Masters Thesis

# A Robotic Platooning Testbed for Cooperative ITS Components

Orientação científica: Ricardo Severino

**Nuno Guedes**

CISTER-TR-191208

# A Robotic Platooning Testbed for Cooperative ITS Components

Nuno Guedes

CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

https://www.cister-labs.pt

## Abstract

Intelligent Transportation Systems are becoming increasingly relevant in the currentand future social and mobility aspects, since they apply information, communication,and sensor technologies to vehicles and transportation infrastructure. They providereal-time information for road users and transportation system operators enablingbetter and more informed and efficient decisions. This technology can be used tomanage road traffic in order to reduce congestion, increase the efficiency of existingtransport infrastructure and improve mobility.Although this technology might be the powerhouse of cooperative autonomousdriving, as others matters, there are still safety concerns to be managed. Thus, itis fundamental to include safety mechanism to assure the required safety level forthese systems.Currently, research in cooperative autonomous systems usually conducted oversimulation frameworks as real experiments are still too costly.A good solution for this problem is to rely on robotic platforms since they arecheaper and replicate with similar functionality real vehicles. In this line, this Thesisfocuses on developing a platooning robotic testbed platform with a 1/10 scale roboticvehicles.To prove it's efectiveness, we validate a cooperative safety mechanism for platooning.

ISEP

Instituto Superior de Engenharia do Porto

# A Robotic Platooning Testbed for Cooperative ITS Components

Master Thesis

To obtain the degree of master at the
Instituto Superior de Engenharia do Porto,
public defend on October by

Nuno Miguel Santos Guedes

Degree in Electronics and Computer Science
Porto, Portugal.

Supervisor:

Prof. Dr. Ricardo Severino

ISBN ++-++++-+++-+

Author email: `1140477@isep.ipp.pt`

# Acknowledgments

First, I would like to thank my supervisor, Ricardo Severino, for this opportunity at CISTER. His advice, supervision and availability were remarkable during the development of this Thesis.

I want to thank all the people that work at CISTER for their support and passion. These attributes show the exciting and challenging workplace that CISTER is.

To Daniel Almeida and Bruno Vieira, I would like to thank them for these five years at ISEP and the last, also in CISTER. They were always their for any problem that would appear, despite having their owns.

I would also like to thank my parents. Without them, I wouldn't be the person I am today.

A special thanks to my dearest, Francisca Santos, who believes most than anyone in me and whose love and support was crucial throughout this master's degree

Last, I would like to leave a dedication to my grandfather, Joaquim Santos, that went on to be my inspiration and who was never able to see me succeed.

# Resumo

Os Sistemas Inteligentes de Transporte estão a tornar-se cada vez mais relevantes nos contextos sociais e de mobilidade atuais e futuros, pois aplicam tecnologias de informação, comunicação e sensores em veículos e infraestrutura de transporte. Estes sistemas fornecem informações em tempo real para condutores e operadores de sistemas de transporte, permitindo decisões melhores, mais informadas e eficientes. Esta tecnologia pode ser usada para controlar o tráfego rodoviário, a fim de reduzir o congestionamento, aumentar a eficiência da infraestrutura de transporte e melhorar a mobilidade.

Embora esta tecnologia possa ser a força motriz da condução autónoma cooperativa, ainda existem problemas de segurança a serem resolvidos. Portanto, é fundamental incluir mecanismos de segurança para garantir o nível de segurança exigido para estes sistemas.

Atualmente, a investigação em sistemas autónomos cooperativos é geralmente realizada em ambiente de simulação, devido ao facto de experiências reais serem ainda muito caras.

Uma boa solução para esse problema é depender de plataformas robóticas, uma vez que são mais baratas e replicam veículos reais com funcionalidade semelhante. Nesta linha, esta Tese concentra-se no desenvolvimento de uma plataforma robótica para "platooning" com veículos robóticos à escala de 1/10.

Para provar sua eficácia, validamos um mecanismo de segurança cooperativo para "platooning".

**Palavras-Chave:** "Platooning" Cooperativo, Plataforma Robótica, Sistemas Inteligentes de Transporte, Mecanismos de Segurança.

# Abstract

Intelligent Transportation Systems are becoming increasingly relevant in the current and future social and mobility aspects, since they apply information, communication, and sensor technologies to vehicles and transportation infrastructure. They provide real-time information for road users and transportation system operators enabling better and more informed and efficient decisions. This technology can be used to manage road traffic in order to reduce congestion, increase the efficiency of existing transport infrastructure and improve mobility.

Although this technology might be the powerhouse of cooperative autonomous driving, as others matters, there are still safety concerns to be managed. Thus, it is fundamental to include safety mechanism to assure the required safety level for these systems.

Currently, research in cooperative autonomous systems usually conducted over simulation frameworks as real experiments are still too costly.

A good solution for this problem is to rely on robotic platforms since they are cheaper and replicate with similar functionality real vehicles. In this line, this Thesis focuses on developing a platooning robotic testbed platform with a 1/10 scale robotic vehicles.

To prove it's effectiveness, we validate a cooperative safety mechanism for platooning.

**Keywords:** Cooperative Platooning, Robotic Testbed, Intelligent Transportation Systems, Safety Mechanisms.

# Contents

# List of Figures

# Acronyms

| | |
|---|---|
| **3D** | three-dimensional |
| **ANN** | Artificial Neural Network |
| **BSM** | Basic Safety Message |
| **CAM** | Cooperative Awareness Message |
| **CCH** | Control Channel |
| **CISTER** | Research Centre in Real-Time & Embedded Computing Systems |
| **CoCPS** | Cooperative Cyber-Physical Systems |
| **DCC** | Decentralized Congestion Control |
| **ESC** | Electronic Speed Controller |
| **ETSI** | European Telecommunications Standards Institute |
| **GNSS** | Global Navigation Satellite System |
| **GUI** | Graphical User Interface |
| **I2C** | Inter-Integrated Circuit |
| **IDE** | Integrated Development Environment |
| **IEEE** | Institute of Electrical and Electronics Engineers |
| **IMU** | Inertial Measurement Uni |
| **IR** | Infrared emitting diode |
| **ITS** | Intelligent Transport System |
| **LiDAR** | Light Detection And Ranging |
| **LOD** | Loss of Direction |
| **OBU** | On-board Unit |
| **OpenCV** | Open Source Computer Vision Library |
| **OSRF** | Open Source Robotics Foundation |
| **PCB** | Printed Circuit Board |
| **PD** | Proportional-Derivative |

| | |
|---|---|
| **PID** | Proportional-Integral-Derivative |
| **PWM** | Pulse Width Modulation |
| **ROS** | Robot Operating System |
| **RSU** | Road-Side Unit |
| **SCH** | Service Channel |
| **SDK** | Software Development Kit |
| **SLAM** | Simultaneous Localization and Mapping |
| **SPI** | Serial Peripheral Interface |
| **UI** | User Interface |
| **uRLLC** | ultra reliable and low-latency communication |
| **USB** | Universal Serial Bus |
| **V2I** | Vehicle-to-Infrastructure |
| **V2V** | Vehicle-to-Vehicle |
| **V2X** | Vehicle-to-Everything |
| **VANET** | Vehicle ad hoc Network |
| **VC-bot** | vehicular cloud robot |
| **VNC** | Virtual Network Computing |
| **WAVE** | Wireless Access in Vehicular Environments |

# 1 Introduction

## 1.1 Motivation

Platooning is a very interesting subject that is in expansion. This approach brings many benefits to society as lower fuel consumption and CO2 emissions, improved safety, since we eliminate human factor that is estimated to be the cause of 90% of accidents, and it is more efficient has the potential to deliver goods faster and reduce traffic jams. When we introduce wireless communication in a platoon, we enable what is called cooperative platooning. In such setup, each following vehicle uses information from its own in-vehicle sensors, plus data received via wireless from the front vehicle, to cooperatively measure and adjust its position, based on the speed, direction and acceleration of the preceding vehicle.

This brings us to the various challenges platooning also presents concerning the reliability of the communications between vehicles and it's impact on road safety. Due to these facts, there is the need to implement monitoring tools or mechanisms in cooperative platooning, to prevent accidents and/or to assure the correct trajectory to a certain target. The Control Loss Warning (CLW) mechanism analyzes real time data of exchanged messages and alerts if some error is detected triggering an emergency action. To validate this technology, we developed a physical cooperative robotic testbed capable of testing and validating this technology.

Robotic testbeds are very important for this matter, since they are a low-cost solution for testing purposes and they replicate real vehicle's functionality and physics. In this Thesis, we show different components that a cooperative platooning vehicle can possess, as their interactions in control algorithm and with other vehicles of a platoon.

## 1.2 Research Context

This project was developed at CISTER in line with the SafeCOP european project [8], that envisages the use of wireless communication, multiple stakeholders, dynamic system definitions, and unpredictable operating environments. The advantages of this project include lower certification costs, increased trustworthiness of wireless communication, better management of increasing complexity, reduced effort for verification and validation, lower total system costs, shorter time to market and increased market share.

In this context, CISTER's participation aimed at participating in the automotive use case on safe cooperative vehicular platooning and particularly in developing a robotic testbed to validate and demonstrate cooperative vehicular platooning applications and safety mechanisms.

## 1.3 Research Objectives

This Thesis main purpose is to create a low-cost robotic cooperative platooning testbed to validate different technologies and algorithms. To achieve this, we first aimed at the implementation of a base example of a platoon that's based on sonars and range finders exclusively. Following this, the inclusion of cameras, on each car, and an improvement of the platooning control. Lastly, the integration of OBUs to support V2V communications and consequently the demonstration of the CLW mechanism.

