomous automatically generates accurate satellite orbital data ("AssistNow Autonomous data") that is usable for future GPS position fixes. AssistNow Autonomous data is reliable for up to 3 days after initial capture.

u-blox' AssistNow Autonomous benefits are:

Faster position fix

No connectivity required

Complementary with Assist Now Online and Offline services

No integration effort, calculations are done in the background

Protocols and interfaces

Protocol	Type
NMEA	Input/output, ASCII, 0183, 2.3 (compatible to 3.0)
UBX	Input/output, binary, u-blox proprietary

Table 4: Available protocols

All listed protocols are available on UART, USB, and DDC. For specification of the various protocols see the u-blox 6 Receiver Description including Protocol Specification.

UART

NEO-6 modules include one configurable UART interface for serial communication (for information about configuration see section 1.15).

USB

NEO-6 modules provide a USB version 2.0 FS (Full Speed, 12Mbit/s) interface as an alternative to the UART. The pull-up resistor on USB_DP is integrated to signal a full-speed device to the host. The VDDUSB pin supplies the USB interface. u-blox provides a Microsoft® certified USB driver for Windows XP, Windows Vista and Windows 7 operating systems.

Serial Peripheral Interface (SPI)

The SPI interface allows for the connection of external devices with a serial interface, e.g. serial flash to save configuration and AssistNow Offline A-GPS data or to interface to a host CPU. The interface can be operated in master or slave mode. In master mode, one chip select signal is available to select external slaves. In slave mode a single chip select signal enables communication with the host.
The maximum bandwidth is 100kbit/s.

Note: NEO 6 M chip supports all of this protocols and interfaces but in our project, we will depend on commercialized NEO-6 M module shown in the figure which supports an interface with UART only.

Figure 10 NEO 6M GPS Module

Power Management

u-blox receivers support different power modes. These modes represent strategies of how to control the acquisition and tracking engines in order to achieve either the best possible performance or good performance with reduced power consumption.

Maximum Performance Mode

During a Cold start, a receiver in Maximum Performance Mode continuously deploys the acquisition engine to search for all satellites. Once the receiver has a position fix (or if pre-positioning information is available), the acquisition engine continues to be used to search for all visible satellites that are not being tracked.

Eco Mode

During a Cold start, a receiver in Eco Mode works exactly as in Maximum Performance Mode. Once a position can be calculated and a sufficient number of satellites are being tracked, the acquisition engine is powered off resulting in significant power savings. The tracking engine continuously tracks acquired satellites and acquires other available or emerging satellites.

Power Save Mode
Power Save Mode (PSM) allows a reduction in system power consumption by selectively switching parts of the receiver on and off.

Operating conditions

All specifications are at an ambient temperature of 25°C.

Parameter	Module	Min	Max	Units
Power supply voltage	NEO-6M	2.7	3.6	V
Supply voltage USB	All	3.0	3.6	V
Antenna gain	All		50	dB
Operating temperature	All	-40	85	°C

Operation beyond the specified operating conditions can affect device reliability.

Default settings

Interface	Settings
Serial Port 1 Output	9600 Baud, 8 bits, no parity bit, 1 stop bit Configured to transmit both NMEA and UBX protocols, but only following NMEA and no UBX messages have been activated at start-up: **GGA, GLL, GSA, GSV, RMC, VTG, TXT** (In addition to the 6 standard NMEA messages the NEO-6T includes **ZDA**).
Serial Port 1 Input	9600 Baud, 8 bits, no parity bit, 1 stop bit Automatically accepts following protocols without need of explicit configuration: **UBX, NMEA** The GPS receiver supports interleaved UBX and NMEA messages.
Power Mode	Maximum Performance mode

2.1.3 U-center 19.04

Product description

The u-center GNSS Evaluation Software provides a powerful platform for product evaluation, configuration, testing and Realtime performance visualization of u-blox GNSS receiver products. u-center provides AssistNow client functionality for A-GNSS. Its unique flexibility makes u-center the ideal tool through the entire system integration process.

Configuration and control options

u-center provides a convenient means to configure the GNSS receiver, to save customized configuration settings in the GNSS receiver Flash memory and to restore factory settings if needed. Toolbar buttons are available to control settings such as to force cold, warm and hot starts.

u-center includes

Support for NMEA and u-blox UBX binary protocol Integrated AssistNow A-GNSS client functionality Structured and graphical data visualization in realtime:

• Satellite summary view

• Navigation summary view

• Compass, speedometer, clock, altimeter

• Chart view of any two parameters of choice

• Data recording and playback functionality

Product Highlights

• Interactive and easy to use

• Extensive GNSS configuration, control features, and output messages

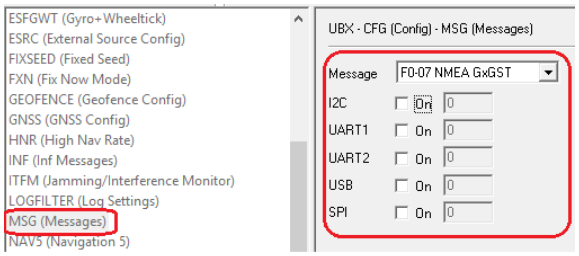
• Supports all u-blox GNSS receivers

• Enables comparative performance analysis of GNSS receivers that output NMEA messages

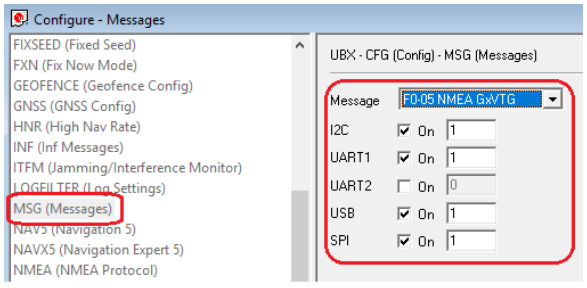
U-center Configuration Steps

1. Disable all messages and Enable GGA/GLL message only

- Disable messages
    - Open u-center.
    - Connect to GPS module (default port is COM3, baudrate 9600):
    - Go to View > Configuration View.
    - Click MSG (Messages) on the left side list.
    - Choose a NMEA dataset at "Message", Uncheck all the boxes:
    - Click **“Send”** button in the lower left corner:



- Enables messages
    - Click MSG (Messages) on the left side list.
    - Choose a NMEA dataset at "Message", Uncheck all the boxes:
    - Click “Send” button in the lower left corner:

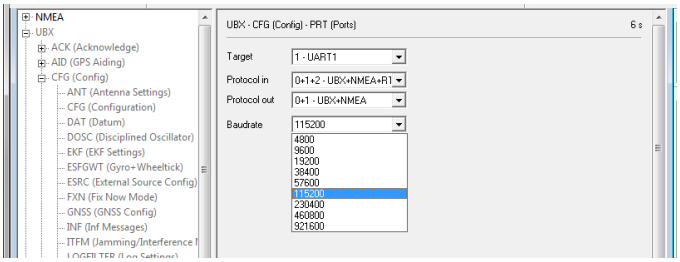


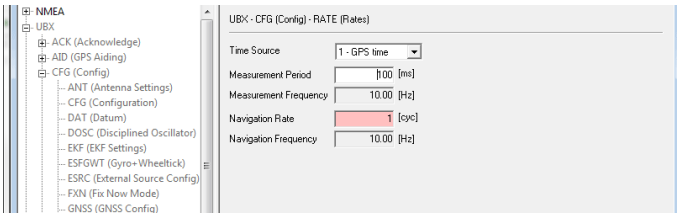
2. change module Baud rate from 9600 to 115200

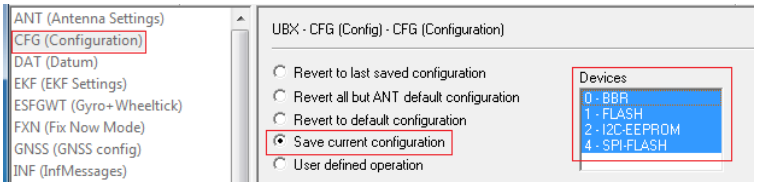
- Navigate to UBX -> CFG -> PRT to change the serial port baud rate
- Click “Send” button in the lower left corner:

3. Change module Data update rate from 1 Hz to 15 Hz.

- Navigate to UBX -> CFG -> RATE to change the GPS data rate
- Save configuration







2.1.4 Module configuration and Messages

In this section we will discuss the Module Configurations and NMEA/UBX protocol messages

2.1.4.1 NMEA protocol

The National Marine Electronics Association (NMEA) has developed a specification that defines the interface between various pieces of marine electronic equipment. GPS receiver communication is defined within this specification. The idea of NMEA is to send a line of data called a sentence that is totally self contained and independent from other sentences. There are standard sentences for each device category and there is also the ability to define proprietary sentences for use by the individual company. All of the standard sentences have a two letter prefix that defines the device that uses that sentence type. (For gps receivers the prefix is GP.) which is followed by a three letter sequence that defines the sentence contents. In addition NMEA permits hardware manufactures to define their own proprietary sentences for whatever purpose they see fit. All proprietary sentences begin with the letter P and are followed with 3 letters that identifies the manufacturer controlling that sentence. For example: PUBX defined by u-blox to control GPS receivers. Each sentence begins with a '$' and ends with a carriage return/line feed sequence and can be no longer than 80 characters of visible text (plus the line terminators). The data is contained within this single line with data items separated by commas.

Here is samples of useful NMEA messeges that we used in our project.

GGA —Global Positioning System Fixed Data

Table 1-3 contains the values for the following example:

*$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M, , , ,0000*18*

GGA Data Format

Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator	1		See Table 1-4
Satellites Used	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
MSL Altitude	9.0	meters	
Units	M	meters	
Geoid Separation		meters	
Units	M	meters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR> <LF>			End of message termination

GLL—Geographic Position - Latitude/Longitude

This Table contains the values for the following example:

$GPGLL,3723.2475,N,12158.3416,W,161229.487,A,A*41

Name	Example	Units	Description
Message ID	$GPGLL		GLL protocol header
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
UTC Time	161229.487		hhmmss.sss
Status	A		A=data valid or V=data not valid
Mode	*A*		*A=Autonomous, D=DGPS, E=DR (Only present in NMEA version 3.00)*
Checksum	*41		
<CR> <LF>			End of message termination

2.1.4.2 UBX protocol

u-blox GPS receivers use a u-blox proprietary protocol (UBX) to transmit and receive GPS data.
This protocol has the following key features:
• Compact - uses 8 Bit Binary Data.
• Checksum Protected - uses a low-overhead checksum algorithm
• Can be used for direct communication and configuration of u-blox GPS receivers.

2.1.5 Contribution

In this part we will discuss the progress of GPS Work during our project.

- Once you a buy a module you can find it in of default configurations mentioned in table 4. The module was on 9600 Baud and supports UART framing of 8 bits, no parity bit, 1 stop bit. Also, it was Configured to transmit both NMEA and UBX protocols. It actually receives 6 types of NMEA frames which is GGA, GLL, GSA, GSV, RMC, VTG.
- Design options:
    - Depend on NMEA protocol frames.
    - Depend on UBX protocol frames.
- Our design option was to depend on NMEA protocol frames for the following reasons:
    - NMEA frames are sent continuously with no need to request data from the receiver each time you need GPS frame.
    - UBX protocol is bidirectional communication protocols, hence we need to send a request for the module and wait for it to respond with the needed frame which consumes double the time taken in NMEA frames reading.
- We investigated the received NMEA frames and found that it's not useful to receive and decode 6 types of frames contains almost same data and it would be better if we can receive only 1 frame that contains the needed data which is Time, latitude, longitude and altitude. Based on this investigation we found that GGA frame supports all needed information and it would be enough for our application.
- GGA is an NMEA frame which consists of 70 to 75 characters contains the needed data and other data which is unneeded. (detailed description for GGA frame mentioned in Module configuration and Messages section)
- We developed a code that decodes the GGA frame and extract data of it and started the testing of module alone. Results as shown in the next figure.

```
$GPGGA,132151.00,3001.52596,N,03112.69253,E,1,05,5.45,70.2,M,15.3,M,,*6F
lat_GGA= 3001.52596
long_GGA= 03112.69253
alt_GGA= 70.2
$GPGGA,132153.00,3001.52860,N,03112.69043,E,1,04,3.09,64.7,M,15.3,M,,*65
lat_GGA= 3001.52860
long_GGA= 03112.69043
alt_GGA= 64.7
$GPGGA,132155.00,3001.53006,N,03112.68926,E,1,04,3.09,62.0,M,15.3,M,,*60
lat_GGA= 3001.53006
long_GGA= 03112.68926
alt_GGA= 62.0
$GPGGA,132157.00,3001.53098,N,03112.68847,E,1,04,3.09,60.1,M,15.3,M,,*60
lat_GGA= 3001.53098
long_GGA= 03112.68847
alt_GGA= 60.1
$GPGGA,132159.00,3001.53157,N,03112.68792,E,1,04,3.10,58.6,M,15.3,M,,*6F
lat_GGA= 3001.53157
long_GGA= 03112.68792
alt_GGA= 58.6
```

Figure 11 GPS Frame reception of type GGA

- After testing the module alone, we developed a code that calculates the distance between 2 GPS modules depending on the coordinates of latitude and longitude of each GPS module.

2.1.5.1 Distance Calculations

To calculate the distance between 2 points in space we need to get latitude and longitude values of 2 GPS modules and use equations to get the distance.

All these formulas are for calculations on the basis of a spherical earth (ignoring ellipsoidal effects) – which is accurate enough* for most purposes.

Here we use the 'haversine' formula to calculate the great-circle distance between two points – that is, the shortest distance over the earth's surface (ignoring any hills they fly over, of course!).

Haversine formula:	$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$
	$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$
	$d = R \cdot c$
where	φ *is latitude,* λ *is longitude,* R *is earth's radius (mean radius = 6,371km); note that angles need to be in radians to pass to trig functions!*

Here is a test of distance calculation and comparison with an online calculator for GPS distance.

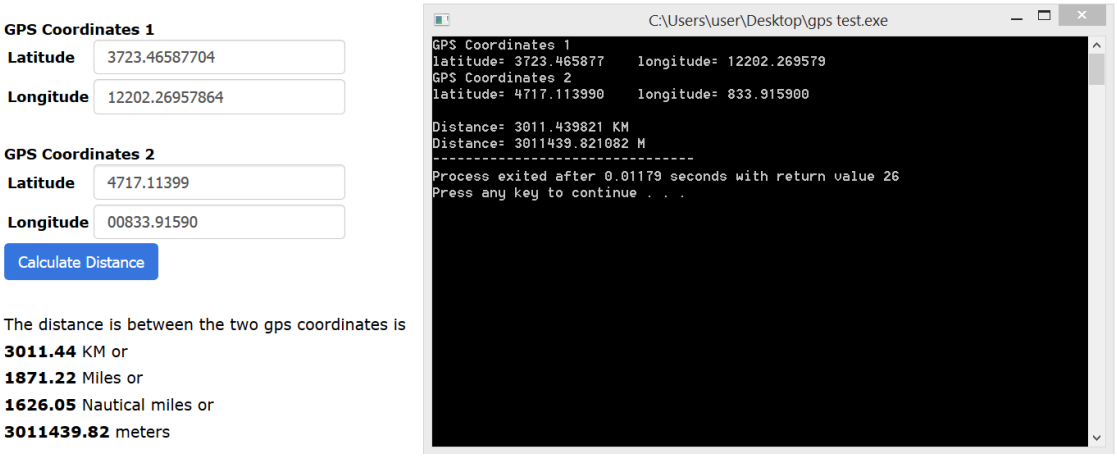


Figure 12 GPS Distance Calculation

Code:

```
double Distance(double lat1, double lon1, double lat2, double lon2, char unit)
{
   int r = 6371; /* Earth reduis */
   double deltaLat=deg2rad(lat2 - lat1);  /* transformation of the difference between Latitudes values from degrees to radian */
   double deltaLon=deg2rad(lon2 - lon1); /* transformation of the difference between Longitudes values from degrees to radian*/
   lat1=deg2rad(lat1); /* transformation of Latitude value from degrees to radian*/
   lat2=deg2rad(lat2);/* transformation of Longitude value from degrees to radian*/
   double v_a;
   double v_c;
   double distance;

   v_a = sin(deltaLat/2) * sin(deltaLat/2) + cos(lat1) * cos(lat2) * sin(deltaLon/2) * sin(deltaLon/2);
   v_c = 2 * atan2(sqrt(v_a),sqrt(1-v_a));
   distance = r * v_c;
   if(unit=='M') return distance*1000;
   else if(unit =='K') return distance;
   else return -1;
    }
```

These equations are taken from an online source.

2.1.5.2 Contribution Results

- After removing unneeded messages and configuring the GPS module for GGA frame only and testing we can achieve results of:
    - Baud rate of 9600
    - New frame received each 1 second (frequency of 1 Hz for reception and navigation.
    - Calculate Distance between 2 coordinates accurately
    - Program average execution time of 1.1 seconds to process a full GGA frame and extract the needed information of it.

2.1.6 optimization

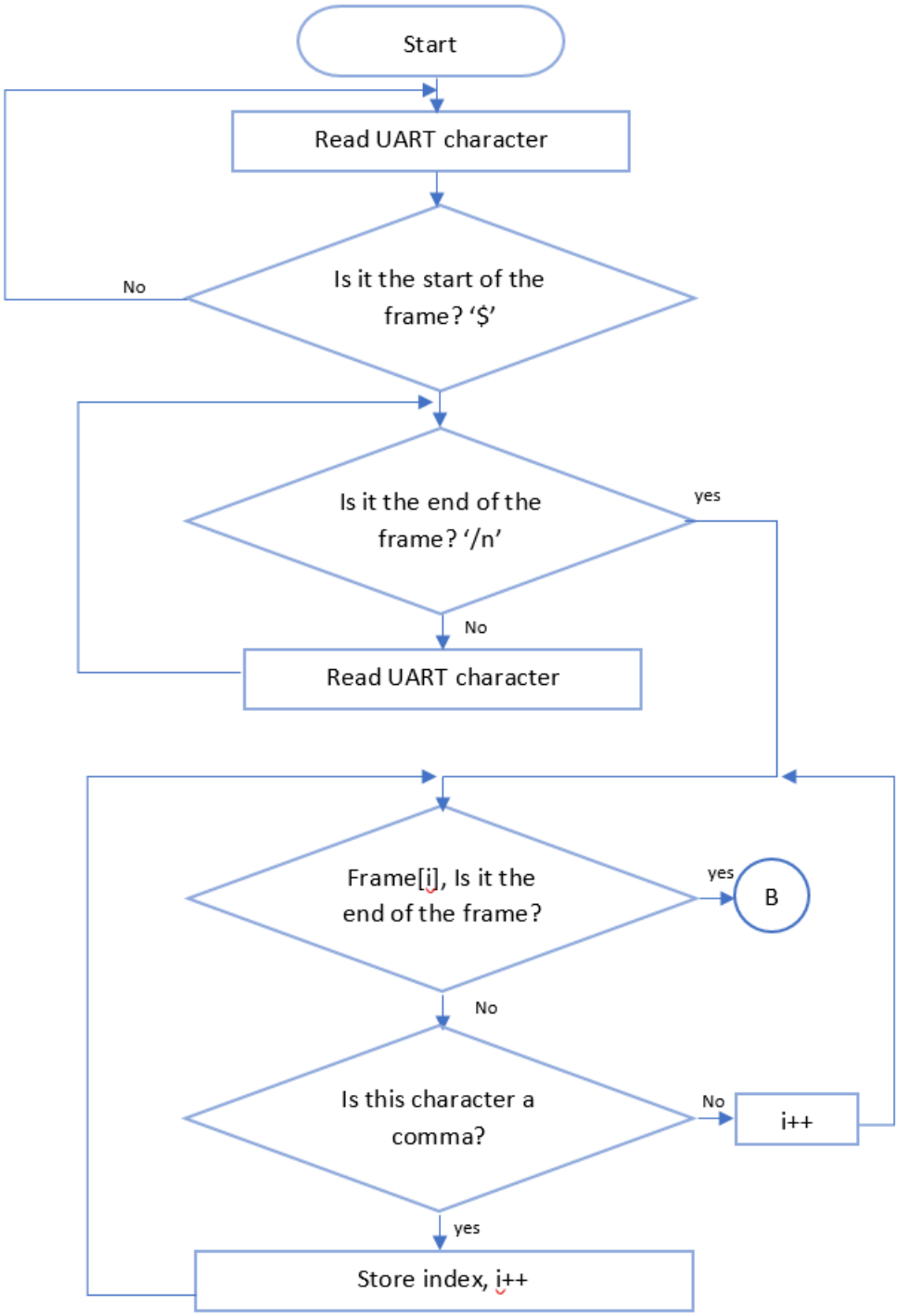
As mentioned in work flow results we could achieve moderate results and in this section, we would discuss the optimization techniques performed in order to get better results.

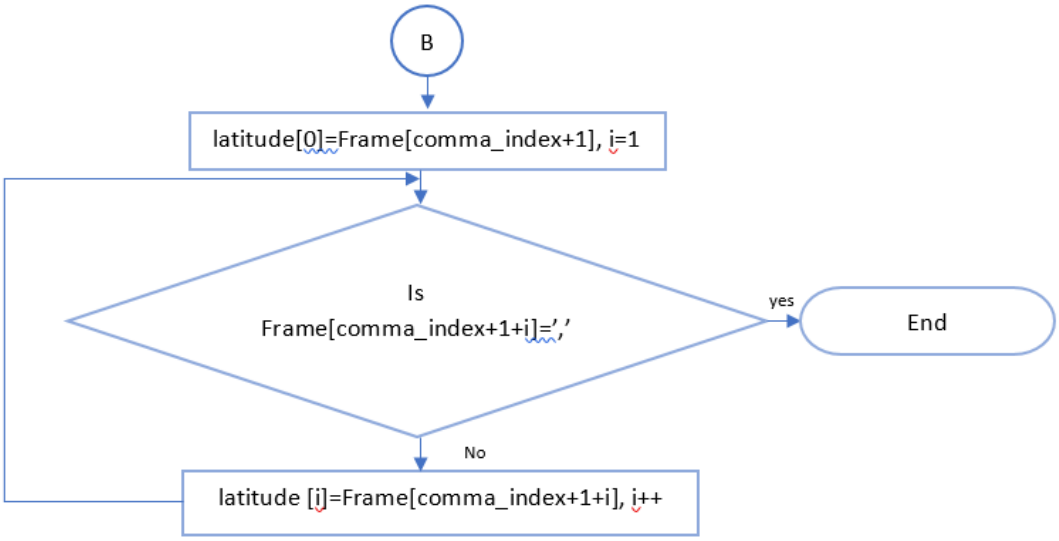
- In order to increase the reception speed, we increased the baud rate of the module and the UART module used on tivac chip. Previously we used 9600 baud rate and currently we are using 115200.
- In order to increase the reception speed, we increased data update rate of the GPS module from 1 Hz to 15 Hz.
- In order to increase the reception speed, Configure the module to Receive GLL frames only instead of GGA frames as GLL frames is more compact and consists of 50 characters only.
- In order to decrease (faster) first time to lock, we changed the module from cold start mode to hot start mode.
- Steps to perform previous techniques are mentioned in U-center section.

2.1.6.1 Optimization results

- After performing this optimization techniques and testing we can achieve results of:
    - Baud rate of 115200
    - New frame received each 65ms second (frequency of 15.4 Hz for reception and navigation.
    - Calculate Distance between 2 coordinates accurately
    - Program average execution time of 100ms to process a full GGA frame and extract the needed information of it.

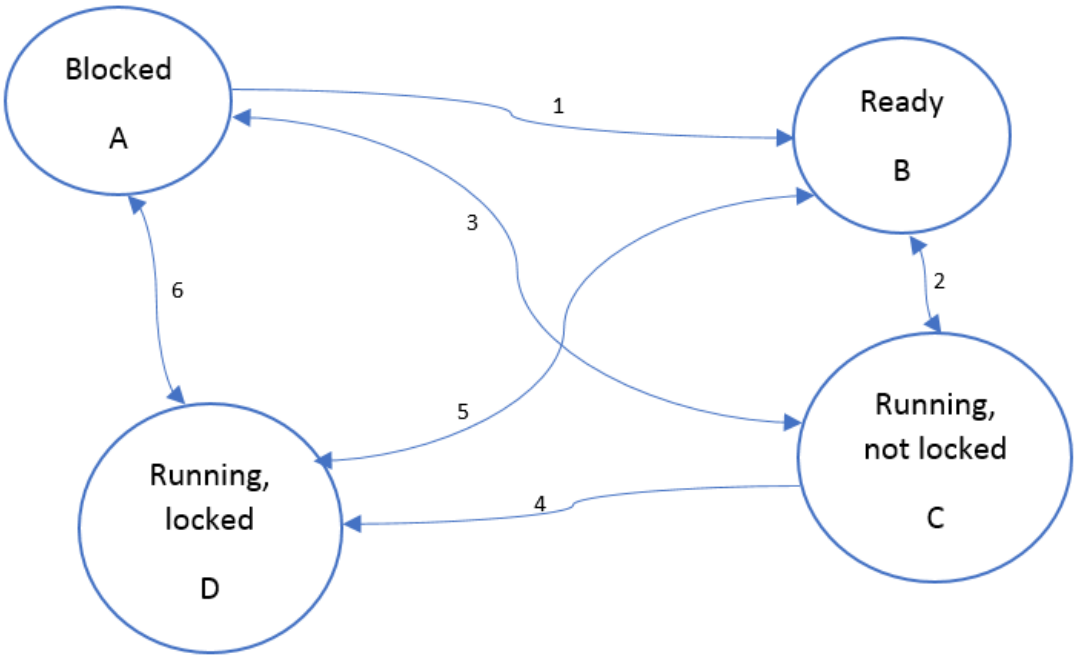
2.1.7 GPS flow chart





B section is repeated for longitude and Altitude

2.1.8 GPS state diagram



A. Blocked state: we enter this state when the task finishes execution and wait for a period of time until it can start again.

    - Arrow 3 (state C to state A), Arrow 6 (state D to state A) the task was running on the CPU and finished execution.

B. Ready state in this state the GPS task can run on CPU but there is a higher priority task running on the CPU so we wait in Ready state until the CPU is free for the task to run.

    - Arrow 1 (state A to state B) the task was blocked for a period of time and the block was removed after it, but there is a higher priority task running on the CPU so the GPS task will wait for it to finish Execution then take the CPU(Arrow 2 and 3).

    - Arrow 2 (state C to state B), Arrow 5 (state D to B) the task was running on the CPU and another Higher priority task preempted it to run on the CPU.

C. Running but not locked state, the task is running on the CPU but GPS module isn't locked on enough satellites so we can't get a valid data in this state.

    - Arrow 2 (state B to state C) the task was ready then it runs on the CPU.

    - Arrow 3 (state A to state C) the task was blocked then it runs on the CPU directly. without going to ready state as the CPU is free.

D. Running and locked state, the task is running on the CPU and GPS module is locked on enough satellites so we can get a valid data in this state.

    - Arrow 4 (C state to State D) GPS module was unlocked then locks to enough satellites.

    - Arrow 5 (state to State D) the task was ready then runs on the CPU.

    - Arrow 6 (state A to State D) the task was blocked then runs on the CPU directly.

2.2 ULTRASONIC SENSOR

2.2.1 Ultrasonic theory of work

The ultrasonic sensor works on the principle of SONAR system which is used to determine the distance to an object. SONAR basically stands for Sound Navigation and Ranging. An ultrasonic sensor generates the high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is sensed by the receiver. By measuring the time required for the echo to reach to the receiver, we can calculate the distance.

2.2.2 HC-SR04 Ultrasonic module

HC-SR04 has an ultrasonic transmitter, receiver and control circuit.

Microcontroller U1

The heart of the unit is
the EM78P153 8-bit
microprocessor.
This handle:

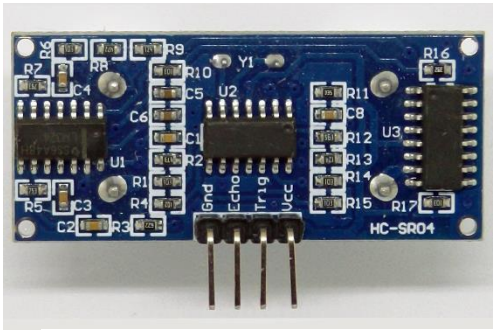


Figure 13 HC-SR04 Ultrasonic module

- Interface to Trigger and Echo pins.
- Timing and sending antiphase burst for ping to send.
- Squelch control, where by during Ultrasonic transmit, a threshold for the incoming receiver is effectively disabled to avoid bogus echoes. This is important as whilst sending vibrations from the TX transducer will be received through PCB and by air between the transducers on the RX transducer.
- Receiving the processed signal from the Receiver section as an interrupt (Rising edge), this is actually a filtered and much amplified version of all the echoes received.

Transmitter U3

- Voltage drive to TX transducer from the antiphase TX signals from the micro (U1). By using antiphase signals, a differential voltage can be driven across the transducer effectively +/-5V across the transducer.
- The other part is two transistors with a common base pin, collectors available on other pins. This transistor forms part of the feedback loop on the final part of the receiver chain, to change an analog signal into a TTL type digital signal

Receiver U2

The quad op-amp IC (U2) is a LM324 is 1 MHz unity gain bandwidth device, with limited range I/O. It is a ubiquitous and cheap device. Considering some of the gain levels used in the stages means that 40 kHz signals are passing through stages with around 100 kHz bandwidth.

Operating Condition

Figure 14 Ultrasonic Operation condition

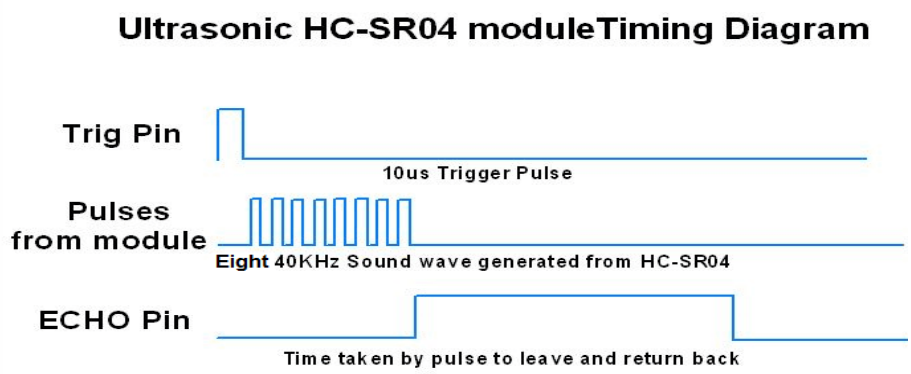
HC-SR04 works at 40 KHz Frequency which lies in ultrasound range, above 20 KHz.

Working voltage is 5V DC. As the current drawn by the sensor is less than 15mA, so won't be affecting the current ratings of the controller. No need for external buffers.

2.2.3 Working Method

We need to transmit trigger pulse of at least 10 us to the HC-SR04 Trig Pin.
10 µs is enough period for the controller, after which it starts to transmit ultrasonic signal.
Then the HC-SR04 automatically sends Eight 40 kHz sound wave pulses and wait for rising edge output at Echo pin. As the number of pulses the amplitude of the received signal increases and saturates at a point. After further increase in number of pulses results in increase in reflection peak, which are not required.
When the rising edge capture occurs at Echo pin, start the Timer and wait for falling edge on Echo pin. As soon as the falling edge is captured at the Echo pin, read the count of the Timer. This time count is the time required by the sensor to detect an object and return back from an object.

Figure 15 Ultrasonic Timing Diagram



Conversion from duration to distance can be done using the following formula:
Distance = (Travel time/2) x speed of sound
We need to divide the travel time by 2 because we have to take into account that the wave was sent, hit the object, and then returned back to the sensor.

If the HC-SR04 does not receive an echo then the output never goes low. Accordingly, some sensors timeout from 28ms to 36ms.

2.2.4 Contribution

In order to measure distance we started by choosing two GPIO Pins. One for sending trigger and other for receiving echo. Both pins are either high or low, so it was better to be used in digital mode.
We start by sending trigger to the module by setting the Trigger pin high for 10 us, then low. After the trigger, the module starts transmitting ultrasound wave and Echo pin in set high until the wave hits and object and return back to the module.
In order to check whether the Echo pin is high or low it was configured to receive interrupts at both rising and falling edges. Using polling wasn't the best solution as it is time consuming and delays other tasks.
Once the Echo pin is high we receive a rising edge interrupt. At this instance, we reset the timer to measure the duration in which Echo pin is high.
When the Echo pin returns low, we receive a falling edge interrupt, disable the timer and calculate the distance.
Since the operating frequency of controller is 80 MHz, so by diving number of ticks by 80 we get the duration at which Echo pin is high in µs. Assuming the speed of transmitted ultrasound wave is approximately 343 m/s, so 343m/s = 0.0343 cm/µS = 1/29.1 cm/µS. Then we divide by 2 to get the required distance in cm.

```
if (!Echo)
  {
    TIMER2_TAV_R  = 0;
    TIMER2_CTL_R= 0x00000001;
    Echo=1;
  }

  else
   {
    pulse = TIMER2_TAR_R;
    TIMER2_CTL_R = 0x00000000;
    Echo=0;
    Reading =(unsigned long)(pulse/80.0);
    Reading = Reading / 58;
```

In order to provide 360^0 awareness for the car and avoid crashes, position and number of sensors had to be taken into consideration. We used 6 Ultrasonic sensors to measure distance between cars in all directions, including blind spot.
As the number of sensors increased we had some constraints such as: Number of GPIO pins and Timers.

- GPIO:

To use less number of GPIO pins, we thought of using only one echo pin for all sensors, but it wasn't the best solution as the controller receive interrupts from more than one sensor at the same time, this problem was slightly solved using a delay function but not for more than three sensors and there was a great inference between sensors readings. Besides, increasing delay between triggering sensors was time consuming and increased task duration.
For improved operation, we connected all sensors to the same GPIO port so that interrupts can be easily enabled and disabled at once if needed.

To ensure we get correct reading before sending trigger to a sensor we enable interrupts on the equivalent Echo pin and disable interrupts on all other Echo pins, to evade simultaneous interrupts.

- Timers:

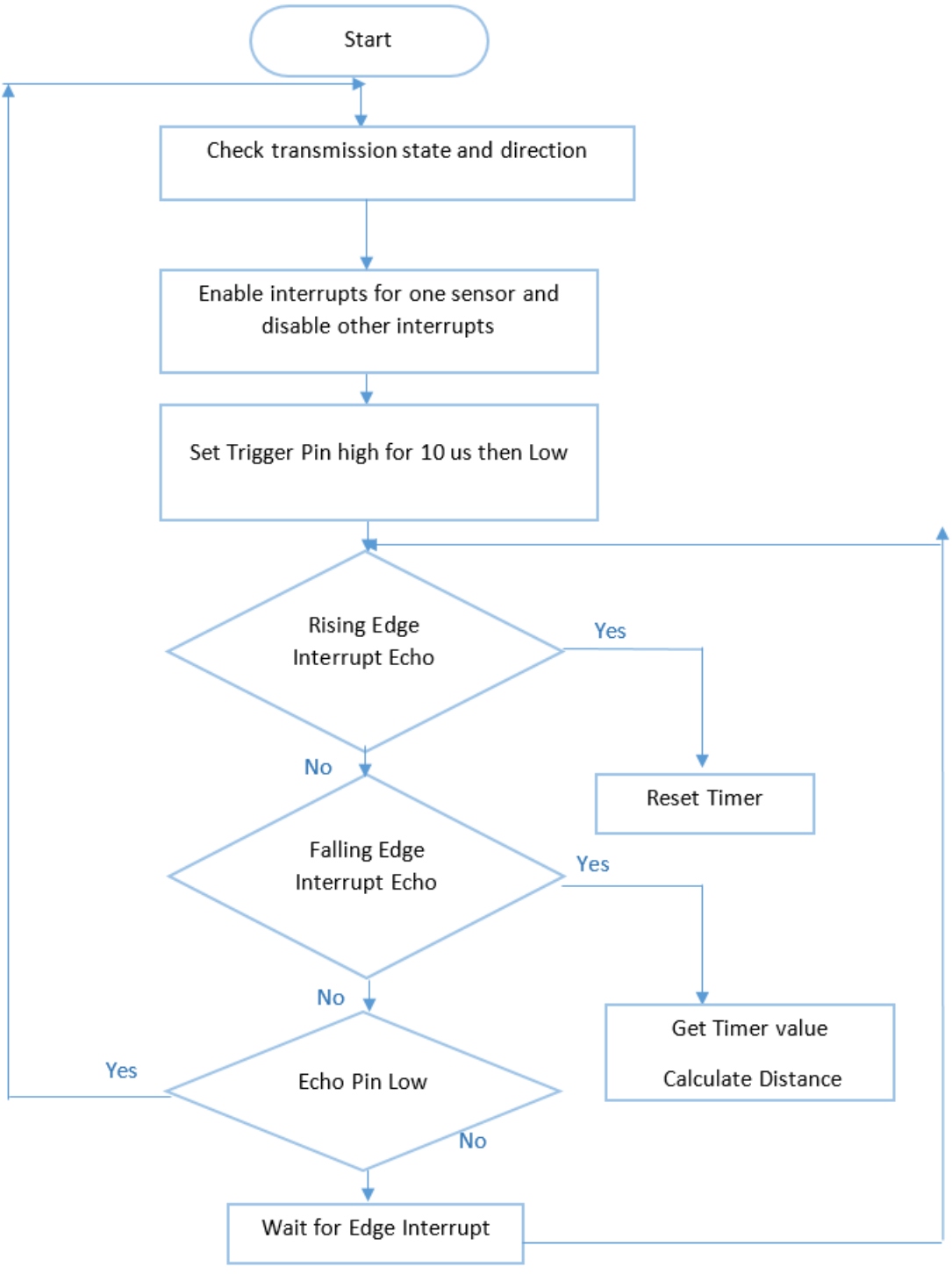
We had to use 32 bit timer to avoid timer overflow and wrong results, at operating frequency of 80 MHz and by using 16 bit timer, the maximum number of counts = $2^{16} = 65536$,
Then time = $\frac{65536}{80*10^6}$ = 0.0008192, so maximum distance is approximately 281 cm which is less than ultrasonic operating range. Thus, it was better to use 32 bit timer.
We thought of using 6 independent timers, one for each sensor, but this solution had some problems as by sending trigger to multiple sensors at the same time you can get multiple interrupts at the same time from the various echo signals, which caused inaccurate values. To avoid wrong readings we agreed on using only one timer, sending trigger to one sensor at a time and loop on other sensors.

2.2.5 Optimization

Since we have 6 Ultrasonic sensors so, we can consider this case. The car is moving in forward direction and there is another one in front of it. Here, we are interested in the front sensor reading as it is the most important. Another case, if the driver wants to turn right, so the right sensor's reading is the most important. Same for turning left or moving backward.
Accordingly, the code was optimized in a way such that the order we get reading from ultrasonic sensor changes according to direction and transmission state.
As a result, the car became more reliable and more responsive to obstacles. Also, we had better control for motors' speed.

```
if( (THIS_CAR_DATA.Transmission_state=='U'||THIS_CAR_DATA.Transmission_state=='X') && THIS_CAR_DATA.Direction=='N')
{
    us_order[0]=6; us_order[1]=4; us_order[2]=5;
    us_order[3]=2; us_order[4]=3; us_order[5]=1;
}
if(THIS_CAR_DATA.Transmission_state=='D' && THIS_CAR_DATA.Direction=='N')
{
    us_order[0]=1; us_order[1]=4; us_order[2]=5;
    us_order[3]=2; us_order[4]=3; us_order[5]=6;
}
if(THIS_CAR_DATA.Transmission_state=='U' && THIS_CAR_DATA.Direction=='R')
{
    us_order[0]=4; us_order[1]=2; us_order[2]=6;
    us_order[3]=5; us_order[4]=3; us_order[5]=1;
}
if(THIS_CAR_DATA.Transmission_state=='U' && THIS_CAR_DATA.Direction=='L')
{
    us_order[0]=5; us_order[1]=3; us_order[2]=6;
    us_order[3]=4; us_order[4]=2; us_order[5]=1;
}
```

2.2.6 Ultrasonic Flowchart



2.3 SPEED SENSOR

Since v2v technology needs complete information about all the vehicles committed to it, it was necessary to calculate the speed of cars so that relative speed between vehicles on the road can be calculated and decisions can be taken according to that.

2.3.1 Speed sensor work Theory:

LM 393 speed sensor is a module that is used to calculate the rotational speed of the rotating motors as follows:

Figure 16 Speed Sensor

Figure() shows the LM393 module which consists of two parts: the sensor part and the control part.