## 1.4 Thesis Structure

In chapter 2, there is an overview of different types of platooning mixing sensor and/or communications and various control models for the followers. There is also an overview of other robotic testbeds relevant to these topics. Chapter 3 contains all technologies and tools that were used to accomplish this project's objectives and their respective clarification. Chapter 4 describes the setup of our testbed and explains how the vehicle's components interact between them and their respective purpose for this application. Chapter 5, one of the main chapters, exhibits the work that was implemented in this Thesis and the results retrieved from it. Chapter 6, similar to the previous chapter, presents the work carried out with GMV Skysoft SA, on platooning related communications and safety features, particularly regarding the CLW module. Chapter 7 concludes the document and points out projects that can and should be carried out in the future to extend this testbed.

# 2 Platooning Overview

## 2.1 General Aspects

The interest in platooning is having an exponential growth in academic and industrial areas and therefore there are increasingly more studies being carried out in such topics. These studies focus on the impact that platoons may have in society, mostly focusing on transportation efficiency and safety.

This section presents the state of the art of platooning focusing on communications and control aspects and it's challenges. One of the main issues to be solved is how to regulate the increase of vehicles in the streets. For this, platoon-based driving can be a viable solution since it improves the traffic flow in the cities and highways.

This approach has many advantages to both traffic and drivers. The cars in a platoon have a constant safety distance that is shorter than in real life situations and this results in less traffic congestion and more road capacity. As the cars are close to each other and in line, the air resistance reduces, forming a slipstream that will result in less energy consumption and emissions to the environment.

It also has a huge impact in communication applications, since there is a constant distance between vehicles. As referred, the drivers also benefit since it's safer and more comfortable driving in a platoon. The vehicles adopt a cooperative autonomous pattern, following the leader's route in a platoon. These patterns consist in some operations that can be done in a platoon, as merging, preserving or splitting a platoon, represented in Figure 2.1.

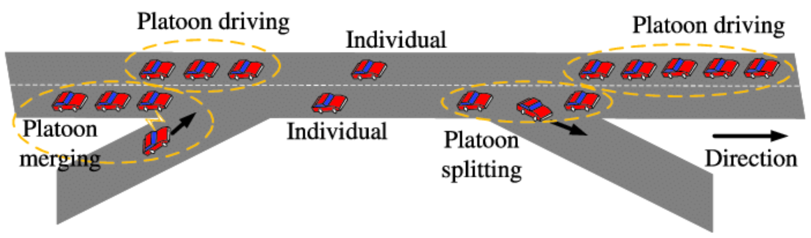


**Figure 2.1:** *Driving Patterns in a Platoon [1]*

The control and behavior of autonomous vehicles can be managed by many sensors in the vehicles. However, wireless communication can improve the safety and reliability of the platoon. The Intelligent Transportation System (ITS) are defined as "*advanced applications which without embodying intelligence as such aim to provide innovative services relating to different modes of transport and traffic management and enable various users to be better informed and make safer, more coordinated and "smarter" use of transport networks*", by the European Union [9]. Each vehicle can carry an On-Board Unit (OBU), responsible for collecting the data from the sensors and send them to the neighbours or to Road Side Units (RSU). Those RSU can be responsible for analyzing the data and redistribute them or even alert the vehicles for conditions of the road.

Considering the vehicles that form a platoon as a complex system that defines the course of action based on sensors and communication with others, it is possible to define them as an instantiation of *Cooperative Cyber-Physical Systems* (CoCPS) [10]. They can be analyzed in two perspectives: intra-vehicle CPS, where the main goal is to optimize the response time and the performance of each car, and inter-vehicular, where the objective is to improve the traffic or the network behavior of the platoon. In both cases, the safety of the platoon is the most important parameter to be guaranteed. The design of the applications and full visualization of platoons as CoCPS is shown in Figure 2.2.

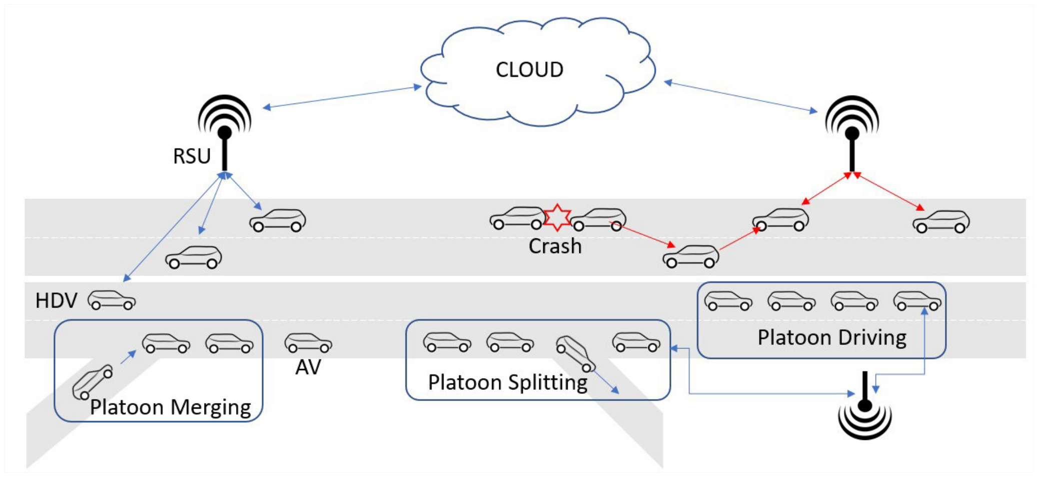


**Figure 2.2:** *Driving Applications*

The most common VP applications focus on the optimization of traffic, reducing energy consumption and service delivery to vehicles, like information about the road. There are many studies on those areas, such as [11], where the authors present an analysis of the traffic flow, considering an autonomous platoon and a mix of human-driven vehicles and autonomous vehicles. Regards the consumption, [12] evaluate the reduction caused by platoons, given the small distance between the cars. There is another study [13], where the decrease in emissions of CO2 is analyzed in platoons. In the service area, the communication with the vehicles can also improve some applications, like internet access and cooperative local service [14, 15].

## 2.2 Communications

Wireless communications are very important in a platoon since they can provide improved reliability and safety, enabling ITS. Researchers developed a technology for this purpose the ITS. For this practical implementation, each car from the platoon must have an OBU, that will receive information from it's sensors and send it to the others or to RSUs. A RSU's purpose is to collect data from cars and/or the road condition and send it to the platoon, so they can adapt their control for a safer path. To have a communication vehicle to vehicle (V2V) or vehicle to infrastructure (V2I) you can establish a wireless communication through a mobile wireless network that is a vehicular ad hoc network. These interactions are shown in Figure 2.3.

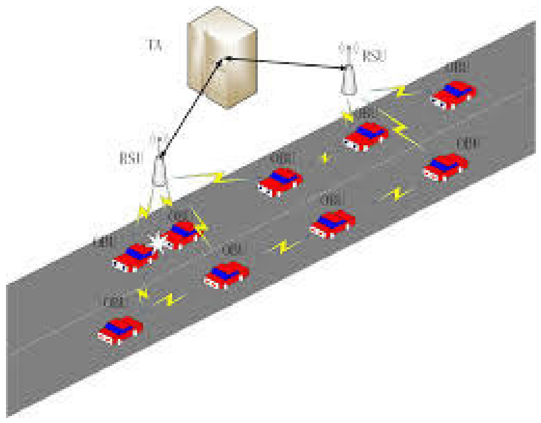


**Figure 2.3:** *Autonomous Vehicle Interaction [2]*

These operations depend on the exchange of messages as state monitoring, services data, control packets and safety messages. The messages' latency can alter from milliseconds to seconds, regarding the intended operation. The data can be disseminated by time, motion or events in the streets and can be distributed in distinctive ways, as unicast, broadcast or multicast.

In this line, some organizations have been developing Vehicle ad hoc Network (VANET) standards. Two big standards are European Telecommunications Standards Institute (ETSI) ITS-G5 from Europe and Wireless Access in Vehicular Environments (Wave) from North America. Both are based on IEEE 802.11p, but they have major differences. A comparison is held in [16], where their main focus is ETSI ITS-G5. It's physical layer is characterized in IEEE 802.11p, but the MAC layer features an additional algorithm, that's named Decentralized Congestion Control (DCC). According to the channel busy rate, this algorithm can adapt the frequency of data transmission dynamically. It can change several parameters of the MAC and physical layer as transmit power, the minimum packet interval, the data rate and the sensitivity of the radio.

One of the main differences between these two standards is that IEEE Wave allows nodes to exchange messages on different channels, the Control Channel (CCH) and one of the Service Channels (SCHs), resulting in the absence of the need for a dual transceiver system. This method is called alternating access. While ETSI ITS-G5 doesn't use this method since it can eliminate the packet loss caused by synchronization effects. Of course this comes with a downside that's the increase of the system's cost due to the need of an on board unit supporting CCH and SCHs applications. There's also a minor difference that consists in the naming of the messages that are distributed through the network. In ITS-G5 these are called

Cooperative Awareness Message (CAM) while in Wave they're named Basic Safety Message (BSM).

These messages, according to ETSI, CAMs are sent systematically from ITS stations to all the neighborsâ stations that are in the communication range. ITS-hosts use this type of messages to help its sensing, as calculating the distance between two vehicles, adding an extra layer for ITS vehicles, despite the fact that these should feature other manners of perceiving their surroundings, like sensors and cameras.