Sensor Part:

an Infrared LED and an NPN Photo Transistor. These two components are placed facing each other is a special housing made of black thermoplastic. This special housing ensures that the Photo Transistor receives light only from the Infrared LED and all the external source of light is eliminated.

Control Part:

LM393 Voltage Comparator and a few passive electronic components. The signal from the photo transistor is given to the LM393 and based on the presence or absence of an object between the Infrared LED and the Photo Transistor, the Output of the LM393 IC will either be HIGH or LOW.

So in a nutshell it's the IR LED always on and passes an IR ray if it meets the motor's wheel slit so IR passes to the NPN photo transistor so the voltage comparator results HIGH, and if the IR ray meets blank areas between motor's wheel slits it blocks the ray so it doesn't reach the photo transistor so the voltage comparator results LOW.

2.3.2 Calculating the resulted counts

As we explained in the work theory section the motor wheel rotation results in passing and blocking IR ray so this counts the slits which meets the ray which is considered as an indication for the speed as we will see in the next section.

An ISR is needed to increase counts by one each time the state of the ray is changed.

```
void PortFIntHandler(void)
{
Counts++;
GPIO_PORTF_ICR_R|= (1<<1);
}
```

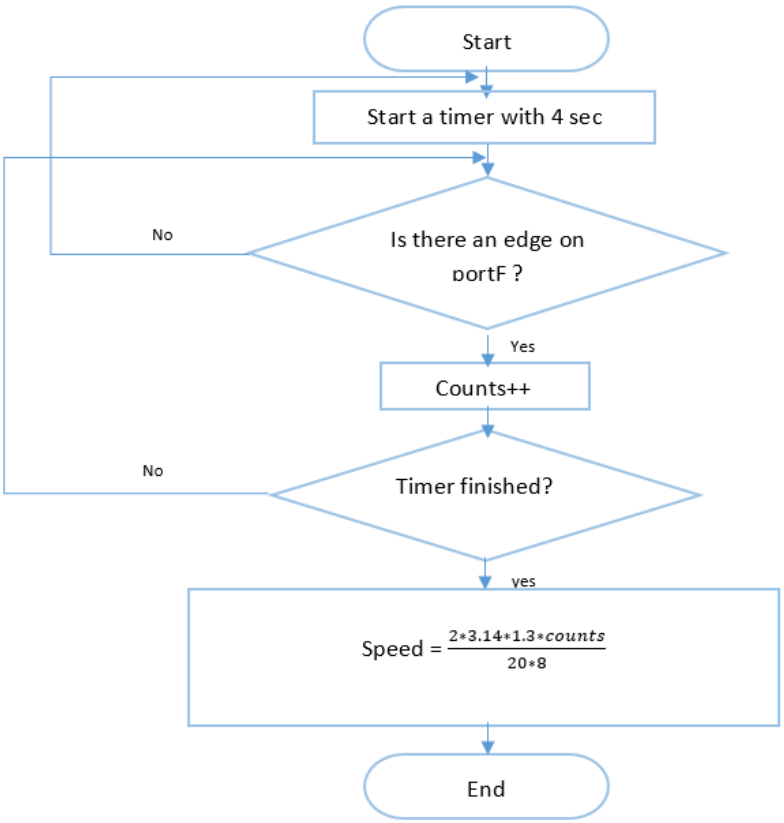
2.3.3Calculating the speed

To calculate the speed we need distance and time: distance is obtained by the perimeter of the motor's wheel and time is obtained by a timer handler, then all the information is ready now to calculate speed as follows:

$$\text{speed} = \frac{\text{distance}}{\text{time}} = \frac{2\pi r * \text{counts}}{\text{no. of slits} * \text{time}} = \frac{2 * 3.14 * 1.3 * counts}{20 * 8}$$

```
void TimerA1_Handler(void)
  {
    TIMER1_ICR_R = TIMER_ICR_TATOCINT;
    THIS_CAR_DATA.speed=((2*3.14*1.3)/160)*Counts;
    Counts=0;
    #ifdef SPEED_PRINT
       Void_Clear_Screen();
       Void_LCD_Print("SPEED=");
       Void_LCD_NUM((int)THIS_CAR_DATA.speed);
    #endif /*SPEED_PRINT*/
  }
```

2.3.4 Speed sensor flowchart



2.4 CAMERA MODULE

2.4.1 About Raspberry Pi Camera

Figure 17 RP Camera

The Raspberry Pi Camera Module v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add - on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated C Si interface, designed especially for interfacing to cameras.

The board itself is tiny, at around 25mm x 2 3 mm x 9mm. It also weighs just over 3g, making it perfect for mobile or other applications where size and weight are important. It connects to Raspberry Pi by way of a short ribbon cable. The high quality S ony IMX219 image sensor itself has a native resolution of 8 megapixel, and has a fixed focus lens on - board. In terms of still images, the camera is capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p90 video.

2.4.2 Features Summary

- 8-megapixel camera capable of taking photographs of 3280 x 2464 pixels and static photos up to 25920X1944 Pixel
- has high resolution and small size that is specially designed for all versions of raspberry Pi
- Capture video at 1080p30, 720p60 and 640x480p90 resolutions
- All software is supported within the latest version of Raspbian Operating System
- It is connected directly to Raspberry using small ribbon -comes with the camera - through the CSI (camera serial interface) port on your Raspberry- Pi.

CHAPTER 3: ROAD BUMP DETECTION

3.1 INTRODUCTION TO MACHINE LEARNING:

Machine Learning can be considered a subfield of Artificial Intelligence since those algorithms can be seen as building blocks to make computers learn to behave more intelligently by somehow generalizing rather that just storing and retrieving data items like a database system would do.

Machine learning can be broken into two broad regimes: supervised learning and unsupervised learning and third type called reinforcement learning. We'll introduce these concepts here, and discuss them in more detail below.

In Supervised Learning, we have a dataset consisting of both features and labels. The task is to construct an estimator which is able to predict the label of an object given the set of features.

Supervised learning is further broken down into two categories, classification and regression.

In classification, the label is discrete - is there a specified object or not-, while in regression, the label is continuous –determining the value of some specified variables-.

Regression is the task of predicting the value of a continuously varying variable (e.g. a price, a temperature, a conversion rate...) given some input variables. Classification is the task of predicting the value of a categorical variable given some input variables.

Unsupervised Learning addresses a different sort of problem. Here the data has no labels, and we are interested in finding similarities between the objects in question. In a sense, you can think of unsupervised learning as a means of discovering labels from the data itself.

Unsupervised learning comprises tasks such as dimensionality reduction, clustering, and density estimation.

"reinforcement learning". Here, the model is learned from a series of actions by maximizing a "reward function". The reward function can either be maximized by penalizing "bad actions" and/or rewarding "good actions". A popular example of reinforcement learning would be the training of selfdriving car using feedback from the environment.

The overfitting issue

Evaluating the quality of the model on the data used to fit the model can lead to overfitting. Consider the following dataset, and three fits to the data:

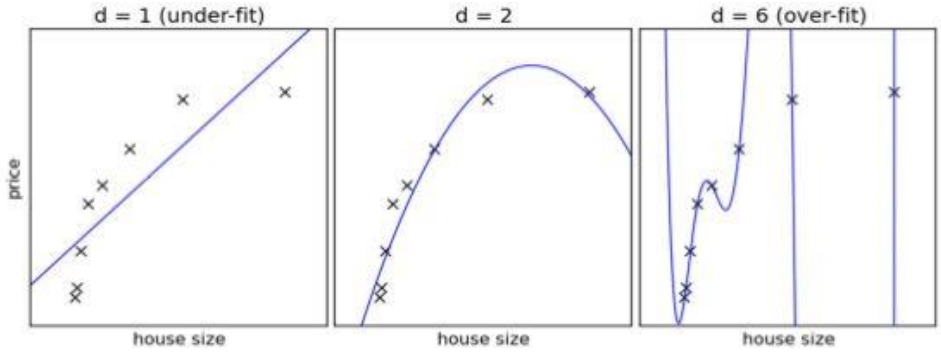


Figure 18 Model Fitting

Evaluating the d = 6 model using the training data might lead you to believe the model is very good, when in fact it does not do a good job of representing the data. The problem lies in the fact that some models can be subject to the overfitting issue: they can learn the training data by heart without generalizing. The symptoms are:

- the predictive accuracy on the data used for training can be excellent (sometimes 100%)
- however, the models do little better than random prediction when facing new data that was not part of the training set.

If you evaluate your model on your training data you won't be able to tell whether your model is overfitting or not.

Solutions to overfitting

The solution to this issue is twofold:

1. Split your data into two sets to detect overfitting situations:
    - one for training and model selection: the training set .
    - one for evaluation: the test set.
2. Avoid overfitting by using simpler models (e.g. linear classifiers instead of gaussian kernel SVM) or by increasing the regularization parameter of the model if available.
3. An even better option when experimenting with classifiers is to divide the data into three sets: training, validation and testing. You can then optimize your features, settings and algorithms for the validation set until they seem good enough, and finally test on the testing set (perhaps after adding the validation set to the training set).

So after this explanation we can apply these concepts on our application as follows:

Since the application is detection of whether there's a speed bump -in the live stream for the road photographed by the camera- or not to warn the driver to slow down if there's a speed bump.

So it's a classification problem (result 1 if there's a bump, and 0 if there is no bump) and mainly it is a supervised learning as the data set is given to the model labeled by their name whether it's a bump image or non bump image.

3.2 Image Classification

as we explained the concept of classification problem, now this section will introduce the details of the image classification and it will lead to our introduced model.

Figure 19 Bump example

Figure() shows an image consists of 278 pixels wide, 181 pixels tall, and has three color channels Red,Green,Blue (or RGB for short). So the image has $278 \times 181 \times 3$ or a total of 150,954 distinct numbers represent the brightness of each pixel in its color plane giving a range of possible values from 0 to 255. Typically zero is taken to be black, and 255 is taken to be white.

Our task is to give a final label for this image if it has a speed bump (1) or don't have a speed bump (0).

This task of image classification encounters many challenges as it's not as easy as for the human being eye which can classify, detect, understand and analyze the detected objects, challenges can be explained as follows:

- Viewpoint variation : which means that each single object can be photographed and viewd from many point of views and so the view differs.
- Scale variation: Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).
- Deformation. Many objects of interest are not rigid bodies and can be deformed in extreme ways.
- Occlusion. The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible.
- Illumination conditions. The effects of illumination are drastic on the pixel level.
- Background clutter. The objects of interest may blend into their environment, making them hard to identify.
- Intra-class variation. The classes of interest can often be relatively broad, such as chair . There are many different types of these objects, each with their own appearance.

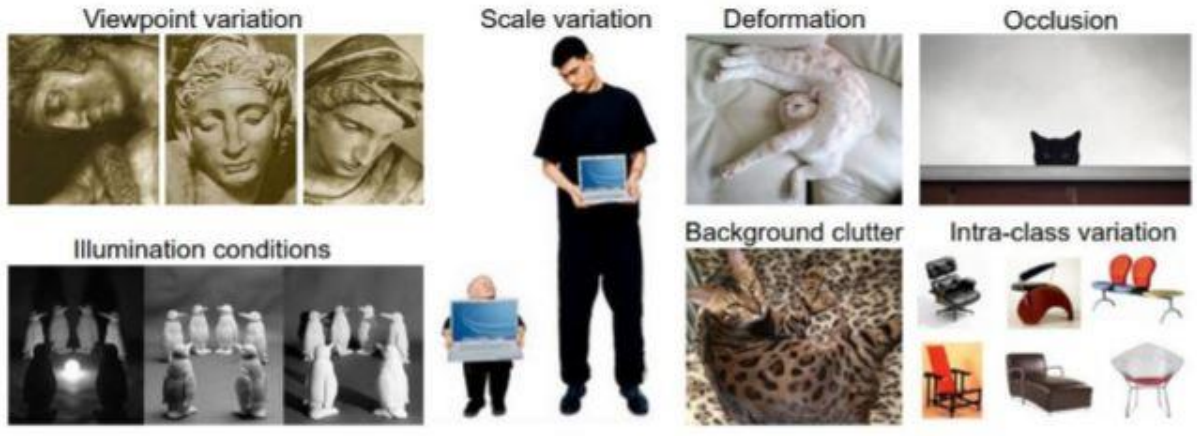


Figure 20 Challenges of Image Classification

Process of Image classification

The main steps to construct a model that results in a final label to classify an object in an image to be one of some specified group of objects are:

- Input dataset: prepare large number N of labeled images according to its class. And this is called the training set.

- Learning: to make the model learn what each class looks like to be able to classify any test image, and this step is called training a classifier or learning a model.
- Evaluation: after training the classifier we test it by entering it a set of test images and make it predict to which class it belongs and then we compare its results by the real one to get the classifier accuracy and of course we want this accuracy percentage to be as large as possible to be a good classifier having a high ability to classify and predict images that it has never seen before in the training set.

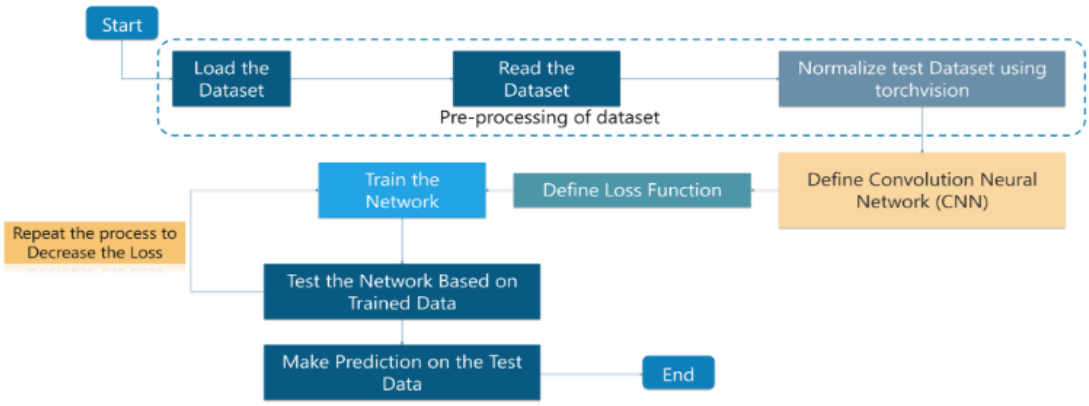


Figure 21 Process of Image Classification

And now after introducing an overview of the main steps of process, the following section will introduce in details the step of preparing the training set.

1-Input dataset

As it's mentioned before that it's necessary to have large number of images as a training set to achieve good performance which makes the model learn by itself the main features of each class and gain the model an ability to predict the image contains which object from the specified objects.

this needed amount of images is very hard to be collected or photographed image by image so image augmentation is usually required to boost the performance of deep networks.

Image augmentation artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc.

Here are some examples of the used image augmentation techniques:

➢ random rotation

it adds some random rotation of value according to uniform distribution so pixels values differ from the original image, and it's implemented as follows:

```
17 def random_rotation(image_array: ndarray):
18     # pick a random degree of rotation between 25% on the left and 25% on the right
19     random_degree = random.uniform(-25, 25)
20     return sk.transform.rotate(image_array, random_degree)
```

And here is the original image and the rotated one:


Figure 22 Random rotation

Horizontal Flip

Mirrors the image so pixels values differ from the original image, and it's implemented as follows:

```
26 def horizontal_flip(image_array: ndarray):
27     # horizontal flip doesn't need skimage, it's easy as flipping the image array of pixels !
28     return image_array[:, ::-1]
```

And here is the original image and the flipped one:


Figure 23 Horizontal flipping

- ## Random noise

it adds some random noise to the original image so pixels values differ from the original image, and it's implemented as follows:

```
22 def random_noise(image_array: ndarray):
23     # add random noise to the image
24     return sk.util.random_noise(image_array)
```

And here is the original image and the noisy one:

Figure 24 Adding random noise

Summary of data collecting:

A number of 1055 images of roads that have speed bumps is collected image by image and number of 253 images of flat roads were photographed image by image then image augmentation was used to increase the amount of the flat roads to reach 1012 images.

2-learning

As mentioned before that this step is to make the model learn by itself the main features of each class to gain the ability to predict any new image which object that it has.

In this section we will discuss why CNN is used, main blocks of CNN and reflect all these concepts on our model.

3.3 Convolutional Neural Networks

Introduction

Convolutional Neural Networks, like neural networks, are made up of neurons with learnableweights and biases**.** Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.These neurons learn how to convert input signals (e.g. picture of a cat) into corresponding output signals (e.g. the label "cat"), forming the basis of automated recognition.

Convolutional Neural Networks (CNN) are neural networks mainly used for image processing and classification.They are also popularly known as ConvNets /CovNets.

Origin of Convolutional Neural Networks:

The intelligence of neural networks is uncanny**.** While artificial neural networks were researched as early in 1960**s** by Rosenblatt, it was only in late 2000s when deep learning using neural networks took off**.** The key enabler was the scale of computation power and datasets with Google pioneering research into deep learning. In July 2012, researchers at Google exposed an advanced neural network to a series of unlabeled, static images sliced from YouTube videos.

To their surprise, they discovered that the neural network learned **a** cat-detecting neuron on its own, supporting the popular assertion that "the internet is made of cats".

Applications of Convolutional Neural Network (CNN)

The applications of CNN are listed as follows:

- Image search
- Self-driving cars
- Automatic video classification systems
- Voice recognition
- Natural language processing

There has been a lot of progress with CNNs in various image tasks like:

- Object detection
- Detecting if an image has a person, animal or vehicle etc
- Object localization

NOTE: In Object localization, the network outputs bounding boxes around various objects. This typically needs a CNN followed by an RNN scheme

Difference between ANN and CNN

In an ANN, each neuron in the network is connected to every other neuron in the adjacent hidden layers.

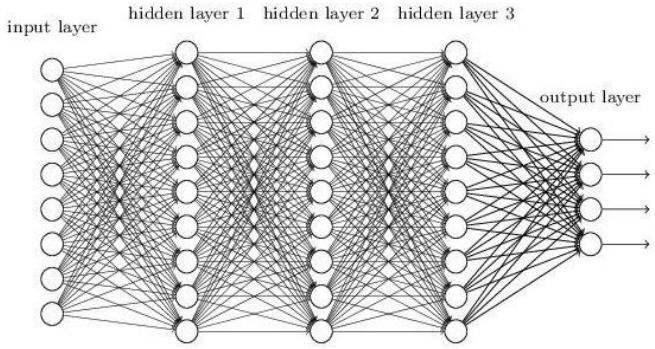


Figure 25 ANN

In a CNN, each neuron in the hidden layer is connected to a small region of the input neurons.

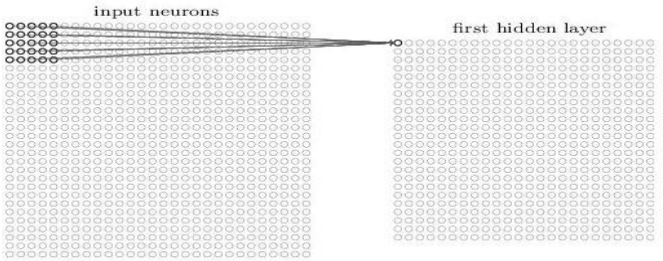


Figure 26 CNN

Why not use regular ANNs for image tasks?
For small images, it might work, but for large images the number of pixels are so many leading to millions of connections between neurons, leading to intractable solutions.

3.4 Most popular Convolutional Neural Network architectures (CNN)

Over the years, many variants of this basic architecture have been developed such as:

- LeNet-5 (1998)
- AlexNet (2012)
- ZFNet (2013)
- GoogLeNet (2014)
- VGGNet(2014)
- ResNet (2015)

LeNet-5

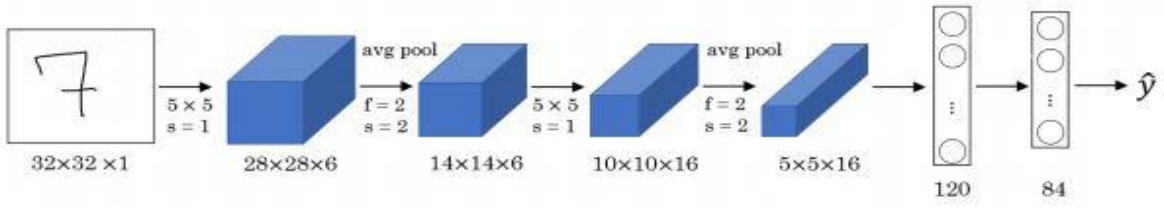


Figure 27 LeNet-5

- The dimensions of the image decreases as the number of channels increases.
- Conv ==> Pool ==> Conv ==> Pool ==> FC ==> FC ==> softmax this type of arrangement is quite common.
- The activation function used in the paper was Sigmoid and Tanh. Modern implementation uses RELU in most of the cases.
- It has 60k parameters.

AlexNet

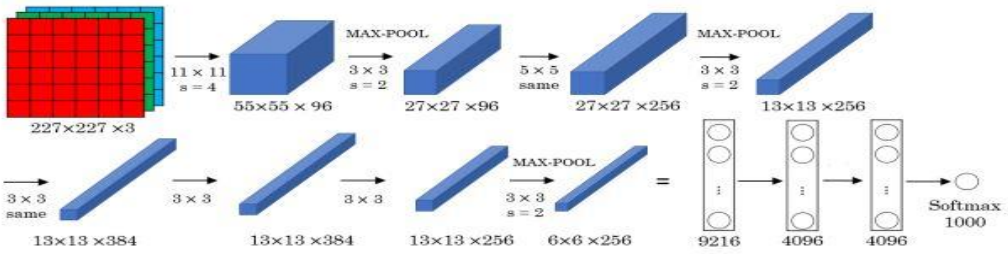


Figure 28 AlexNet

- The goal for the model was the ImageNet challenge which classifies images into 1000 classes.
- Conv => Max-pool => Conv => Max-pool => Conv => Conv => Conv => Max-pool ==> Flatten ==> FC ==> F
- Similar to LeNet-5 but bigger.
- Has 60 Million parameter compared to 60k parameter of LeNet-5.
- It used the RELU activation function.

VGG-16

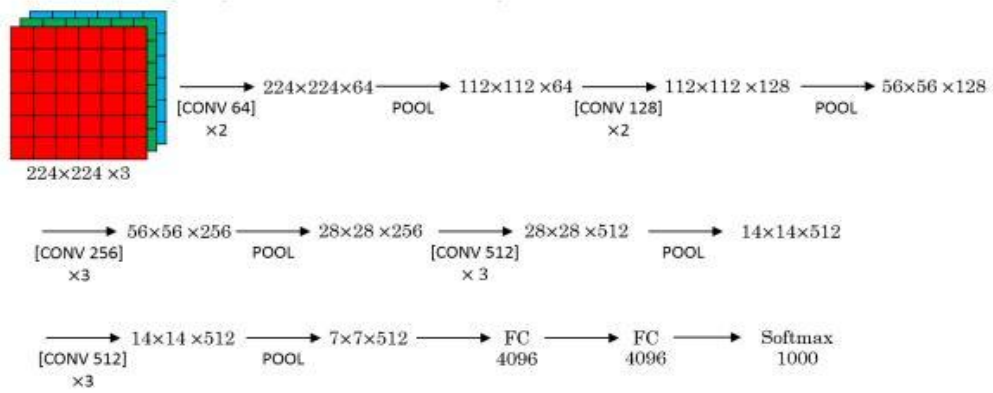


Figure 29 VGG-16

- This network is large even by modern standards. It has around 138 million parameters.
- Most of the parameters are in the fully connected layers.
- There are another version called VGG-19 which is a bigger version. But most people uses the VGG-16 instead of the VGG-19 because it does the same.

Residual Networks (ResNets)

- Very, very deep NNs are difficult to train because of vanishing and exploding gradients problems.
- Residual networks make us train the large deep NNs by skipping connections in some layers and taking the activation from one layer to feed it to another layer.

Residual block

- ResNets are built out of some Residual blocks.

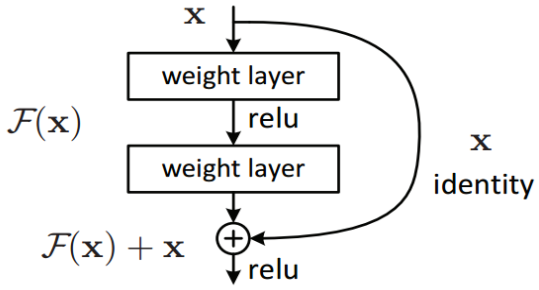
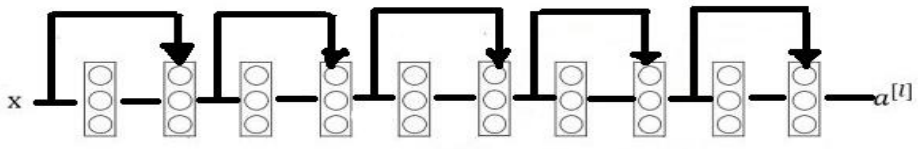


Figure 30 Residual Networks

- They add a shortcut/skip connection before the second activation.

Residual Network

- Are a NN that consists of some Residual blocks.



3.5 Convolutional Neural Network (CNN) main blocks

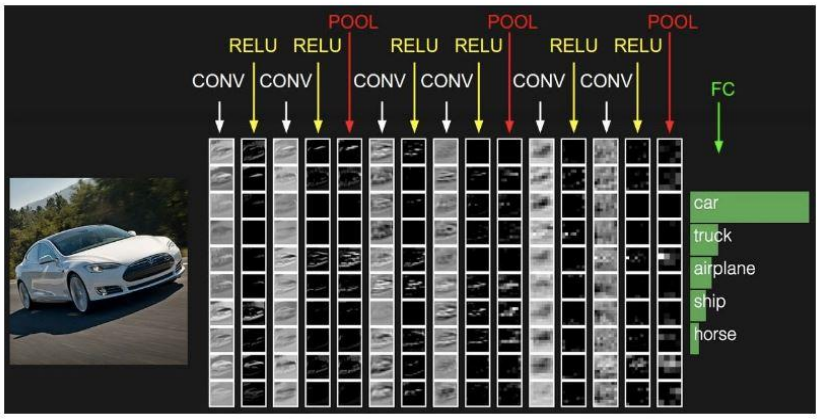


Figure 31 CNN Architecture

CNN architecture has the following configuration:

- Convolutional Layer
- Pooling Layer
- Fully connected Layer

Most normal configuration is the convolutional layer, followed by ReLU layer, followed by a pooling layer. The sets then keep repeating.
The final layer is the fully connected layer, which precedes the final image classification.

3.5.1 Convolutional Layer

Here neurons in the first hidden/convolutional layer are connected only to a subset of neurons in the prior layer, and so on in the next convolutional layer. This area in the input image is called Receptive Field.

CNN layers with rectangular local receptive fields are shown below:

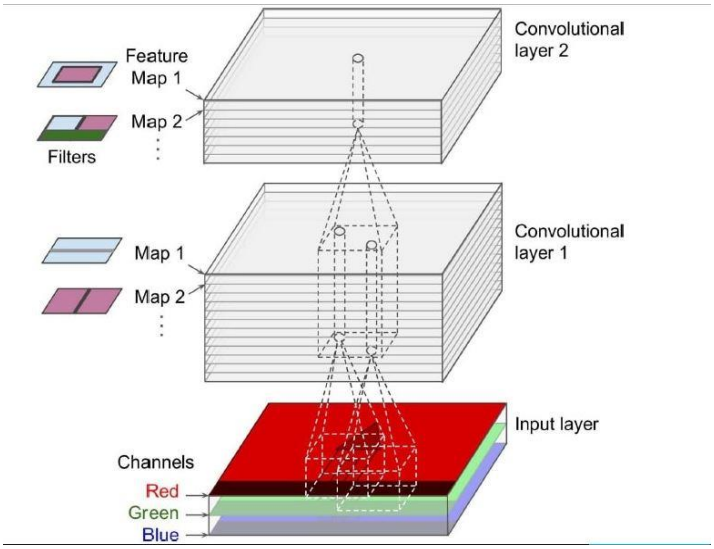


Figure 32 CNN layers with rectangular local receptive

High And Low-level Features

First, the hidden layer focuses on lower-level features and the later layers assemble them into higher-level features.

Consider the image of a dog whose features can be categorized as:

- High-level features - Eyes/ears/mouth/paws
- Low-level features - Edges/curves

This is done by 3 techniques: Vertical edge detection, horizontal edge detection and full edge detection.

Vertical And Horizontal Filters

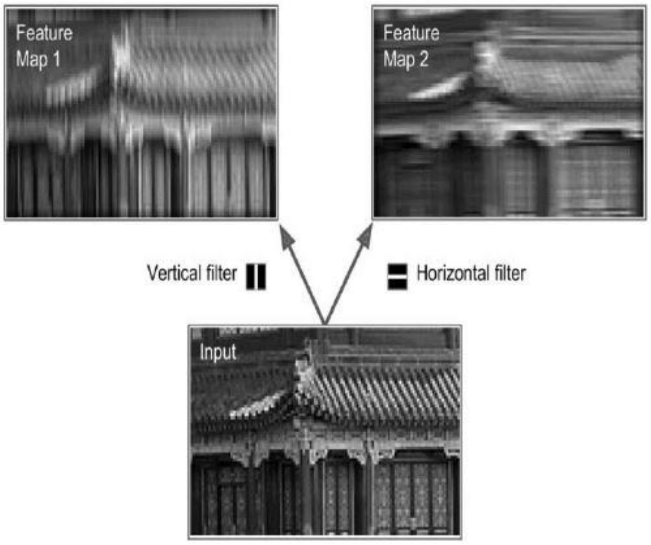
Example:

The image shows two kernels – vertical and horizontal filters. Each is a 5x5 matrix with all 0s, except 1 in the vertical line for vertical filter and 1 in a horizontal line in a horizontal filter

.

Vertical Filter:

0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0



Horizontal Filter:

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0

The effect of multiplying with vertical kernel filter is that all pixels except the vertical lines get subdued. Similarly, with horizontal kernel filter, it accentuates the horizontal lines.

The output image has a feature map, which highlights the areas in the image that are most similar to the filter.

In this fashion, a CNN finds low-level features first and then combines them in higher layers to detect complex features at points where both filters are active.

In the next section, let us focus on Zero Padding to adjust the sizes of the output of each layer.

3.5.2 Zero Padding

A matrix nxn is convolved with fxf filter/kernel give us n-f+1,n-f+1 matrix.

Ex: 6 × 6 matrix convolved with 3 × 3 filter/kernel results in a 4 × 4 matrix, so we see that the size shrinks as convolution is done so to apply this convolution operation multiple times without having effect on the size of the output matrix padding is needed and in the following section padding is introduced in details with an example.

Padding is done by adding some rows and columns to the image in the top, bottom, right and left of the image, and usually the values of these extra values are zero.
Example: if a matrix nxn is convolved with fxf filter/kernel and padding p give us
n+2p-f+1 × n+2p-f+1 matrix.
e.g.: if n=6, f=3 and p=1 output image will have n+2p-f+1 × n+2p-f+1 matrix (6 × 6)
(6+2× 1 − 3 + 1 = 6).
Same convolutions uses padding -of size (P = (f-1) /2) where f is the size of filter/kernel- to have the **same** size in the output as in input, whereas Valid convolutions don't use padding so size is affected.

Strided convolution

When we are making the convolution operation we use S to tell us the number of pixels we will jump when we are convolving filter/kernel.
Example: if a matrix nxn is convolved with fxf filter/kernel and padding p and stride S give us
(n+2p-f) /S+1 × (n+2p-f) /S+1 matrix.
e.g.: if n=6, f=3, p=1 and S=1 output image will have (n+2p-f) /S+1 × (n+2p-f) /S+1 matrix (6 × 6)
(6+2× 1 − 3 + 1 = 6).
In case (n+2p-f)/s + 1 is fraction we can take floor of this value.

Process of Convolution

Imagine a small patch being slid across the input image. This sliding is called convolving.
This patch is called the filter/kernel. The area under the filter is the receptive field.
The idea is to detect local features in a smaller section of the input space, this receptive field slides and convolve section by section to eventually cover the entire image.
In other words, the CNN layer neurons depend only on nearby neurons from the previous layer. This has the impact of discovering the features in a certain limited area of the input feature map.

Example

Assume the filter/kernel is a weight matrix "wk". For example, let's assume a 3X3 weighted matrix.

The weight matrix is a filter to extract some particular features from the original image. It could be for extracting curves, identifying a specific color, or recognizing a particular voice.

0	1	1
1	0	0
1	0	1

Assume the input to be a 6X6 image.

As the filter/kernel is slide across the input layer, the convolved output layer is obtained by adding the values obtained by element-wise multiplication of the weight matrix.

81	2	209	44	71	58
24	56	108	98	12	112
91	0	189	65	79	232
12	0	0	5	1	71
2	32	23	58	8	209
49	98	81	112	54	9

Input layer:

81	2	209	44	71	58
24	56	108	98	12	112
91	0	189	65	79	232
12	0	0	5	1	71
2	32	23	58	8	209
49	98	81	112	54	9

Filter/Kernel (Weighted matrix):

0	1	1
1	0	0
1	0	1

Output

515			

For example, when the weighted matrix starts from the top left corner of the input layer, the output value is calculated as:
(81x0+2x1+209x1)+(24x1+56x0+108X0)+(91x1+0x0+189x1) = 515
The filter then moves by 1 pixel to the next receptive field and the process is repeated. The output layer obtained after the filter slides over the entire image would be a 4X4 matrix.
This is called an activation map/ feature map.

3.5.3 Bias

it is a real number added to all the elements in the resulted window from the convolution operation. and it's used to improve the model accuracy.

3.5.4 Dropout

```
model.add(keras.layers.Dropout(rate=0.3))
```

Dropout consists in randomly setting a fraction `rate` of input units to used value at each update during training time, which helps prevent overfitting.

Arguments

- **rate**: float between 0 and 1. Fraction of the input units to drop.

3.5.5 Activation functions

Activation functions adds –or introduce- some non lineareties to the netword to make it learn the complicated functional mappings between inputs and outputs. Their main purpose is to convert a input signal of a node in a A-NN to an output signal to be entered to the next layer.

It is applied after summing the product of weights and pixels' values.

If we do not apply a Activation function then the output signal would simply be a simple linear function.A linear function is just a polynomial of **one degree.** A Neural Network without Activation function would simply be a **Linear regression Model,** which has limited power and does not performs good most of the times.

And since we need a Neural Network Model to learn and represent almost anything and any arbitrary complex function which maps inputs to outputs, activation functions are needed in the model, and in the following section the types of the used functions will be introduced with their details.

Most popular types of Activation functions

1. Sigmoid or Logistic
2. ReLu -Rectified linear units

Sigmoid Activation function: It is a activation function of form f(x) = 1 / (1 + exp(-x)). Its Range is between 0 and 1. It is a S — shaped curve. It is easy to understand and apply but it has major reasons which have made it fall out of popularity

- Vanishing gradient problem
- Secondly , its output isn't zero centered. It makes the gradient updates go too far in different directions. 0 < output < 1, and it makes optimization harder.
- Sigmoids saturate and kill gradients.
- Sigmoids have slow convergence.

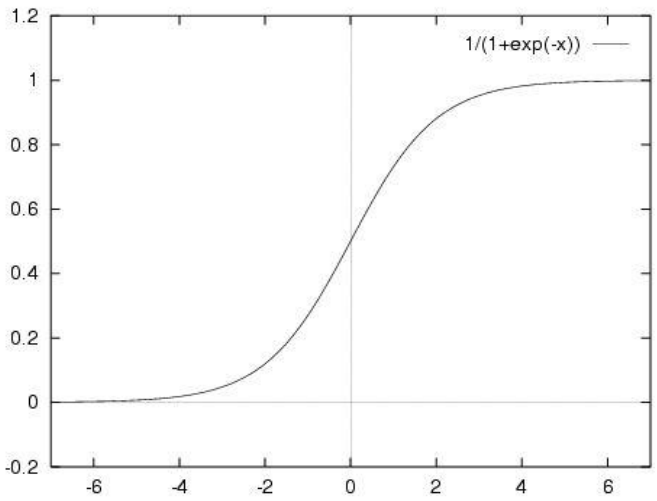


Figure 33 Sigmoid Activation function

Rectified Linear Unit (ReLU) transform function only activates a node if the input is above a certain quantity, while the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable.

it avoids and rectifies **vanishing gradient** problem . Almost all deep learning Models use **ReLu** nowadays. But its limitation is that it should only be used within Hidden layers of a Neural Network Model.

Hence for output layers we should use a Softmax function for a Classification problem to compute the probabilites for the classes, and for a regression problem it should simply use a linear function.

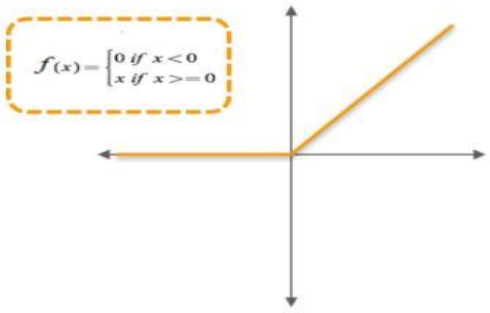


Figure 34 ReLU Activation function

In the shown figure,an illustrated example:

It made the negative values equal zero, and when it came to the positive values it performed a linear relationship

x	f(x)=x	F(x)
-3	f(-3) = 0	0
-5	f(-5) = 0	0
3	f(3) = 3	3
5	f(5) = 5	5

Another example for the RELU function is shown below:

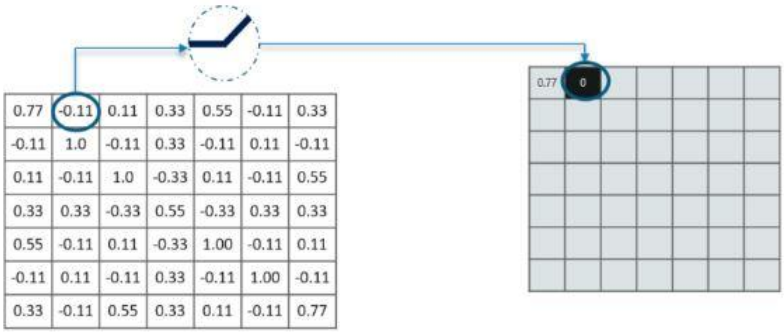


Figure 35 ReLU in Process

3.5.6 Pooling Layer

In this layer we shrink the image stack into a smaller size. Pooling is done after passing through the activation layer. We do this by implementing the following 4 steps:

- Pick a window size (usually 2 or 3)
- Pick a stride (usually 2)
- Walk your window across your filtered images
- From each window**,** take the maximum value for the maximum pooling or take the average in the average pooling

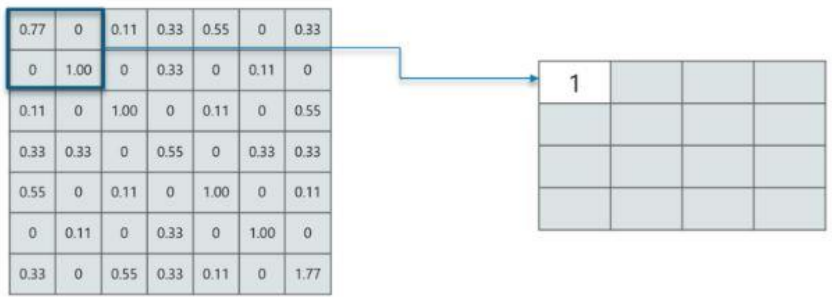
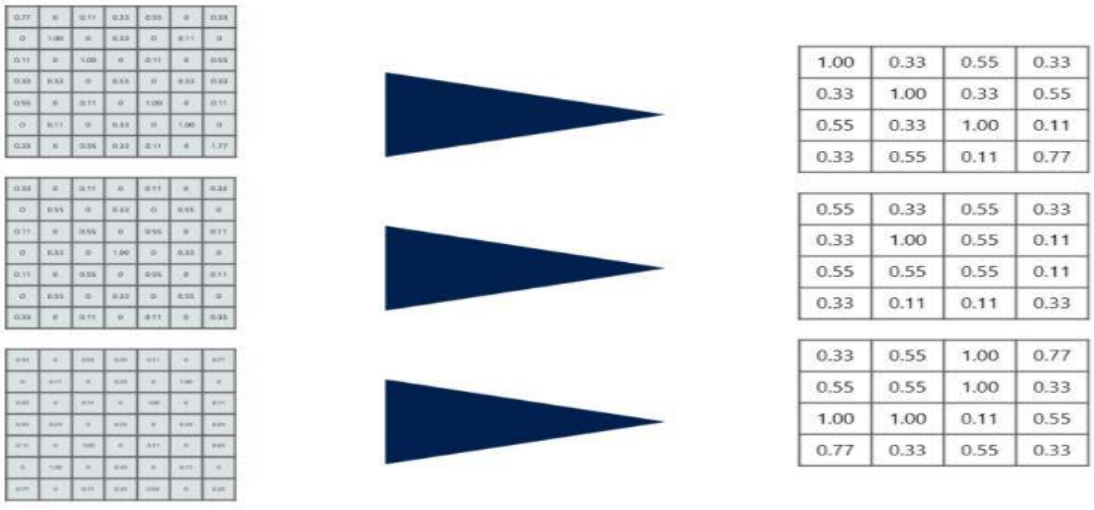


Figure 36 Pooling layer

Then apply this technique on the 3 color layers (RGB) to get the following:



3.5.7 Fully connected layer

The last layers in the network are **fully connected,** Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. It is worth noting that the only difference between FC and CONV layers is that the neurons in the CONV layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical.