Information sent through CAMs is commonly about the ITS host's position and status. The frequency of sending CAMs is a very important quality requirement, since they should be sent periodically and might express an improvement of services of applications. ETSI set some requirements to the generation of CAMs for most ITS scenarios [17]:

- maximum time interval between CAM generations: $1s$;
- minimum time interval between CAM generations is $0,1s$.
- generate CAM when absolute difference between current and last heading (towards North) CAM heading $\geq \pm 4°$;
- generate CAM when distance between current position and last CAM position $\geq 5m$;
- generate CAM when absolute difference between current and last speed CAM speed $\geq 1m/s$;
- These rules are checked at most every $100ms$;

## 2.3 Control Strategies

As communications are not completely reliable, each car will have it's own control based on data received from sensors that are attached to it as sonars, Light Detection And Ranging (LiDAR), cameras, infrared sensors among others. We will only focus in the control of the followers as that's the main purpose of this project.

Since communications may fail due to interference's or information congestion, the authors of this work [18] try to compensate the failures using an adaptive Proportional-Derivative (PD) control to achieve stability in the platoon. They implement a dynamic information flow among the vehicles, applying a predecessor-follower mode. This way, the information is sent from the preceding vehicle to the following two vehicles. There are also sensors to detect the distance and position of the previous car.

In the following project [19], the authors also consider the predecessor-follower control problem. In this case study, they contempt a platoon in a track that also has static obstacles. They adopt a sensor-based platoon which input derives from an on board camera on each vehicle to detect the previous vehicle and a laser scanner to detect surrounding object in the track. With this, each vehicle calculates his own control signal, acquired from the previous sensors. By obtaining these information, collisions with obstacles and other successive vehicles are avoided.

In [20], the project consists in a platoon of two buses, were the follower was tested with one longitudinal control and four lateral controls. The longitudinal control was Proportional-Integral-Derivative (PID) based and there were performed tests were the input distances were in between 2 m and 10 m. They tested this control with speed variation of 10 m/s from the leader and different payloads on each bus. The lateral controls are Pure Pursuit, Spline Pursuit, Circular Pursuit and Modified Pure Pursuit.

The Pure Pursuit uses the point located at the center of the leader rear axle as the look ahead point. This control law aims to drive the follower on a circular arc in such a way that the center of the follower rear axle coincides with the trace of the center of the leader rear axle.

The second one, Spline Pursuit, takes into account the leaders heading with respect to the follower. In this study, they fitted a cubic spline between the rear axles of the buses. The position of the target point was defined the same way as in Pure Pursuit method. They defined at the follower's rear axle a, the target steering angle was derived after the curvature of the spline.

In the Circular Pursuit method, the steering angle of the follower aims to pursue a circular arc matching with the leaders front axle and tangential to the leaders heading. The control law intends to steer the follower front axle center on an arc defined by the leaders front axle trace. This circular arc passes through the leader and follower front axle centers and is simultaneously coincident with the leaders heading.

The Modified Pure Pursuit method is refitted version of the Pure Pursuit method. This modification takes into account the relative heading of the leader. The difference between both approaches is in the angle between the virtual followers heading and the look ahead vector. In transient conditions, the follower has a different heading in comparison to the virtual follower. Therefore, there is a term to align the followers and the virtual followers headings.

## 2.4 Platooning Testbeds

There are several works on platooning and one major aspect is a practical implementation as a robotic testbed to validate your hypothesis'. In [21], they present a testbed that uses 5G ultra reliable and low-latency communication (uRLLC). They designed a V2X, based on software defined radio, that features flexible reconfiguration in short frame structure and numerology, rapid real-time processing, flexible synchronization and that is easily deployed. The authors examine the system's design and technical enablers for use cases and communication requirements for future cooperative autonomous driving. This testbed consists in an optimized base-band processing and a reconfigurable RF front-end on general purposed CPUs. The technical enablers include a new OFDM-like waveform based on Pulse-shaping, a flexible and self-contained frame-structure design, Global Navigation Satellite System (GNSS) aided hybrid synchronization and low-latency scheduled multiple-access.

Their radio subsystem consists of:

- The NI/Ettus USRP X310 SDR platform as the RF Unit (RFU).
- High performance Intel CPU based PC.
- Lightweight linux distribution.
- Real-time processing framework implemented using C/C++.
- Algorithm simulation environment in Matlab.
- A self-built RF external (RFE) module with power amplifier and TDD switch.

This testbed was integrated with the autonomous car prototypes to test the important use cases of in the cooperative driving such as semi-simultaneous emergency brake. Their initial experiments demonstrated that the emergency brake is successfully triggered from one car to the other ones with 100% successful rate and less than one millisecond delay. This work represents proof of concept of their testbed in a scenario of platooning and has wider applications in the automotive industry.

The work done in [21] has a similar approach to our testbed when comparing to the safety mechanism implemented by CISTER. They propose a emergency brake mechanism for improved road safety. In both testbeds are used OBUs to communicate the control warning to result in full stop of the following vehicles. While in our project we use a Jetson TX2 to process the information, in their project is used high performance Intel CPU based PC. Each project uses a Linux distribution optimized for run-time performance.

At Arizona State University, a group of researchers [3] present vehicular cloud robots (VC-bots) which are designed to ensure an open platform for both research experiments and education services on VANET, vehicular cloud computing infrastructure and future smart vehicles applications, such as on road or indoor. They developed four standard robot vehicles. These are called VC-truck, VC-van, VC-sedan and VC-compact, as shown in Figure 2.4. The VC-truck is displayed at bottom left, VC-van is at bottom right, the VC-sedan is situated at top right and VC-compact is at top left.

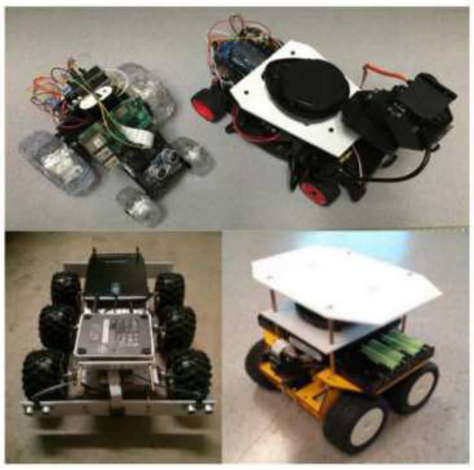
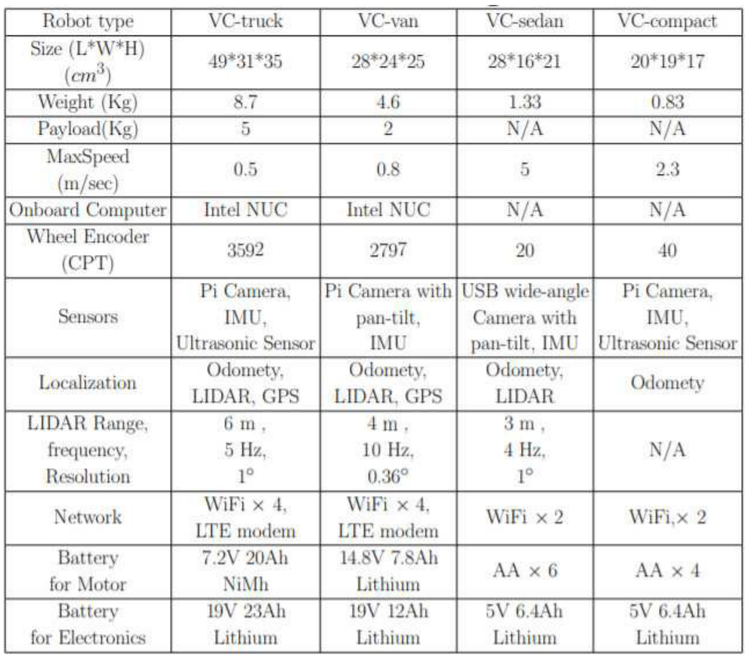


**Figure 2.4:** *VC-Bots*

They have different size, weight, payload and maximum speed but our focus is on the hardware that was used. All robots have IMU and wheel encoders but the VC-sedan is the only one without a Pi Camera. Instead it has a USB wide-angle camera. The VC-truck and VC-compact have a Ultrasonic Sensor. For localization, they all have odometry, but the VC-Compact doesn't have LiDAR, while the other three have it. The VC-Truck and VC-Van also are equipped with a GPS. These characteristics are represented in Figure 2.5



| Robot type | VC-truck | VC-van | VC-sedan | VC-compact |
|---|---|---|---|---|
| Size (L*W*H) ($cm^3$) | 49*31*35 | 28*24*25 | 28*16*21 | 20*19*17 |
| Weight (Kg) | 8.7 | 4.6 | 1.33 | 0.83 |
| Payload(Kg) | 5 | 2 | N/A | N/A |
| MaxSpeed (m/sec) | 0.5 | 0.8 | 5 | 2.3 |
| Onboard Computer | Intel NUC | Intel NUC | N/A | N/A |
| Wheel Encoder (CPT) | 3592 | 2797 | 20 | 40 |
| Sensors | Pi Camera, IMU, Ultrasonic Sensor | Pi Camera with pan-tilt, IMU | USB wide-angle Camera with pan-tilt, IMU | Pi Camera, IMU, Ultrasonic Sensor |
| Localization | Odomety, LIDAR, GPS | Odomety, LIDAR, GPS | Odomety, LIDAR | Odomety |
| LIDAR Range, frequency, Resolution | 6 m , 5 Hz, 1° | 4 m , 10 Hz, 0.36° | 3 m , 4 Hz, 1° | N/A |
| Network | WiFi × 4, LTE modem | WiFi × 4, LTE modem | WiFi × 2 | WiFi,× 2 |
| Battery for Motor | 7.2V 20Ah NiMh | 14.8V 7.8Ah Lithium | AA × 6 | AA × 4 |
| Battery for Electronics | 19V 23Ah Lithium | 19V 12Ah Lithium | 5V 6.4Ah Lithium | 5V 6.4Ah Lithium |

**Figure 2.5:** *VC-Bots Hardware [3]*

The motor control signals are generated on Raspberry Pi with a two level PID control algorithm based on current motion states and desired states. Then the control signals are handled by an Arduino, that measures wheel speed with rotary encoder, which generates Pulse Width Modulation (PWM) signal to drive the motors.