Also, fully connected layer is the final layer where the classification actually happens. Here we take our filtered and shrinked images and put them into one single list as shown below:

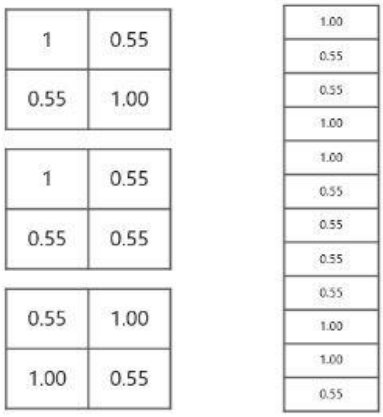


Figure 37 Image Flattening

3.6 Tools and libraries in the model

- Kaggle is used as an IDE (integrated development environment) to train the model and upload the training and test sets on it.
- Keras is used as a high-level neural networks API

3.6.1 About Keras

- Keras is written in Python and capable of running on top of TensorFlow.
- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

3.6.2 Libraries:

- numpy as it is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

```
x_train.append((np.array(temp) - np.mean(temp)) / np.std(temp))
x_train.append((np.array(temp.rotate(90)) - np.mean(temp)) / np.std(temp))
x_train.append((np.array(temp.rotate(180)) - np.mean(temp)) / np.std(temp))
x_train.append((np.array(temp.rotate(270)) - np.mean(temp)) / np.std(temp))
```

- Pandas as it is for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- PIL as it adds support for opening, manipulating, and saving many different image file formats.

```
from PIL import Image as img
```

In[5]:
```
img.open("../input/lastt-isa/trai_n/trai_n/bump (1).jpg").resize((64, 64)).convert("L")
```

- glob as our model needs to look for a list of files on the filesystem with names matching a pattern – images are named with bump and nonbump and numbered-

In[]:
```
bump_train_list = glob.glob("../input/traindata/train/train/bump*.jpg")
nonbump_train_list = glob.glob("../input/traindata/train/train/nonbump (*.jpg")
```

- tqdm to use loops in smart way.

```
for i in tqdm(bump_train_list):
    temp = img.open(i).resize((64, 64))
    temp = temp.convert("L")
```

- matplotlib.pyplot to plot accuracy of training.

3.6.3 Evaluation

As it's mentioned before, in this step we test the model by entering some test set to it and make it predict the class of the object inside it and calculate the accuracy by comparing the model's results by the real ones.

The test is done on 100 images and the accuracy is found 90%

3.7 Used model and results

here we import the mentioned used libraries:

In[1]:
```
from IPython.display import HTML
import keras
import numpy as np
import pandas as pd
from PIL import Image as img
import base64
```

```
Using TensorFlow backend.
```

In[2]:
```
import glob
from tqdm import tqdm
import matplotlib.pyplot as plt
```

Here we used (keras.backend.image_data_format) to return the default image data format convention.

In[3]:
```
keras.backend.image_data_format()
```
Out[3]:
```
'channels_last'
```

```
keras.backend.set_image_data_format("channels_first")
keras.backend.image_data_format()
```
Out[36]:
```
'channels_first'
```

Here is to open one image of those which have speed bumps, resize it, convert it to grey scale and display it just to see it:

In[5]:
```
img.open("../input/data-1/train/train/bump (1).jpg").resize((64, 64)).convert("L")
```
Out[5]:

Here is to open one image of those which don't have speed bumps, resize it, convert it to grey scale and display it just to see it:

In[6]:
```
img.open("../input/data-1/train/train/unbump (128).jpg").resize((64, 64)).convert("L")
```
Out[6]:

here to follow the pattern of images numbers to store them in list called bump_train_list and unbump_train_list to perform operation after that on them:

In[7]:
```
bump_train_list = glob.glob("../input/data-1/train/train/bump (*.jpg")
unbump_train_list = glob.glob("../input/data-1/train/train/unbump (*.jpg")
```

Here to open all the bump images, resize them to be 64*64 and convert them to grey scale then perform image augmentation on them to increase the number of training set entered then store the final list in x_train and create y_train and store 1 in it for all the images to be labeled by 1 (bump images label = 1):

In[8]:

```
x_train = []

for i in tqdm(bump_train_list):
    temp = img.open(i).resize((64, 64))
    temp = temp.convert("L")

    x_train.append((np.array(temp) - np.mean(temp)) / np.std(temp))
    x_train.append((np.array(temp.rotate(90)) - np.mean(temp)) / np.std(temp))
    x_train.append((np.array(temp.rotate(180)) - np.mean(temp)) / np.std(temp))
    x_train.append((np.array(temp.rotate(270)) - np.mean(temp)) / np.std(temp))

#    if not idx % 200:
#        print(idx)

y_train = np.tile(1, len(bump_train_list)*4)
print("bump's images loading is done")
```

```
100%|██████████| 871/871 [00:11<00:00, 74.29it/s]

bump's images loading is done
```

Here to open all the bump images, resize them to be 64*64 and convert them to grey scale then perform image augmentation on them to increase the number of training set entered then store the final list in x_train and create y_train and store 1 in it for all the images to be labeled by 1 (bump images label = 1):

In[9]:

```
for i in tqdm(unbump_train_list):
    temp = img.open(i).resize((64, 64))
    temp = temp.convert("L")

    x_train.append((np.array(temp) - np.mean(temp)) / np.std(temp))
    x_train.append((np.array(temp.rotate(90)) - np.mean(temp)) / np.std(temp))
    x_train.append((np.array(temp.rotate(180)) - np.mean(temp)) / np.std(temp))
    x_train.append((np.array(temp.rotate(270)) - np.mean(temp)) / np.std(temp))

y_train = np.concatenate((y_train, np.tile(0, len(unbump_train_list)*4))).astype("uint8")
print("unbump's images loading is done")
```

```
100%|██████████| 1000/1000 [02:24<00:00,  5.29it/s]

unbump's images loading is done
```

Here to convert x_train to an array:

In[10]:

```
a = np.asarray(x_train)
x_train = a.reshape(a.shape[0], 1, a.shape[1], a.shape[2])
```

To delete a array:

```
In[11]: del(a)
```

to display the final resulted images after augmentation and the dimensions of each one after resizing:

```
In[12]: x_train.shape
Out[12]: (7484, 1, 64, 64)
```

To perform Relu function on the images by setting `f(x) = alpha * x for x < 0`, `f(x) = x for x >= 0`:

```
In[13]: LeakyReLU = keras.layers.LeakyReLU(alpha=0.01)
```

To construct the model layers:

1st layer:

- convolution with 32 filters each of size 2*2
- perform Relu function on them.
- Add dropout with rate= 0.3

2nd layer:

- convolution with 32 filters each of size 3*3
- perform Relu function on them.
- Add dropout with rate= 0.3

3rd layer:

- convolution with 64 filters each of size 3*3
- perform Relu function on them.
- Add maxpooling layer.
- Add dropout with rate= 0.3

4th layer:

- Flatten the output of the latter layer to feed it as 1D array for the fully connected layer.
- Perform Dense for the output to be (output = activation(dot(input, kernel) + bias))

In[14]:

```
model = keras.models.Sequential()

model.add(keras.layers.Conv2D(filters=32, kernel_size=(2, 2), input_shape=(1, 64, 64)))
model.add(LeakyReLU)
model.add(keras.layers.Dropout(rate=0.3))

model.add(keras.layers.Conv2D(filters=32, kernel_size=(3, 3)))
model.add(LeakyReLU)
model.add(keras.layers.Dropout(rate=0.3))

model.add(keras.layers.Conv2D(filters=64, activation="relu", kernel_size=(3, 3)))
model.add(keras.layers.MaxPooling2D(pool_size=(3, 3)))
model.add(keras.layers.Dropout(rate=0.3))

model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(units=12, activation="relu"))
model.add(keras.layers.Dense(units=1, activation="sigmoid"))
```

To use an optimizer as it is one of the two arguments required for compiling a Keras model:

In[15]:

```
model.compile(optimizer=keras.optimizers.SGD(), loss=keras.losses.binary_crossentropy, metrics=["binary_accuracy"])
```

To display the summary of model:

- As the input size decreases over layers while the number of filters increases.
- A CNN usually consists of one or more convolution
- followed by a pooling.
- Fully connected layers has the most parameters in the network.

In[16]:

```
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 63, 63)        160
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    multiple                  0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 63, 63)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 61, 61)        9248
_________________________________________________________________
dropout_2 (Dropout)          (None, 32, 61, 61)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 59, 59)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 19, 19)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 64, 19, 19)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 23104)             0
_________________________________________________________________
dense_1 (Dense)              (None, 12)                277260
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 13
=================================================================
Total params: 305,177
Trainable params: 305,177
```

here to divide the training set to 10 epochs to be easy for the model to train and learn from them, and to result the training accuracy:

In[17]:
```
model.fit(x=x_train, y=y_train, epochs=10, validation_split=0.1, shuffle=True)
```

```
Train on 6735 samples, validate on 749 samples
Epoch 1/10
6735/6735 [==============================] - 5s 725us/step - loss: 0.6528 - binary_accuracy: 0.6003 - val_loss: 0.5317
- val_binary_accuracy: 0.7437
Epoch 2/10
6735/6735 [==============================] - 2s 313us/step - loss: 0.5618 - binary_accuracy: 0.7137 - val_loss: 0.7710
- val_binary_accuracy: 0.4526
Epoch 3/10
6735/6735 [==============================] - 2s 318us/step - loss: 0.4876 - binary_accuracy: 0.7724 - val_loss: 0.5815
- val_binary_accuracy: 0.6382
Epoch 4/10
6735/6735 [==============================] - 2s 315us/step - loss: 0.4416 - binary_accuracy: 0.8040 - val_loss: 0.7168
- val_binary_accuracy: 0.5634
Epoch 5/10
6735/6735 [==============================] - 2s 315us/step - loss: 0.4088 - binary_accuracy: 0.8254 - val_loss: 0.6950
- val_binary_accuracy: 0.5821
Epoch 6/10
6735/6735 [==============================] - 2s 314us/step - loss: 0.3894 - binary_accuracy: 0.8330 - val_loss: 0.6785
- val_binary_accuracy: 0.5861
Epoch 7/10
6735/6735 [==============================] - 2s 314us/step - loss: 0.3656 - binary_accuracy: 0.8448 - val_loss: 0.5210
- val_binary_accuracy: 0.6969
Epoch 8/10
6735/6735 [==============================] - 2s 313us/step - loss: 0.3583 - binary_accuracy: 0.8462 - val_loss: 0.4931
- val_binary_accuracy: 0.7156
```

```
Epoch 8/10
6735/6735 [==============================] - 2s 313us/step - loss: 0.3583 - binary_accuracy: 0.8462 - val_loss: 0.4931
- val_binary_accuracy: 0.7156
Epoch 9/10
6735/6735 [==============================] - 2s 316us/step - loss: 0.3384 - binary_accuracy: 0.8572 - val_loss: 0.3292
- val_binary_accuracy: 0.8144
Epoch 10/10
6735/6735 [==============================] - 2s 315us/step - loss: 0.3281 - binary_accuracy: 0.8595 - val_loss: 0.4779
- val_binary_accuracy: 0.7156
```

To save the weights and load them to the model:

In[18]:
```
model.save("bump_unbump_model_01.h5")
```

In[19]:
```
model = keras.models.load_model("bump_unbump_model_01.h5")
```

here to divide the training set to 2 epochs to be easy for the model to train and learn from them, and to result the training accuracy:

```
In[20]: model.fit(x=x_train, y=y_train, epochs=2, validation_split=0.1, shuffle=True)
```

```
Train on 6735 samples, validate on 749 samples
Epoch 1/2
6735/6735 [==============================] - 2s 350us/step - loss: 0.3316 - binary_accuracy: 0.8555 - val_loss: 0.2669
- val_binary_accuracy: 0.8385
Epoch 2/2
6735/6735 [==============================] - 2s 311us/step - loss: 0.3050 - binary_accuracy: 0.8656 - val_loss: 0.2811
- val_binary_accuracy: 0.8438
Out[20]:
<keras.callbacks.History at 0x7fca955e9128>
```

To display the history of the training accuracy:

```
In[21]: len(model.history.history["binary_accuracy"])
Out[21]:
2
```

```
In[22]: np.arange(1, len(model.history.history["binary_accuracy"])+1, 1)
Out[22]:
array([1, 2])
```

To plot the training accuracy and loss as shown:

```
In[23]:
plt.figure(figsize=(20, 7))
plt.subplot(1, 2, 1)
plt.plot(model.history.history["binary_accuracy"])
plt.plot(model.history.history["val_binary_accuracy"])
plt.title("Model Accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Train", "Val"], loc="upper left")
#plt.xticks(np.arange(0, len(model.history.history["binary_accuracy"]), 1))

plt.xticks(np.arange(len(model.history.history["binary_accuracy"])), np.arange(1, len(model.history.history["binary_acc

plt.subplot(1, 2, 2)
plt.plot(model.history.history["loss"])
plt.plot(model.history.history["val_loss"])
plt.title("Model Loss")
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.legend(["Train", "Val"], loc="upper right")
plt.xticks(np.arange(len(model.history.history["loss"])), np.arange(1, len(model.history.history["loss"])+1, 1))
plt.show()
```

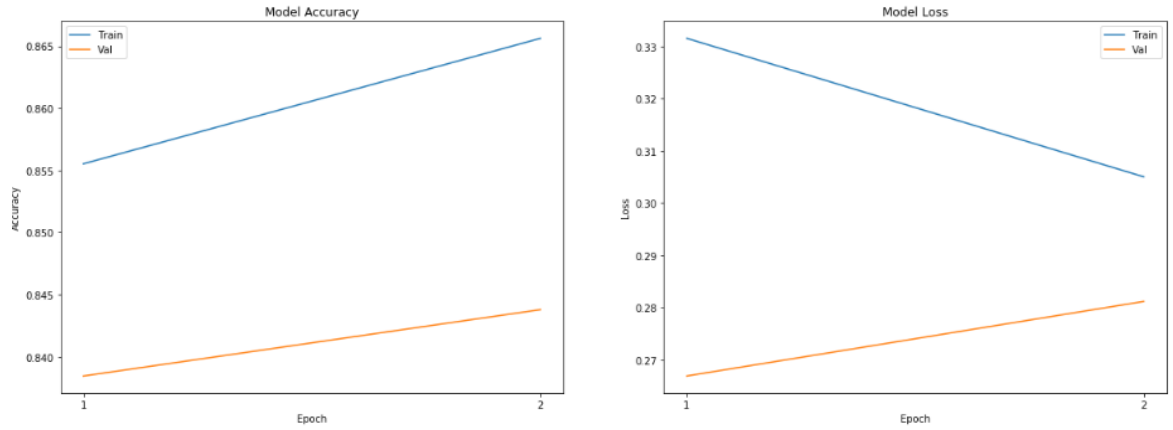


Figure 38 Model Accuracy and Loss

here to follow the pattern of test images numbers to store them in list called test_list to perform operation after that on them:

In[24]:

```
test_list = glob.glob("../input/wednesday/test1/test/test (*.jpg")
```

Here to open all the bump images, resize them to be 64*64 and convert them to grey scale then store the final list in x_test:

In[25]:

```
x_test = []

for i in tqdm(test_list):
    temp = img.open(i).resize((64, 64))
    temp = temp.convert("L")
    x_test.append((np.array(temp) - np.mean(temp)) / np.std(temp))

print("test images loading is done")
```

```
100%|██████████| 59/59 [00:13<00:00,  4.44it/s]

test images loading is done
```

Here to convert x_test to an array:

In[26]:

```
a = np.asarray(x_test)
x_test = a.reshape(a.shape[0], 1, a.shape[1], a.shape[2])
```

To delete a array:

In[27]:
```
del(a)
```

To predict the test images (x_test) whether they have speed bumps or not and store the result:

In[28]:
```
result = model.predict(x=x_test)
```

In[29]:
```
idx = []
for i in test_list:
    idx.append(i[21:-4])
```

To ask if the resulted label for the test image greater than the threshold (0.5) set it as 1 (to be bump image) and if less than the threshold(0.5) set it as 0 (to be non bump image):

In[30]:
```
result = result.reshape(result.shape[0])
result[result>0.5] = 1
result[result<0.5] = 0
```

In[31]:
```
submission = {"id": idx, "label": result}
```

To store the final result in .csv file to be able to download it:

In[32]:
```
pd.DataFrame(submission).to_csv("submission.csv", index=False)
```

In[33]:
```
pd.DataFrame(submission)
```

Here is a function to give us a link to download the csv file containing the results:

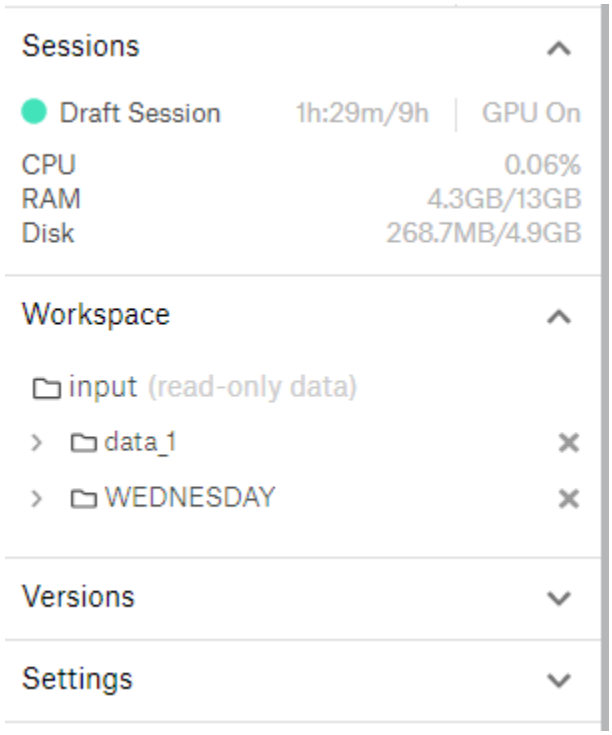
In[34]:

```
# function that takes in a dataframe and creates a text link to
# download it (will only work for files < 2MB or so)
def create_download_link(df, title = "Download CSV file", filename = "data.csv"):
    csv = df.to_csv()
    b64 = base64.b64encode(csv.encode())
    payload = b64.decode()
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}" target="_blank">{title}</a>'
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)

# create a random sample dataframe
df = pd.DataFrame(submission)

# create a link to download the dataframe
create_download_link(df)
```

Out[34]:

Download CSV file

Here's a screenshot for the used CPU, RAM and disk storage for the model on Kaggle:



Evaluation

As it's mentioned before, in this step we test the model by entering some test set to it and make it predict the class of the object inside it and calculate the accuracy by comparing the model's results by the real ones.

The test is done on 59 images and resulted in 90% accuracy.

3.8 The Raspberry Pi 3 Model B Board

The latest and most updated version of the Raspberry Pi series, namely, the Raspberry Pi 3 Model B Board; is used as the robot's "brain". It controls all other boards, establishes communications, receives instructions from the user's PC, and processes information from the camera that is compatible with it to take video from the road and divide it to frames about 100 frames to enter these images to the classification model on raspberry as the test to compute the accuracy .

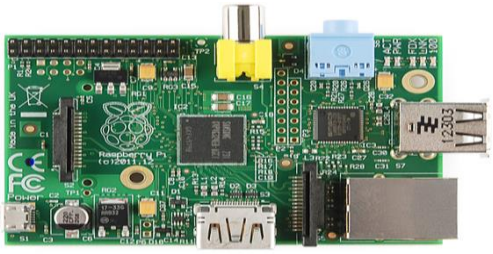


Figure 39 Raspberry pi 3 model B with the camera

Specifications of raspberry 3 model Specifications of the used Raspberry Pi 3 Model B board.

SoC	Broadcom BCM2837
CPU	4× ARM Cortex-A53, 1.2GHz
GPU	Broadcom VideoCore IV
RAM	1GB LPDDR2 (900 MHz)
Networking	10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth	Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage	microSD
GPIO	40-pin header, populated
Ports	HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

From these specifications, our project needs to port of camera serial interface to make the camera is on and takes the frames as the images for testing

Model on raspberry :

To run the model on raspberry ,that required install some libraries on it to able to run successfully. From these libraries Keras ,Numpy and Pandas

Here are instructions that used to install the libraries on raspberry

For Keras

sudo apt-get install python3-numpy
sudo apt-get install libblas-dev
sudo apt-get install liblapack-dev
sudo apt-get install python3-dev
sudo apt-get install libatlas-base-dev
sudo apt-get install gfortran
sudo apt-get install python3-setuptools
sudo apt-get install python3-scipy
sudo apt-get update
sudo apt-get install python3-h5py
sudo pip3 install keras

Test the installation by
python -c 'import keras; print(keras.__version__)'# python 2
python3 -c 'import keras; print(keras.__version__)' # for Python 3

for numpy

sudo apt-get install python3-numpy or

pip3 install numpy

for pandas

sudo apt-get install python3-pandas

for pillow /pil

(sudo) pip3 install pillow

After installing all required libraries we can apply this code to capture the video and divide it into frames

```
import cv2
if __name__ == "__main__" :
    cap=cv2.VideoCapture('project.mp4')
    ret,img=cap.read()
    count=20
    x=0
    while ret :
        if count == 20: #number of frame dropped (not saved)
            x=x+1
            cv2.imwrite('path of the image '+str(x)+'.jpg',img)
            count = 0
        ret,img=cap.read()
        count=count+1

    cap.release()
    cv2.destroyAllWindows()
```

And adding the classification model code to train and test on raspberry and this is the capture from the model

```
model = keras.models.Sequential()

model.add(keras.layers.Conv2D(filters=32, kernel_size=(2, 2), input_shape=(1, 64, 64)))
model.add(LeakyReLU)
model.add(keras.layers.Dropout(rate=0.3))

model.add(keras.layers.Conv2D(filters=32, kernel_size=(3, 3)))
model.add(LeakyReLU)
model.add(keras.layers.Dropout(rate=0.3))

model.add(keras.layers.Conv2D(filters=64, activation="relu", kernel_size=(3, 3)))
model.add(keras.layers.MaxPooling2D(pool_size=(3, 3)))
model.add(keras.layers.Dropout(rate=0.3))

model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(units=12, activation="relu"))
model.add(keras.layers.Dense(units=1, activation="sigmoid"))
```

Conclusion:

By using the camera of raspberry that takes the video of the road and tests the frames as images to detect the object (bump) or not.

CHAPTER 4: COMMUNICATION PROTOCOLS

Vehicle-to-vehicle technology uses many technologies like WIFI, LTE and dedicated short-range communications (DSRC), the best of the is DSRC but the module isn't available so we used WIFI in order to transfer frames between cars.
In the second section of this chapter we will talk about Bluetooth that we used to control car movement remotely.

4.1 WIFI MODULE

4.1.1 About the module

ESP8266-01 is the module we used because it's available, inexpensive, compact and powerful Wi-Fi Module.

The module can operate in station mode, soft access point, and both at the same time

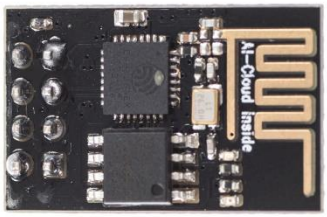


Figure 40 ESP8266

The module has a frequency range 2.4G - 2.5G, supporting different WIFI protocols 802.11.

The module supports WPA/WPA2 WIFI security and WEP/TKIP/AES encryption

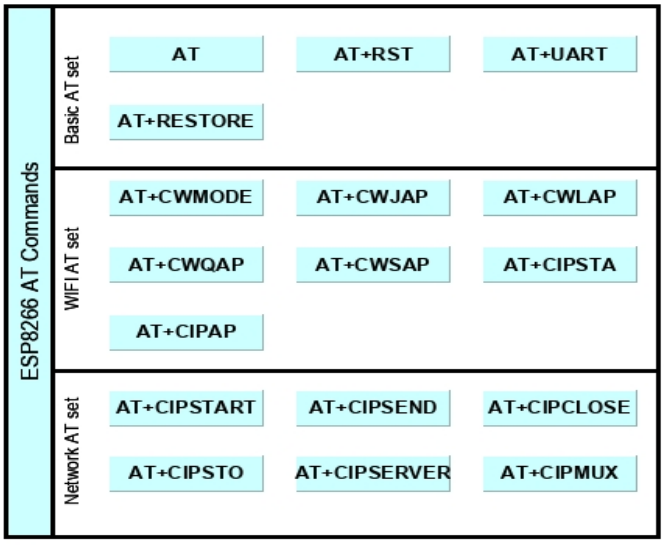
The module supports IPV4, TCP, UDP and HTTP.'

The module has a controller with 2 GPIO pins for small applications

The module operating voltage is 2.5V – 2.6 V, and operating current of 80mA on average

The module firmware can be loaded with UART by many programs like ESP8266Flasher and ESP8266 DownloadTool, this firmware determines the AT commands that the module will understand and the stability of the module, the firmware is made by LUA programming language so we could build our own firm ware but we didn't have to, there are many firmwares available on the internet, after many trails with the different firmwares the best firmware we found is ESP8266_NONOS_SDK.

This module has large set of AT commands at different layers we will discuss the ones we used.



Basic AT set

1- AT

this command is used to make sure that the module is working

AT - Test AT startup	
Response	OK
Parameters	null

2- AT+RST

This command is used to reset the module, some of the configuration is saved at the module's flash memory, and some don't get saved, this command erases the commands that got saved.

All the AT commands we use get stored at the flash memory, for example if we use AT+CWJAP This command join a certain access point, when we reset the module, after starting again the module will try to join this access point automatically, this is similar to "Connecting automatically" option that we choose when we connect to any WIFI from a laptop or mobile.

Resetting the module doesn't occur just due to this command, if the power fed to the module is low this module keeps resetting, every time we send an AT command to it, we noticed that from our work,

AT+RST - Restart module	
Response	OK
Parameters	null

3- AT+RESTORE

This command is used to erase the Flash memory bringing the factory configurations

AT+RESTORE - Factory reset	
Response	OK
Notes	Restore factory default settings. The chip will restart.

4- AT+UART

This command is used to set the UART configurations needed to interface it with the micro-controller

AT+UART=<baudrate>,<databits>,<stopbits>,<parity>,<flow control> This command is deprecated, please use AT+UART_CUR or AT+UART_DEF instead.		
Example	AT+UART=115200,8,1,0,3	
Response	OK	
Parameters	<baudrate>	Baudrate range: 110 to 115200*40 (4.608 Mega)
	<databits>	5: 5 bits data 6: 6 bits data 7: 7 bits data 8: 8 bits data
	<stopbits>	1: 1 bit stop bit 2: 1.5 bit stop bit 3: 2 bit stop bit
	<parity>	0: None 1: Odd 2: EVEN
	<flow control>	0: disable flow control 1: enable RTS 2: enable CTS 3: enable both RTS and CTS
Notes	• This configuration will also store the baudrate as the default rate in the user parameter area in the Flash for boot up. • Flow control needs hardware support: MTCK is UART0 CTS and MTDO is UART0 RTS.	

WIFI AT set

1- AT+CWMDOE

this command set mode of the operation either access point or station or both, each mode has his own separate IP

AT+CWMODE - WiFi mode	
This command is deprecated. Please use AT+CWMODE_CUR or AT+CWMODE_DEF instead.	
Command	**AT+CWMODE=?**
Response	+CWMODE:(value scope of <mode>) OK
Parameters	Please refer to AT command settings.
Command	**AT+CWMODE?**
Response	+CWMODE:<mode> OK
Parameters	Please refer to AT command settings.
Command	**AT+CWMODE=<mode>**
Response	OK
Parameters	<mode> 1 : station mode 2 : softAP mode 3 : softAP + station mode
Notes	This setting will be stored in the flash system parameter area. It won't be erased even when the power is off and restarted.

As mentioned before this module can operate in softAP + Station mode, but we discovered that it doesn't work in such mode, we will discuss this problem later on.

2- AT+CWJAP

This command is used when the module operate at station mode for joining an access point, similar to connecting to any WIFI from a mobile

This command requires the WIFI name and password, the response time and the response itself will vary depending on whether this network name exist or not and the password entered is correct or not

AT+CWJAP - Connect to AP [@deprecated]. Please use AT+CWJAP_CUR or AT+CWJAP_DEF instead.	
Example	• AT+CWJAP ="abc", "0123456789" • If SSID is "ab/,c" and password is "0123456789"/" AT+CWJAP ="ab///,c", "0123456789/"//" • If several APs have the same SSID as "abc",target AP can be found by bssid: AT+CWJAP ="abc","0123456789","ca:d7:19:d8:a6:44"
Command	**AT+CWJAP?**
Response	+CWJAP:<ssid>,<bssid>,<channel>,<rssi> OK
Parameters	<ssid> string, AP's SSID
Command	**AT+CWJAP=<ssid>,<pwd>[,<bssid>]**
Response	OK or +CWJAP:<error code> FAIL
Parameters	<ssid> string, AP's SSID <pwd> string, MAX: 64 bytes ASCII [<bssid>] string, AP's MAC address, for several APs may have the same SSID <error code> only for reference,it's not reliable <error code> 1 connecting timeout <error code> 2 wrong password <error code> 3 can not found target AP <error code> 4 connect fail This command needs station mode enable. Escape character syntax is needed if "SSID" or "password" contains any special characters (',', '"'and'/')
Notes	This configuration will store in Flash system parameter area.

3- AT+CWLAP

This command is used at station mode for listing nearby access points, this is similar to when you open the "WIFI-Networks" at any mobile it takes a second or 2 to list all the WIFI-Networks around you

AT+CWLAP - Lists available APs	
Example	• AT+CWLAP List of all available AP's detected by ESP8266 • AT+CWLAP="wifi","ca:d7:19:d8:a6:44",6 Find AP with specific SSID and MAC at specific channel. • AT+CWLAP="wifi" Find AP with specific SSID
Command	**AT+CWLAP=<ssid>,<mac>,<ch>**
Response	+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<ch>,<freq offset> OK ERROR
Parameters	<ecn> 0 OPEN 1 WEP 2 WPA_PSK 3 WPA2_PSK 4 WPA_WPA2_PSK <ssid> string, SSID of AP <rssi> signal strength <mac> string, MAC address <freq offset> frequency offset of AP, unit: KHz. <freq offset> / 2.4 to get unit "ppm"
Command	**AT+CWLAP**
Response	+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<ch>,<freq offset> OK ERROR
Parameters	The same as above

4- AT+CWQAP

This command is used at station mode to quit the access point that the module connected to

AT+CWQAP - Disconnect from AP	
Command	**AT+ CWQAP**
Response	OK
Parameters	null

5- AT+CWSAP

This command is used at softAP mode to set the IP of the access point

AT+ CWSAP - Configuration of softAP mode [@deprecated]. Please use AT+CWSAP_CUR or AT+CWSAP_DEF instead.	
Example	AT+CWSAP="ESP8266","1234567890",5,3
Command	**AT+CWSAP?**
Response	+CWSAP:<ssid>,<pwd>,<chl>,<ecn>
Parameters	<ssid> string, ESP8266 softAP' SSID <pwd> string, range: 8 ~ 64 bytes ASCII <chl> channel id <ecn> 0 OPEN 2 WPA_PSK 3 WPA2_PSK 4 WPA_WPA2_PSK
Command	**AT+CWSAP=<ssid>,<pwd>,<chl>,<ecn>**
Response	OK ERROR
Parameters	The same as above.
Notes	This CMD is only available when softAP mode enable. ESP8266 softAP don't support WEP. This configuration will store in Flash system parameter area.

6- AT+CIPSTA

this command changes the IP of the station mode of the module, and this change get stored into Flash memory

AT+ CIPSTA - Set IP address of ESP8266 station [@deprecated]. Please use AT+CIPSTA_CUR or AT+CIPSTA_DEF instead.	
Example	AT+CIPSTA="192.168.6.100","192.168.6.1","255.255.255.0"
Command	**AT+CIPSTA?**
Response	+CIPSTA:<IP> OK
Parameters	<IP> string, IP address of ESP8266 station
Command	**AT+CIPSTA=<IP>[,<gateway>,<netmask>]**
Response	OK
Parameters	<IP> string, IP address of ESP8266 station [<gateway>] gateway [<netmask>] netmask
Notes	This configuration interacts with AT+CWDHCP related AT commands: • If enable static IP, DHCP will be disabled; • If enable DHCP, static IP will be disabled; • This will depend on the last configuration.

7- AT+CIPAP

This command changes the IP of the access point, and this change get stored into Flash memory

AT+ CIPAP - Set IP address of ESP8266 softAP [@deprecated]. Please use AT+CIPAP_CUR or AT+CIPAP_DEF instead.	
Example	AT+CIPAP="192.168.5.1","192.168.5.1","255.255.255.0"
Command	**AT+CIPAP?**
Response	+CIPAP:<IP> OK
Parameters	<IP> string, IP address of ESP8266 softAP
Command	**AT+CIPAP=<IP>[,<gateway>,<netmask>]**
Response	OK
Parameters	<IP> string, IP address of ESP8266 softAP [<gateway>] gateway [<netmask>] netmask
Notes	This configuration interacts with DHCP related AT commands (AT+CWDHCP related): • If enable static IP, DHCP will be disabled; • If enable DHCP, static IP will be disabled; • This will depend on the last configuration.

Network AT set

1- AT+CIPSTART

This command is used to start a connection with other devices on the network, this connection could be TCP or UDP, in order for this command to work we have to guarantee few things

- Make sure the device we are trying to connect to exist on this network
- Know the IP of this device
- Know the port number of this device for TCP and UDP connections

This is similar to when we open a browser and type www.google.com, we are starting a connection with google servers

AT+CIPSTART - Establish TCP connection or register UDP port, start connection	
Example	AT+CIPSTART="TCP","192.168.101.110",1000
Single connection	(+CIPMUX=0) **AT+CIPSTART=<type>,<remote IP>,<remote port>[,<UDP local port>,<UDP mode>][,<TCP keep alive>]**
Multiple connection	(+CIPMUX=1) **AT+CIPSTART=<link ID>,<type>,<remote IP>,<remote port>[,<UDP local port>,<UDP mode>][,<TCP keep alive>]**
Response	OK or ERROR If connection already exists, returns ALREAY CONNECT
Parameters	<link ID> ID of the connection (0~4), for multi-connect <type> string, "TCP" or "UDP" <remote IP> string, remote IP <remote port> string, remote port [<UDP local port>] for UDP only [<UDP mode>] In UDP transparent transmission, it has to be 0. [<UDP mode>] 0 : destination peer entity of UDP will not change. [<UDP mode>] 1 : destination peer entity of UDP can change once. [<UDP mode>] 2 : destination peer entity of UDP is allowed to change. Note: [<UDP mode>] can only be used when [<UDP local port>] is set. [<TCP keep alive>] default 0. unit: 500 milliseconds. [<TCP keep alive>] 0 : disable TCP keep-alive [<TCP keep alive>] 1 ~ 7200 : TCP keep-alive interval

This command works if the module is operating at station mode or softAP mode but, it doesn't work if the module is operating at station + sotAP mode, we will discuss this problem in details later.

TCP protocol is based on circuit switching, which means we need to establish a link between the devices before sending, but UDP is based on packet switching so we don't need to build a link, then how is there a start connection command for both of them.

On the physical network there is, indeed, no connection needed. However, on socket object level they have deliberately made UDP communications look more like TCP connections. And, since the UDP "connection" is nothing more than sock object's abstraction, we use the word "connection" in quotation marks.

2- AT+CIPSEND

Once the connection has started this command is used to send the data over this link, this command takes some time to send the data during this time we can't send any at commands to the module

AT+CIPSEND - Send data	
Single connection	(+CIPMUX=0) **AT+CIPSEND=<length>**
Multiple connection	(+CIPMUX=1) **AT+CIPSEND=<link ID>,<length>**
UDP Transmission	**AT+CIPSEND=[<link ID>,]<length>[,<remote IP>,<remote port>]**
Response	Wrap return ">" after set command. Begins receive of serial data, when data length is met, starts transmission of data. If connection cannot be established or gets disconnected during send, returns ERROR If data is transmitted successfully, returns SEND OK
Parameters	<link ID> ID of the connection (0~4), for multi-connect <length> data length, MAX 2048 bytes [<remote IP>] UDP transmission can set remote IP when send data [<remote port>] UDP transmission can set remote port when send data
Command	**AT+CIPSEND**
Response	Wrap return ">" after execute command. Enters unvarnished transmission, 20ms interval between each packet, maximum 2048 bytes per packet. When single packet containing "+++" is received, it returns to normal command mode. This command can only be used in transparent transmission mode which require to be single connection mode. For UDP transparent transmission, <UDP mode> has to be 0 in command "AT +CIPSTART"

3- AT+CIPCLOSE

After the connection has started and the data got sent, we can't leave this TCP link active we need to close it, this is the command used for closing

AT+CIPCLOSE - Close TCP or UDP connection	
Multiple connection	**AT+CIPCLOSE=<link ID>**
Response	OK or ERROR
Parameters	<link ID> ID no. of connection to close, when ID=5, all connections will be closed. (ID=5 has no effect in server mode)
Single connection	**AT+CIPCLOSE**
Response	OK or If no such connection, returns ERROR

4- AT+CIPSTO

This command is used when the module operate as a server to set a connection timeout

AT+ CIPSTO - Set TCP server timeout	
Example	AT+CIPMUX=1 AT+CIPSERVER=1,1001 AT+CIPSTO=10
Command	**AT+CIPSTO?**
Response	+ CIPSTO:<time> OK
Parameters	The same as below.
Command	**AT+CIPSTO=<time>**
Response	OK
Parameters	<time> TCP server timeout, range 0~7200 seconds
Notes	ESP8266 as TCP server, will disconnect to TCP client that didn't communicate with it even if timeout. If AT+CIPSTO=0, it will never timeout. We don't recommend that.

5- AT+CIPSERVER

This is the command used to make the module a server on certain network so that other device could start connection with it, google company and other websites use something similar to this concept

AT+ CIPSTO - Set TCP server timeout	
Example	AT+CIPMUX=1 AT+CIPSERVER=1,1001 AT+CIPSTO=10
Command	**AT+CIPSTO?**
Response	+ CIPSTO:<time> OK
Parameters	The same as below.
Command	**AT+CIPSTO=<time>**
Response	OK
Parameters	<time> TCP server timeout, range 0~7200 seconds
Notes	ESP8266 as TCP server, will disconnect to TCP client that didn't communicate with it even if timeout. If AT+CIPSTO=0, it will never timeout. We don't recommend that.

6- AT+CIPMUX

This command is used to enable multiple connections, should be used if we are trying to set the module to operate as a server

AT+ CIPMUX - Enable multiple connections or not	
Example	AT+CIPMUX=1
Command	**AT+CIPMUX?**
Response	+ CIPMUX:<mode> OK
Parameters	<mode>0 single connection <mode>1 multiple connection
Command	**AT+CIPMUX=<mode>**
Response	OK If already connected, returns Link is builded
Parameters	The same as above.
Notes	1. "AT+CIPMUX=1" can only be set when transparent transmission disabled ("AT+CIPMODE=0") 2. This mode can only be changed after all connections are disconnected. 3. If TCP server is started, has to delete TCP server first, then change to single connection is allowed.

4.1.2 The local Network

The network between two of these modules, without an access point between them is the challenge in our project, in order to build such a network there are some commands that are useless and some commands will be more important.

In order to build a V2V communication with this module we need to have these module to send and receiver data from each other with minimum communication delay and no intermediate router or access point, so we came up with a work flow for the modules, to fulfill these requirements, but first let's discuss how to send a data from one WIFI module to another

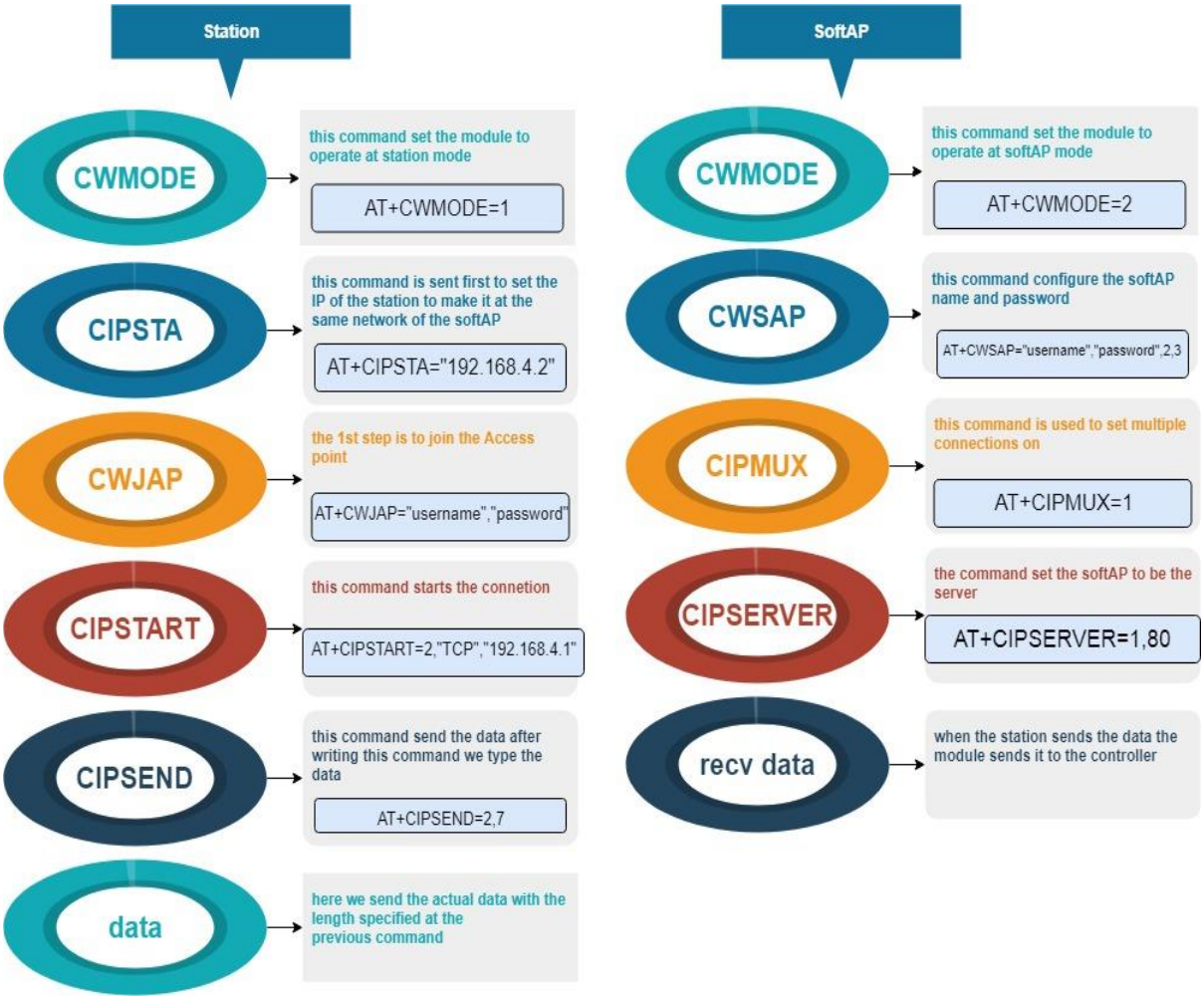


Figure 41 WIFI Send and Receive configurations

This figure shows the flow of AT commands at both the station and access point, first we set the mode of operation by the AT command "AT+CWMODE" if it equals 1 this means it operates as station, and If it equals 2 it operates as access point.

At the station module we send "AT+CIPSTA" to set the IP of the station to an IP that exist on the same network, because the Initial IP of the module as a station is "0.0.0.0", and the initial IP of the module as a softAP is "192.168.4.1".

Now that the mode and the IP are set we can start the communication by joining the Access point with the command "AT+CWJAP", we type the username and the password we typed at the softAP at the command "CWSAP", the other two parameters at "CWSAP" are to set the security level to WPA_2 and channel ID to 3.

After adding both of them are on the same network, both of them can now send to each other, one should be the server and the other should start "TCP" connection with that server, we choose that the station will start the connection and the softAP will be the server, because as we will show later the softAP mode will be the receiver part of the car and the station mode will be the sending part of the car, this way whenever the car wants to send it's frame to other cars it will join their access points and start a connection with their servers and send its data then close this connection, if we make the SoftAP the sending part this

means we need the other car to join this car at the moment we want to send the data, which is unguaranteed.

The Station module can now send the data to the softAP module using "AT+CIPSEND", the parameters are the channel ID and that has to be the same channel ID that the station started the connection at, it equals 2 in this example, the other parameter is the length of the data we want to send, it equals 7 at this example After sending this command to the module, the module is now waiting for us to send the data, after the data is sent, the module send it to the other module.

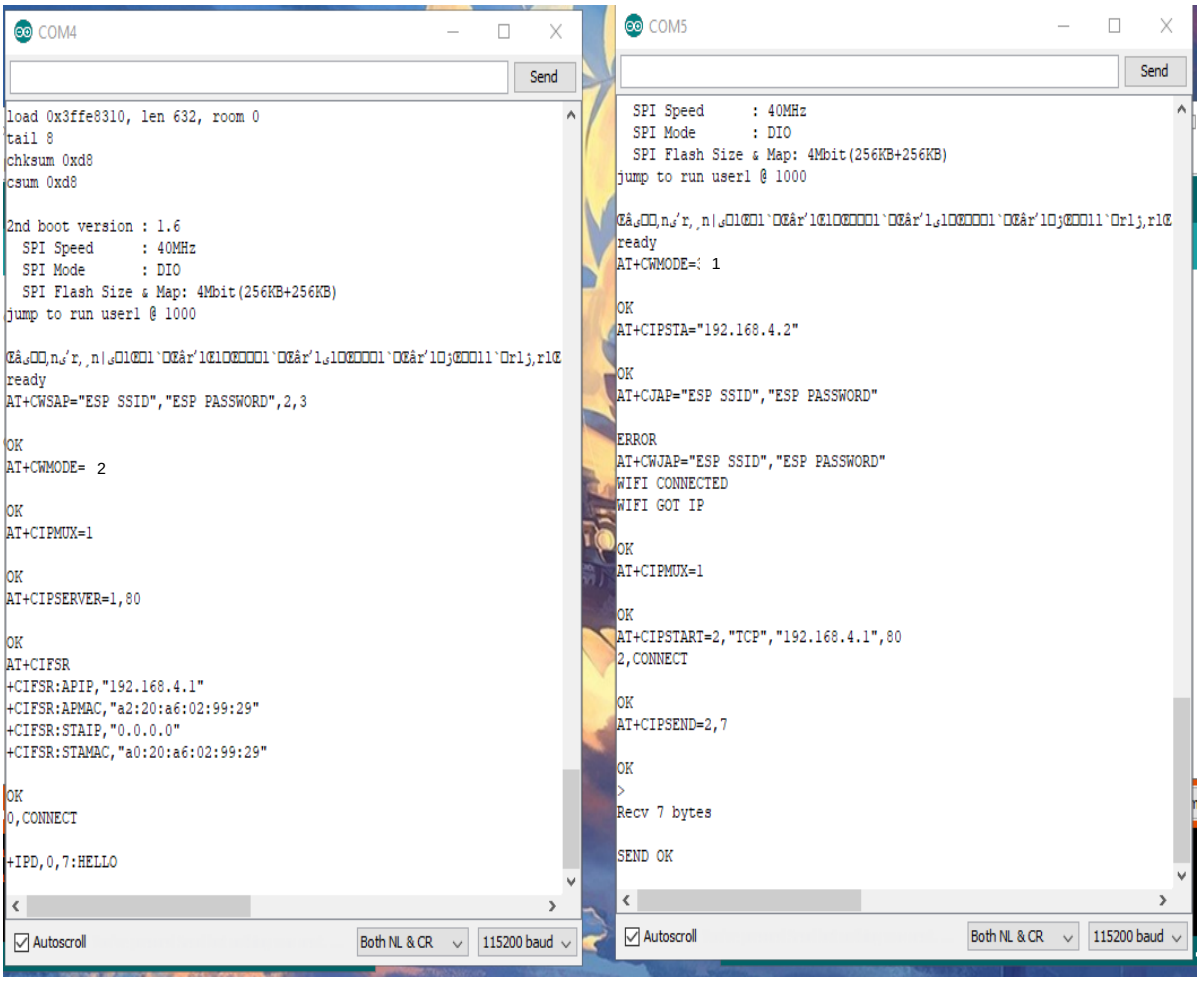


Figure 42 WIFI send and receive example

Based on the previous example, we implemented the local network between the cars to be as follows:

- each car will operate at both modes, softAP and station
- the car will use the station mode to send its data to other cars
- the car will use the SoftAP mode to receive other cars data
- the controller will handle the 2 modes and handles sending and receiving data

the following figure demonstrate the structure.

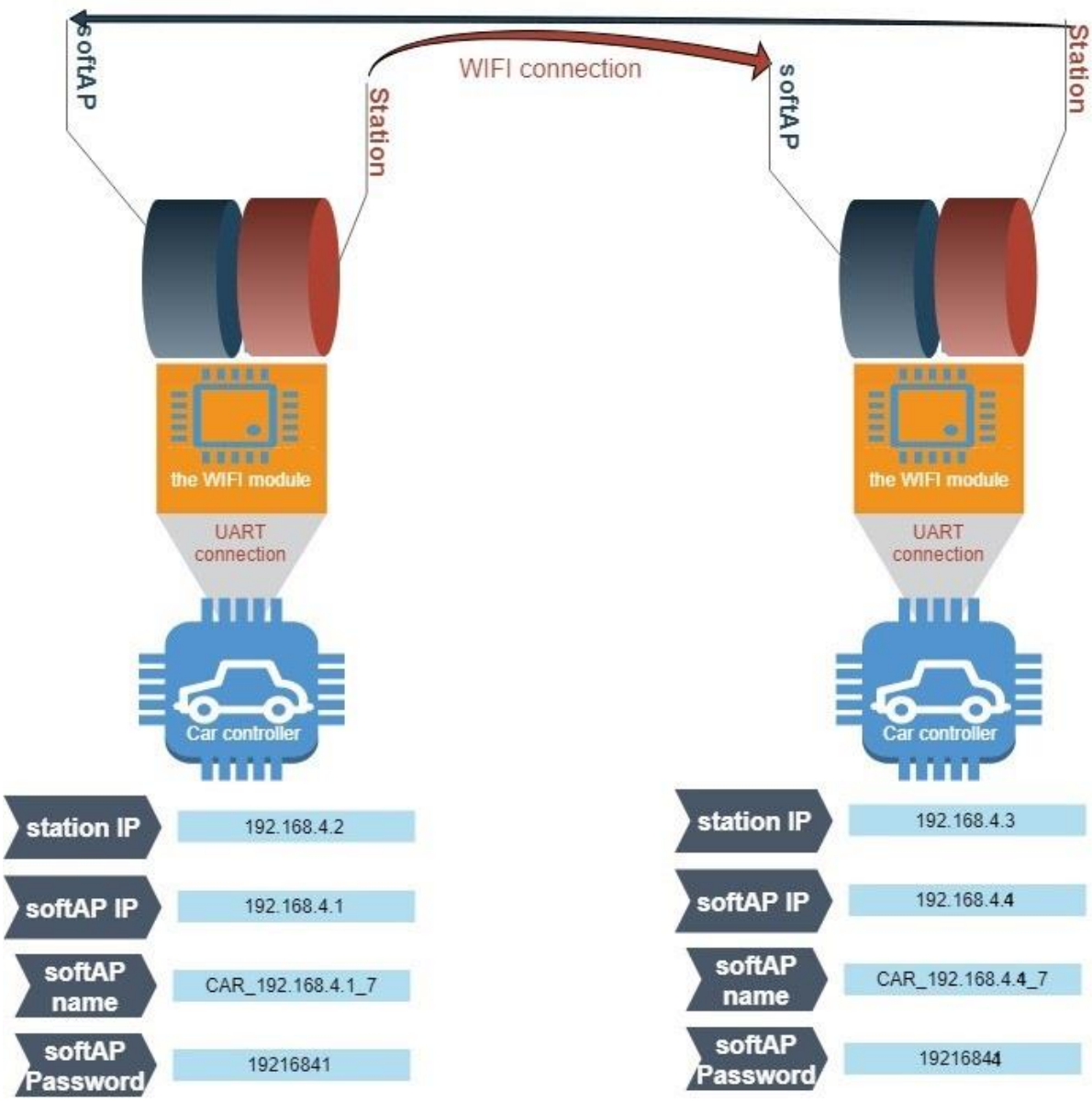


Figure 43 Our Local Network architecture

To determine the password and port number of any other car so that we can connect to it and send the data to it, we implemented a system, each car softAP name will be CAR_it's IP_it's port number for example if the car IP is "192.168.4.5" and port number is 7, the CAR softAP name will be "CAR_192.168.4.5_7", accordingly the password will be the IP without the dots We need to change the station and softAP IP for all the cars, because if we had two identical station IPs the softAp won't know how to separate them, but if there are two identical softAP IP the stations won't be to join both

4.1.3 WIFI functions

Based on this structure the controller has to handle the following tasks

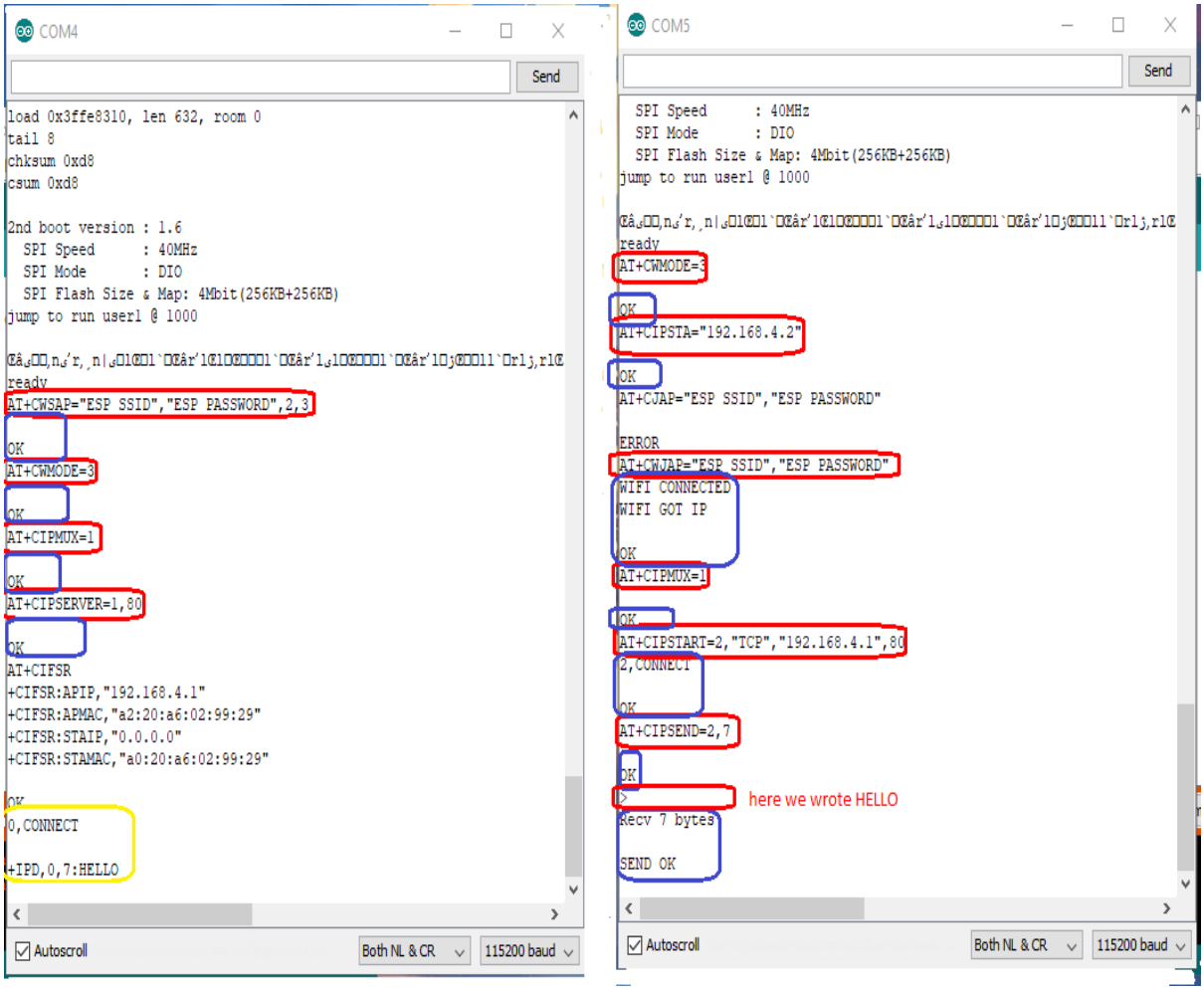
- initializing the module
- sending the data
- receiving the data when we don't send the data
- receiving other cars data while sending my data

since we don't know when will the other cars send data to me, the controller must be ready to receive data at any time.

Based on this the work flow of the controller will be as follows:

Sending my data and receive the data while sending

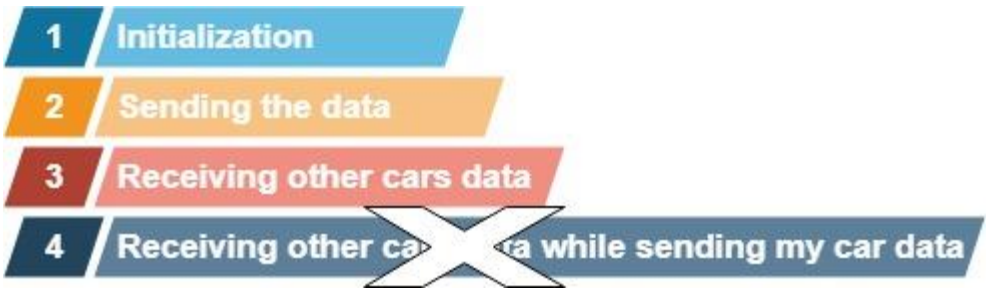
Sending the data is simply sending some AT commands in order and the data is among them, to receive the data while sending, the controller should be ready to receive the data instead of an AT command, for example at the previous communication example at the station module we were sending the AT commands for the module to send the data to the softAP module ,we were connecting both modules ourselves so we arranged the AT commands so that we would see the message at the softAP while we weren't sending any AT commands to Module,



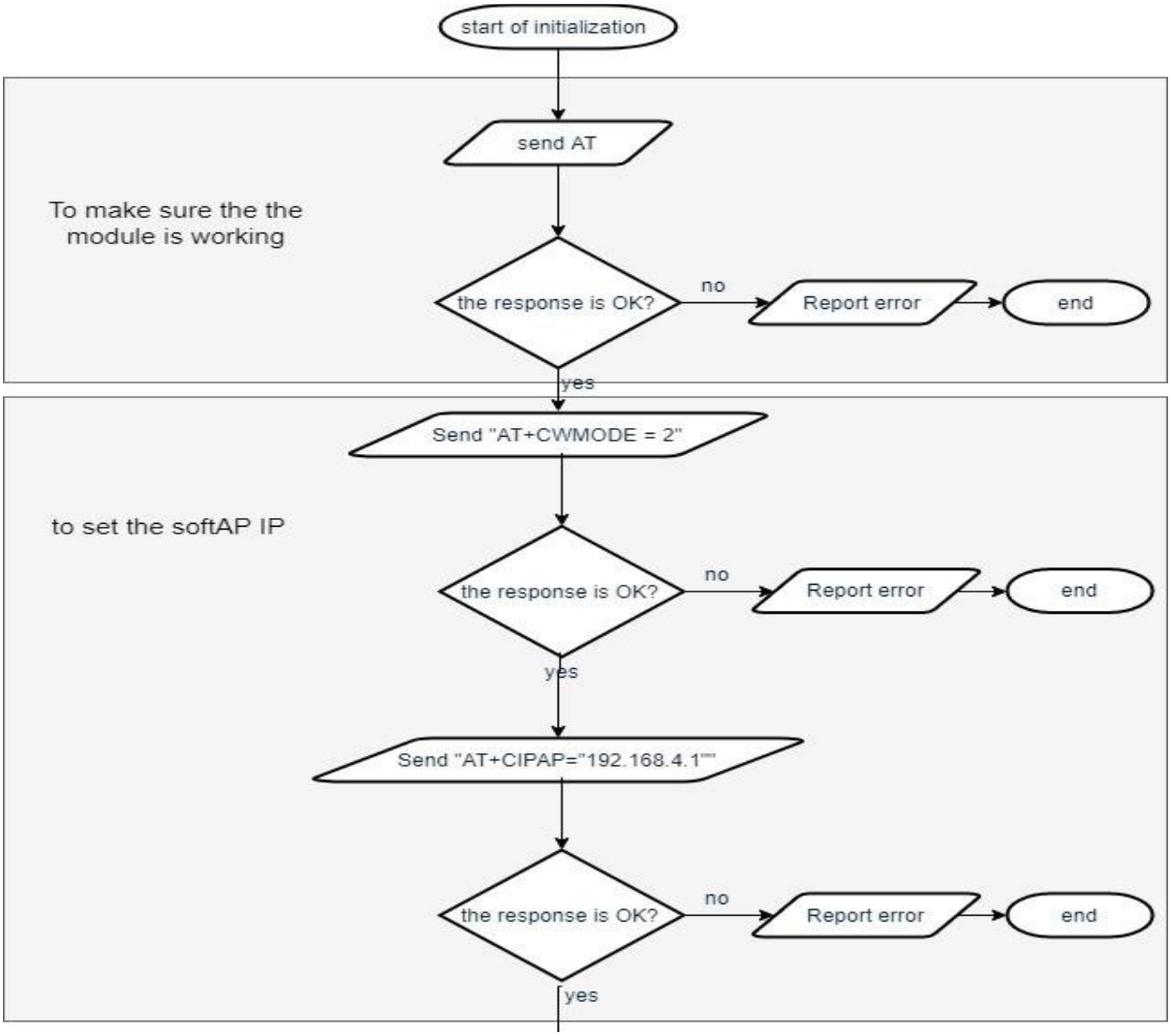
In other word when we don't know when will other cars send their messages, we have to deal with the fact that the yellow square might come right before the blue squares at the previous example.

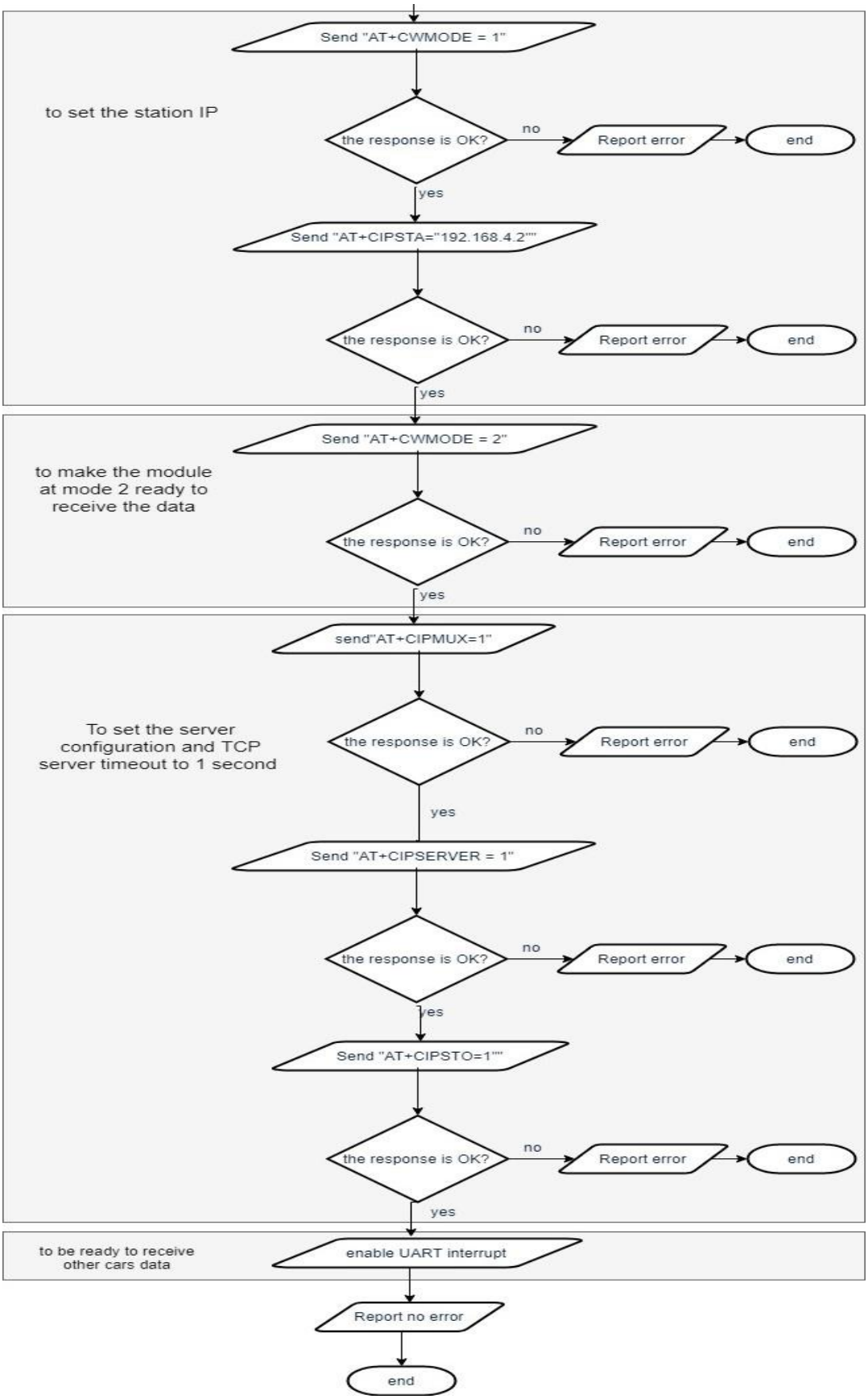
And also, in order for the module to be able to send while receiving, the module has to operate at the third mode softAP and station, after many trails and asking some people we discovered that the module can't start connection while operating at the third mode, so we had to drop the third task of the controller since the module itself can't support it.

Due to that problem we can't receive while sending, that put a constrain of the sending time to be as short as possible, so this car won't be invisible to the others for long time.



1- Initialization

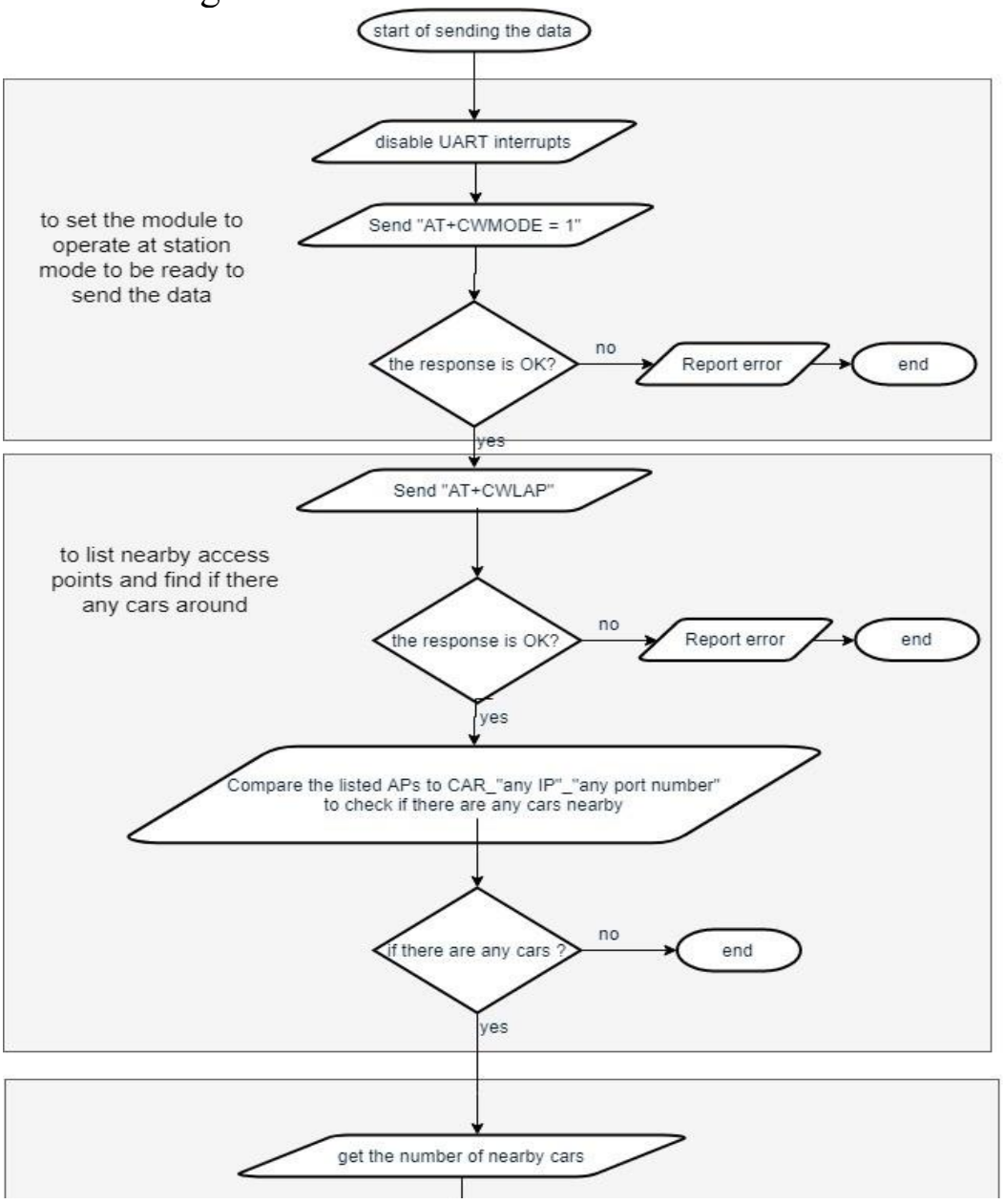


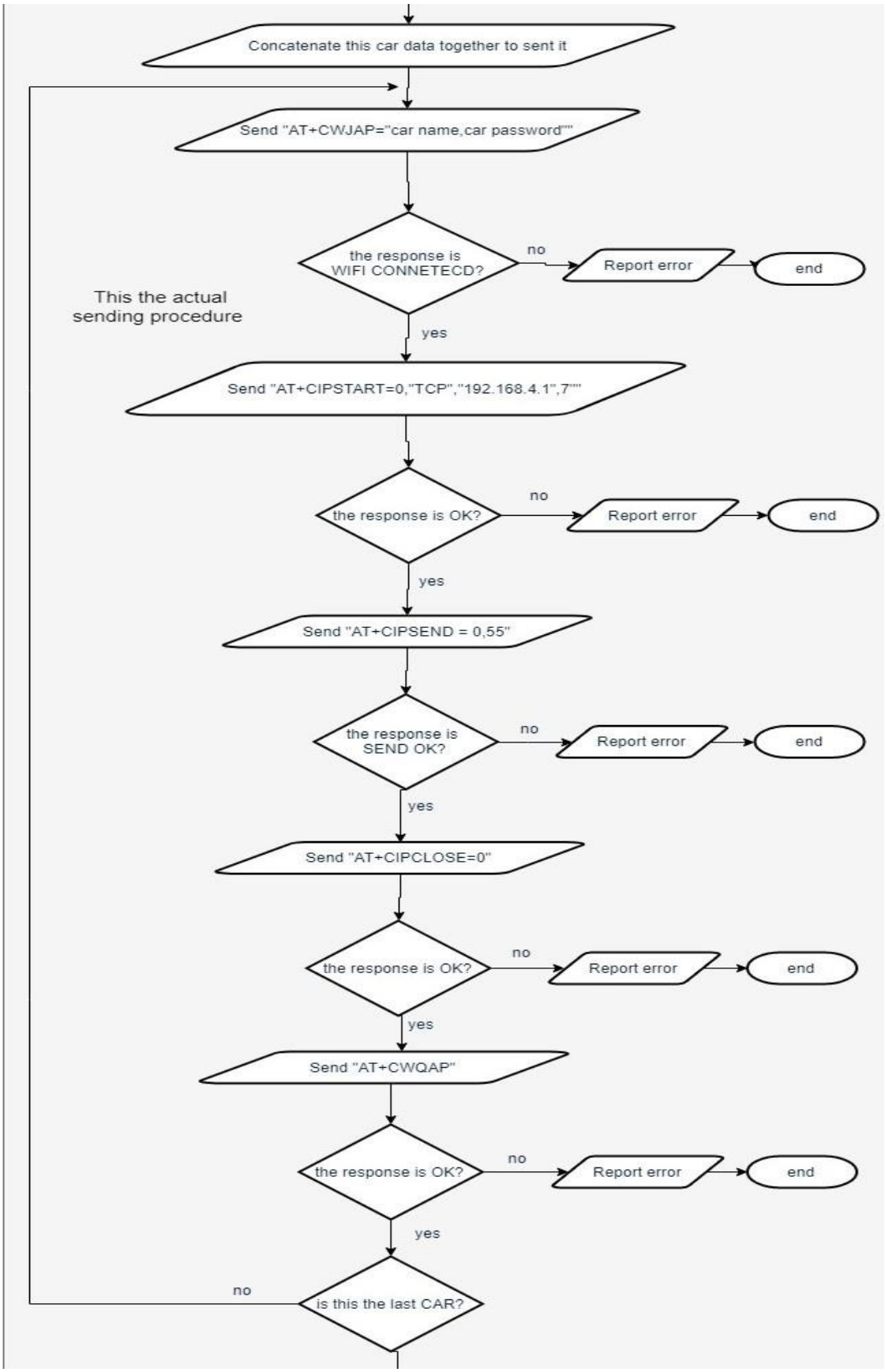

Send "AT+CWMODE = 1"
to set the station IP
the response is OK?
no
Report error
end
yes
Send "AT+CIPSTA="192.168.4.2""
the response is OK?
no
Report error
end
yes
Send "AT+CWMODE = 2"
to make the module at mode 2 ready to receive the data
the response is OK?
no
Report error
end
yes
send"AT+CIPMUX=1"
To set the server configuration and TCP server timeout to 1 second
the response is OK?
no
Report error
end
yes
Send "AT+CIPSERVER = 1"
the response is OK?
no
Report error
end
yes
Send "AT+CIPSTO=1""
the response is OK?
no
Report error
end
yes
to be ready to receive other cars data
enable UART interrupt
Report no error
end

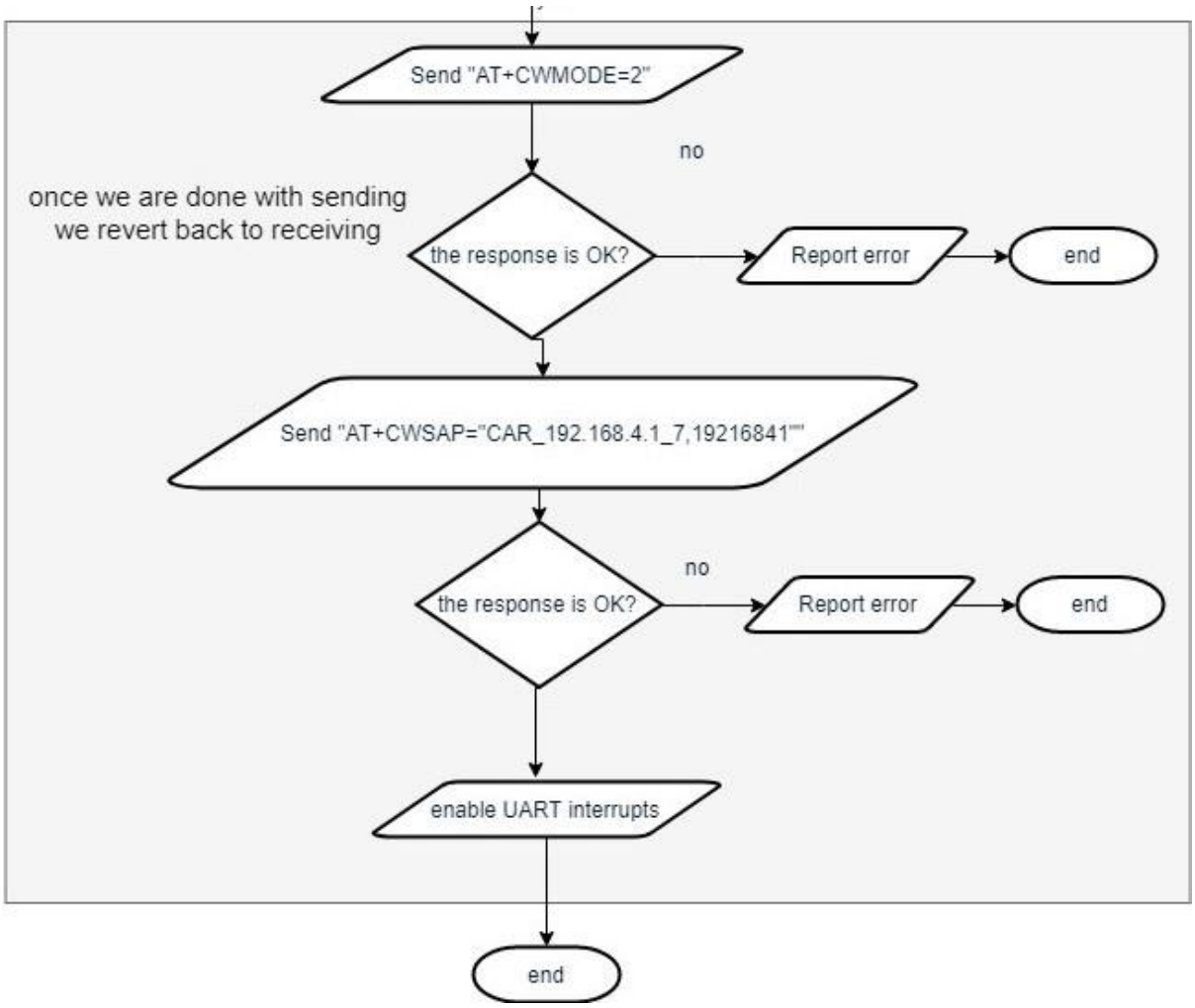
This initiation flow guarantee:

- To set the station IP to "192.168.4.2" and different IPs for different cars
- To set the softAP of all cars to "192.168.4.1"
- To set the module to work at softAP initially and enable the UART interrupt to receive the data, when we send the data, we change the mode to station and change back when are finished with sending the data

2- Sending the data




Concatenate this car data together to sent it
Send "AT+CWJAP="car name,car password""
the response is WIFI CONNETECD?
no
Report error
end
This the actual sending procedure
yes
Send "AT+CIPSTART=0,"TCP","192.168.4.1",7""
the response is OK?
no
Report error
end
yes
Send "AT+CIPSEND = 0,55"
the response is SEND OK?
no
Report error
end
yes
Send "AT+CIPCLOSE=0"
the response is OK?
no
Report error
end
yes
Send "AT+CWQAP"
the response is OK?
no
Report error
end
yes
is this the last CAR?
no



This is the flow chart for sending the data, it consists of many things:

- 1st stage is the change of modes to be able to send the data:
    1. Disabling the UART interrupts
    2. Sending AT+CWMODE=1 to set the module to station mode
- 2nd stage is listing nearby Access points and finding if there are any cars in these Access points
- If there are any cars around with the sofAP "CAR_192.168.4.1_7"then concatenate this car data, which consists of
    1. My IP
    2. My speed
    3. My transmission state -> forward, reverse, braking
    4. My direction -> left, right, non
    5. My Latitude
    6. My longitude
    7. My Altitude

    The frame consists of 55 characters which are 11 for IP, 3 for speed, one for transmission state, one for direction for,11 for latitude, 11 for longitude, 11 for altitude and 6 spaces to separate them for example,

192.168.4.1 300 F L 092725.0000 4717.113990 00833.91590

- 3rd stage the actual sending of data, iterate on the following for all the cars:
    1. Join their access points with the password known form their name
    2. Start TCP connection with them at the port number known form their name
    3. Send the data collected in the previous stage
    4. Close this TCP connection
    5. Quit this access point
- 4th stage when we are finished with all the sending, we are ready to witch back to softAP to receive the data by
    1. Changing the mode to softAP
    2. Setting the softAP name and password to "CAR_192.168.4.1"and"19216841"
    3. Enable UART interrupts

3- Receiving other cars data

Receiving the data of the other cars occurs on two stages

1. The actual receiving of the data which happens at the UART ISR, the module sends to the controller 3 messages
    a. Channel ID, Connect for example "0,CONNECT", this message is sent form the module when another WIFI module starts a TCP connection with it
    b. The message itself, this message is sent form the module when another WIFI module sends a data using "AT+CIPSEND"
    c. Channel ID, CLOSED for example "0,CLOSED", this message is sent form the module when another WIFI module close a TCP connection using "AT+CIPCLOSE"

    The total size of these messages is 88 characters, and the module send them with no pauses with 115200 baud rate, we need not to lose any data in the middle, and we don't know when will the other cars send their data so we have make it interrupt based, the kit we are using has a FIFO buffer of just 16 characters so we have a problem with the size, because there will be a time wasted when the buffer get full and the micro-controller store the data and then wait for new interrupt, there will be a loss of data in the middle of the frame, this is a hardware constrain on the systems and the only way to solve it with this controller is to implement a busy wait mode inside the ISR, meaning that when an interrupt happens the controller will stay at the ISR receiving all 88 character at one shot, this will make the ISR very long, and make some other tasks wait, but we have no choice, The interrupt can be on 2 characters or 4 ,6, 8, 16, the best option is 2 char because there will be a wasted time at context switching between the task and the ISR, so we need to make some free spaces for the module to spend his data until we get to the ISR, once we get to the ISR, the micro-controller will poll on the UART with its full speed, since the controller is faster than the module, the characters in the buffer won't accumulate, they will be served as soon as they get to the buffer, in order to do that we have make a fixed data size, which is the case.

```
340 void UART7_Handler()
341 {
342     int i=0,j=0;
343     /*88=0,connet+frame+o,closed*/
344     while(wait_Buffer[i]!=0)
345     {
346         i++;
347     }
348     while(j<88)
349      {
350          wait_Buffer[i]=UART7_InChar();
351          i++;j++;
352      }
353 }
```

2. Reading this data, the previous stage stores the data at a waiting buffer, so need to get the buffer and extract the important data form it, this stage is a simple string cutting function, each data sent start with "0,CONNECT"and then +IPD, channel ID, length of the data: the data itself then "0,CLSOED" , for example

0,CONNECT +IPD,0,7:HELLO0CLOSED

4.1.4 WIFI optimizations

When we measured the Tasks time for the first time, we were shocked, the time taken by the tasks was:

TASk	Time
WIFI_Frame_RCEV	510 us
WIFI_Frame_Send(sending to one car)	5.5 s
WIFI_Frame_Send(sending to two car)	8.5 s
WIFI_Frame_Send(just WIFI listing)	2.5 s

This is a very long time, our first trail to reduce it was by splitting the one big function into smaller functions to be able to run other tasks in between.