The VC-bots support multiple network access methods. where each robot vehicle is equipped with multiple WiFi interfaces and several WiFi routers are deployed as RSU which are connected to the Internet. One interface is setup as WiFi access point and other interfaces can either connect to another robot vehicle for V2V communication, or connect to an RSU for V2I communication. The robots form a connection to other vehicles by sending commands over the management network, which is used by the User Interface (UI) to allow user to setup network topology.

Their motion states include three types, position and orientation, linear speed and angular speed, and linear acceleration. The rotary encoders sense wheel rotation speed, which further derive through kinematics model and by dead reckoning. Due to the accumulated error in dead reckoning, they use a LiDAR sensor with simultaneous localization and mapping (SLAM) software to correct position and an IMU to correct orientation. Based on these sensors, they designed multiple controllers as cruise control and planar stabilizing.

They have implemented a longitudinal platooning algorithm with their testbed, that uses V2V communications. The platooning software component registers three methods. These methods are join, follow, and a state report event source on its local message broker. By using V2V communications, other platooning elements directly link to the platooning header and call the headerâs join method as well as subscribe the state reporting event source. The return of join method is the name of the robot vehicle at platoon tail. The new platooning member also connects to the platoon tail and calls its predecessorâs follow method and subscribes the state reporting event source, since the platooning algorithm requires each robot to send its acceleration and speed to its successor in the cluster. After successfully joining the platoon, the algorithm systematically calculates the desired speed based on predecessorâs states, distance to the leader as well as the headerâs state to maintain steadiness of the whole platoon. Platoon leaving is implemented in the reverse order by calling the unfollow method on the predecessor, a leave method on the head and unsubscribing the event sources.

When comparing this project done at Arizona State University to the work done at CISTER, it is possible to highlight that both project use ROS as a baseline. They have four different platforms, of which two use on board computer, Intel NUC, but since they do not refer which one, it is not possible to perform a comparison to our testbeds processing power. The other two platforms that don't use an on board computer use remote cloud computing to control their movement. They use WiFi, while in our safe mechanism the communication is done via ETSI ITS-G5, enabling a higher safety-critical reliability for our scenario.

Despite our communications being more reliable, their testbed offers a wider sensory input analysis, coming from LiDAR, wheel encoders and GPS.

They also have a larger test flexibility since they have several WiFi routers deployed as road side unit, simulating a better real life scenario.

# 3 Technologies and Tools

This chapter presents a description of all technologies that were used and implemented for this project. First, we approach the hardware components followed by a description of the software.

## 3.1 Hardware

### 3.1.1 Traxxas RC Car

In the interest of developing a physical testbed, a car model was necessary to be altered and adapted to have all the components placed and organized. The car model used is a Traxxas Fiesta ST Rally [22], a 1/10 scale of a real car, like the one in Figure 3.1. The versatility of the RC model, allows it to be adjusted at will, creating a well structured platform to test different scenarios.

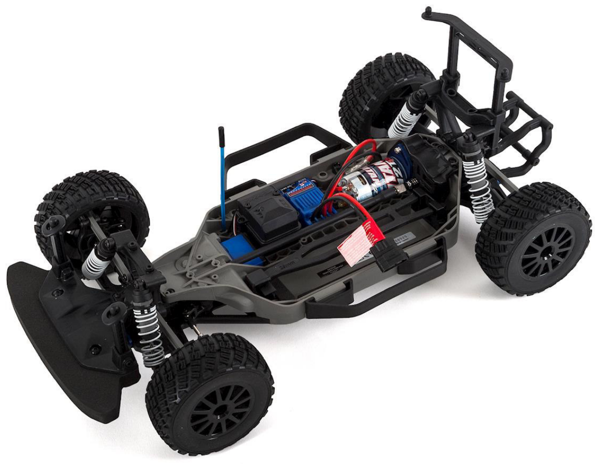
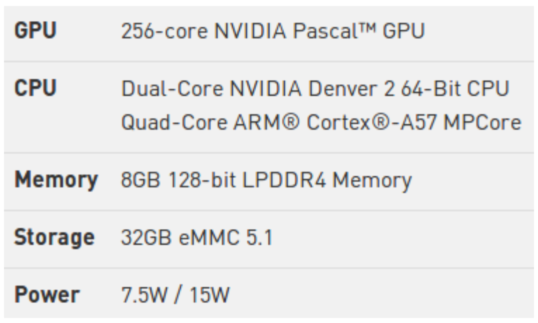


**Figure 3.1:** *Traxxas RC CAR [4]*

This RC car comes with a Titan 12T Waterproof DC Motor, up to 8.4 V, a XL-5 Electronic Speed Controller (ESC), a steering servo, a RC receiver and it's remote controller. Besides the components that come mounted on the car, a Lipo battery is necessary to power up the car, in this build case a 2-cell 5800 maH 7.4V Traxxas Lipo battery was used.

### 3.1.2 Jetson TX2 and Developer Kit

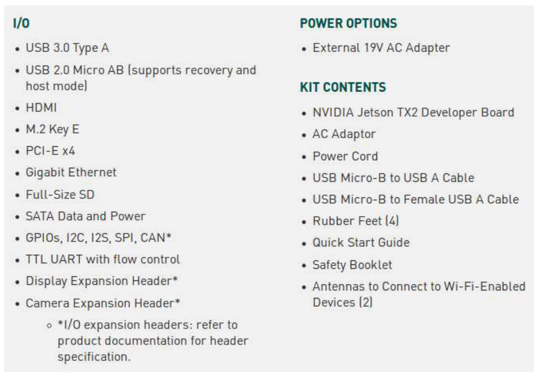
Jetson TX2 is a very fast, power-efficient embedded AI computing device. This 7.5-watt has a 256-core NVIDIA Pascal GPU and is loaded with 8GB of memory and 59.7GB/s of memory bandwidth. It has a eMMC 5.1 storage with 32 GB and a Dual-Core NVIDIA Denver 2 64-Bit CPU and also a Quad-Core ARM® Cortex®-A57 MPCore as shown in Figure 3.2.



| | |
|---|---|
| **GPU** | 256-core NVIDIA Pascal™ GPU |
| **CPU** | Dual-Core NVIDIA Denver 2 64-Bit CPU<br>Quad-Core ARM® Cortex®-A57 MPCore |
| **Memory** | 8GB 128-bit LPDDR4 Memory |
| **Storage** | 32GB eMMC 5.1 |
| **Power** | 7.5W / 15W |

**Figure 3.2:** *Jetson TX2 Technical Specifications*

The Jetson TX2 Developer Kit gives you a fast, easy way to develop hardware and software for the Jetson TX2 on a module. It exposes the hardware capabilities and interfaces of the developer board, comes with design guides and other documentation, and is pre-flashed with a Linux development environment (Figure 3.3). It also supports NVIDIA Jetpack  a complete software development kit (SDK) that includes libraries for deep learning, computer vision, GPU computing, multimedia processing, and much more.



**I/O**

- USB 3.0 Type A
- USB 2.0 Micro AB (supports recovery and host mode)
- HDMI
- M.2 Key E
- PCI-E x4
- Gigabit Ethernet
- Full-Size SD
- SATA Data and Power
- GPIOs, I2C, I2S, SPI, CAN*
- TTL UART with flow control
- Display Expansion Header*
- Camera Expansion Header*
    - *I/O expansion headers: refer to product documentation for header specification.

**POWER OPTIONS**

- External 19V AC Adapter

**KIT CONTENTS**

- NVIDIA Jetson TX2 Developer Board
- AC Adaptor
- Power Cord
- USB Micro-B to USB A Cable
- USB Micro-B to Female USB A Cable
- Rubber Feet (4)
- Quick Start Guide
- Safety Booklet
- Antennas to Connect to Wi-Fi-Enabled Devices (2)

**Figure 3.3:** *Developer Kit Tech Spec*

### 3.1.3 Teensy 3.2

Teensy is a Universal Serial Bus (USB) based micro-controller that features a 32 bit ARM processor with great processing power and has 24 I/O, from which are digital, analog, Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), PWM, serial and touch pins. These characteristics are shown in Figure 3.4. This board is ideal for this project since it is compatible with Arduino software and libraries, that interfaces with the Robot Operating System (ROS).

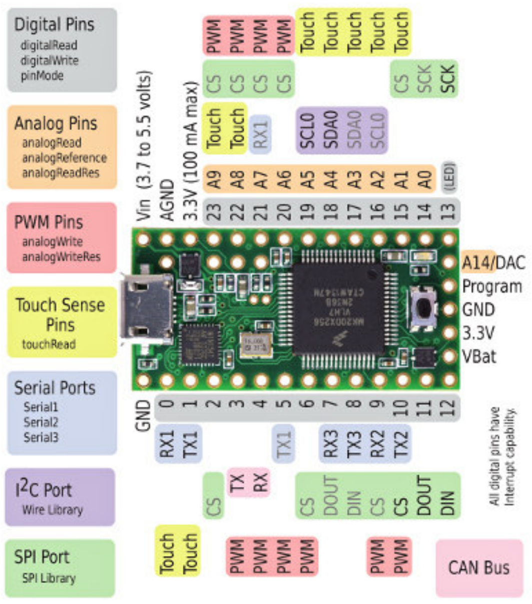


**Figure 3.4:** *Teensy 3.2*

### 3.1.4 Inertial Measurement Unit (IMU)

The Sparkfun 9 dof Razor IMU, version sen 14001, equal to the one in Figure 3.5 has 9 degrees of freedom and combines 3 different types of sensors, accelerometer, gyroscope and magnetometer. These have the ability to sense linear acceleration, angular rotation speed and magnetic field vector respectively. Combining all three sensors, it can obtain the orientation of a given object, in this case, a vehicle.