Figure 44 WIFI sending Task First optimization

This splitting is just cutting the 1st function into 2 functions just one modification made
At the end of the first part we enable the UART interrupt, and change the mode to softAP, and the beginning of the second function we disable the interrupt and change the mode to station, this allow us to be able to receive data between these tasks, the measured time of the new two functions was

TASk	Time
ESP_SEND_THIS_CAR_DATA__CARS_LISTING	2.5 s
ESP_SEND_THIS_CAR_DATA___SENDING_TO_SINGLE_CAR	3.2 s

This splitting was good but not good enough we still have a very big task, we had to make farther optimization, not just splitting we had to investigate to find what is taking so long, we discovered that there are two at commands responsible for this huge delay

1. AT+CWLAP
   This command response is the largest response, it gets all the nearby APs and their MAC number, for example

```
+CWLAP:(3,"CVBJB",-71,"f8:e4:fb:5b:a9:5a",1)
+CWLAP:(3,"HT_00d02d638ac3",-90,"04:f0:21:0f:1f:61",1)
+CWLAP:(3,"CLDRM",-69,"22:c9:d0:1a:f6:54",1)
+CWLAP:(2,"AllSaints",-88,"c4:01:7c:3b:08:48",1)
+CWLAP:(0,"AllSaints-Guest",-83,"c4:01:7c:7b:08:48",1)
+CWLAP:(0,"AllSaints-Guest",-83,"c4:01:7c:7b:05:08",6)
+CWLAP:(4,"C7FU24",-27,"e8:94:f6:90:f9:d7",6)
+CWLAP:(2,"AllSaints",-82,"c4:01:7c:3b:05:08",6)
+CWLAP:(3,"QGJTL",-87,"f8:e4:fb:b5:6b:b4",6)
+CWLAP:(4,"50EFA8",-78,"74:44:01:50:ef:a7",6)
+CWLAP:(0,"optimumwifi",-78,"76:44:01:50:ef:a8",6)
+CWLAP:(3,"BHQH4",-95,"18:1b:eb:1a:af:5b",6)
+CWLAP:(3,"NETGEAR49",-86,"84:1b:5e:e0:28:03",7)
+CWLAP:(3,"ngHub_319332NW00047",-56,"20:e5:2a:79:b1:2f",11)
+CWLAP:(3,"N16FU",-53,"20:cf:30:ce:60:fe",11)
+CWLAP:(3,"ITS",-82,"90:72:40:21:5f:76",11)
+CWLAP:(3,"ITS",-79,"24:a2:e1:f0:04:e4",11)
```

We thought the delay is coming from receiving all this huge response but it turns out, this only takes about 100 ms and module takes 2 seconds to search for the APs, We split the listing function into two functions, the 1st one just send the AT command, the other one receive the response, and between these two functions the system can run other functions unlike before.

2. AT+CWJAP

   Joining an access point turns out be the largest delay at the sending, it takes about 2.8 seconds, at this time the controller is waiting for the module to respond, we took advantage of that time, of that, we split the second function into three functions, the 1stfunction just send the AT command "AT+CWJAP", the 2nd function receive the response, the 3rd function continue the sending flow

After this optimization the sending flow became

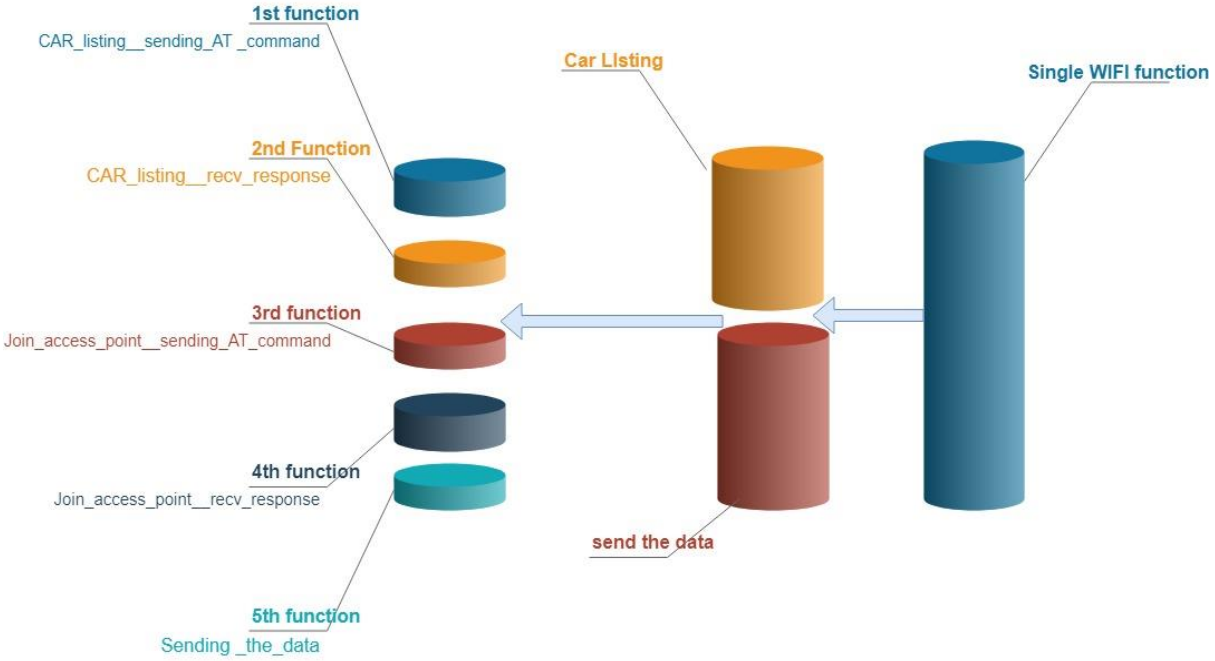
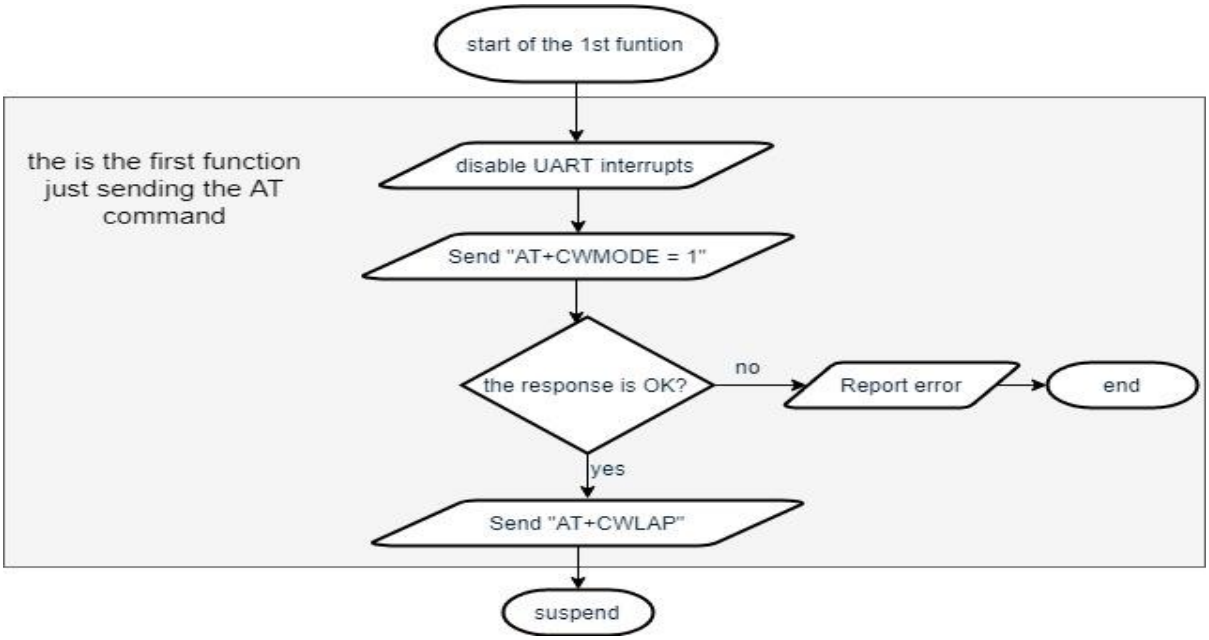
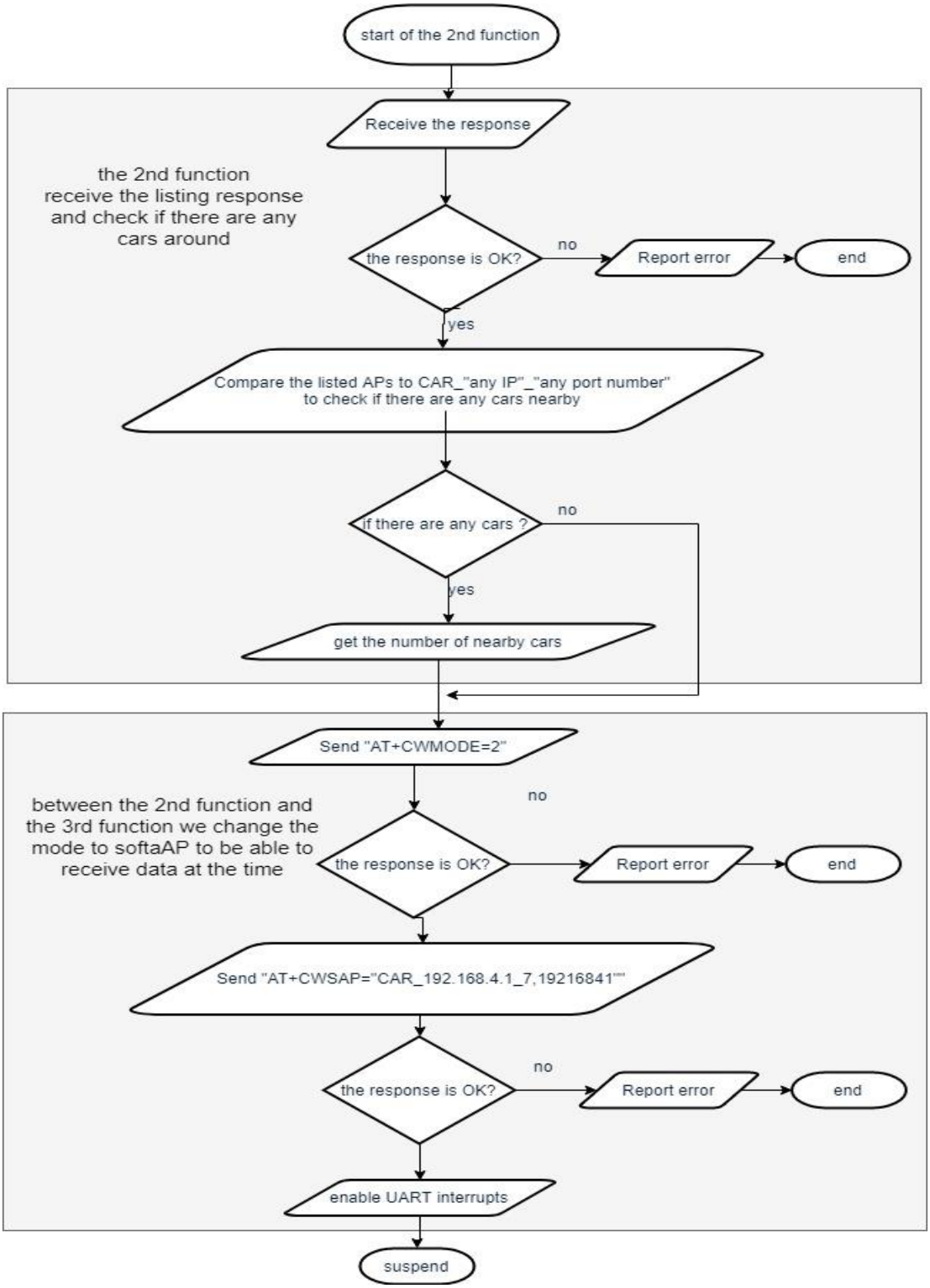


Figure 45 WIFI Send Task second optimization

The 1st function is just sending the AT command "AT+CWLAP", before this optimization we would stay at the function waiting two seconds for the module to respond, now we will use these two seconds to something else in the system.

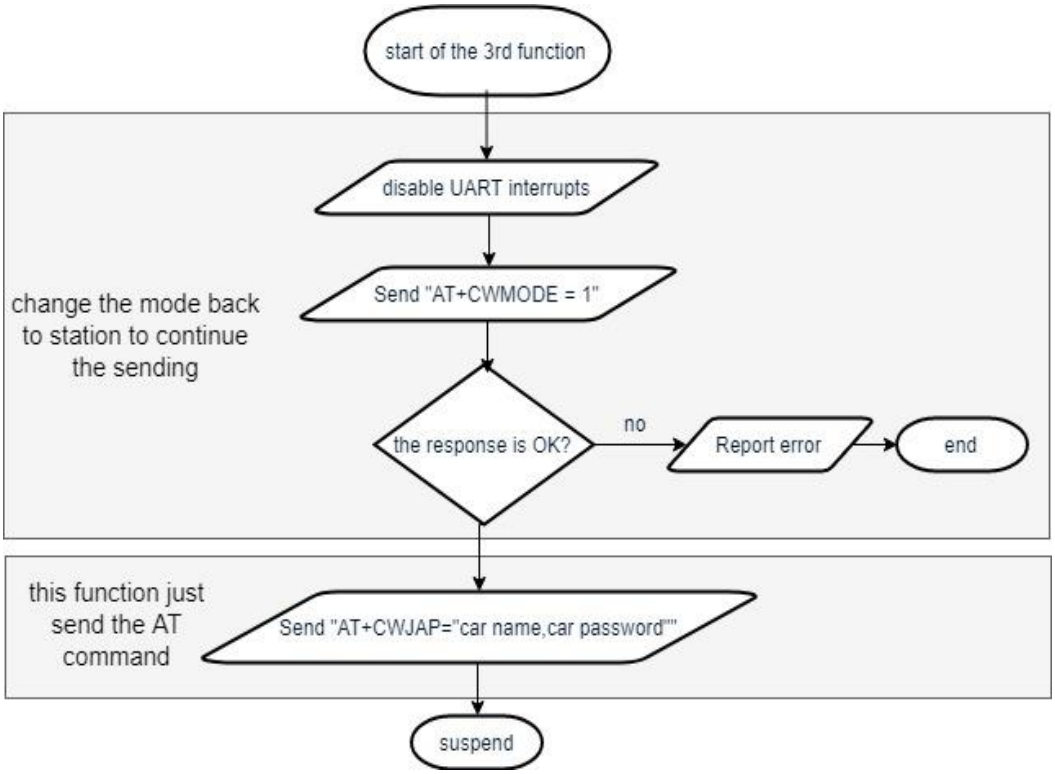


After two seconds, the 2nd function starts, it receives the module response, then we change the mode of operation of the module to softAP at the end of the function so that we can receive data during this time between 2nd function and 3rd function.
between the 3rd function and the 2nd function we don't have a constrain form the module on this time, it shouldn't be very long so that the car don't get out of the WIFI range.

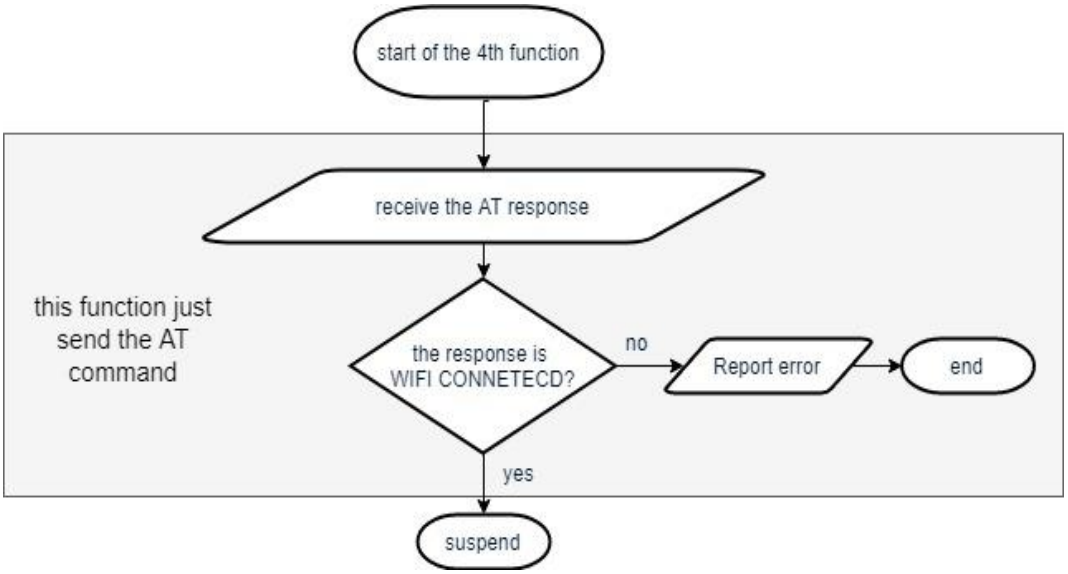


The 3rd function is similar to the 1st function, it just sends the AT command "AT+CWJAP="CAR_192.168.4.1_7"."19216841"", then the module takes 2.8 seconds to respond during this time, the controller switch to some other task, because we changed the mode of operation at the end

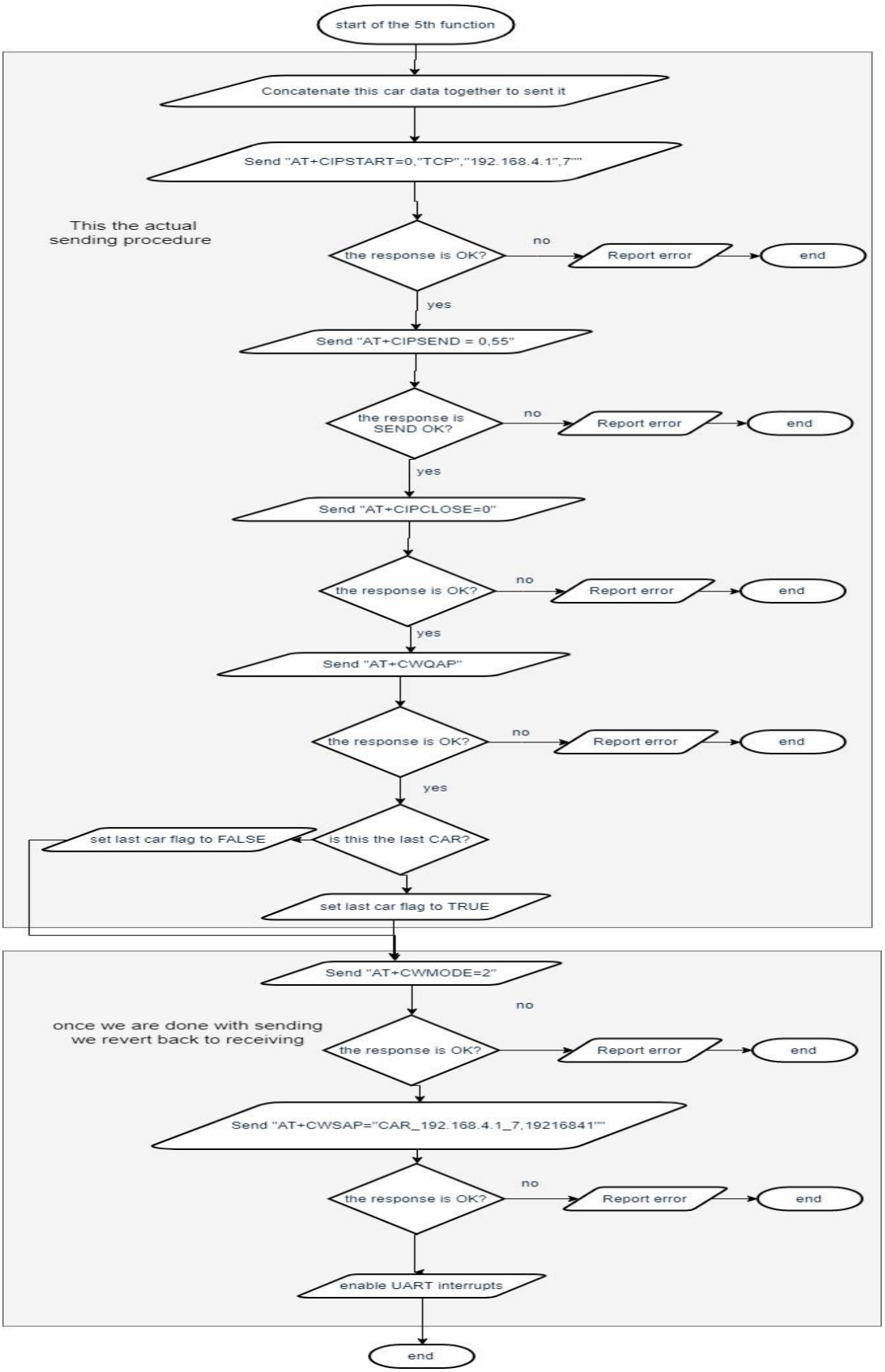
of the second function to softAP so that we could receive other cars data, we have to switch back to station mode at the start of the 3rd function.



When this time is over the 4th function starts, it receives the response of the previous AT command, Unlike the previous functions we can't change the mode of operation, because changing the mode to softAP will shut station mode, and the access point we just connected to will disconnect, accordingly the time between 4th and 5th function has to be small because we can't receive any data during this time, and also because that the car doesn't get out of range and disconnect.



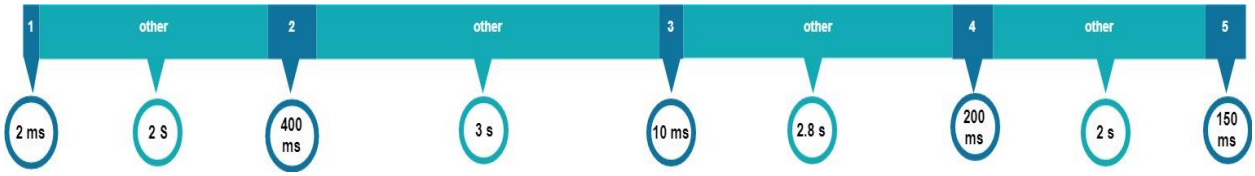
The 5th function is last one, it's responsible to send data to other cars, at the end we change the mode to softAP again to receive the data.


start of the 5th function
Concatenate this car data together to sent it
Send "AT+CIPSTART=0,"TCP","192.168.4.1",7""
This the actual sending procedure
the response is OK?
no
Report error
end
yes
Send "AT+CIPSEND = 0,55"
the response is SEND OK?
no
Report error
end
yes
Send "AT+CIPCLOSE=0"
the response is OK?
no
Report error
end
yes
Send "AT+CWQAP"
the response is OK?
no
Report error
end
yes
set last car flag to FALSE
is this the last CAR?
set last car flag to TRUE
Send "AT+CWMODE=2"
no
once we are done with sending we revert back to receiving
the response is OK?
Report error
end
Send "AT+CWSAP="CAR_192.168.4.1_7,19216841""
the response is OK?
no
Report error
end
enable UART interrupts
end

After this optimization the functions had the following timing:

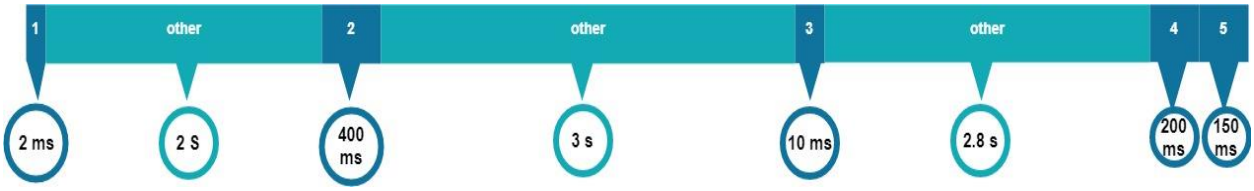
Task	Time
ESP_SEND_THIS_CAR_DATA__CARS_LISTING__Command_Sending	2ms
Time between	2 s
ESP_SEND_THIS_CAR_DATA__CARS_LISTING__Response_recv	400ms
Time between	3 s
ESP_SEND_THIS_CAR_DATA___CONNECTING_TO_SINGLE_CAR__Command_ sending	10ms
Time between	2.8 s
ESP_SEND_THIS_CAR_DATA___CONNECTING_TO_SINGLE_CAR__Response_ Recv	200ms
Time between	2 s
ESP_SEND_THIS_CAR_DATA___SENDING_TO_SINGLE_CAR	150ms

This new task time is much better for the system, but whole sending time has increase to the sum of all the times,



the sending time = 10.28 seconds

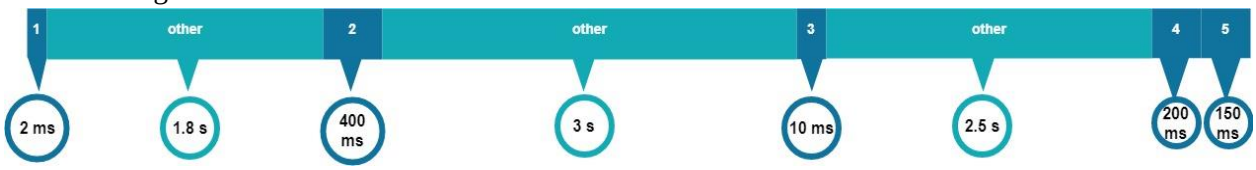
So far, we have very good result, but as mentioned between the 4th and the 5th function we can't swap the mode of operation, and since both functions combined have 350 ms delay which is less that the 2nd function, we removed the delay between them.



the sending time = 8.562 seconds

after this optimization when we tried the WIFI tasks with other task, we found that we need to at some margin at the time between the tasks, so that the functions responsible for receiving the response don't miss any of the data.

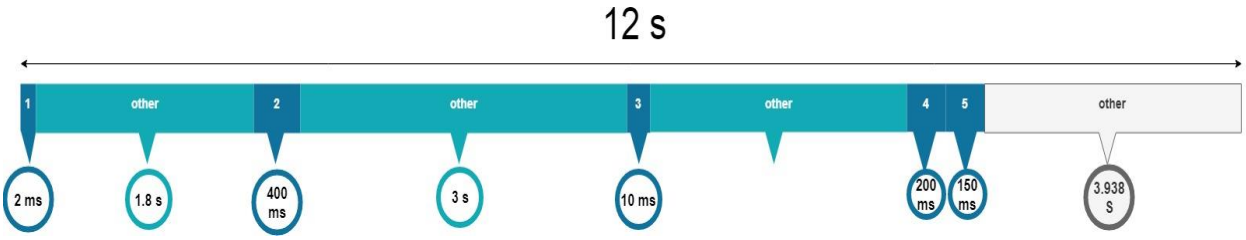
the sending time = 8.062 seconds



as we mentioned before we can't send while receiving, but with this new function architecture, we receive data at the three seconds interval.

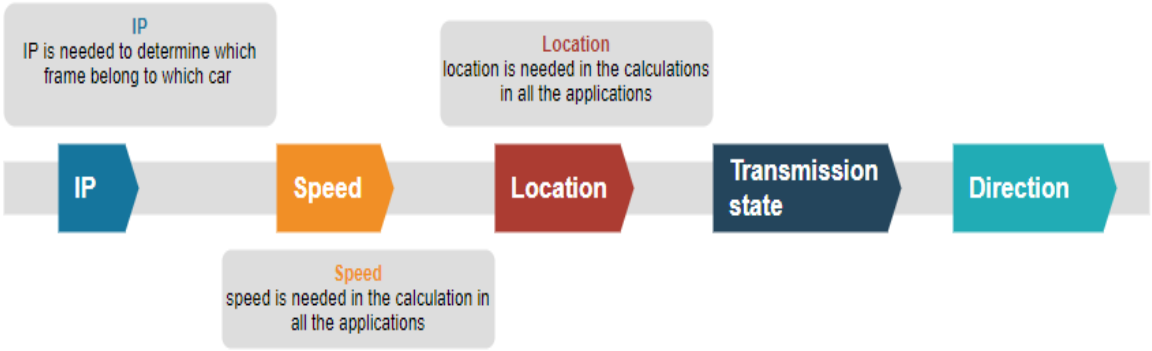
So, this leave us with 2.202 seconds sending then 3 seconds receiving then 2.86 sending, which means we send 62.7 % of the time and receive only 37.21%.

This ratio is bad, the probability to miss other cars data is very high, the solution is to add some other time at the end of the sending cycle, in other works increase the periodic time of the WIFI task ,but this increase come at the expense of frequency of sending the data, after many trails we found that 12 seconds periodic time for the WIFI task, we get good comptonization between the probability to miss the data and the frequency of sending my car data.
the WIFI now receives = 57.81% of the time.



4.1.5 Standard message format

Vehicle-to-vehicle (V2V) communications comprises a wireless network where automobiles send messages to each other with information about what they're doing. This data would include speed, location, direction of travel, car transmission state.



4.2 BLUETOOTH MODULE

4.2.1 Bluetooth theory of work

Bluetooth technology is a short-range wireless communications technology to replace the cables connecting electronic devices. The Bluetooth RF transceiver (or physical layer) operates in the unlicensed ISM band centered at 2.4 GHz (the same range of frequencies used by microwaves and Wi-Fi). The core system employs a frequency-hopping transceiver to combat interference and fading.

Bluetooth's short-range transmitters are one of its biggest plus points. They use virtually no power and, because they don't travel far, are theoretically more secure than wireless networks that operate over longer ranges. Bluetooth devices automatically detect and connect to one another and up to eight of them can communicate at any one time. They don't interfere with one another because each pair of devices uses a different one of the 79 available channels. If two devices want to talk, they pick a channel randomly and, if that's already taken, randomly switch to one of the others.

When a group of two or more Bluetooth devices are sharing information together, they form a kind of ad-hoc, mini computer network called a piconet. Other devices can join or leave an existing piconet at any time. One device (known as the master) acts as the overall controller of the network, while the others (known as slaves) obey its instructions. Two or more separate piconets can also join up and share information forming what's called a scatternet.

4.2.2 About the Module

- The HC series is a serial connection over Bluetooth based on BC417 Chip. The HC06 is set at the factory into master or slave mode, and can be ordered in either variation. The HC05 can be set to either master or slave mode using the 'STATE' pin.
- Modules operate in the industrial, scientific and medical (ISM) radio bands: 2.40GHz—2.48GHz.
- Operating voltage ranges from 3.3 to 6v and operating current 40 mA.
- Modules can reach a range up to 9 meters.

Figure 46 Bluetooth Module

4.2.3 Module AT Commands

HC 06

In case of HC 06 AT commands are limited as it comes in either master or slave mode and can't be configured as HC 05

Command	Response	Function
AT	OK	Used to test the UART connection between the host controller and Bluetooth Module.
AT+BAUD<P>	OK<r>	Change Baud Rate. For 9600, P = 4 , r = 9600
AT+NAME<name>	OK<name>	To change the device name.
AT+PIN<1234>	OK<1234>	Passkey (PIN Code) is a 4-digit code shared with a master Bluetooth Device (e.g. PC) to prevent unauthorized pairing.

- AT+BAUD

<p>	<r>	Remarks
1	1200	set to 1200bps
2	2400	set to 2400bps
3	4800	set to 4800bps
4	9600	set to 9600bps (Default)
5	19200	set to 19200bps
6	38400	set to 38400bps
7	57600	set to 57600bps

Example1: Set baud rate to 57600bps

From Host controller:
AT+BAUD7
EGBT-046S Response
OK57600

- **AT+NAME**

Example1: Set device name as EGBT-04

From Host controller:
AT+NAMEEGBT-04
EGBT-046S Response
OKEGBT-04

- **AT+PIN**

Example1: Set PASSKEY to 5995

From Host controller:
AT+PIN5995
EGBT-046S Response
OK5995

HC 05

Some extra commands in HC 05

Command	Function
AT+ROLE	See role of Bluetooth module[1=master/0=slave]
AT+ORGL	Restore factory settings
AT+UART	See Baudrate

4.2.4 Contribution

Once you buy the module it comes with its default name and baud rate, so we didn't have to use AT commands for configuration. The first step was configuration of UART with same baud rate as the Bluetooth module

Next we had to search for an appropriate mobile application, which is compatible with our Bluetooth module, to control the car movement.

At first we used an application that sends continuous stream of characters to the module. The problem was that upon finger release, it sends a constant character 'S'. Since the stream is continuous, we had to implement a function that reads all buffer characters to get the majority character in order to determine the required direction or speed.

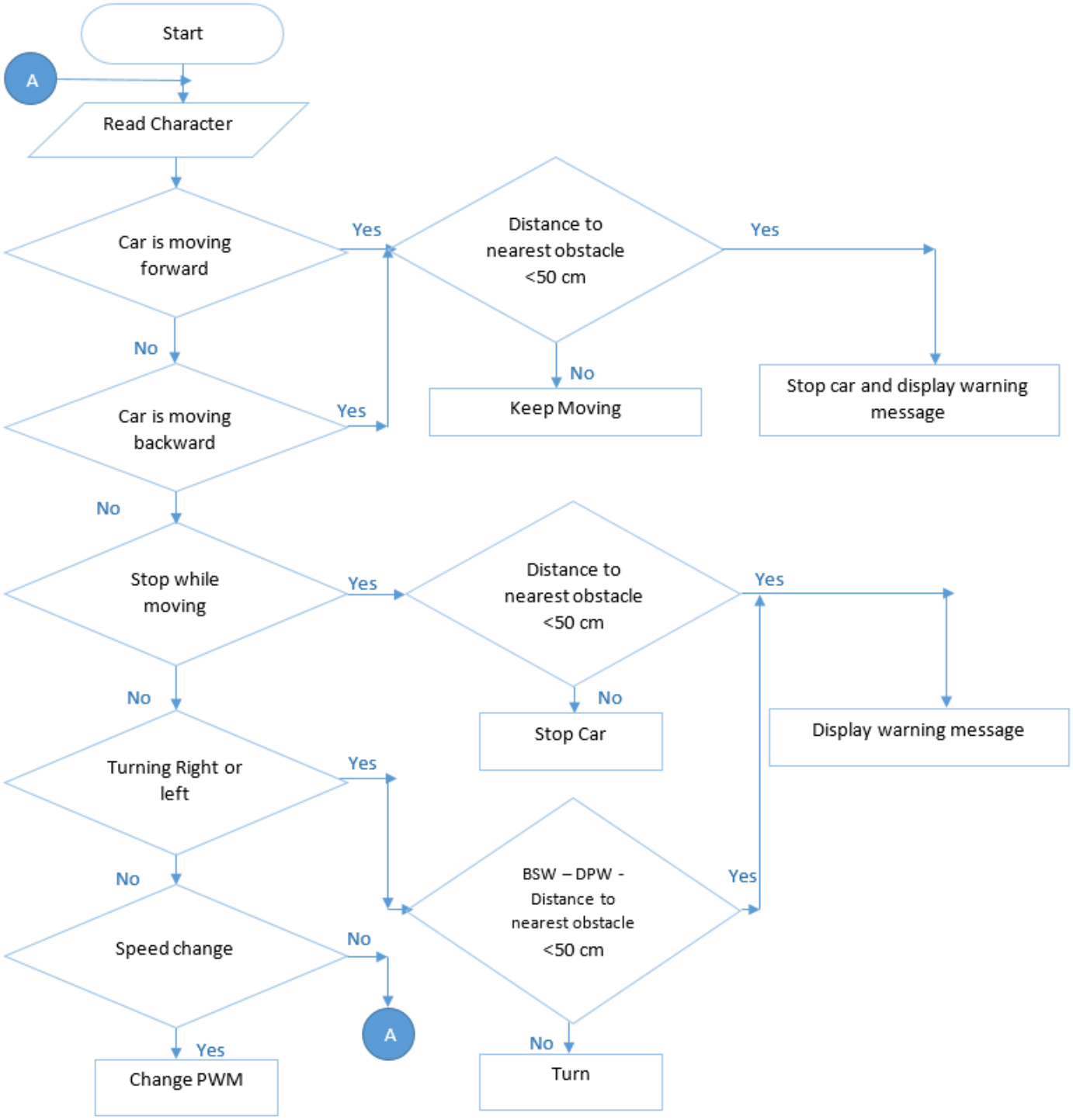
This solution had some problems as the number of tasks increased, causing delay in motors' action and the car became slightly unresponsive. We thought of decreasing the time during which Bluetooth task is delayed, but this didn't help either.

4.2.5 Optimization

For a faster response, we agreed on using interrupts. But as the application sends continuous stream of bits, the controller will be receive an interrupt every 50 ms, which was not applicable as it interrupts other tasks.

UART2 was configured in a way such that we receive an interrupt when the buffer is 1/8 full and this is the minimum interrupt level. Fortunately, we found a mobile application that sends two characters at a time, and since the buffer is 16 bits, we have an interrupt when the characters are sent. Using interrupts improved car response and facilitated motors control, as it is done directly after reading character in ISR.

4.2.6 Flowchart



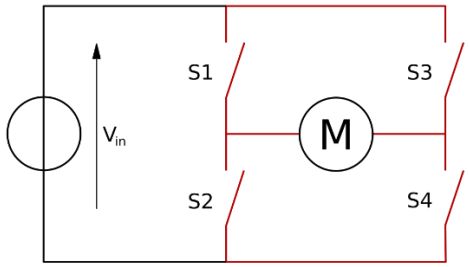
CHAPTER 5: ACTUATORS

In this chapter we are going to cover the actuators that we used in our project.
Firstly we will talk about Motors and H-bridge used in car movement then we will cover the LCD display we used to show warning messages in critical situations.