**Figure 3.5:** *Sparkfun Razor IMU*

### 3.1.5 ZED stereo camera

The ZED, Figure 3.6, is a passive stereo vision based camera that reproduces the way human vision works. Using it's two cameras and through triangulation, the ZED understands it's surroundings and creates a three-dimensional (3D) model of the scene it observes. The ZED camera outputs a high resolution side-by-side color video on USB 3.0. that can go up to 2.2K at 15 frames per second. In our case we use at most 1080p at 30 fps. It has a depth of 0.5 to 20 meters and has an accuracy of 1 mm / 0.1. It has the capacity perform real-time depth-based visual odometry and SLAM and supports ROS, Open Source Computer Vision (OpenCV), Matlab, Unreal Engine 4 and Unity.

**Figure 3.6:** *ZED Camera*

### 3.1.6 Range Finders

The Range Finder GP2Y0A02YK0F from Sharp, Figure 3.7, is a distance measuring sensor unit. This unit is composed by a position sensitive detector, an infrared emitting diode (IR) and a signal processing circuit. As this sensor uses the triangulation method, it is not very sensitive to the variation of the reflectivity of the object, the environmental temperature and the operating duration. These devices output the voltage corresponding to the distance detection and they can measure distances from 20 cm up to 150 cm.

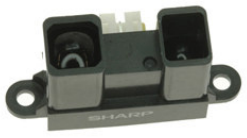


**Figure 3.7:** *Range Finder GP2Y0A02YK0F*

### 3.1.7 Sonar

The SRF08 sonar, Figure 3.8, is a high performance ultrasonic range finder that measures an amazingly wide range from 3 cm to 6 m. This sonar interfaces via the industrial I2C protocol. This module offers two modes, a ranging and a artificial neural network (ANN) mode.

In the ranging mode, it can register up to 17 echo's and the default time for completion of ranging is 65 ms, however it is possible to shorten this by writing to the range register before issuing a ranging command. This is a major benefit since there is the possibility to acquire more detection's in the same amount of time.

The ANN mode provides a multi-echo in a manner that is easier to input that data into a neural network.

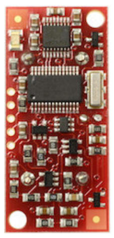


**Figure 3.8:** *Sonar SRF08*

### 3.1.8 Cohda On Board Unit

The MK5, Cohda's fifth-generation OBU [5], shown in Figure 3.9, is a small, low-cost module that is adapted to fit in vehicles for aftermarket deployment or field trials and it's designed for automotive production and Smart City. This unit performs in safety-critical scenarios or potential hazards since it exchanges information at a high speed over wide distances. It's radio performance fulfil purpose in challenging outdoor conditions where no line-of-sight is available. It offers a robust, secure foundation for the intelligent transport systems and can be applied in V2X trials. The MK5 works on Dual IEEE 802.11p radio and has a powerfull processor running Cohda software applications, a GNSS, integrated security, hardware acceleration and has a tamper-proof key storage. Also supports DSRC (IEEE 802.11p), Ethernet 100 Base-T.

**Figure 3.9:** *Cohda Wireless MK5 OBU [5]*

## 3.2 Software

### 3.2.1 NVIDIA Jetpack 3.3

NVIDIA offers solution for building applications, the NVIDIA JetPack SDK which allows to flash the development board with latest OS image, install developer tools, libraries and samples. This package contains CUDA toolkit for the host (Ubuntu) and target platform, the latest NVIDIA Developer Tools (Tegra Graphics Debugger and NVIDIA System Profiler), VisionWorks, cuDNN, Multimedia Application Programming Interface (API), OpenCV Library 3.3.1, and TensorRT [23].

### 3.2.2 Robot Operating System

ROS framework, maintained by Willow Garage and Open Source Robotics Foundation (OSRF) since 2007, is an open-source middleware that has undergone rapid development and has been widely used to design robotics applications. With it's many software frameworks it provides a variety of tools, libraries and conventions that facilitates the creation of robotic applications and further encourages the sharing and reusing codes and problem solving through the robotic community. Although ROS is not a real-time framework, it is possible to integrate it with real-time code.

Basically, it consists in establishing communication between two or more nodes and it will help reusing code that has already been implemented. For example, the following diagram, Figure 3.10, represents a demo system, where ellipses represent ROS nodes, which publish and/or subscribe to a topic "/example" whose message type is "std_msgs/String".

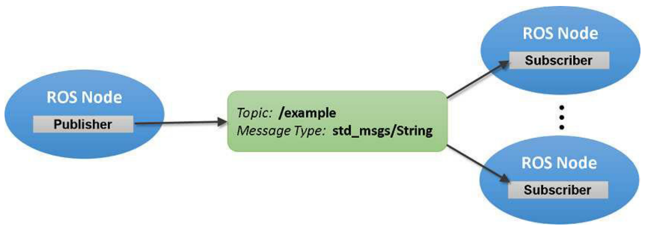


**Figure 3.10:** *Publish and Subscribe Example*

This kind of diagrams is generated by using rqt_graph tool, which automatically presents a diagram representing the currently running ROS system.

As explained in [24], ROS has 3 levels of concepts: The Filesystem level, The Computation Graph level, and the Community level.

The Filesystem level covers the overall resources of ROS and includes Packages, Messages and Services. The Packages are the main unit for organizing software in ROS. A package may contain ROS runtime processes (nodes), a ROS-dependent library, datasets, configuration files or anything else that is usefully organized together. Packages are the most atomic build and release item in ROS. Meaning that lowest level thing you can build and release is a package. The Messages description store the messages used in each package. Service description define the request and response data structures for services in ROS.

The Computation Graph level is what establishes the communication between ROS processes that are computing data together. The fundamental concepts of this level are:

- Master [25] - The ROS Master provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other they communicate with each other peer-to-peer.

- Nodes [26] - Nodes are processes that perform computation. A robot control system will usually include many nodes, that one can be in charge of controlling a laser range finder, another reading wheel odometry, other performing localization among others.

- Messages [27] - A message is a data structure, containing typed fields and is used by nodes to pass information between each other. For example a message can carry the values of a node that reads the scan of a sonar and needs to share to another node to use that distance in order to apply a longitudinal control algorithm.

- Topics [28] - Messages need to be transported trough a defined route. To do that Topics are in charge of guiding the messages via a system with publish / subscribe semantics. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish or subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence. The idea is to decouple the production of information from it's consumption.

- Bags [29] - When developing and testing the necessity of reproducing the same environment is crucial. Bags, created from the tool ROS Bags, are a format that store serialized message data, generally from sensors as it's received. This data is then saved in the bag and can be played back the same way as any other node. This is usually used for debug or demonstrations.

The ROS Master is the central node in the ROS Computation Graph. It stores topics and services registration information for ROS nodes. Nodes communicate with the Master to report their registration information and to receive information about other registered nodes. The Master will also make callbacks to these nodes when this registration information changes, which allows nodes to dynamically create connections as new nodes are run.

Nodes connect to other nodes directly, being the master only responsible to store and associate information. Nodes that subscribe to a topic will request connections from nodes that publish that topic and will establish that connection over an agreed upon connection protocol. The most common protocol used in a ROS is called TCPROS, which uses standard TCP/IP sockets.

The third and last level of ROS, the Community level is responsible for sharing and providing resources, knowledge and software with distinct communities. This level provides distributions, repositories, a wiki, Q&A web page for answering dedicated ROS Questions. In this project the distribution used was ROS Kinetic Kame.

### 3.2.3 OpenCV 3.4.2

OpenCV is a cross-platform library that enables programming targeting computer vision. This library is written in C++, however there are bindings for other different types of programming languages, such as Python. OpenCV provides an extensive area of applications, such as facial recognition, object identification, motion tracking and many others. As result of it's functions, OpenCV was vital tool in order to create a platoon, as the follower car searches for an image from the car in front of it an applies the necessary controls to follow it.

### 3.2.4 ZED Python API

Since the focal programming language used in this project was Python, to obtain the information necessary from ZED and process it a dependency was required to install. This dependency is the ZED Python API and allows to use certain functions that are indispensable for integrating in the image detection.

### 3.2.5 Virtual Network Computing (VNC)

The purpose of this program was to visualize the NVIDIA Jetson TX2 data from a remote computer. Since the cars will be running autonomously, there was in need a way to not only monitor receiving and sending information, but also a remote way to stop the cars. This way, VNC Viewer was installed both on the Jetson TX2 and another remote computer and created an ad hoc network, to ensure that the connection to the car was never lost.

# 4 Robotic Platooning Testbed Architecture

This chapter overviews the overall testbed architecture and details the workings of each implemented component and their interactions.

## 4.1 System's Architecture

As explained before, this car was setup based on the F1/10 build [30] with some modifications.

In Figure 4.1, we can observe, on a lower level, that the teensy is our interface between the Jetson and the motor and servo. By sending PWM signals, from the teensy to the motor and servo, we can control the car's speed and steering. There are also two switches so that we can opt between controlling the car with it's original controller or by software that we developed.

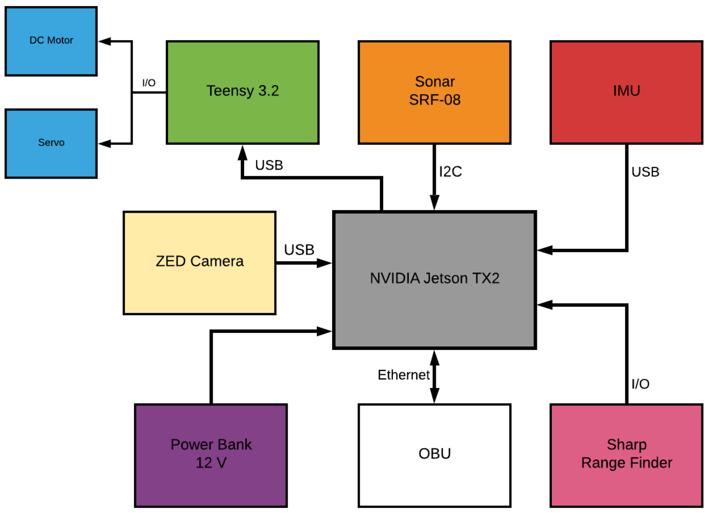


**Figure 4.1:** *Hardware Architecture*

We can also observe that the IMU, the teensy and the ZED camera connect to the Jetson through USB. As the Jetson only provides one USB port, there was the need to add an USB hub to create further slots.

The sonars connect via I2C communication protocol so that their echo is transmitted as a signal. The range finders that were implemented have their voltage supply coming from Jetson, but their signal is sent to digital pins from teensy, then processed through the arduino IDE and published to ROS.

We have a power bank that supplies energy to the Jetson and the OBUs at 12 V. The OBUs, that are placed on the top of each vehicle with their respective antenna are used to communicate to other vehicles of the platoon. The Jetson board has an Ethernet connection to the OBUs in order to send data from the robot to the OBU and consecutively to the platoon's network.

As the hardware architecture is presented, there is also a representation of the software architecture in Figure 4.2.

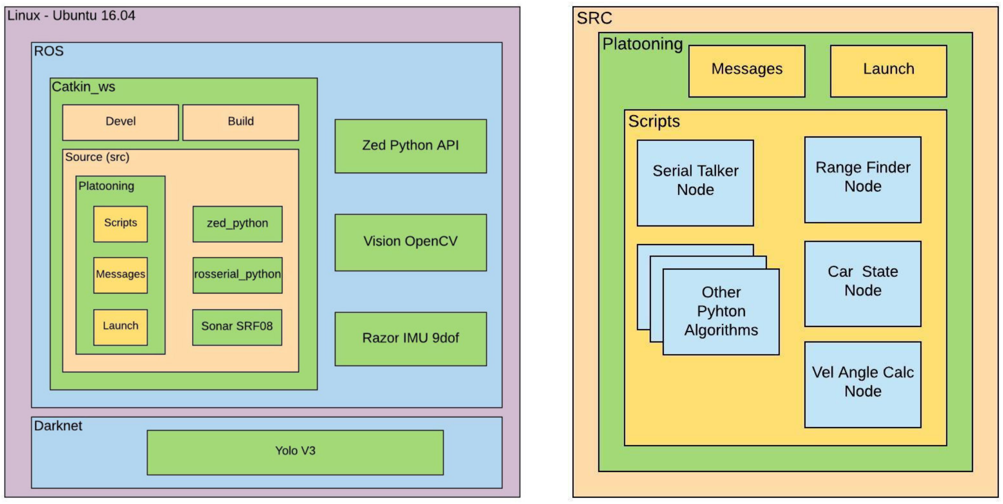


**Figure 4.2:** *Software Architecture*

The operating system running on the Jetson TX2 is Linux Ubuntu 16.04. The ROS-based system has a workspace, Catkin_ws, that contains three folders: Devel, Build and Source. It also has additional ROS packages such as Zed Python API, Vision OpenCV and Razor IMU 9dof, that complement our coding necessities.

In the Source folder contains folders for various control mechanism. In the platooning mechanism are included the developed scripts, messages and launch files necessary to implement the platooning on our testbeds.

These are also other control mechanism, as zed_python, rosserial_python and Sonar SRF08. The zed_python mechanism is responsible for providing camera image processing, while the rosserial_python establishes connection to the teensy and the Sonar SRF08 triggers and reads the sonars used.

The image processing is aided by an external package, Darknet, that is used to classify images in a fast and easy way. Yolo V3 is the object detection algorithm chosen for this project.

## 4.2 ZED Camera mount

A support was required to secure the ZED Camera in front of the car. It was attached in the front part of the car, to analyze the environment. This support has in it's constitution 3D printed pieces and the smallest acrylic bases.

Firstly, remove the stands in front of the car, that were supporting the cars carcass. Secondly, as shown in Figure 4.3, attach on each side of the base both 3D pieces. If it struggles to fit in, carefully file down the 3D pieces holes and/or the part from the base that fits in the holes. Finally, insert the ZED camera in the assembled part.

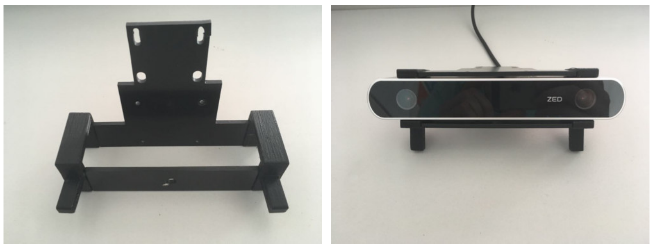


**Figure 4.3:** *Mount of the ZED camera base*

## 4.3 Teensy PCB Board

The Teensy, purpose explained in section 3.1.3, must be fixed with two switches in order to vary our purpose to control the vehicle via software or the original controller. For this purpose, we developed a PCB board that would contain those elements and the wiring to the motor and servo of the car. The PCB's connection are represented in Figure 4.4.

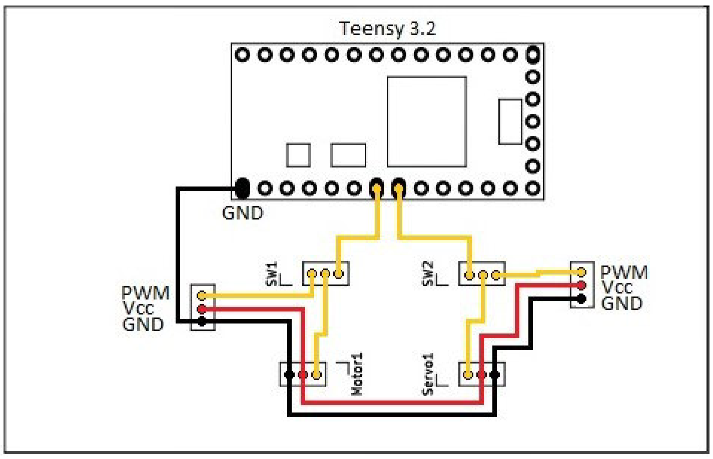
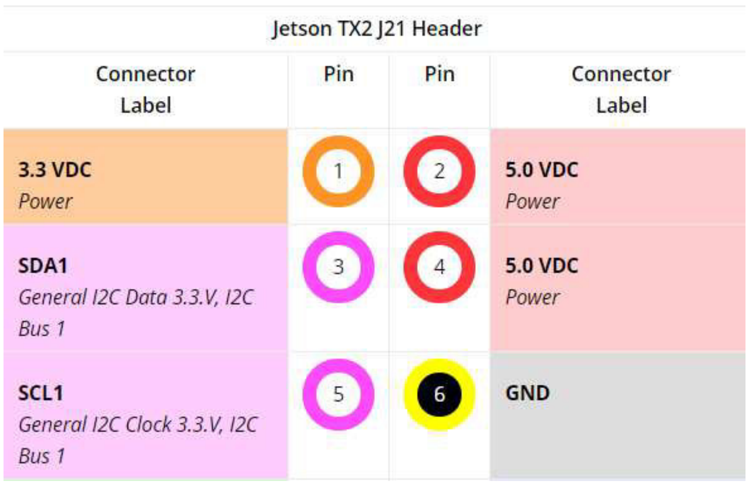


**Figure 4.4:** *Teensy PCB Board Schematic*

## 4.4 Sensor Board

The developed board contains four range finders and three sonars. The four range finders are situated above the sonars and are vertically symmetrical as the sonars are. The range finders edges have holes that will be screwed into this board in order to make them stable enough to transmit and receive their signal. On the other of the side, the screws will be held with nuts. In order to pass the sensor's cables through the board, there were made holes to pass them. These sensors are vertically oriented, since the recommendation in the datasheet is to set the sensor in the moving direction of the object and the line between emitter center and detector center should be vertical.

Each range finders has three connections that have to be set. The signal wire connects to the an analog pin in Teensy. The pins designated from Teensy for the range finders are the 14, 15, 16 and 17. The mass wire connect all between themselves and the Jetson TX2 mass pin. Lastly, the power supply wire, identical as the mass wire, connects to the other power supply wires from the other range finders and the 5 V pin from Jetson TX2. In Figure 4.5, there is a presentation of the Jetson TX2 pinout, where we can observe the pins that were used for this project.



Jetson TX2 J21 Header

| Connector Label | Pin | Pin | Connector Label |
|---|---|---|---|
| **3.3 VDC** *Power* | 1 | 2 | **5.0 VDC** *Power* |
| **SDA1** *General I2C Data 3.3.V, I2C Bus 1* | 3 | 4 | **5.0 VDC** *Power* |
| **SCL1** *General I2C Clock 3.3.V, I2C Bus 1* | 5 | 6 | **GND** |

**Figure 4.5:** *Jetson TX2 Pinout [6]*

The "SDA1" and "SCL1" are important when referring to the sonars as they use the I2C protocol. The "SCL" is the clock line that is used to synchronize all data transfers over the I2C bus, while the "SDA" is the data line. Each sonar has a "SDA" and a "SCL" pin that connect to the respective pins on the Jetson. They also have a power supply pin and a ground pin that connect to the respective pin on the Jetson as the range finders. Figure 4.6 shows the sonar SRF08 pinout to better comprehend the previous explanation. There is also a pin marked as "Do not connect" that is only used by manufacturer to program the PIC16F872. The sonars were placed on the bottom part of the board with the execution of openings in the same so that they can measure the distance through sent beams.