5.1 MOTORS

5.1.1 H-Bridge Theory of work

The term H-Bridge is derived from the typical graphical representation of such a circuit. An H bridge is built with four switches. When the switches S1 and S4 are closed (and S2 and S3 are open) a positive voltage will be applied across the motor. By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed, allowing reverse operation of the motor.
Using the nomenclature above, the switches S1 and S2 should never be closed at the same time, as this would cause a short circuit on the input voltage source. The same applies to the switches S3 and S4. This condition is known as shoot-through.

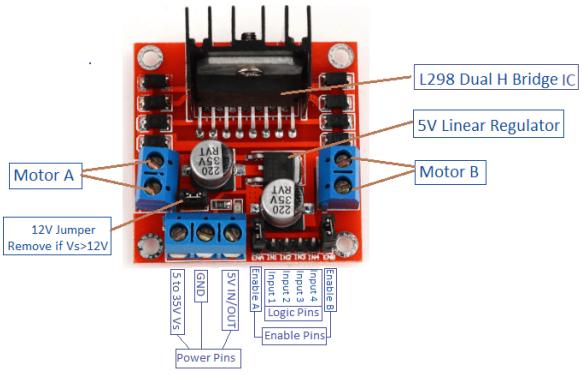


5.1.2 H-bridge module

L298N H-bridge is a dual Motor Controller Module, used to control motors' speed and direction. The recommended voltage ranges from 7 V to 12 V. For each motor we have two digital inputs to control whether motor rotates clockwise or anti-clock wise, according to the table below:

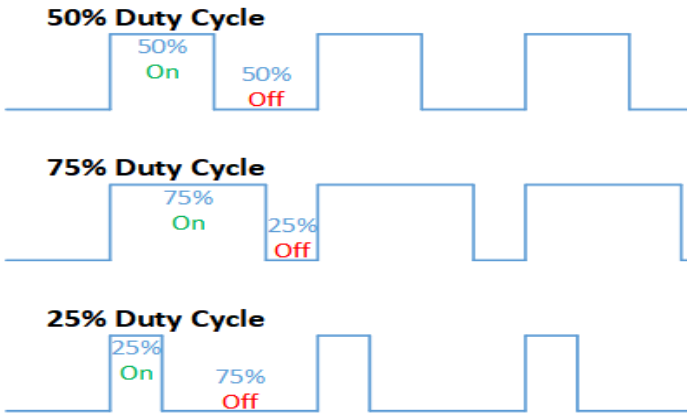
5.1.3 DC motors theory of work

Input A	Input B	Motor State
High	Low	Turns clockwise
Low	High	Turns anti-clockwise
High	High	Braking occurs
Low	Low	Braking occurs



A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor.

Motor's direction of rotation can be changed by reversing the polarity of applied voltage. One way to control speed is using timer in PWM mode.
The pulse width modulation method is a very efficient method and is the most commonly used method. The PWM method drives the motor by applying a series of ON and OFF pulses. It varies the duty cycle by applying the ON or OFF time greater or lower while keeping the frequency constant. The pulse width modulation signal is a square signal which has a high frequency (more than 1 KHz). If we want to vary the power supplied then we have to vary the duty cycle of this square wave. Duty cycle is the fraction of time in which the signal was high. The duty cycle can be varied from 0% to 100%.



The formula for calculating the duty cycle of square wave:

$$\%\text{Duty Cycle} = (T_{ON} / (T_{ON} + T_{OFF})) * 100$$

TON is the time in which the square wave will be on.
TOFF is the time in which the square wave will be off.

5.1.4 Contribution

To control motor speed and direction we need to interface H-Bridge module with controller, but direct interfacing causes serious damage to the controller as the module draw high current.
We used 74LS125N Tri-State buffer for all output pins from the controller to the module to ensure safe interfacing. It contains four independent gates, which is enough to interface 4 motors. Each gate performs a non-inverting buffer function. The outputs have the 3-STATE feature. When enabled, the outputs exhibit the low impedance characteristics of a standard LS output with additional drive capability to permit the driving of bus lines without external resistors. When disabled, both the output transistors are turned off presenting a high-impedance state to the bus line. Thus the output will act neither as a significant load nor as a driver.

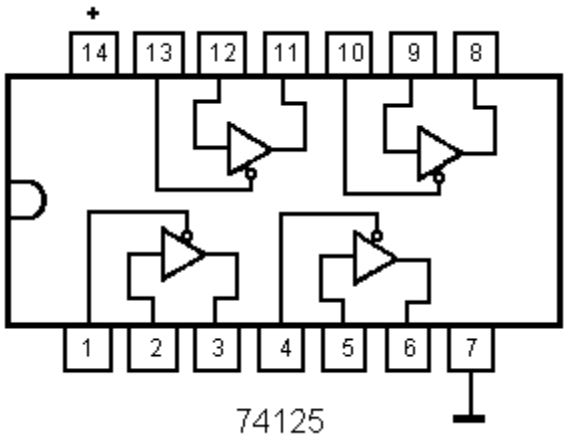


Figure 47 Tri-state Buffer

- We used 2 modules to control 4 DC motors. Since all motors are moving in the same direction either forward or backward, only two digital GPIO pins are needed and connected to H-Bridge control.
- To control the speed we used Timer0 in PWM mode and change the duty cycle according to speed received from Bluetooth module.
- In order to control right and left rotation we used Timer0A for two motors on the right side of the car and Timer 0B for the left side.
- As an example, to turn right, motors on the right side move with half speed of left side. Accordingly, we had to use different timers, one for each module.

5.2 LCD

5.2.1 LCD Theory of work

A liquid-crystal display (LCD) is a flat panel display, electronic visual display, or video display that uses the light modulating properties of liquid crystals. Liquid crystals do not emit light directly. 20x4 means that 20 characters can be displayed in each of the 4 rows of the 20x4 LCD, thus a total of 80 characters can be displayed at any instance of time.

LCD accepts two types of signals, one is data, and another is control. These signals are recognized by the LCD module from status of the RS pin. Now data can be read also from the LCD display, by pulling the R/W pin high. As soon as the E pin is pulsed, LCD display reads data at the falling edge of the pulse and executes it, same for the case of transmission. There are two types character LCD and graphical LCD.

5.2.2 About the module

We used monochromatic 20x4 alphanumeric LCD.

- 5x8 dots
- +5V power supply
- 1/16 duty cycle
- LED Backlight

Figure 48 LCD Module

-Pins Configuration

Data pins D7-D0 : Bi-directional data /command pins alphanumeric characters are sent in ASCII format

RS (register select)

RS =0 (command register is selected)

RS =1 (data register is selected)

R/W (Read or Write)

0 (Write) /1(Read)

E (Enable data)

Used to enable the LCD display

V0 (contrast control)

For maximum brightness V0=0

-Display Data RAM (DDRAM) Addresses

- Display Data RAM (DDRAM) Addresses inside LCD stores display data represented in 8-bit character codes.
- Its extended capacity is 80*8 bits or 80 characters.
- The area in Display Data RAM (DDRAM) that is not used for display can be used as general data RAM.

Pin No.	Pin Name	Descriptions
1	VSS	Ground
2	VDD	Supply voltage for logic
3	V0	Input voltage for LCD
4	RS	H : Data signal, L Instruction signal
5	R/W	H : Read mode, L : Write mode
6	E	Chip enable signal
7	DB0	Data bit 0
8	DB1	Data bit 1
9	DB2	Data bit 2
10	DB3	Data bit 3
11	DB4	Data bit 4
12	DB5	Data bit 5
13	DB6	Data bit 6
14	DB7	Data bit 7
15	LED_A	Backlight Anode
16	LED_K	Backlight Cathode

- So whatever you send on the DDRAM is actually displayed on the LCD.

		COLUMN										
		1st	2nd	3rd	4th	5th	. . . . .	16th	17th	18th	19th	20th
ROW	1st	00h	01h	02h	03h	04h	. . . . .	0Fh	10h	11h	12h	13h
	2nd	40h	41h	42h	43h	44h	20 x 4 Characters (5x8 dots font)	4Fh	50h	51h	52h	53h
	3rd	14h	15h	16h	17h	18h		23h	24h	25h	26h	27h
	4th	54h	55h	56h	57h	58h	. . . . .	63h	64h	65h	66h	67h

DDRAM Address Map

- GROM Memory

Character Generator ROM memory contains a standard character map all characters that can be displayed on the screen .Each character is assigned to one memory location.

- LCD commands

Command	Function
0F	LCD ON, Cursor blinking ON, Cursor ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Cursor blinking OFF, Display ON
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
08	Display OFF, Cursor OFF
83	Cursor line 1 position 3
3C	Activate second line
38	Use 2 lines and 5x7 matrix
C1	Jump to second line, position1
OC	Cursor OFF, Display ON
C2	Jump to second line, position2

5.2.3 Contribution

At first we interfaced the LCD with controller in 8 it mode, which caused problems later as the number of modules used increased.

To use LCD some of the above commands must be sent for initialization before sending data, so we implemented initialization function and a function to clear LCD screen. For better code organization we have 2 independent functions one to print string variables, other to print numeric variables. Besides a function to set the cursor position to start writing at desired location.

5.2.4 Optimization

In order to decrease number of GPIO pins used to interface between LCD and controller we used LDC in 4 bit mode in which data is sent in two cycles and we modified the code accordingly.

CHAPTER 6: SOFTWARE ARCHITECTURE

6.1 AUTOSAR SOFTWARE ARCHITECTURE

AUTOSAR or Automotive Open System Architecture was developed to create a common standardized software architecture for designing automotive electronic control units (ECUs). The AUTOSAR architecture is based on a 3-layered architecture model,

6.1.1 The 3-Layered AUTOSAR Architecture

The AUTOSAR specifies a three-layer architecture, which are categorized into following modules:

- **Basic software (BSW):** can be defined as standardized software module offering various services necessary to run the functional part of the upper software layer. This layer consists of the ECU specific modules along with the generic AUTOSAR modules.

  It is divided into three sub layers namely the Services layer, ECU (Electronic Control Unit) abstraction layer, and the Microcontroller Abstraction Layer (MCAL).

  The MCAL is a software module that abstracts all the upper layers (the application layer and the BSW) Microcontroller. Thus, MCAL helps in making the upper layers independent of the low-lying hardware platform.

- **Runtime environment (RTE):** acts as a middleware between the AUTOSAR application layer and the lower layers. Basically, the RTE layer manages the inter- and intra-ECU communication between application layer components as well as between the BSW and the application layer.
- **Application layer:** The AUTOSAR application layer includes various application specific software components that are designed to execute specific set of tasks, as per the use-case.

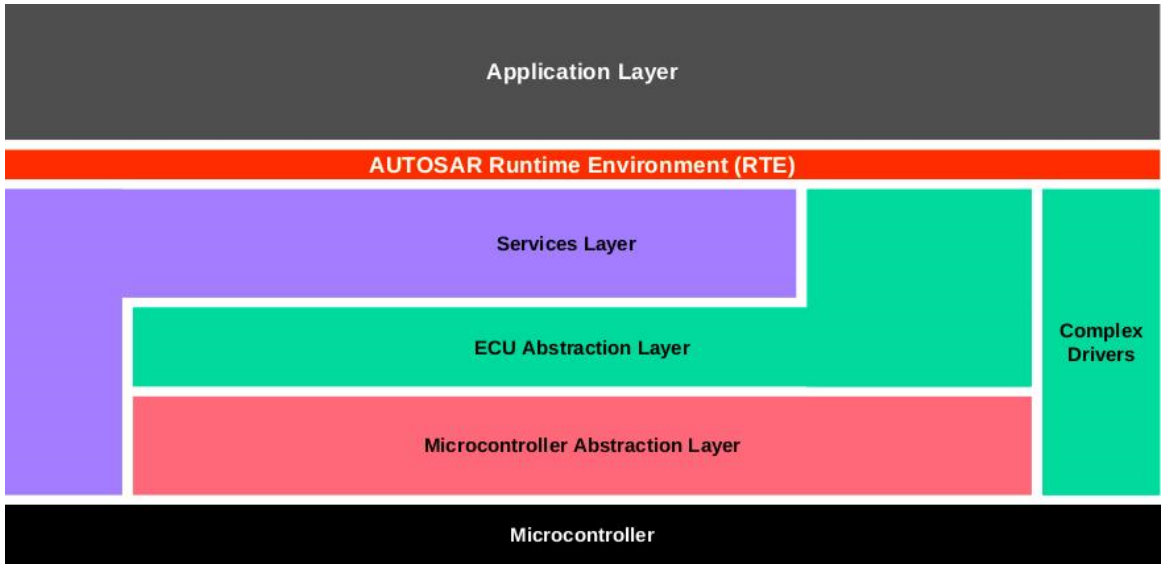


Figure 49 AUTOSAR Layers

6.1.2 Contribution

We Mapped this architecture model to our project in order to achieve the following goals:

1. Modular design
    a. Being able to divide the project into small software modules easy to implement, test and develop.
2. Standardization of Basic software
    a. Having the Application layer independent of basic software layers as to reduce the time and money to be spend on BSW development. AUTOSAR defines the HDL (High level design) and structure of BSW so that only LLD (Low level design) Needs to be taken care by the company to have a standard BSW and concentrate its major focus on development of application software and earn good profit.
3. Software Component Re-Usability:
    a. Being able to use most of software components on different microcontroller structures. Previously, the software components were written and were dependent on underlying hardware and hence each software specific to its ECU. But AUTOSAR has made software components independent.

In order to achieve this goals we divided our software into 3 layers

1. Application layer which contains the logic for the 3 applications mentioned before.
2. Basic software layer which is divided into 3 layers
    a. MCAL layer which abstracts upper layer components and contains implementation of
        i. UART2 driver, with baud rate of 9600 and in interrupt mode support Bluetooth module
        ii. UART3 driver with baud rate of 115200 in normal model to support GPS
        iii. UART7 driver with baud rate of 115200 in both modes to support WIFI module
        iv. Phase locked loop driver to increase ARM frequency from 16MHz to maximum frequency of 80 MHz.
        v. Timer1 sub timer A driver to support speed sensor
        vi. Timer0 A and Timer0 drivers B in PWM mode to interface with motors
        vii. Timer 2 driver to support ultrasonic module
        viii. GPIO driver Init for all pins on the Microcontroller.
    b. Hard ware abstraction layer which abstracts the interfacing with MCAL layer drivers in order to get raw date to use in upper layers.
        i. WIFI HAL implements the needed simple operations of sending AT commands to WIFI module and receiving response.
        ii. GPS HAL implements the operation of reading the needed frame using MCAL layers.
        iii. Bluetooth HAL handles the reception of sent bytes by the module in an interrupt service routine and produces the needed signals on Motors GPIO pins.
        iv. Ultrasonic HAL handles the operation of distance measurements inside echo pins ISR.
        v. Speed Sensor HAL Handles the operation on calculating speed inside Timer1 ISR and PORTF ISR.

c. Service layer which offers a full abstraction of Basic software layers by offering some logic operated over the raw data from HAL layer to make it useful to be used directly by OS and Application layer, and includes the services of:
    i. WIFI send/receive Full car frame
    ii. getting GPS Coordinates of my car
    iii. Display any needed text over LCD
    iv. Getting ultrasonic Reading
    v. Motor control information.

3. Real time operating system layer, which is used to manage system tasks and make use of the defined services in service layer.
4. Application layer, which contains logic implementation of whole project.

Note: there might be some interference between layers (not fully abstracted) due to some SW/HW constrains.

Project Architecture model

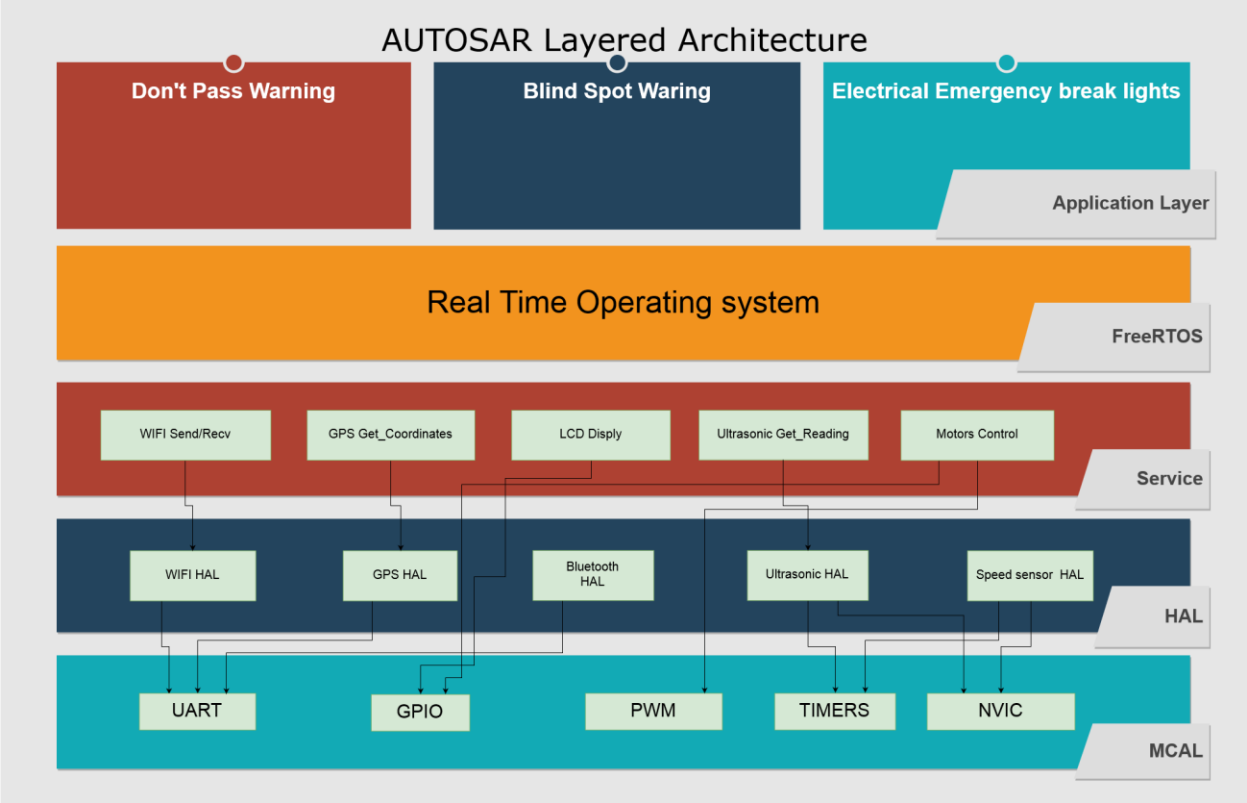


Figure 50 Our AUTOSAR layered Architecture

6.2 REAL TIME OPERATING SYSTEM

6.2.1 Introduction to Real time operation systems

An operating system is a computer program that supports a computer's basic functions, and provides services to other programs (or applications) that run on the computer. The applications provide the functionality that the user of the computer wants or needs. The services provided by the operating system make writing the applications faster, simpler, and more maintainable. If you are reading this web page, then you are using a web browser (the application program that provides the functionality you are interested in), which will itself be running in an environment provided by an operating system.

Most operating systems appear to allow multiple programs to execute at the same time. This is called multi-tasking. In reality, each processor core can only be running a single thread of execution at any given point in time. A part of the operating system called the scheduler is responsible for deciding which program to run when, and provides the illusion of simultaneous execution by rapidly switching between each program.

The type of an operating system is defined by how the scheduler decides which program to run when. For example, the scheduler used in a multi user operating system (such as Unix) will ensure each user gets a fair amount of the processing time. As another example, the scheduler in a desk top operating system (such as Windows) will try and ensure the computer remains responsive to its user.
The scheduler in a Real Time Operating System (RTOS) is designed to provide a predictable (normally described as deterministic) execution pattern. This is particularly of interest to embedded systems as embedded systems often have real time requirements. A real time requirement is one that specifies that the embedded system must respond to a certain event within a strictly defined time (the deadline). A guarantee to meet real time requirements can only be made if the behavior of the operating system's scheduler can be predicted (and is therefore deterministic).

Traditional real time schedulers, such as the scheduler used in FreeRTOS, achieve determinism by allowing the user to assign a priority to each thread of execution. The scheduler then uses the priority to know which thread of execution to run next. In FreeRTOS, a thread of execution is called a task.

6.2.2 FreeRTOS

FreeRTOS is a class of RTOS that is designed to be small enough to run on a microcontroller - although its use is not limited to microcontroller applications.

A microcontroller is a small and resource constrained processor that incorporates, on a single chip, the processor itself, read only memory (ROM or Flash) to hold the program to be executed, and the random-access memory (RAM) needed by the programs it executes. Typically, the program is executed directly from the read only memory.

Microcontrollers are used in deeply embedded applications (those applications where you never actually see the processors themselves, or the software they are running) that normally have a very specific and dedicated job to do. The size constraints, and dedicated end application nature, rarely warrant the use of a full RTOS implementation - or indeed make the use of a full RTOS implementation possible. FreeRTOS therefore provides the core real time scheduling functionality, inter-task communication, timing and

synchronization primitives only. This means it is more accurately described as a real time kernel, or real time executive. Additional functionality, such as a command console interface, or networking stacks, can then be included with add-on components.

FreeRTOS is a real-time operating system kernel for embedded devices that has been ported to 35 microcontroller platforms. It is distributed under the MIT License.
FreeRTOS is designed to be small and simple. The kernel itself consists of only three C files. To make the code readable, easy to port, and maintainable, it is written mostly in C, but there are a few assembly functions included where needed (mostly in architecture-specific scheduler routines).

FreeRTOS provides methods for multiple threads or tasks, semaphores and software timers. A tick-less mode is provided for low power applications. Thread priorities are supported. FreeRTOS applications can be completely statically allocated. Alternatively, RTOS objects can be dynamically allocated with different schemes of memory allocation provided includes but not limited to:

1. allocate only;
2. allocate and free with a very simple, fast, algorithm no protection
3. C library allocate and free with some mutual exclusion protection.

There are none of the more advanced features typically found in operating systems like Linux or Microsoft Windows, such as device drivers, advanced memory management, user accounts, and networking. The emphasis is on compactness and speed of execution. FreeRTOS can be thought of as a 'thread library' rather than an 'operating system',

FreeRTOS implements multiple threads by having the host program call a thread tick method at regular short intervals. The thread tick method switches tasks depending on priority and a round-robin scheduling scheme. The usual interval is 1/1000 of a second to 1/100 of a second, via an interrupt from a hardware timer, but this interval is often changed to suit a particular application.

How to use FreeRTOS?
FreeRTOS is supplied as source code. The source code should be included in your application project. Doing so makes the public API interface available to your application source code.

When using FreeRTOS your application should be written as a set of independent tasks. This means your main() function does not contain the application functionality, but instead creates the application tasks, then starts the RTOS kernel. See the main.c and project files (makefile or equivalent) included with each port for examples.

6.2.3 Basic operations of FreeRTOS

Task Creation

- BaseType_t xTaskCreate(TaskFunction_t pvTaskCode,
  const char * const pcName,
  configSTACK_DEPTH_TYPE usStackDepth,
  void *pvParameters,
  UBaseType_t uxPriority,
  TaskHandle_t *pxCreatedTask
  );

Task control

```
void vTaskPrioritySet( TaskHandle_t xTask,UBaseType_t uxNewPriority );
void vTaskDelay( const TickType_t xTicksToDelay );
void vTaskSuspend( TaskHandle_t xTaskToSuspend );
void vTaskResume( TaskHandle_t xTaskToResume );
```

RTOS Kernal control

```
void taskENTER_CRITICAL( void );
void taskEXIT_CRITICAL( void );
void vTaskStartScheduler( void );
```

semaphores

```
SemaphoreHandle_t xSemaphoreCreateBinary( void );
```

6.2.4 FreeRTOS Configuration

```
#define configUSE_PREEMPTION                1
#define configCPU_CLOCK_HZ                  80000000
#define configTICK_RATE_HZ                  1000 //1000 ticks in second
#define configMAX_PRIORITIES                5
#define configMINIMAL_STACK_SIZE            128
#define configUSE_TIME_SLICING              0
/* Memory allocation related definitions. */
#define configSUPPORT_STATIC_ALLOCATION     0
#define configSUPPORT_DYNAMIC_ALLOCATION    1
#define configTOTAL_HEAP_SIZE               20000
```

6.2.5 Tasks Description

Task name	Ultrasonic_1_Reading
Description	Handles the operation of ultrasonic modules through: Reordering Ultrasonic modules priority, triggers module signal to start measurments
Semaphores	Task is blocked with a semaphore which is down after sending trigger and released when echo pin is high to allow the task to resume.
Data sharing	The task shares it's data with other tasks through a global struct Ultrasonic_reading, which stores the 6 ultrasonic readings.
Periodicity	N/A
Priority	1
Allocated stack	150

Task name	GPS_Task
Description	Receive full GLL frame and extracts latitude, longitude and altitude values out of the received frame.
Semaphores	N/A
Data sharing	The task shares it's data with other tasks through a global struct Ultrasonic_reading, which stores the 6 ultrasonic readings.
Periodicity	4 seconds
Priority	2
Allocated stack	128

Task name	WIFI_Frame_Send
Description	Handles the operation of sending a frame of data to other cars through, listing them, connection establishment and sending data.
Semaphores	N/A
Data sharing	The task shares it's data with other tasks through a global struct Ultrasonic_reading, which stores the 6 ultrasonic readings.
Periodicity	12 seconds
Priority	4
Allocated stack	150

Task name	Applications_Detection
Description	
Semaphores	N/A
Data sharing	The task shares it's data with other tasks through a global struct Ultrasonic_reading, which stores the 6 ultrasonic readings.
Periodicity	5 seconds
Priority	3
Allocated stack	180

6.3 APPLICATION LAYER

6.3.1 EMERGENCY ELECTRONIC BRAKE LIGHTS (EEBL)

when there is a hard-braking vehicle in the pass ahead,
three vehicles are traveling in the same lane you are driving the last vehicle and you can't see the first vehicle because it's been blocked by the car in front of you ,unexpectedly the vehicle slams on its brakes ,duo to v2v communication the car can produce a waning of the braking car ahead

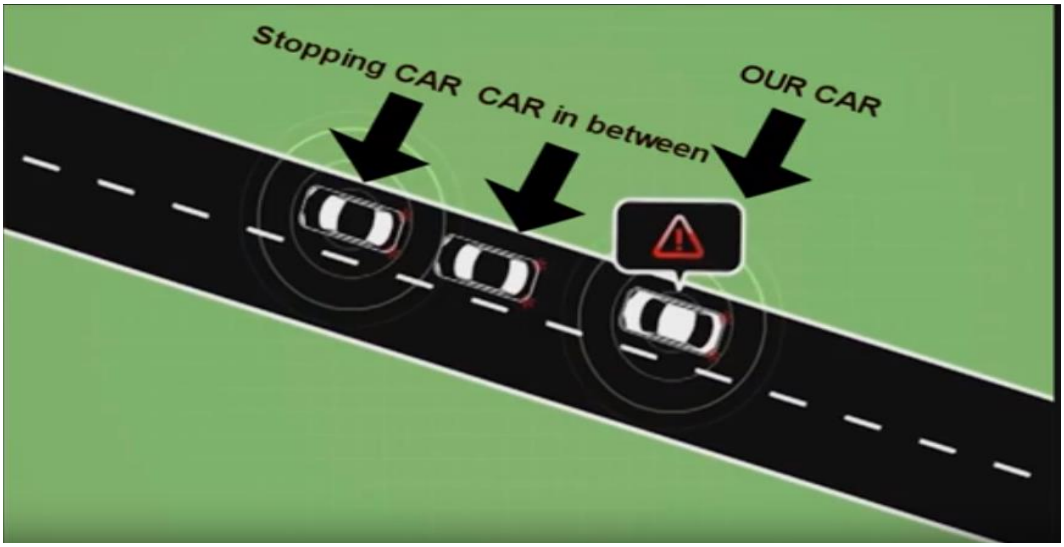


Figure 51 EEBL Application

6.3.2 BLIND SPOT WARNING(BSW)

when there is a car that may not be visible to the car driver, because of v2v communication a warning message is issued to make you aware of the presence of this vehicle, should you attempt to lane change this warning message will tell you that it's not safe to lane change

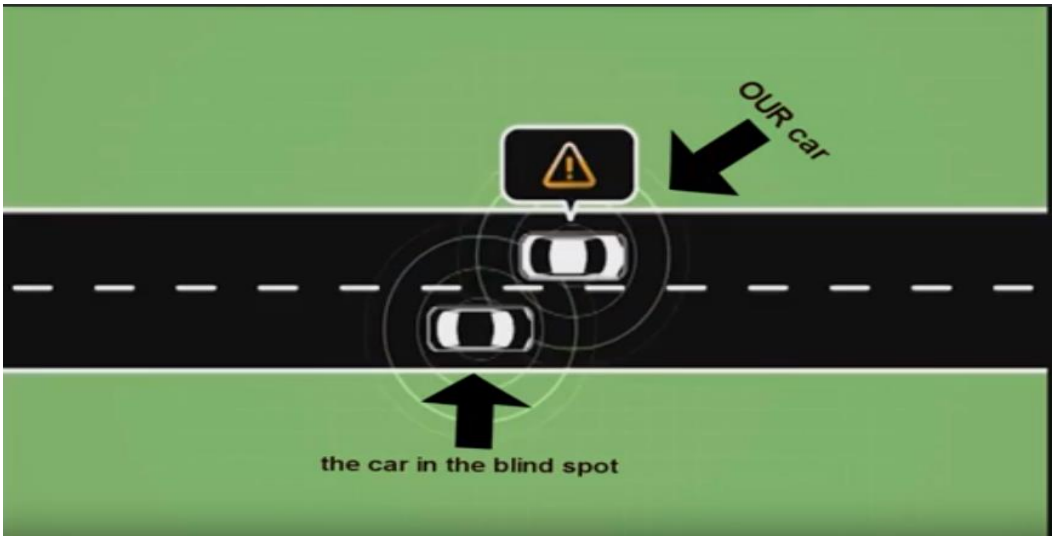


Figure 52 BSW Application

6.3.3 DO NOT PASS WARNING(DNPW)

a safety application is intended to let the driver know that it's not safe to attempt to pass a slower moving vehicle because of incoming traffic in the passing zone, if a vehicle is detected in the passing done a warning message is provided letting you know that the passing situation is potentially unsafe

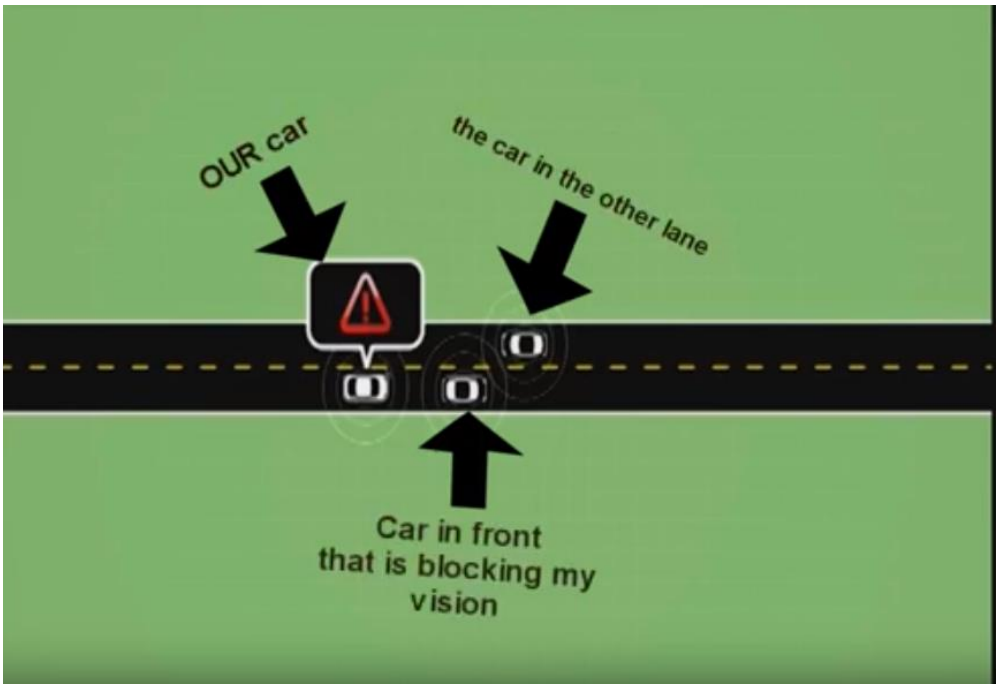
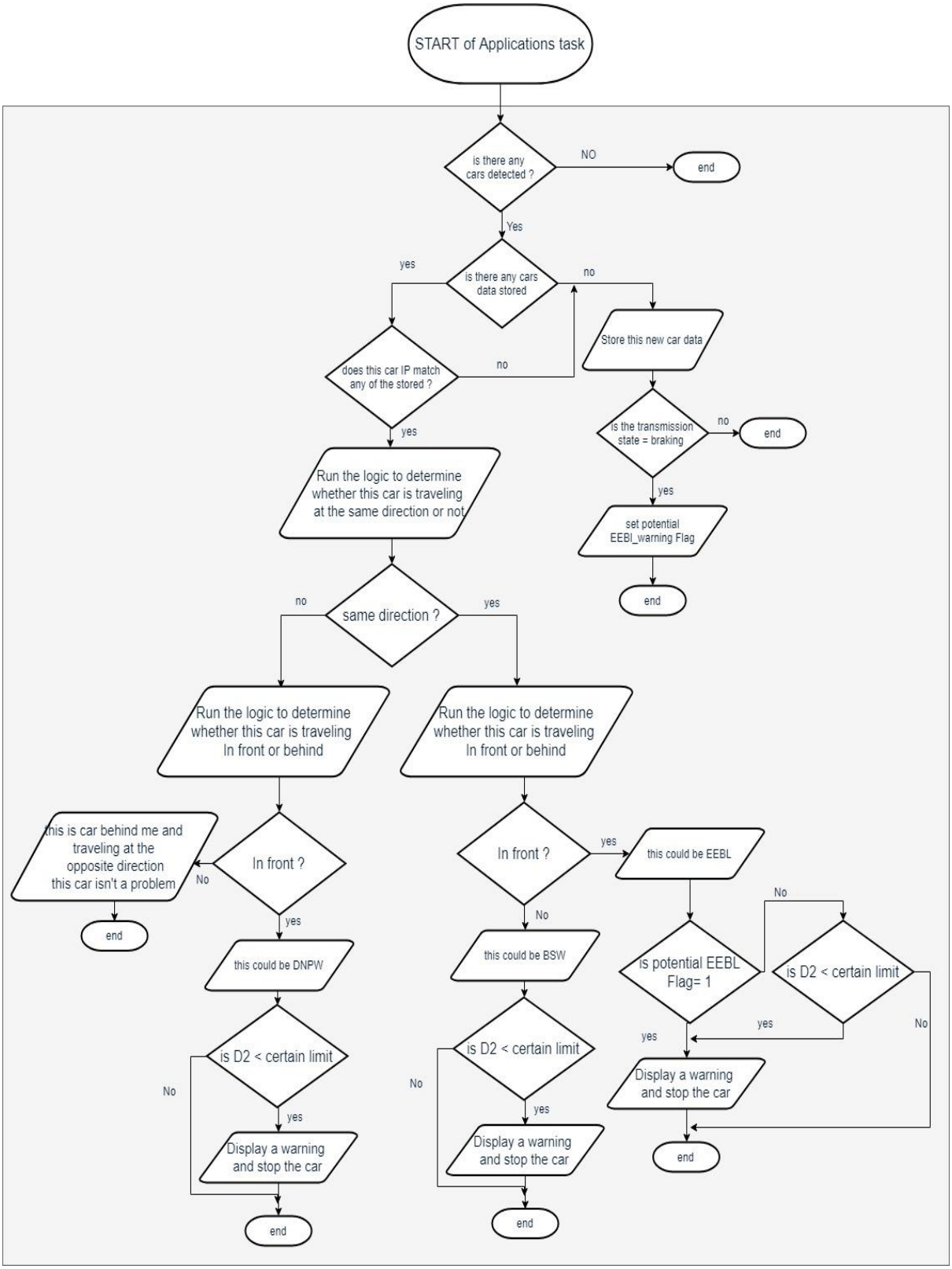


Figure 53 DNPW Application

these are some of the application but there are a lot of other applications

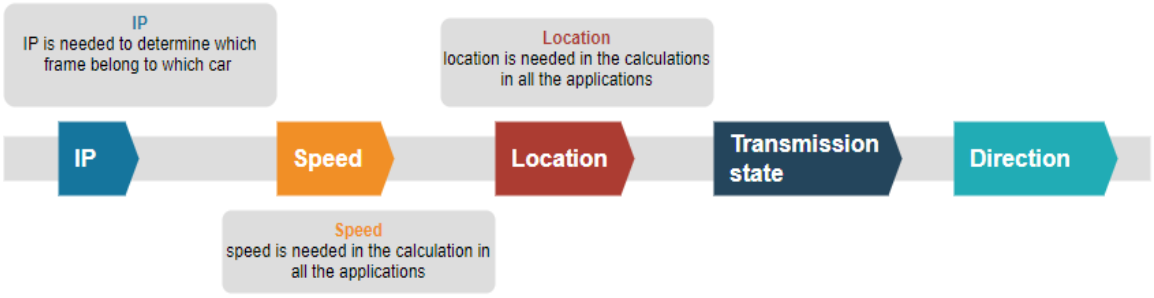
6.3.4 Implementation

We have an Application task with a frequency of 5 seconds, this application task does the following



We will discuss this flow chart in details.

We mentioned in previous chapters that we use the frame that other cars send and our data to determine whether there is a potential crash or not, this frame consists of:



So, the question is how do we know that there is a potential accident based on this data, for start we won't be able to determine anything based on a single frame data, because this frame tell us the speed of the other vehicle and the distance between us, but doesn't tell whether this car is behind me of Infront of me or whether this car is traveling at the same direction or the opposite direction

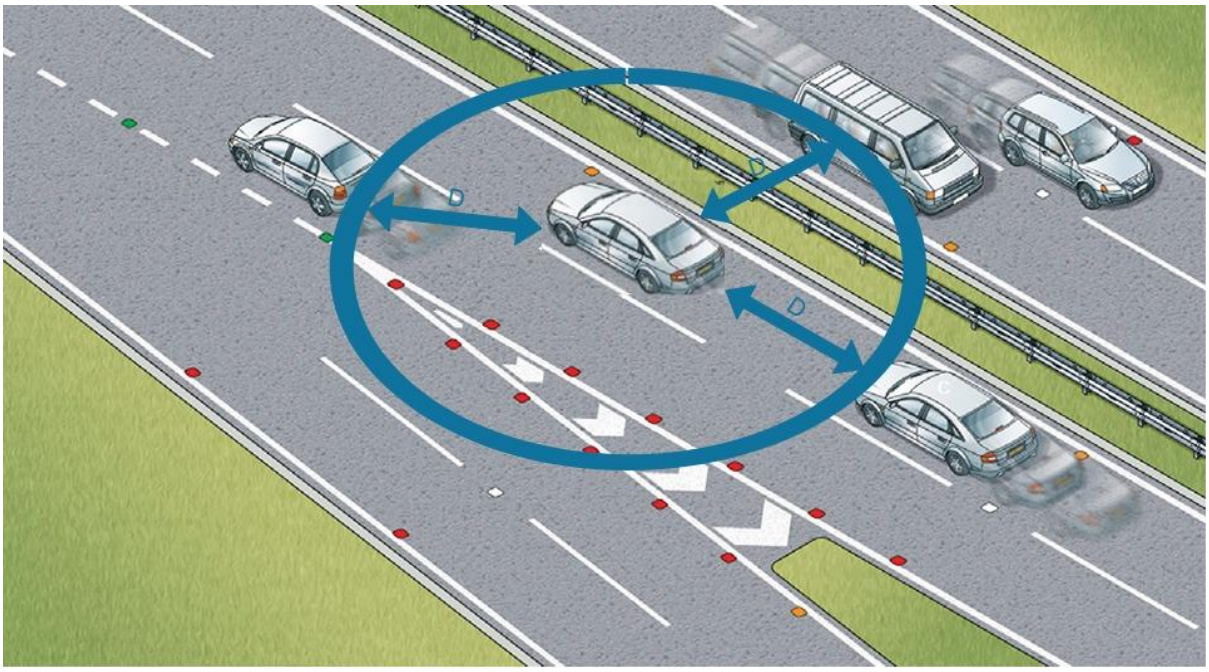


Figure 54 1st Single frame problem

As shown in the picture the distance between our car and the three cars is the same, but each one of them is traveling in a different direction and one of them is behind me and the other is in front of me, the distance we get from the GPS is a radius of a circle, our car is in the middle and the other car could be anywhere on the circumference of this circle.