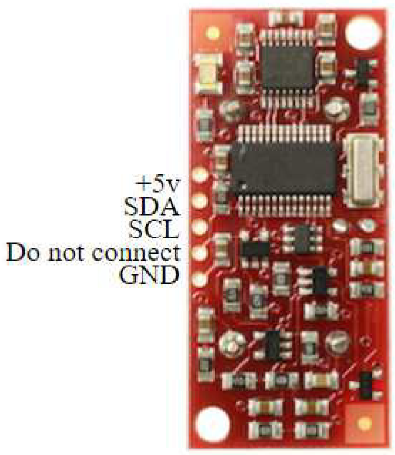


**Figure 4.6:** *Sonar Pinout [7]*

In Figure 4.7, we can observe the final aspect of the developed board throughout the purpose of establishing a platoon based on sensory input. As mentioned before, it is noticeable the positioning of the sensors symmetrically. We can also observe the perforations made for the sonars and the range finder's cables.

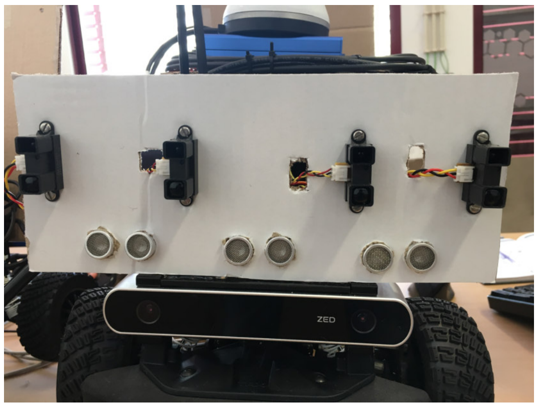


**Figure 4.7:** *Sensor Board*

In Figure 4.8, we can observe the basic function of the sensors that were used on this board, as well as, their reach and, in the case of the sonars, the beam width.

In red, we observe the lasers transmitted by the range finders, that are limited by their minimum and maximum reach of 20 cm and 150 cm respectively. In grey, we represented the sonars beams. They can measure between 1 cm and 2 meters with our configuration.

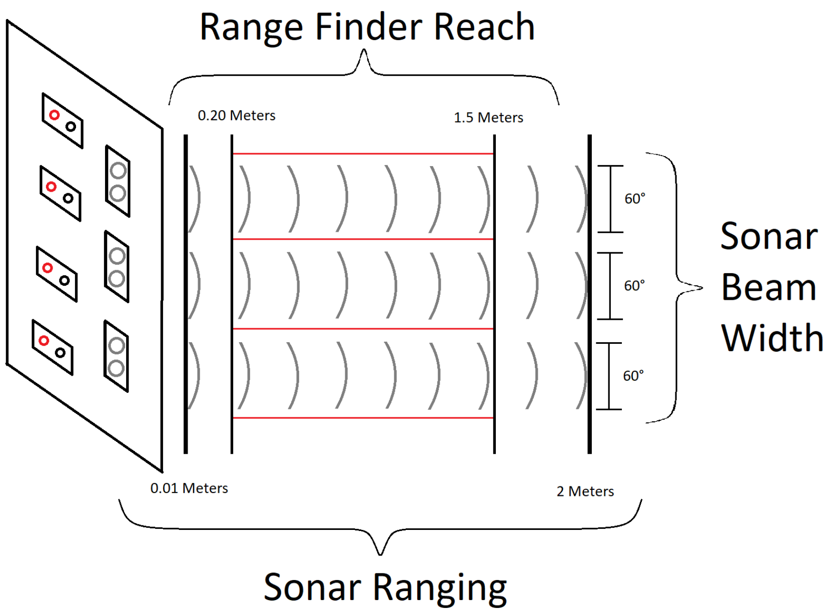


**Figure 4.8:** *Sensor Board Representation*

## 4.5 MK5 ROS Bridge

The MK5 ROS bridge provides a bi-directional bridge between ROS systems implementing a platoon testbed, whether they are simulated or not, and the MK5 OBUs from Cohda Wireless. This allows a ROS environment to connect and communicate with others ROS environments through MK5 802.11p platforms.

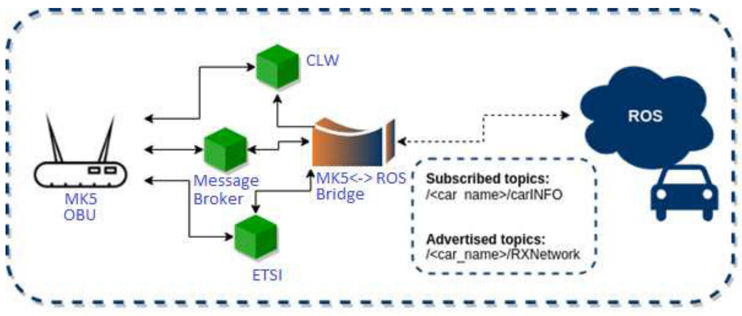


**Figure 4.9:** *ROS Bridge Architecture*

In Figure 4.9, ROS represents a physical platooning testbed. The MK5 ROS Bridge end-side subscribes to a topic coming from the ROS-based system, in order to provide their information through the bridge to feed the OBUs. In the opposite direction, it also, publishes a topic filled with information coming from the Bridge, so the vehicles can properly acknowledge this. On the OBUs end-side of the Bridge, all the information coming from it is fed into the Message Broker module, which segregates and processes this data, in order to, feed it into the remaining OBU modules, while CLW and ETSI are safety mechanism modules.

## 4.6 Software Setup

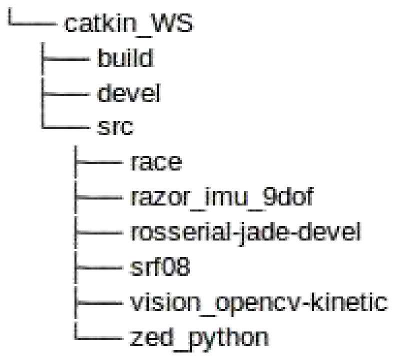
At first, we started by flashing Jetson TX2 with the JetPack 3.3 and Ubuntu 16.04. We downloaded the SDK from NVIDIA's web page and proceeded to the setup. In the setup, we selected which platform was flashed (Jetson TX2). Then, we chose the packages that should be installed and if we wanted to flash the operating system into the Jetson. We proceeded with a full installation, as we want to use Jetson's full potential for this project. Subsequently, we selected the network layout between our personal computer and Jetson TX2. In this case, the devices are connected via router. Then, we followed a simple 5 step guide:

1. Jetson TX2 must be fully powered down.
2. Connect a micro-B plug to the USB micro-b port on the Jetson and a USB port on the personal computer.
3. Connect the power adapter to Jetson.
4. Press and release the power button to start the Jetson. Press and hold the force recovery button. While pressing, press and release the reset button.

Then wait 2 seconds to release the force recovery button.

5. Check on your personal computer, that when you type, in the terminal, the lsusb command for a line that refers "NVidia Corp". If it does, press Enter and it will start flashing your Jetson.

Then we installed ROS for Kinetic distribution [31]. Firstly, we had to configure our Ubuntu repositories to allow "restricted," "universe," and "multiverse". After, we setup up sources.list to accept packages from ROS and our keys. Confirmed that our Debian package index is up-to-date. Subsequently, we installed ROS-Base, that includes only the basics of ROS, no Graphical User Interface (GUI) tools. Then, we initialized rosdep that permits us to easily install system dependencies to compile and run core components in ROS. Lastly, we setup our work environment, were all the work was implemented and stored. The final aspect is in Figure 4.10.



```
└── catkin_WS
    ├── build
    ├── devel
    └── src
        ├── race
        ├── razor_imu_9dof
        ├── rosserial-jade-devel
        ├── srf08
        ├── vision_opencv-kinetic
        └── zed_python
```

**Figure 4.10:** *Work Environment*

After, we learned how to establish a connection between the Jetson TX2 and the motor and servo through the teensy 3.2. For this, we programmed the teensy with a code provided by F1tenth [32] using the Arduino Integrated Development Environment (IDE) on a personal computer. This was done because there is no Arduino IDE available for the Jetson's architecture. As we were programming, there was the need to install some additional packages for this communication as the teensy is not recognized by the Jetson [33]. The code provided by F1tenth envisages a ROS subscription of a topic /drive_pwm that contains the speed and steering values already computed, so that teensy only needs to send two PWM signals to the servo and the motor.

Further on, we added a implementation to the code in teensy with the Arduino IDE that would trigger the range finders, register their output and calculate the distance that was detected in each. Those values would be published through ROS.

For the sonars, as, by default, they all have the same address, there was the need to develop a code in Python that would change their address. To change the I2C address of the SRF08 we connected only one sonar on the bus at each time. Then, we sent 3 sequence commands in the correct order followed by the address. The sequence was sent to the command register at location 0, which means 4 separate write transactions on the I2C bus. In the first command we transmitted the value "0xA0", followed by "0xAA" and "0xA5". The last value is the address the we pretend [7]. Also, there was the necessity to change their time ranging, as we want the maximum output values in the shortest amount of time possible. This value was adjusted to 20 ms, since we want to capture a maximum of 3 m.