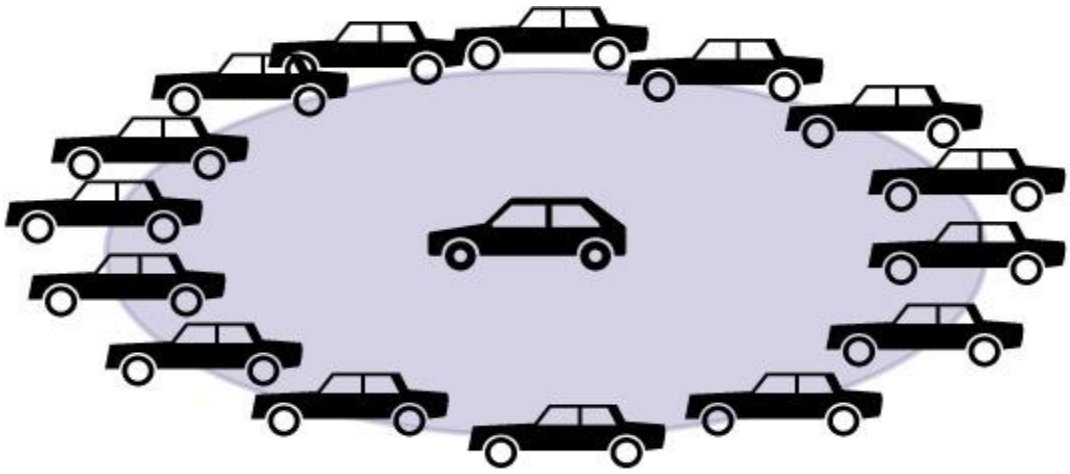


Figure 55 2nd Single frame problem

So, we need at least two frames to determine the location of the other car, based on that in order to determine that there is a potential accident of the three cases we talked about we have to determine the following:

1. Is this traveling at the same direction or opposite to my direction
2. Is this car Infront of me of behind me

if we could determine this information about a certain car, we will be able to tell if there will be a potential accident or not.

Is this car traveling at the same direction or opposite to my direction

Assume:

1-that the frequency of sending a frame is T

2-the distance that our car has traveled between receiving two frames is d1

3-the distance that the other car has traveled between sending two frames for our cars is d2

4-the distance between our car and the other car at time of the first frame is D1

5- the distance between our car and the other car at time of the second frame is D2

To get the condition for determining whether the other car is traveling at the same direction or the opposite, we draw the scenarios.

1- The other car is traveling at the same direction

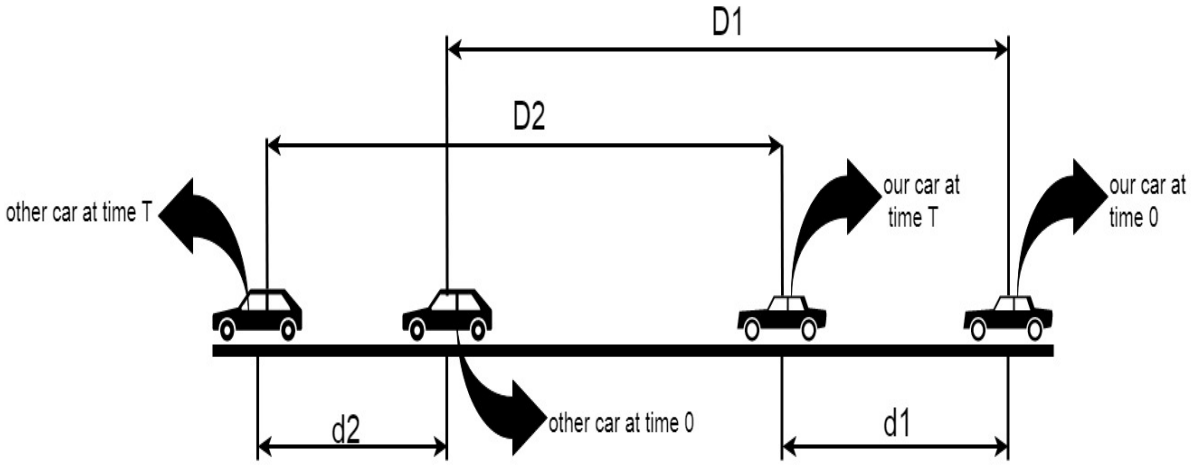


Figure 56 Both Cars at the same direction

We know that our car is moving and from the transmission state of the other car we know that it is moving, even if our car and the other car are moving with negative acceleration,

$$d1 \ \& \ d2 > 0$$

from the drawing

$$D1 + d2 = D2 + d1$$

Then

$$D2 = (D1 - d1) + d2 \rightarrow \boxed{1}$$

2- The other car is traveling at the opposite direction

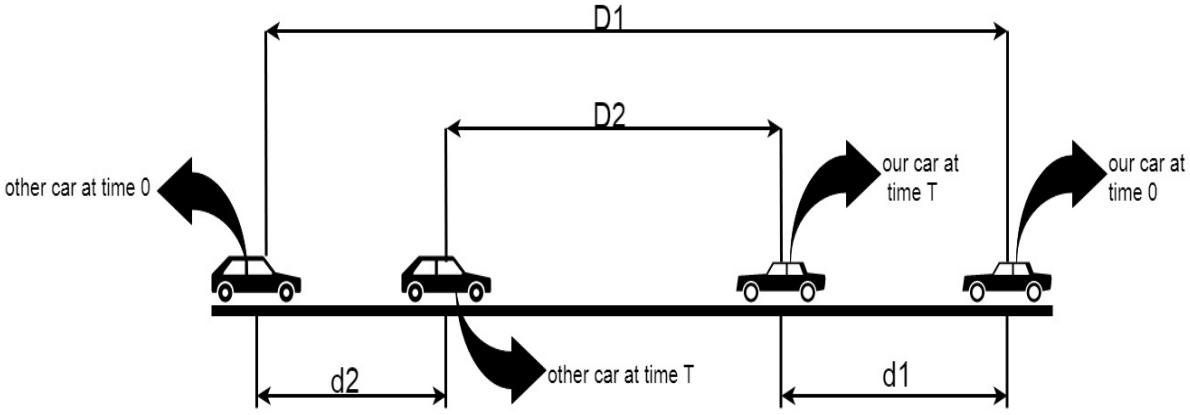


Figure 57 Cars at opposite direction

As before

$$d1 \,\&\, d2 > 0$$

from the drawing

$$D1 = D2 + d1 + d2$$

Then

$$D2 = (D1 - d1) - d2 \rightarrow \boxed{2}$$

Our car knows D1, D2 and d1 which we will show later how to calculate it, but it doesn't know d2, from equation 1 & 2 our car doesn't need to know d2, it can compare D2 to the threshold value (D1-d1) and since we know that d2 > 0, this comparison is valid.

$$\boxed{D2 \leqslant (D1 - d1)}$$

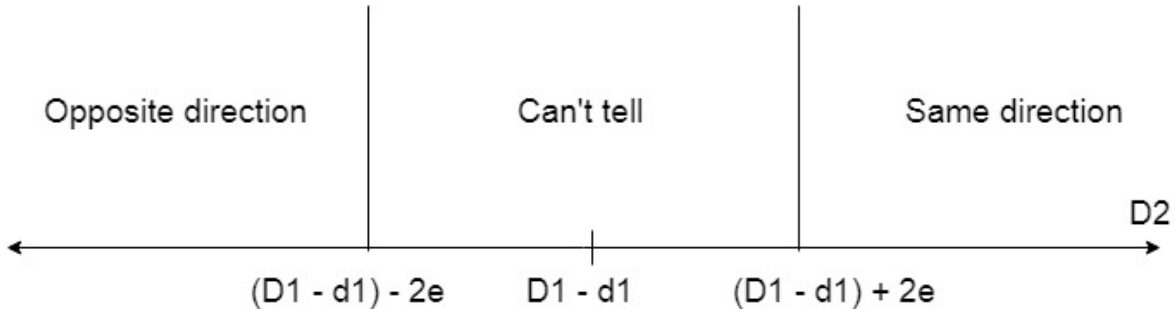
If D2 is greater than (D1-d1) then the other car is definitely traveling at the same direction, and if D2 is less than (D1-d1) then the other car is definitely traveling at the opposite direction.

If we consider the GPS accuracy,

$$D2 \pm e \leqslant (D1 \pm e - d1)$$

Then

$$\boxed{D2 \leqslant (D1 - d1) \pm 2e}$$



If D2 is greater than (D1 – d1) + 2e then the other car is traveling at the same direction, and vice versa, but at this area in between we can't tell where is the other car, the ideal case for perfect GPS, this area become zero.

What if D2 = D1 -d1, this is the case at which the other isn't traveling neither at the same direction nor the opposite direction, this other has stopped before the first frame.

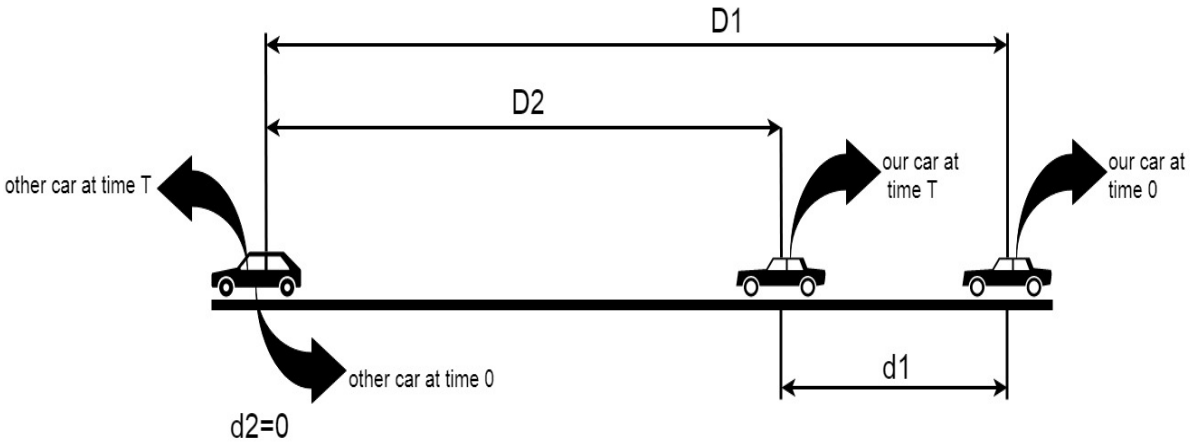
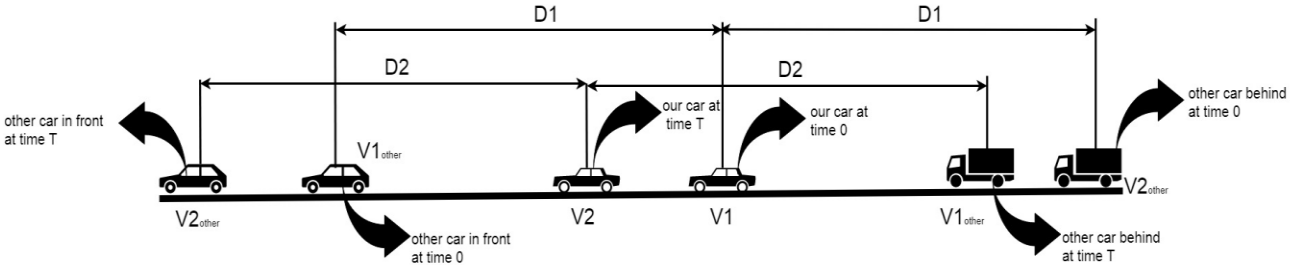


Figure 58 other car has stopped

At this case our car receives at the first frame that the other car has stopped, we handle this special case alone.

Is this car Infront of me of behind me
This depends on whether the other car is traveling at the same or the opposite direction

Same direction



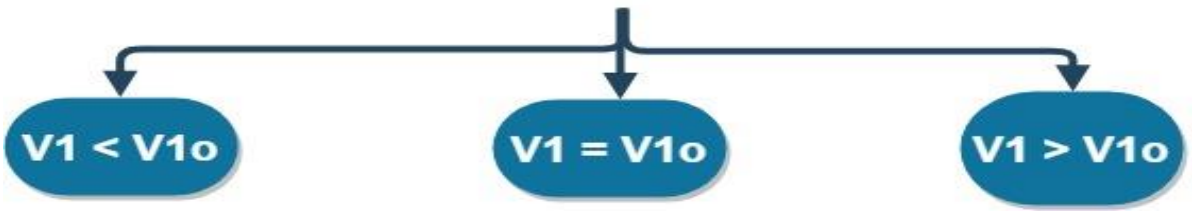
V1 is our car speed at the 1st frame

V2 is our car speed at the 2nd frame

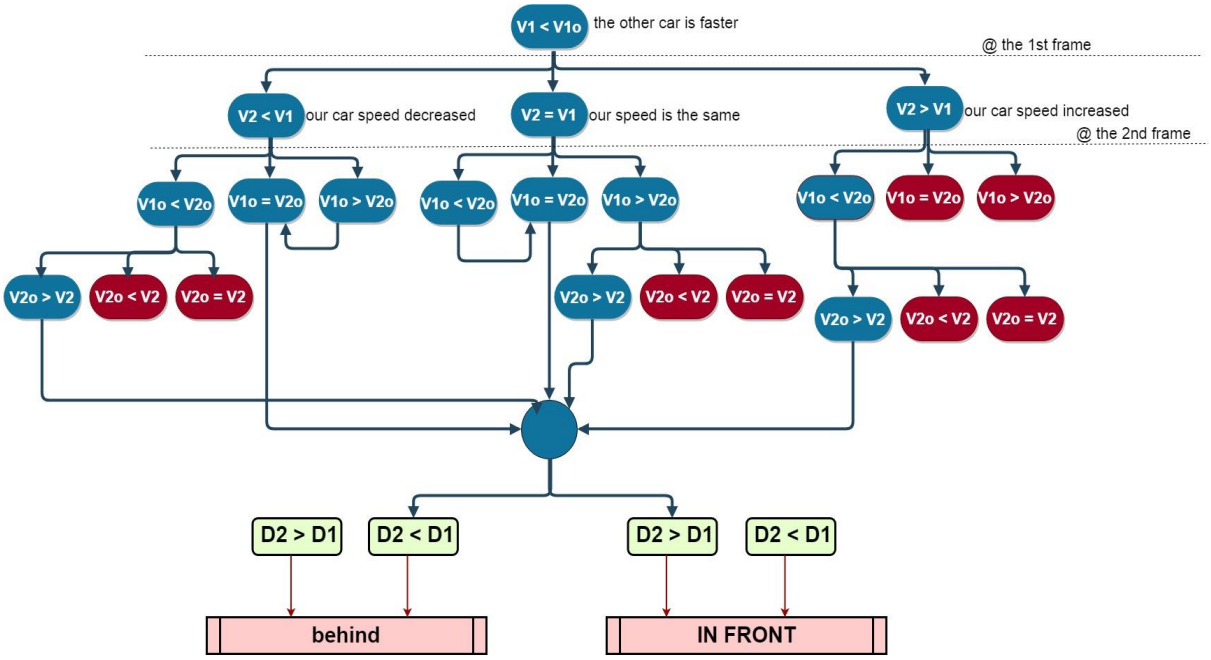
V1o is the other car speed at the 1st frame

V2o is the other car speed at the 2nd frame

To determine whether the other car is in front or behind, we will depend on our car speed and other car speed at time 0 and time T, and also the distance D1 and D2, our car now has 6 values for a single car which means a lot of comparisons, we divide this comparison into three large comparisons



1- If the other car is faster at the 1st frame



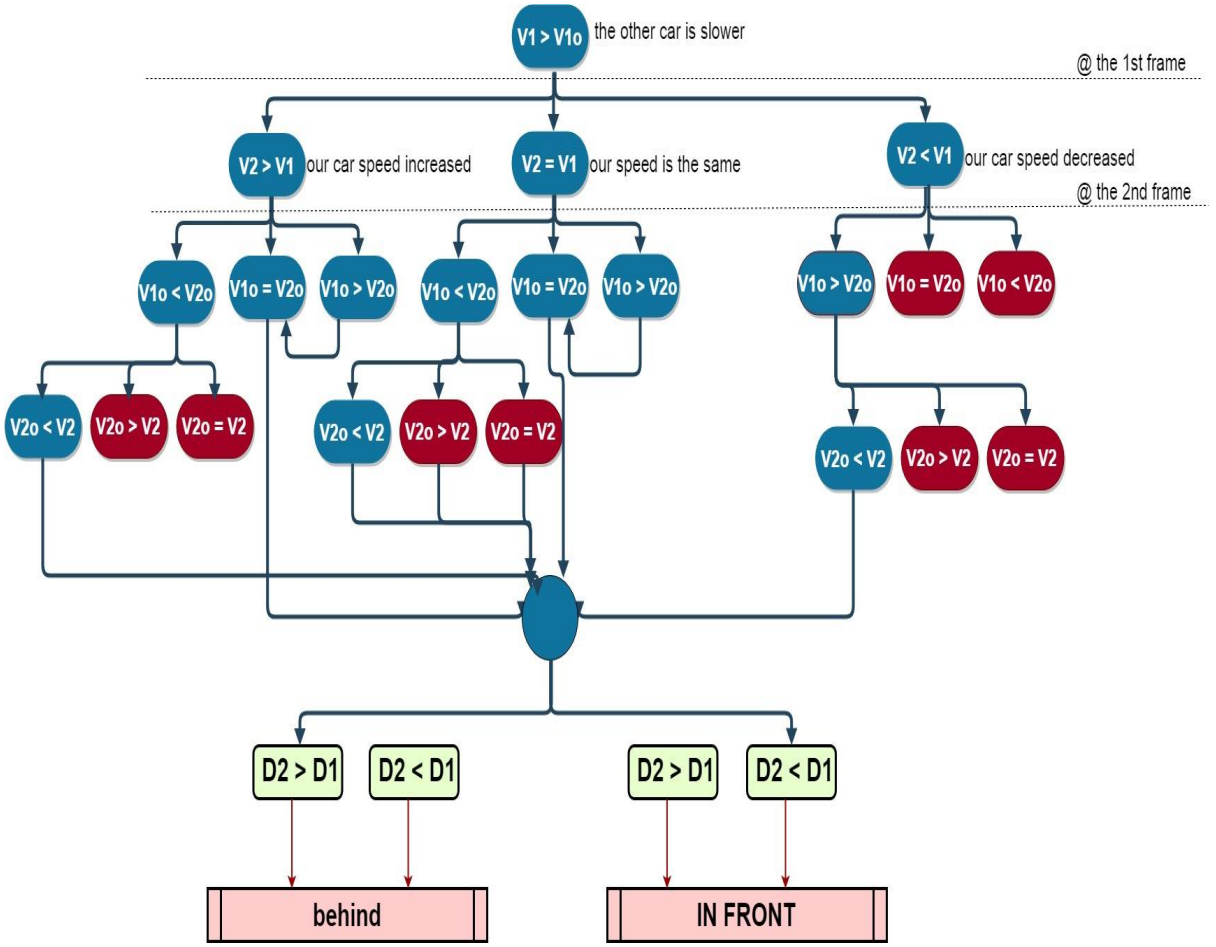
This chart explains the different combinations of V1, V2, V1o, V2o, D1 and D2, the red circles are the case at which we can't tell whether the other is in front or behind us, these case require more information about the other car, like the acceleration but we didn't implement that due to short time, for example the case at which

1- V1 < V1o meaning the other car is faster at the 1st frame
2- V2 = V1 meaning that our car speed didn't change
3- V1o > V2o meaning that the other car speed decreased
   Here we don't whether the other car speed has become less that our car speed or not
   If the other car speed has decreased but still greater than our speed then there is no problem, the problem happens when it becomes less than our car speed, at this case we need more information about the other car, like the deceleration.

The blue circles are the conditions that our car can solve with the current frame data, for example the case at which

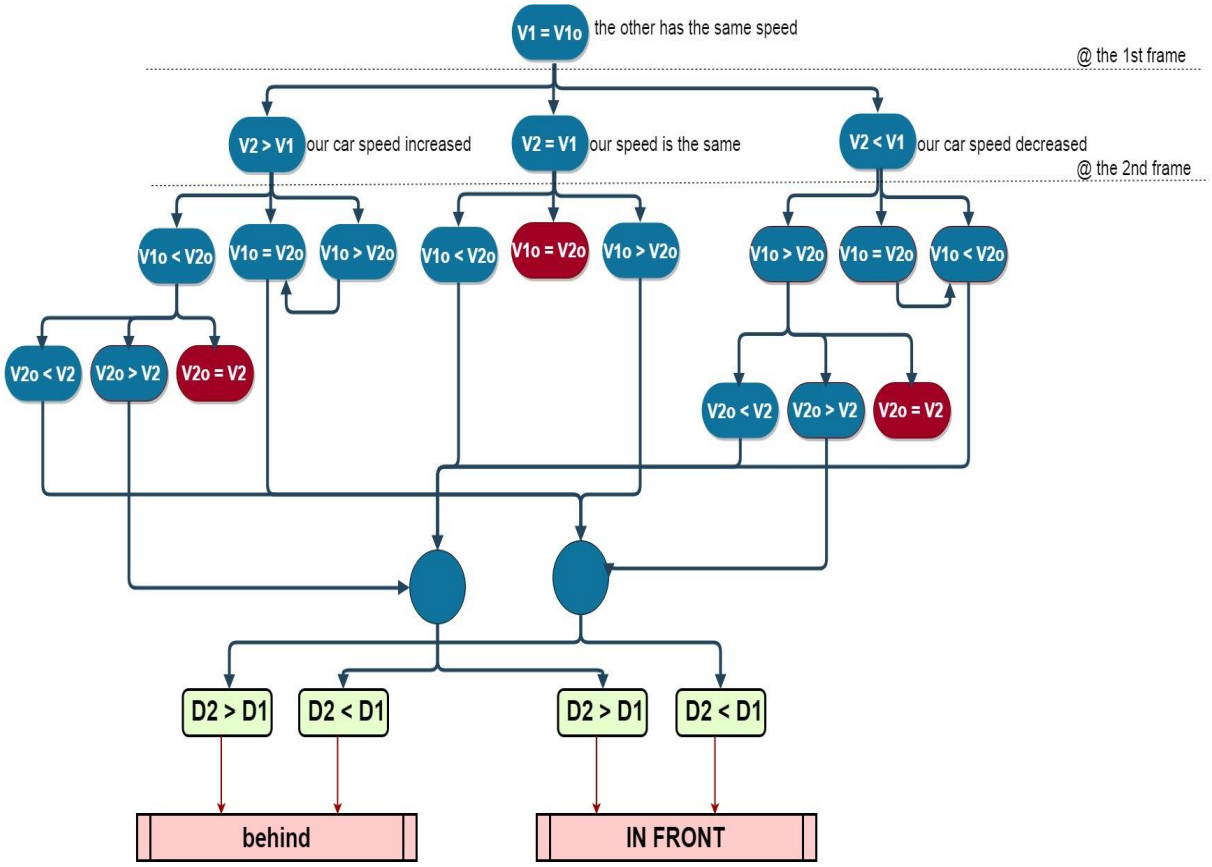
1- V1 < V1o meaning the other car is faster at the 1st frame
2- V2 = V1 meaning that our car speed didn't change
3- V1o = V2o meaning that the other car speed didn't change, it still greater than our car speed
   This means that overall the other car is faster than our car, if the distance decreases then, the other car is definitely behind us, and if the distance increase then the other car is definitely in front of us.

2- If the other car is slower at the 1st frame



This is an exact mirror of the previous case

3- If the other car is traveling with the same speed at the 1st frame



Here the red condition are the conditions at which the speed is the same at the first and the second frame.

opposite direction

here the condition is much easier, if the distance increases then the other car is definitely behind our car, and if the distance decrease then the other car is definitely Infront of us.

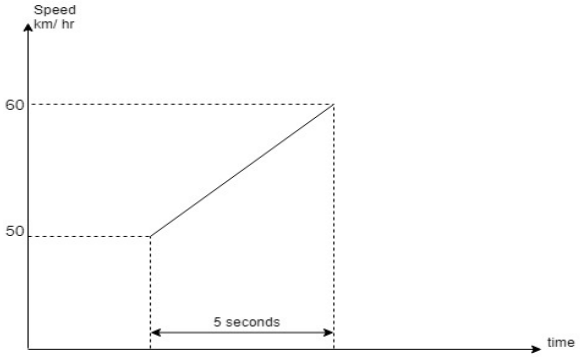
$$D2 \lesseqgtr D1$$

Now we can determine whether the other car is Infront of us or behind us and determine whether the other car is traveling in the direction or the opposite direction based on the equation

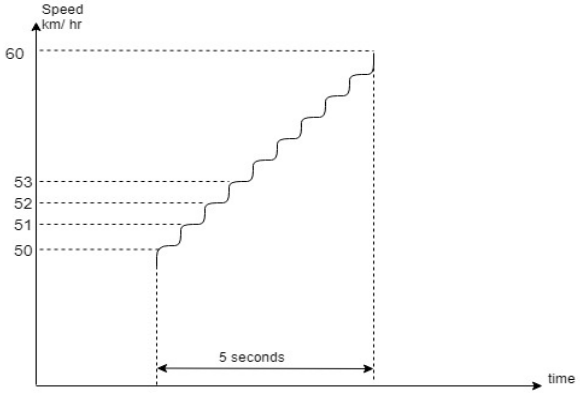
$$D2 \lesseqgtr (D1 - d1)$$

At this equation our car knows D2, D1 and can calculate d1, as we will show later, we have a task that check for some conditions that will determine whether there is an accident or not. This task will have a frequency of 5 seconds, meaning that every 5 seconds the car check for a potential accident, this task will calculate the distance d1, we assume an assumption to calculate that:

During this 5 second period the transition between different speeds happens in 0 time, meaning that if car normally increase the speed from 50 km/hr to 60 km/hr it will increase linearly like this.



We assume it is like this



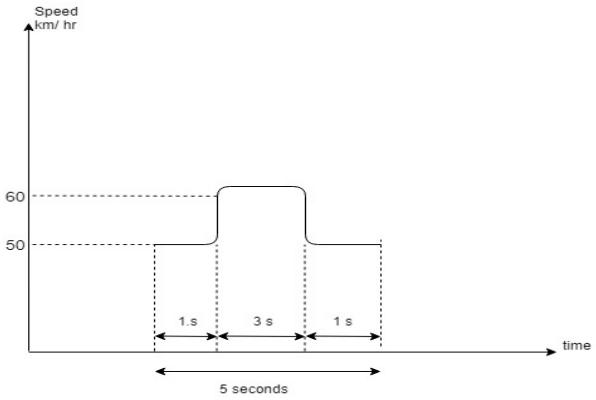
This assumption is valid because:

1- 5 seconds period is too small for large changes
2- We excluded the braking from this, meaning that if the other car has stopped, we will deal with it with different method, and won't use this calculated distance
3- We controlle our car with a program that has 10 speed levels from 0 to 9, and our car max speed is 140 cm/sec

Based on this assumption, we can calculate d1 using the simple equation

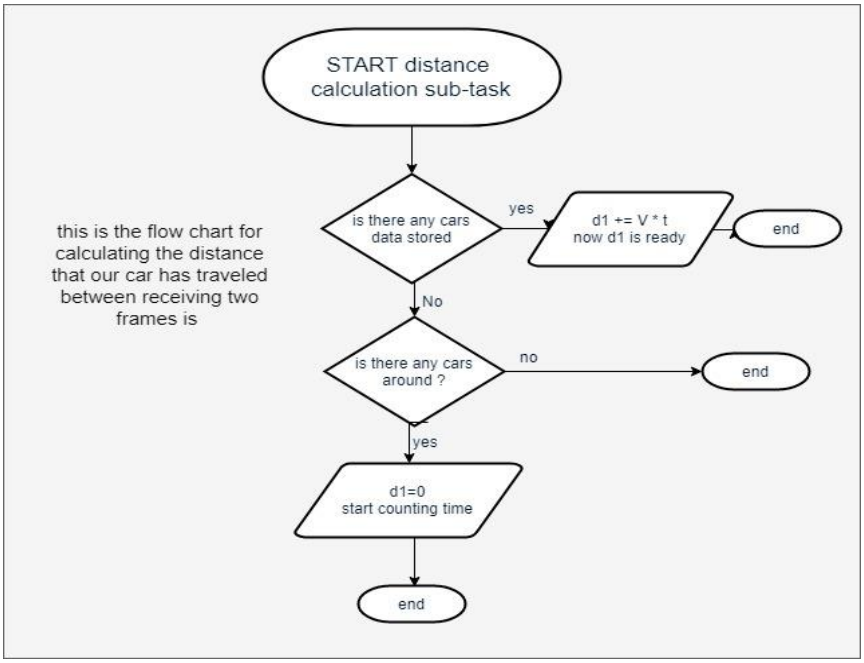
$$d1 = V * t$$

whenever the speed changes multiply the old speed by the time it took then add it to the total distance. For example, let the speed graph at this 5 second period is the following



The distance = 1 * 50 *(1000/3600) + 3 *60*(1000/3600) + 1 *50*(1000/3600) = 77.7m

This distance d1 calculation starts at the Applications task that has a frequency of 5 seconds, if the speed changed, we multiply the old speed with the time it took, at the next call for this task , we multiply the time from the last change of the speed by the current speed, the flowchart for this will be



When we change the speed, we apply this

- d1 += previous speed * time
- reset the timer for counting

now we can determine where is the other car exactly based on the previous explanation, determining the use case now is easy.

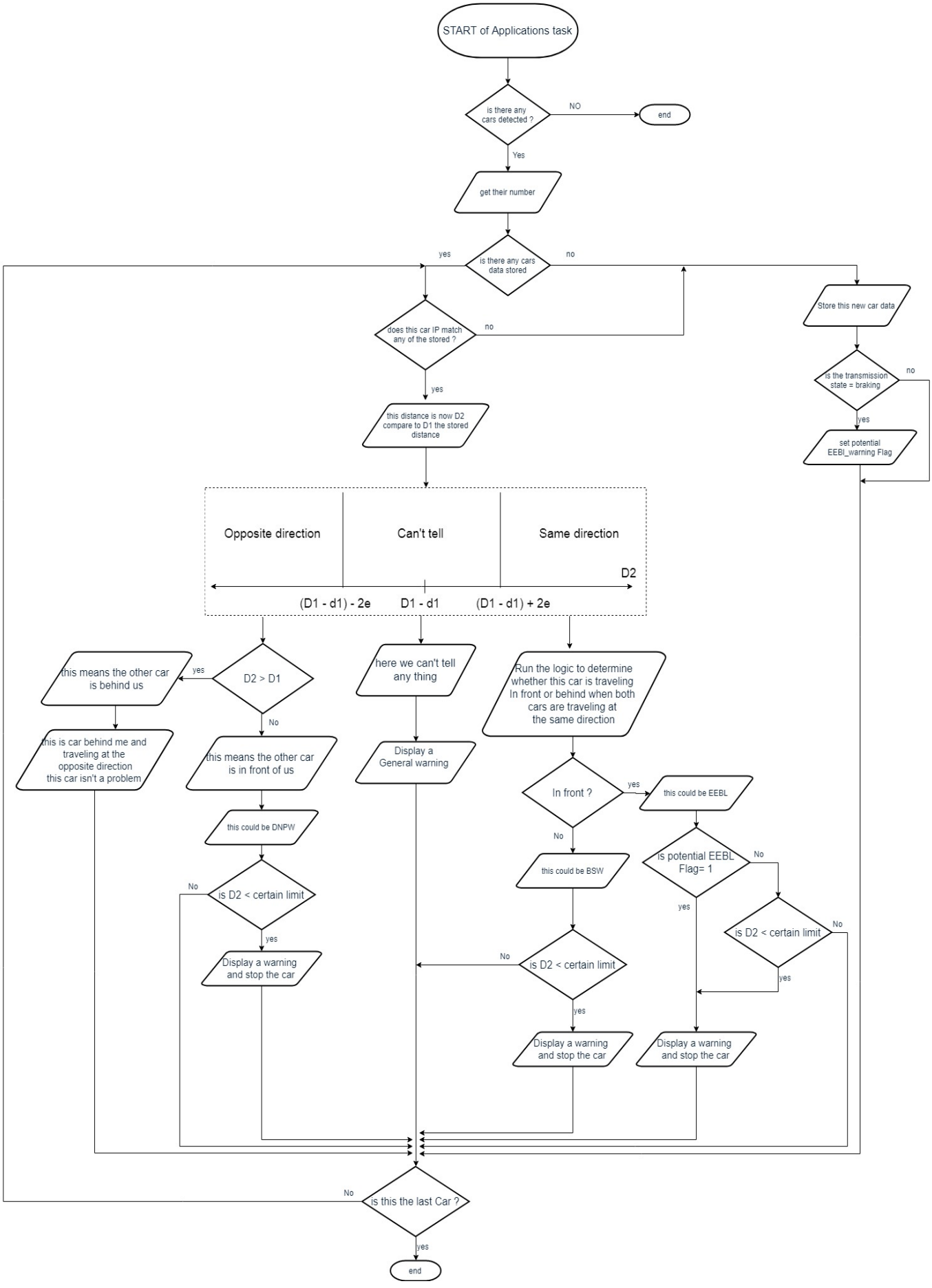
EMERGENCY ELECTRONIC BRAKE LIGHTS (EEBL) is the use case at which the other car is in front and traveling at the same direction but with the distance decreasing or with the braking transmission state.

So, we get 2 frames of the other car to verify that the other car is at EEBL use case.

There is a case for EEBL at which the other has stopped from the first frame, we can't wait for the second frame to verify that this car is in front of us, at this case we issue a potential EEBL warning, letting the driver know that there might be an EEBL accident, and the car can't tell because we only have one frame from this car, at this use case we depend on the front ultrasonic to detect any vehicle ahead, with detection range higher than the normal range for the ultrasonic task.

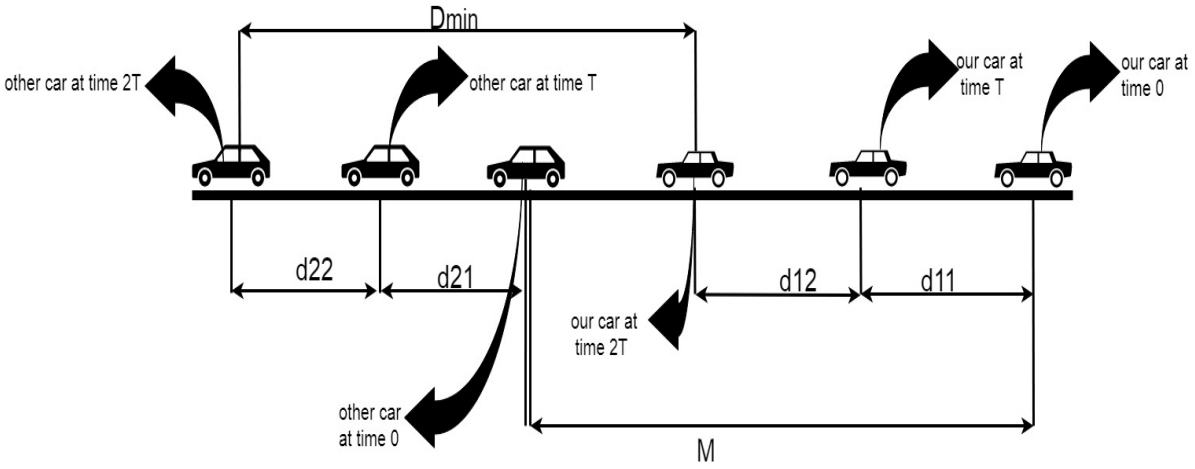
BLIND SPOT WARNING(BSW) is the use case at which the other car is traveling at the same direction, and travel behind us, DO NOT PASS WARNING(DNPW) is the use case at which the other car is traveling the opposite direction, and travel Infront of us.

The flow chart for the Application task, in details is the following:


START of Applications task
is there any cars detected ?
NO
end
Yes
get their number
is there any cars data stored
yes
no
does this car IP match any of the stored ?
no
yes
this distance is now D2 compare to D1 the stored distance
Store this new car data
is the transmission state = braking
no
yes
set potential EEBL_warning Flag
Opposite direction
Can't tell
Same direction
D2
(D1 - d1) - 2e
D1 - d1
(D1 - d1) + 2e
D2 > D1
yes
this means the other car is behind us
this is car behind me and traveling at the opposite direction this car isn't a problem
No
this means the other car is in front of us
this could be DNPW
is D2 < certain limit
No
yes
Display a warning and stop the car
here we can't tell any thing
Display a General warning
Run the logic to determine whether this car is traveling In front or behind when both cars are traveling at the same direction
In front ?
yes
this could be EEBL
No
this could be BSW
is potential EEBL Flag= 1
No
yes
is D2 < certain limit
No
yes
is D2 < certain limit
No
yes
Display a warning and stop the car
Display a warning and stop the car
is this the last Car ?
No
yes
end

6.3.5 Max speed calculations

We need to calculate the max speed of the vehicle based on our design, at EEBL and BSW use cases where other cars are traveling at the same direction, we can calculate the max speed as the following



M is the max range of the WIFI module which is 300m at our module, at this figure we assume the worst case, we assume that the other car has entered the WIFI range just after our car has finished car listing we have to wait time T for the first frame and another T so that our car can know the location of the other car.

We need to determine the max relative speed between both cars so that

$$D2 \geq D\,min$$

D min is the minimum distance the car can brake and not hit the other car, which means

$$M = d11 + d12 + Dmin - d21 - d22$$

Assume the easy case that there is no acceleration from time 0 to time 2T, then

$$d11 = d12 = d1 \;\&\; d21 = d22 = d2$$

And also

$$d1 = V1 * T \;\&\; d2 = V2 * T$$

Where V1 and V2 are the speeds of our car and the other car

Then the equation will be

$$M = 2 * d1 + Dmin - 2 * d2$$

$$M = 2 * V1 * T + Dmin - 2 * V2 * T$$

$$V1 - V2 = \frac{M - Dmin}{2 * T}$$

Then the relative speed between the two cars when both are traveling at the same direction

$$V1 - V2$$

then

$$\boxed{V1{,}2_{max} = \frac{M - Dmin}{2T}}$$

this is the easy case when there is no acceleration, but assume we have acceleration since this is the realistic case, at this case we have V_0 , V_2 , Vo_0 , Vo_2 , a and ao

V_0 is the speed of our car at time 0.

V_2 is the speed of our car at time 2T.

Vo_0 is the speed of other car at time.

Vo_2 is the speed of other car at time 2T.

a is our car acceleration, ao is the other car acceleration.

from newton 2nd law of motion

$$X = \frac{a\ t^2}{2} + V_o\ t$$

The distances d11, d12, d21, d22

$$d1 = d12 + d11 = \frac{a\ \ 4T^2}{2} + V_0\ 2T$$

$$d2 = d22 + d21 = \frac{ao\ 4T^2}{2} + Vo_o\ 2T$$

and the equation will be

$$M = \frac{a\ \ 4T^2}{2} + V_0\ 2T + Dmin - \frac{ao\ 4T^2}{2} - Vo_o\ 2T$$

Multiply by 2

$$2M = a\ \ 4T^2 + (V_0 - Vo_o)\ 4T\ \ + 2Dmin - \ ao\ 4T^2$$

Based on this equation

$$a = \frac{2M - (V_0 - Vo_o)4\ T - 2Dmin + ao\ 4\ T^2}{4T^2}$$

From newton's 3rd low of motion

$$V_f = \sqrt[2]{V_i^2 + 2\ a\ d1}$$

The speed of our car will be

$$V_2 = \sqrt[2]{V_0^2 + 2\left(\frac{2M - (V_0 - Vo_o)4\,T - 2Dmin + ao\,4\,T^2}{4T^2}\right)\left(\frac{a\;4T^2}{2} + V_0\;2T\right)}$$

And the speed of the other car will be

$$Vo_2 = \sqrt[2]{Vo_0^2 + 2\;ao\;d}$$

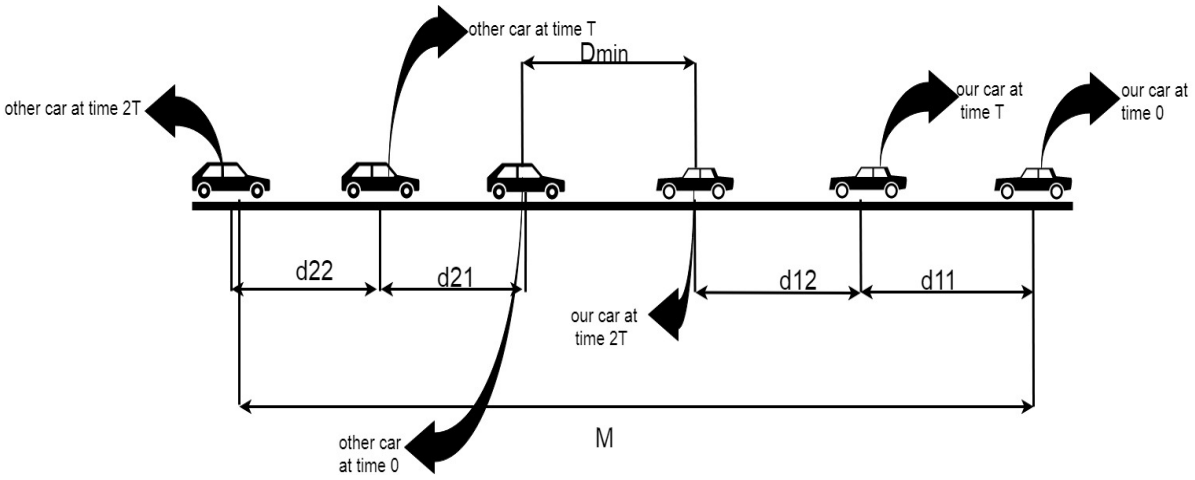
And the relative speed will be

$$V_2 - Vo_2 = \sqrt[2]{V_0^2 + 2\left(\frac{2M - (V_0 - Vo_o)4\,T - 2Dmin + ao\,4\,T^2}{4T^2}\right)\left(\frac{a\;4T^2}{2} + V_0\;2T\right)} - \sqrt[2]{Vo_0^2 + 2\;ao\;d}$$

$$\boxed{V_{2,o2} = \sqrt[2]{V_0^2 + 2\left(\frac{2M - (V_0 - Vo_o)4\,T - 2Dmin + ao\,4\,T^2}{4T^2}\right)\left(\frac{a\;4T^2}{2} + V_0\;2T\right)} - \sqrt[2]{Vo_0^2 + 2\;ao\;d}}$$

This expression is much more complicated.

at the DNPW use case when the cars are traveling at opposite directions, we can calculate the max speed using the following



Similar to the previous case the equation will be

$$M = d11 + d12 + Dmin + d21 + d22$$

Assume the easy case that there is no acceleration from time 0 to time 2T, then

$$d11 = d12 = d1 \;\&\; d21 = d22 = d2$$

And also

$$d1 = V1 * T \;\&\; d2 = V2 * T$$

Where V1 and V2 are the speeds of our car and the other car

Then the equation will be

$$M = 2 * d1 + Dmin + 2 * d2$$

$$M = 2 * V1 * T + Dmin + 2 * V2 * T$$

$$V1 + V2 = \frac{M - Dmin}{2 * T}$$

Then the relative speed between the two cars when both are traveling at the opposite direction is

$$V1 + V2$$

Then

$$\boxed{V1{,}2_{max} = \frac{M - Dmin}{2T}}$$

Notice that we got exactly the same expression, the difference is the expression of the relative speed velocity, we can deduce directly the expression at the complicated case

$$\boxed{V_{2,o2} = \sqrt[2]{V_0^2 + 2\left(\frac{2M - (V_0 + Vo_o)4\,T - 2Dmin - ao\,4\,T^2}{4T^2}\right)\left(\frac{a\ 4T^2}{2} + V_0\,2T\right)} + \sqrt[2]{Vo_0^2 + 2\,ao\,d}}$$

let's take an example using the simple case

M=300, Dmin=15m and T=12 second, the relative speed will be

$$V1{,}2_{max} = \frac{M - Dmin}{2T} = \frac{300 - 15}{12} = 23.75\ m/s$$

$$= 85.5\ km/h$$

Let the other car speed be 30 km/h then at the EEBL and BSW cases the max speed of our car is

85.5 + 30 = 115.5 km/h

And at the DNPW case the max speed will be

85.5 – 30 = 55.5 km/h which make sense.

CHAPTER 7: PRINTED CIRCUIT BOARD

7.1 PCB specifications

We began our hardware implementation using wires and breadboards but as the number of modules interfaced with controller increased, we suffered from some problems. The number of wires was very large and although we did our best to keep connections neat, it was very difficult to detect connection problems and track wires. Besides, the area of connection increased, we had to use many breadboards, and this wasn't applicable because they wouldn't fit on the car. Connections weren't stable enough with the car movement. So we decided to use PCB design to solve the problems as follows:

- Compact size

A typical PCB contains a large number of electronic components, most of which are very small in size. Without a PCB, it would be nearly impossible to connect such components together with wires. A PCB provides a convenient platform to arrange the electronic components in a compact and efficient way. This compactness allows development of large and complicated electronic circuits in small form factors, taking less space in devices.

- Immunity to Movement

Since components on a PCB are held fixed to the board by solder flux, they do not move, irrespective of the movement of the board. This enables the electronic circuit to be placed in devices that are moving or shaking without worrying about the possibility of component displacement and subsequent electronic short circuits.

- Ease in Diagnostics and Repair

PCBs are helpful in performing diagnostics for a number of reasons. The components and their polarities on a well-designed PCB are clearly labeled on the board, which is convenient for installation as well as repair. For diagnostics, one often needs to trace signal paths, which would be very difficult to perform if the traces were not exposed and well organized.

- Low Electronic Noise

When properly laid out, a PCB minimizes electronics noise that could significantly degrade performance. The electrical components on a PCB are organized in such a way that the path lengths of the electrical current between them are minimized, leading to low radiation and pickup of electromagnetic waves. This ensures lower cross-talk between components and between different traces, which is a major concern in electronic circuits.

To start our PCB design, we had a variety of tools to choose from, we decided on Altium designer as it is more professional and flexible to use.

7.2 Altium Designer

Altium Designer is a PCB and electronic design automation software package for printed circuit boards. Altium Designer's suite encompasses four main functional areas: schematic capture, 3D PCB design, Field-programmable gate array (FPGA) development and release/data management. Altium makes designing complex, high-quality projects easier, faster and more accurate from concept to production.

The strength of Altium's design tools begins with their foundation, a solid platform built on a unified design environment. Another core strength of Altium that it provides the most useful library and up-to-date user interface for design workflow. Both interactive and automatic, routing tools help to finish routing in less time and with better results.

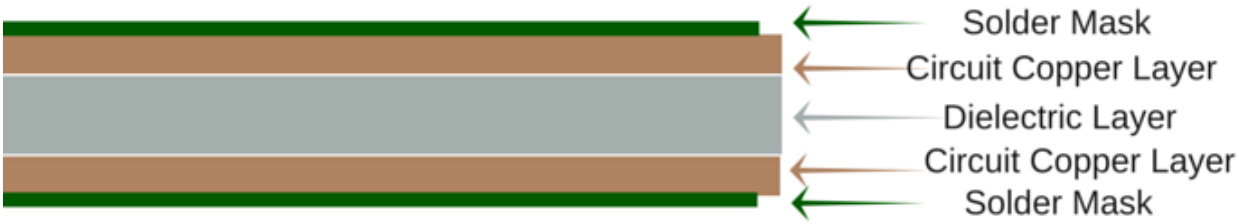
It gives complete error checking and design verification. The best rules driven design, as well as the ability to verify all aspects of your design including your circuit board's power delivery add speed to the workflow.

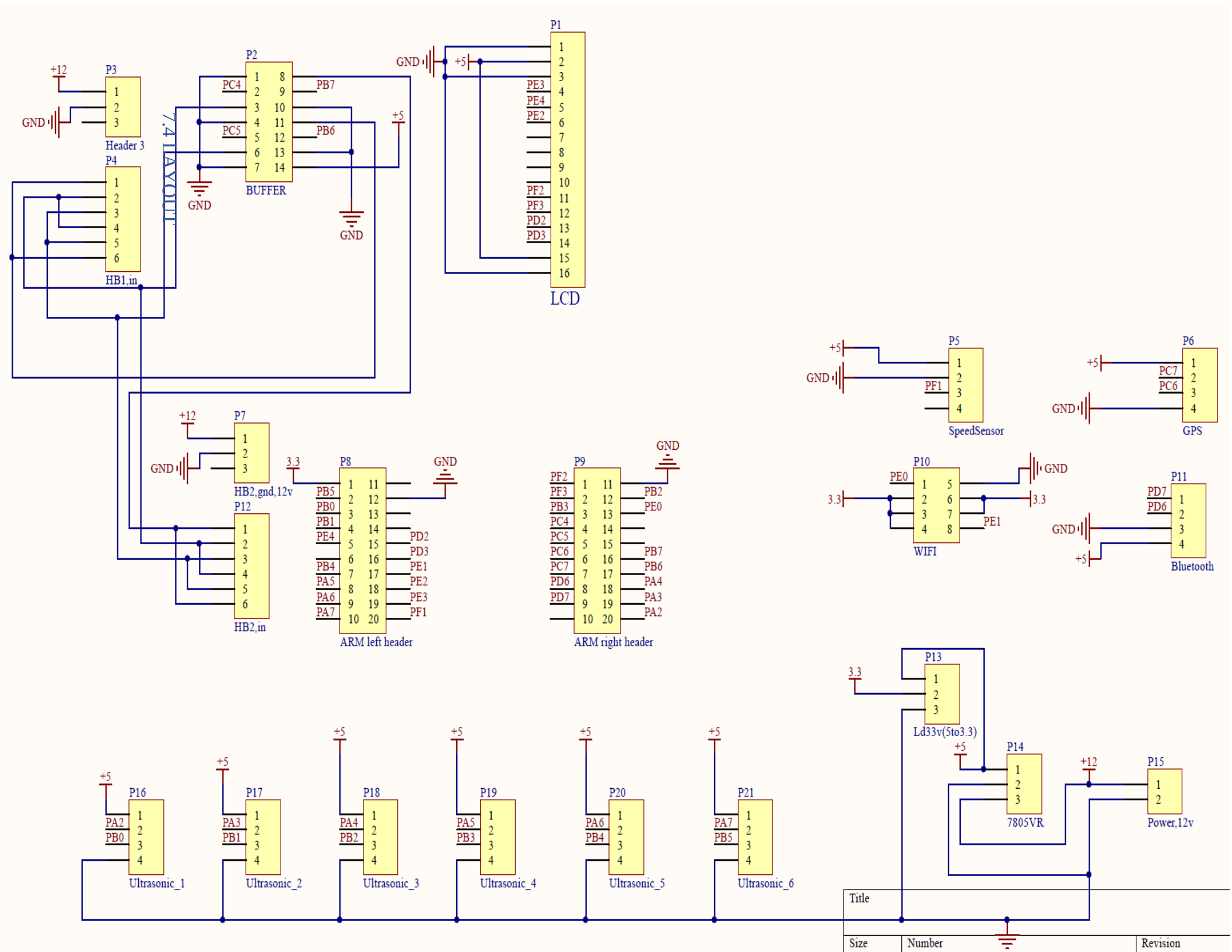
7.3 Schematic

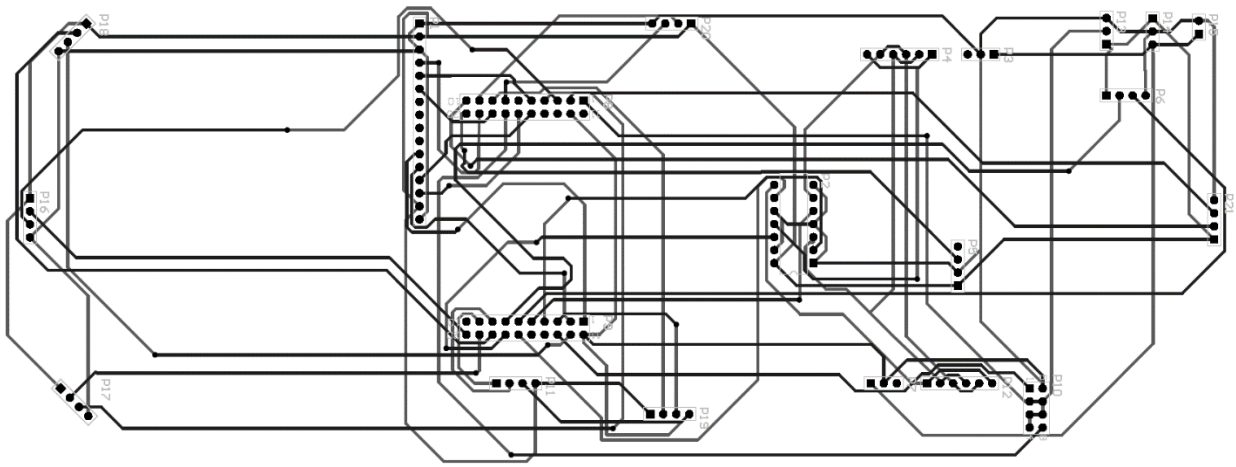
We started out design by schematic implementation and checked all connections between the modules and controller. Then, we it was time for layout, we started by one-layer PCB as it is easier to implement, but as we are limited with car's dimensions, we couldn't achieve optimum routing on only one layer.

So, we decided to use double layer PCB, such that tracks are well arranged and we get optimum routing in the car's area. We began with auto-routing then we applied some modifications manually for best fit design.

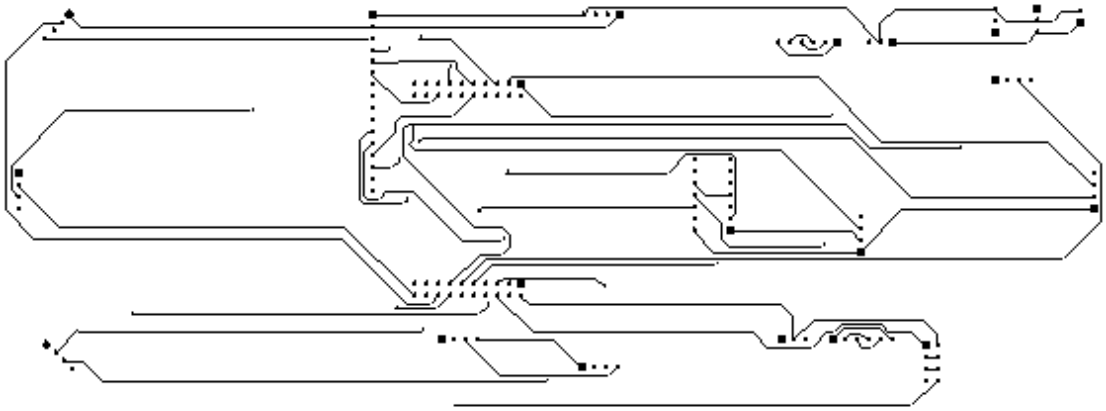
Double Sided PCBs allow closer routing traces by alternating between a top and bottom layer using vias. Double Sided PCBs allow a higher density of component. This is because you have a whole extra layer where you can add tracks to connect the components, freeing space on the other to place the components closer together. Besides, it reduces the board size.



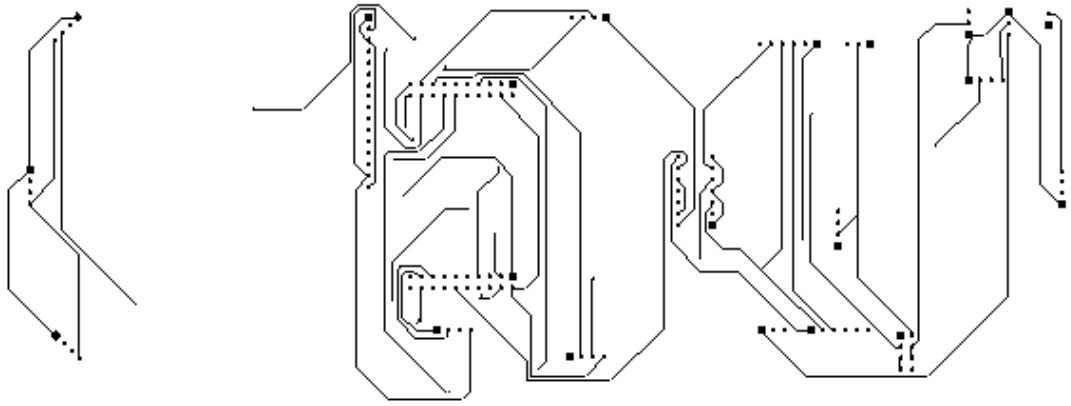

P1
LCD
P2
BUFFER
P3
Header 3
P4
HB1,in
P5
SpeedSensor
P6
GPS
P7
HB2,gnd,12v
P8
ARM left header
P9
ARM right header
P10
WIFI
P11
Bluetooth
P12
HB2,in
P13
Ld33v(5to3.3)
P14
7805VR
P15
Power,12v
P16
Ultrasonic_1
P17
Ultrasonic_2
P18
Ultrasonic_3
P19
Ultrasonic_4
P20
Ultrasonic_5
P21
Ultrasonic_6
74LAYOUT
GND
+5
+12
3.3
Title
Size
Number
Revision



7.4.1 Top layer



7.4.2 Bottom layer



CHAPTER 8: OVARALL PERFORMANCE OPTIMIZATION

After the development of a full prototype we made studies around 2 important points that may affect the overall performance of the systems. In this chapter we are going to discuss these points.

8.1 CPU Load and utilization

We get the idea of this optimization after we tested our RTOS schedule, we asked some questions,

- Are we wasting time in idle task instead of running useful tasks on CPU?

To answer the first question, we need to find our CPU utilization factor which is defined as the sum of work handled by a Central Processing Unit.
In optimum case we find CPU utilization factor of 100% which means the we never enter idle task and all the time is consumed in useful tasks.

U = 100% - (Percentage of time that is spent in idle task)

But having 100% utilization doesn't mean our CPU is in a good situation, we might have utilized CPU time but missed some tasks in ready state for a while or forever! So we asked the second quuestion.

- Are we overloading the CPU with tasks that may lead to task blocking or deadlock for some of them?

To answer the second question, we need to be aware of CPU loading by performing some CPU load calculations which we can define by the maximum number of tasks waiting in ready queue where another task is running on CPU.

In this step, to be able to detect the problem of CPU over loading we started to calculate each Task execution time and we found the following results

Task	Execution time
GPS task	1 s
Ultrasonic task	120 ms
Bluetooth reading task	10 ms
Speed sensor interrupt handler	3 ms
UART7 interrupt handler	0.9
WIFI_Frame_Send(actually sending to one car)	5.5 s
WIFI_Frame_Send(actually sending to two car)	8.5 s
WIFI_Frame_RCEV(actually receiving)	1 ms

From this analysis we found that WIFI execution time is disastrous, it will be blocking the system for 5.5 seconds while sending for a single car. So, in order to optimize it we divided WIFI send task into to Functions and we get the following results:

TASk	Time
ESP_SEND_THIS_CAR_DATA__CARS_LISTING	2.5 s
ESP_SEND_THIS_CAR_DATA___SENDING_TO_SINGLE_CAR	3.2 s

We get a better CPU load than previous analysis as the system would be blocked for 3.2 seconds only but it still very bad CPU load so we divided the task more and more

TASk	Time
ESP_SEND_THIS_CAR_DATA__CARS_LISTING__Command_Sendi ng	2ms
Time between	2 s
ESP_SEND_THIS_CAR_DATA__CARS_LISTING__Response_recv	400ms
Time between	---(3s)
ESP_SEND_THIS_CAR_DATA___CONNECTING_TO_SINGLE_CA R__Comman d_sending	10ms
Time between	2.8 s
ESP_SEND_THIS_CAR_DATA___CONNECTING_TO_SINGLE_CA R__Response _Recv	200ms
Time between	(0)
ESP_SEND_THIS_CAR_DATA___SENDING_TO_SINGLE_CAR	150ms

Here we got awesome results, WIFI task is not a problem anymore!
Here we found that the systems is blocked for 1 second on average waiting for GPS task so we made the 2nd optimization aiming to reduce CPU load.
In this step we optimized GPS performance by increasing baud rate to 115200 instead of 9600 and increased data update rate from 1 HZ to 15 Hz.
Here is our final Tasks execution time of all tasks where we could notice that there are no tasks blocking the system.

Task	Execution time
GPS task	100ms
Ultrasonic task	120ms
Bluetooth reading task	10ms
Speed sensor interrupt handler	3ms
UART7 interrupt handler	0.9s
WIFI_Frame_RCEV(actually receiving)	1 ms
ESP_SEND_THIS_CAR_DATA__CARS_LISTING__Command_Sending	2ms
ESP_SEND_THIS_CAR_DATA__CARS_LISTING__Response_recv	400ms
ESP_SEND_THIS_CAR_DATA___CONNECTING_TO_SINGLE_CAR_ _Comman d_sending	10ms
ESP_SEND_THIS_CAR_DATA___CONNECTING_TO_SINGLE_CAR_ _Response _Recv	200ms
ESP_SEND_THIS_CAR_DATA___SENDING_TO_SINGLE_CAR	150ms

Currently we got better CPU load as we reduced average execution time of all tasks.

Also, we need to assure 100% CPU utilization so in order to prevent the system from entering idle task we changed it from periodic mode to a background task with lower priority so that it always running and when any other task is ready it takes the CPU.

8.2 Power dissipation analysis

In order to be able to develop a stable prototype we need to be aware of each module power consumption to achieve 2 goals

1. Circuit protection from high current spikes.
2. Determine needed batteries and maximum available usage time.

To achieve these goals, we measured the current consumption of each module

Module	Current in Ampere
Motors initial step	1.2~1.4
Motors in movement	0.34~0.4
WIFI	0.1
GPS	0.1
Ultrasonic	0.1
Average current per car	0.6 ~0.7

Here we can notice the motors requests very high current at initial step compared to other modules which may cause damage to them when this current flows in the circuit.

So, we used separate batteries for motors and other PCB components.
Motors batteries needs high voltage/current so we used 3 series 3.7V batteries, where each battery is composed of 2 batteries in parallel to increase current support ability and to last for longer time.

For other modules we used 2 series 3.7V batteries where each battery is composed of 2 batteries in parallel to increase current support ability and to last for longer time.

8.3 Speed measurement enhancement

While testing the prototype we found that the reading of speed sensor not very accurate so in order to improve it's accuracy we developed a mathematical model that maps car movement as a function of control speed from 2 to 11 and found the following results

Control speed	Actual speed
0-5	0
5	13.33 cm/sec
10	43.3 cm/sec
11	43.4 cm/sec

From this analysis we can draw a linear relation between control speed and actual speed

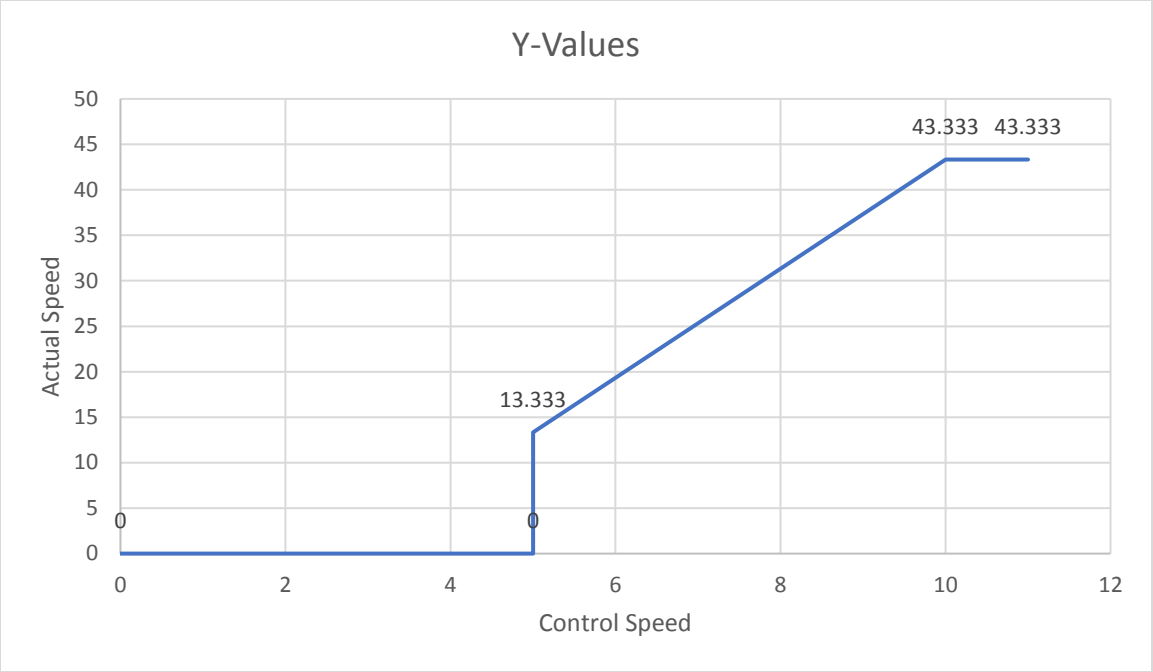


Figure 59 Speed Graph

Slope of the line in the interval of 5 to 10
Speed factor = slope = (43.3-13.3)/(10-5) =6
Then we get valid speed equation in the interval 5 to 10

Actual speed = 6* Control speed -16.7

Where -16.7 is the intersection of the equation with Actual speed axis

Using this equation to produce car speed gives more accurate results than using speed sensor.

8.4 Microcontroller Frequency

In order to get better response time of overall system and decreasing processing time of each task we increased the Frequency of the Microcontroller from 16MHz to 80 MHz by using internal PLL module that is used to lock the frequency on different values.

Appendix

Modules Driver Functions

Clock configuration	
void PLL_Init(void);	configure the system to get its clock from the PLL and set clock frequency to 80 MHz.
NEO-6m GPS module	
void get_Buffer();	Frame segmentation and extraction of Longitude and latitude.
const char * get_location(uint8 index);	Get comma locations in GPS frame.
double Distance(double lat1, **double** lon1, **double** lat2, **double** lon2, **char** unit);	Calculate distance given GPS coordinates, return distance in meters or km.
double deg2rad(double);	Conversion from degree to radians.
double rad2deg(double);	Conversion from radians to degree.
void UART3_INIT();	Initialize UART 3 with baud rate 115200 for communication with GPS
uint8 **UART3_recieveByte(void);**	Return received character form UART buffer to get GPS frame.
Ultrasonic module	
void Configure_Trigger(void);	Configure pins connected to ultrasonic trigger as digital output pins.
void Configure_Echo(void);	Configure pins connected to ultrasonic echo as digital input and interrupt pins.
void Captureinit(void);	Initialize Timer 2 to measure the time during which echo pin is high.
void Enable_Ultrasionc_Interrupts(void);	Enable interrupts for port B pins, connected to ultrasonic echo.
void Disable_Ultrasionc_Interrupts(void);	Disable interrupts for port B pins, connected to ultrasonic echo.
void Send_Trigger (unsigned short int);	Set Trigger pin high for 10µs then low.
void inputInt(void);	ISR for detecting edge transitions on ultrasonic Echo.
LCD	

void Void_LCD_INIT (**void**);	Send initialization sequence for LCD, clear screen, set cursor at first row and column.
void Void_LCD_CMD (u8 cmd);	Send command to LCD, when RS pin is low and command resister is selected.
void Void_Clear_Screen ();	Clear LCD screen, using (0x01) command.
void Void_LCD_DATA (s8 data);	Send data to LCD, when RS pin is high and data resister is selected.
void Void_LCD_Location (u8 x,u8 y);	Set cursor position to specific row and column
void Void_LCD_Print(s8* str);	Print string on LCD
void Void_LCD_NUM (s32 num);	Print signed integer
u32 **U32_POW**(u8 U8_NUM,u8 U8_PWR);	Calculate power
void Void_Send_Init_SEQ(u8 U8_INIT);	Send initialization sequence
void Void_Wait_Busy_Flag ();	Wait till processing ends after sending command or data.
Speed sensor	
void PortFIntHandler(**void**);	ISR for detecting rising edge on PF1 pin connected to speed sensor digital output.
void Configure_Speed_pin(**void**);	Configure pin connected to speed sensor as digital input and interrupt pin.
void Enable_Speed_Interrupt(**void**);	Enable Port F interrupts.
void Disable_Speed_Interrupt(**void**);	Disable port F interrupts.
void TimerA1_Handler(**void**);	ISR to detect when timer finishes the required duration for measuring speed.
void Timer1A_Init(**unsigned long** period);	Configure timer 1A and initialize it with the required period for speed measurement.
Motors and PWM	
void Motors_init(**void**);	Set motor control pins connected to H-bridge as digital output pins.
void Direction(**char** input_char);	Control motor's movement, forward, backward and stop.
void Timer0B_PWM_Init(u32 period, u32 high); **void Timer0A_PWM_Init**(u32 period, u32 high);	Configure timer in PWM mode and set duty cycle.
void Timer0B_PWM_Duty(u32 high); **void Timer0A_PWM_Duty**(u32 high);	Change duty cycle to control motor's speed.
Bluetooth	

void uart2_init(void);	Configure UART2 with baud rate 9600 for communication with Bluetooth module.
void uart2_interrupt_enable(void);	Enable UART2 interrupt when 1/8 of the buffer is full.
void uart2_interrupt_disable(void);	Disable UART2 interrupts.
void UART2_Handler(void);	ISR entered upon receiving characters from Bluetooth module.

ESP8266 WIFI Module

char* ESP_Send_AT_Command_and_Receive_Response(**char*** ATCommand);

bool ESP_init(**const char*** IP_IN_SAP_MODE,**const char*** IP_IN_STA_MODE,**const char*** THIS_SAP_SERVER_PORT_NUMBER);

bool ESP_SAP_AND_SERVER_Configuration(**const char*** SSID,**const char*** PWD);

bool ESP_STA_Configuration(**void**);

void ESP_List_Nearby_APs__Command_sending(**void**);

bool ESP_List_Nearby_APs__APs_recv(**void**);

void APs_CLEAR(**void**);

void ESP_Join_AP__Command_sending(uint8 AP_INDIX,**char*** PWD);

bool ESP_Join_AP__Response_Recv();

bool ESP_Quit_AP(**void**);

bool ESP_START_CONNECTION_AT_STA_MODE(**char*** DEST_AP_PORT_NUMBER,**char*** DEST_IP,**char*** CHANNEL_ID);

bool ESP_SEND_DATA_AT_STA_Mode(**char*** DATA,**char*** CHANNEL_ID,**char*** DATA_LEN);

bool ESP_END_CONNECTION_AT_STA_MODE(**char*** CHANNEL_ID);

bool ESP_RECEIVE_DATA_AT_SERVER_MODE(**void**);

void ESP_RECEIVE_BUFFER_CLEAR(**void**);

bool ESP_GET_OTHER_CARS_DATA(**void**);

void ESP_FDATA_CLEAR(**void**);

bool ESP_SEND_THIS_CAR_DATA__CARS_LISTING__Command_Sending(**void**);

sint8 **ESP_SEND_THIS_CAR_DATA__CARS_LISTING__Response_recv**(**void**);
bool ESP_SEND_THIS_CAR_DATA___CONNECTING_TO_SINGLE_CAR__Command_sending(uint8 CAR_INDEX);

bool ESP_SEND_THIS_CAR_DATA___CONNECTING_TO_SINGLE_CAR__Response_Recv();

```
bool ESP_SEND_THIS_CAR_DATA___SENDING_TO_SINGLE_CAR(uint8 CAR_INDEX,bool LAST);
```

Prototype cost

Component	Cost
ARM® Cortex®-M4F Based MCU TM4C123G Launchpad	600
Raspberry pi controller	1550
RP Camera	290
NEO-6m GPS module	550
ESP8266 WIFI module	90
6 Ultrasonic modules	240
LCD 20 x 4 Character	100
Car body	225
4 Motors	40
2 H-bridge	60
Speed sensor	75
Bluetooth	135
Total cost	3955

References

WIFI

- ESP8266EX, wifi module datasheet (2018) chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
- ESP8266 AT Instruction Set (2018) chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://ieee-sensors.org/wp-content/uploads/2018/05/4a-esp8266_at_instruction_set_en.pdf
- Texas Instruments support (2019) https://e2e.ti.com/support/microcontrollers/other/f/908/p/798412/2954736#2954736

GPS

- Global Positioning System (2019) https://en.wikipedia.org/wiki/Global_Positioning_System
- World Geodetic System (2019) https://en.wikipedia.org/wiki/World_Geodetic_System
- NEO-6 seriesVersatile u-blox 6 GPS modules (2019) https://www.u-blox.com/sites/default/files/products/documents/NEO-6_ProductSummary_%28GPS.G6-HW-09003%29.pdf
- u-blox 6 GPS ModulesData Sheet (2019) https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf
- u-centerGNSS evaluation software Product summary (2019) https://www.u-blox.com/sites/default/files/u-center_ProductSummary_%28UBX-13003929%29.pdf
- u-centerGNSS evaluation software for Windows, User Guide (2019) https://www.u-blox.com/sites/default/files/u-center_UserGuide_%28UBX-13005250%29.pdf
- Stanly, How to change U-Blox GPS default baud rate and data rate (2015) https://freematics.com/forum/viewtopic.php?f=11&t=1759&sid=3e9b9a9f158afa2a0fce64f7b82de273
- Handheld Group, Enable/Disable NMEA dataset's (2018) https://www.handheldgroup.com/support-rugged-computers/knowledgebase-KB/24331/?fbclid=IwAR0nDd0E_SjwHyXis39oVSTOETtf4w1GsNydBEmU4nsE_m9lzhCuUYLP6-8
- Distance Between Coordinates Theory https://www.movable-type.co.uk/scripts/latlong.html
- Distance Between Coordinates Calculator (2019) https://gps-coordinates.org/distance-between-coordinates.php?fbclid=IwAR1jl8zPtho8G6MrTzghCOecO859HTLVc-MCzoDI4KLngQ98uEIwGwprP2o

Ultrasonic

- Science Direct, Ultrasonic sensors (2019), https://www.sciencedirect.com/topics/engineering/ultrasonic-sensors
- Electronic Wings, Sensors & Modules, Ultrasonic Module HC-SR04 (2019), https://www.electronicwings.com/sensors-modules/ultrasonic-module-hc-sr04
- Introductory guide to sensors, Ultrasonic sensors (2019), https://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/
- PC Services, Circuit Diagram Ultrasonic Distance Sensor HC-SR04 (2018), http://www.pcserviceselectronics.co.uk/arduino/Ultrasonic/electronics.php#faults
- Luis Rafael, Google Sites, GPIO timers (2014), https://sites.google.com/site/luiselectronicprojects/tutorials/tiva-tutorials/tiva-general-purpose-timers/timer-periodic-mode---srf04
- Cytron Technologies, "Product User's Manual – HCSR04 Ultrasonic Sensor" (2013), http://raspoid.com/download/datasheet/HCSR04

Bluetooth

- Scientific American, How does Bluetooth work? (2019), https://www.scientificamerican.com/article/experts-how-does-bluetooth-work/
- Components 101, HC-06 Bluetooth module (2019), https://components101.com/wireless/hc-06-bluetooth-module-pinout-datasheet
- e-Gizmo Mechatronix Central, Bluetooth Modules, "Hardware Manual & AT Commands Reference Manual Rev. 1r0" (2019), https://cdn.instructables.com/ORIG/FQ1/CUVZ/HXA9PUVQ/FQ1CUVZHXA9PUVQ.pdf
- Module143, HC06 and HC05 (Bluetooth Module) (2019), http://invent.module143.com/hc06-and-hc05-bluetooth-module-how-to-use-it/
- Instructables Circuits, "HC-03/05 Embedded Bluetooth Serial Communication Module AT command set" (2014) , https://cdn.instructables.com/ORIG/FKY/Z0UT/HX7OYY7I/FKYZ0UTHX7OYY7I.pdf

Motors and H-bridge

- Electrical4u, Operating principle of DC motor(2019), https://www.electrical4u.com/working-or-operating-principle-of-dc-motor/
- Roland Pelayo, Microcontroller tutorials, How to Use L298N Motor Driver (2019), https://www.teachmemicro.com/use-l298n-motor-driver/
- All about circuits, Pulse width modulation (2019), https://www.allaboutcircuits.com/textbook/semiconductors/chpt-11/pulse-width-modulation/
- STMicroelectronics, "DUAL FULL-BRIDGE DRIVER-L298"(2019), https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf
- Fairchild Semiconductor,"74125 Quad 3-STATE Buffer" (2019), http://www.datasheetcatalog.com/datasheets_pdf/7/4/1/2/74125.shtml

Altium-PCB

- PCB way, Advantages of double sided PCB (2019), https://www.pcbway.com/blog/Engineering_Technical/Advantages_of_Double_Sided_Pcb.html
- Amitron, Double sided PCB (2019), https://www.amitroncorp.com/printed-circuit-boards/double-sided.html
- Robert Ferance, Fedvel academy (2011),

https://www.fedevel.com/welldoneblog/2011/12/5x-why-to-choose-altium-designer-for-electronic-hardware-design/

LCD

- Lumisense Technologies, Chennai,"20*4 LCD" (2019), http://www.lumisense.in/pdfs/20-4%20LCD.pdf
- Elprocus, LCD Interfacing (2019), https://www.elprocus.com/lcd-interfacing-with-8051-microcontroller/
- Winstar, 20x4 Character LCD Display (2018), https://www.winstar.com.tw/products/character-lcd-display-module/20x4-lcd-display.html
- VISHAY INTERTECHNOLOGY," LCD-020N004L" (2017), https://www.vishay.com/docs/37314/lcd020n004l.pdf
- Hitachi," Systronix 20x4 LCD" (2000), http://www.systronix.com/access/Systronix_20x4_lcd_brief_data.pdf

Speed sensor

- Electronics Hub, Interfacing LM393 Speed Sensor (2019), https://www.electronicshub.org/interfacing-lm393-speed-sensor-with-arduino/

FreeRTOS

- API Reference (2019) https://www.freertos.org/a00106.html

AUTOSAR

- Autosar Layered Architecture https://www.embitel.com/tag/autosar-layered-architecture

Machine learning

- Simplilearn, Convolutional neural networks (2019), https://www.simplilearn.com/convolutional-neural-networks-tutorial
- Edureka, Developing An Image Classifier In Python Using TensorFlow (2019), https://www.edureka.co/blog/convolutional-neural-network/
- Ehab Essa, Neural network lectures (2019), https://drive.google.com/drive/folders/1Dv3mzafx9l3Ogwyi1y50vsFMVZCvMfJj?fbclid=IwAR08Cq1DPrn__cB5Y19NDSoMQrK9k7uHElDrpZ7xoLfYQLNVGvv5yT4m3n4
- CS231n Convolutional Neural Networks for Visual Recognition (2019), http://cs231n.github.io/convolutional-networks/#fc
- Keras, The python deep learning library(2019),
- https://keras.io/?fbclid=IwAR14aZ5mkbJZ-cRHA5PrbQvmK4nLRFNI6MO3GnK1GHfYM_RApziFsaaSrIw
- Mahmoud Badry, Convolutional neural networks (2019), https://drive.google.com/drive/folders/19VGXK1DGXjAscY8Og4LvwxY2FDD4IjF8?fbclid=IwAR1wh2X-CiFw8R6AX4FP3TMALMqmDTPVT3kuY8eD4ay_augbVr4VxHM96JU
- Sabyasachi Sahoo, Towards data science, Residual blocks — Building blocks of ResNet (2018), https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
- Anish Singh, Towards Data Science, Activation functions (2017), https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

Raspberry

- Raspberry library,"keras" , (2018), https://medium.com/@abhizcc/installing-latest-tensor-flow-and-keras-on-raspberry-pi-aac7dbf95f2
- Raspberry specifications (2017), https://mail.google.com/mail/u/0/#search/nor/FMfcgxwBVqRJxqQDRqghZNtJrTrZzXGd?projector=1&messagePartId=0.1
- Raspberry library,"numpy".(2018) https://www.raspberrypi.org/forums/viewtopic.php?t=207058
- Raspberry library,"pandas".(2018) https://raspberrypi.stackexchange.com/questions/17073/how-do-i-install-pandas-on-raspberry-pi
- Raspberry libraries,(2017) https://www.raspberrypi.org/forums/viewtopic.php?t=66740
- Raspberry Pi 3 Model B with Camera Module v2,(2019) https://www.element14.com/community/docs/DOC-83171/l/raspberry-pi-3-model-b-with-camera-module-v2