When we successfully established the previous connection, we began to analyse what was needed to implement the ZED camera. By searching their web page [34], we downloaded the ZED SDK 2.8.0 for Jetson TX2 and proceeded to the installation. As we were working in Python, there was the need to install a dependency for the ZED SDK, the ZED Python API [35] and other dependencies as numpy and cython. After testing the tutorials with success, we then installed OpenCV 3.4.2, so that we could process and manipulate images as we intended. For this, we followed instructions [36] for python 2.7 and without using a virtual environment as we concluded there was no need.

# 5 Platooning Algorithms

In Chapter 5, there is a presentation and description of the platooning algorithms that were implemented, together with the corresponding results.

## 5.1 Sensor Board based Platooning

As our base line was to have a cheap and practical implementation of a platoon that could be tested in a safe environment as our workplace, we started with 3 sonars and 4 range finders for data input to control both steering and speed through PWM signals. The range finders, positioned symmetrically, are used to determine the steering by knowing which one of them is capturing the car in front.

This platoon consists in a group of several nodes, each responsible to calculate a different property of the autonomous vehicle. These nodes are represented in Figure 5.1.

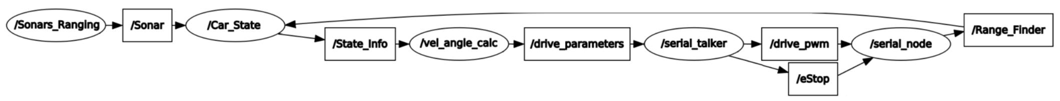


**Figure 5.1:** *Baseline Rqt Graph*

There is one independent node, meaning it only publishes topics and doesn't subscribe any. This node is /Sonars_Ranging, as we can perceive in Figure 5.2, that is responsible for starting the sonars ranging and calculate the respective values in cm that will be used for the longitudinal control. These values are published to a topic called /Sonar.

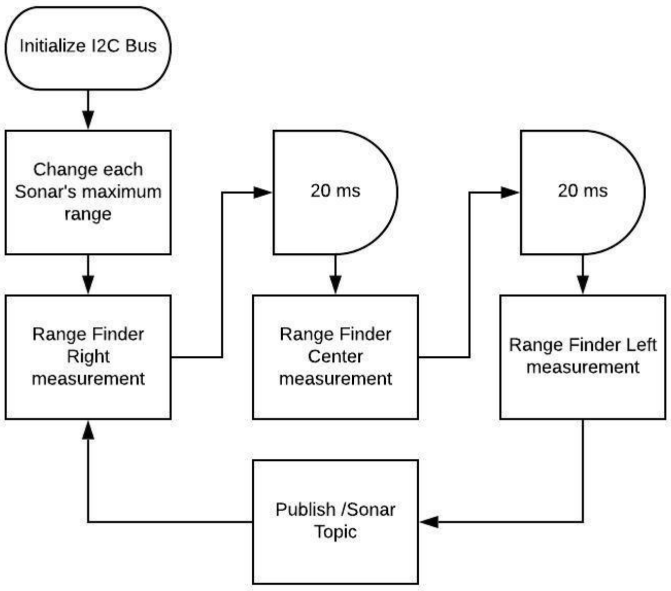

Initialize I2C Bus
Change each Sonar's maximum range
Range Finder Right measurement
20 ms
Range Finder Center measurement
20 ms
Range Finder Left measurement
Publish /Sonar Topic

**Figure 5.2:** */Sonars_Ranging Node Flowchart*

Then there is the /Car_State node, that subscribes the previous topic, /Sonar, and the topic /Range_Finder. This last topic is published by the /serial_node node with the values of the range finders that are calculated on teensy in cm. The /Car_State node decides, based on the information that is being subscribed, what's the car's status. This definition is explained in sub chapter 5.1.1. It publishes a topic, /State_Info, that contains the information of the distance to the car in front and it's status. At that point, the node /vel_angle_calc subscribes this topic and is responsible for determining the vehicle's steering, based on the vehicle's state and applying a PD control to calculate the vehicle's desired speed, as represented in Figure 5.3. When these computations are done, it publishes them to the topic named /drive_parameters.

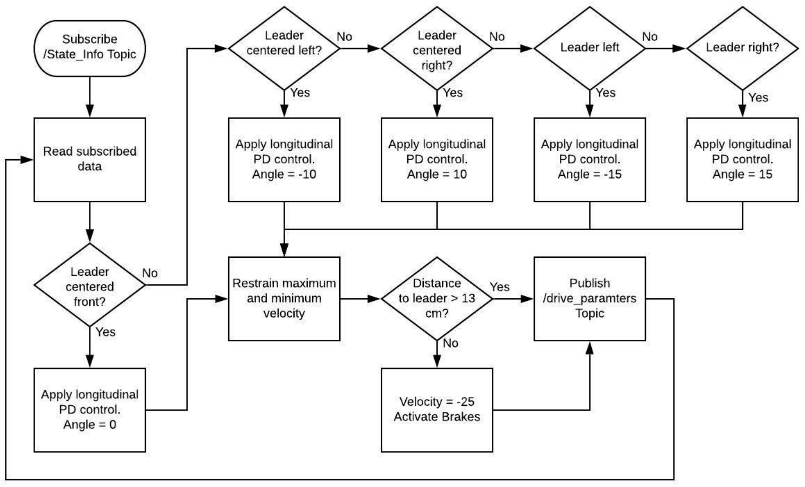

Subscribe /State_Info Topic
Read subscribed data
Leader centered front?
No
Yes
Apply longitudinal PD control. Angle = 0
Leader centered left?
No
Yes
Apply longitudinal PD control. Angle = -10
Leader centered right?
No
Yes
Apply longitudinal PD control. Angle = 10
Leader left
No
Yes
Apply longitudinal PD control. Angle = -15
Leader right?
Yes
Apply longitudinal PD control. Angle = 15
Restrain maximum and minimum velocity
Distance to leader > 13 cm?
Yes
No
Publish /drive_paramters Topic
Velocity = -25 Activate Brakes

**Figure 5.3:** */vel_angle_calc Flowchart without camera*

The /serial_talker node subscribes the /drive_parameters and converts the speed and steering values in to numbers that will be interpreted as PWM signals, as demonstrated in Figure 5.4.

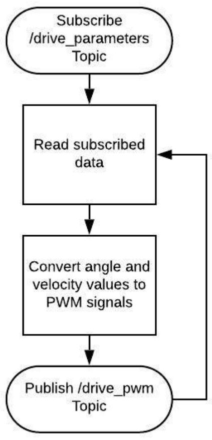

Subscribe /drive_parameters Topic
Read subscribed data
Convert angle and velocity values to PWM signals
Publish /drive_pwm Topic

**Figure 5.4:** */serial_talker Node Flowchart*

These are published to /drive_pwm topic that is subscribed by the /serial_node node, running on teensy, and are sent to the motor and servo, as shown in Figure 5.5.

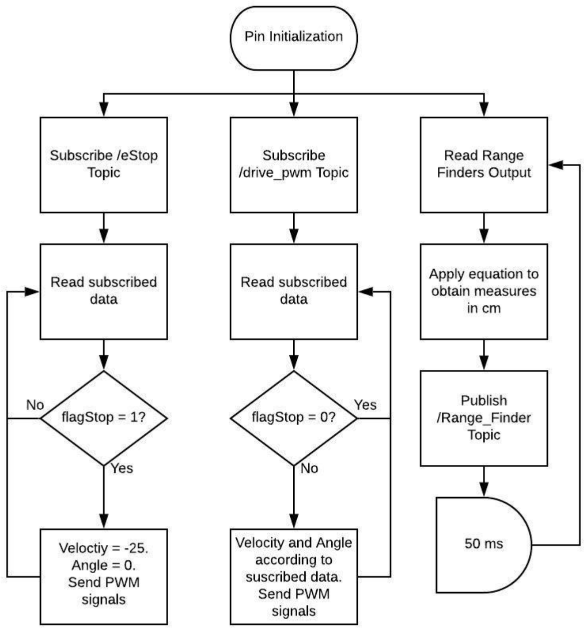

Pin Initialization
Subscribe /eStop Topic
Subscribe /drive_pwm Topic
Read Range Finders Output
Read subscribed data
Read subscribed data
Apply equation to obtain measures in cm
No
flagStop = 1?
Yes
flagStop = 0?
Yes
No
Publish /Range_Finder Topic
50 ms
Velociy = -25. Angle = 0. Send PWM signals
Velocity and Angle according to suscribed data. Send PWM signals

**Figure 5.5:** */serial_node Node Flowchart*

There is also a safety node called /kill_switch that whenever someone wants to stop the car, only has to press the "delete" button and activates the topic /eStop. The car can be restarted, in other words, resume it's control path by entering the "home" button. This sequence is shown in Figure 5.6

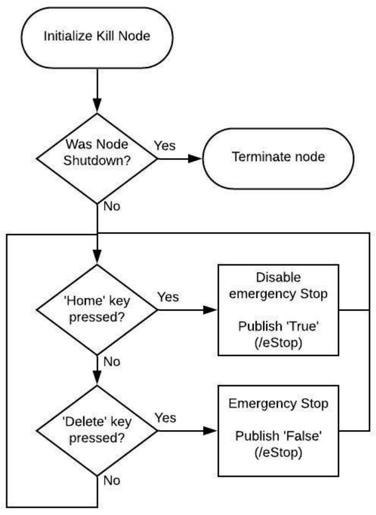


**Figure 5.6:** */kill_switch Node Flowchart*

### 5.1.1 Lateral Control

In Figure 5.7, we can observe the algorithm function. The Car State node subscribes the sonars and range finders data to further decide the vehicle's state.

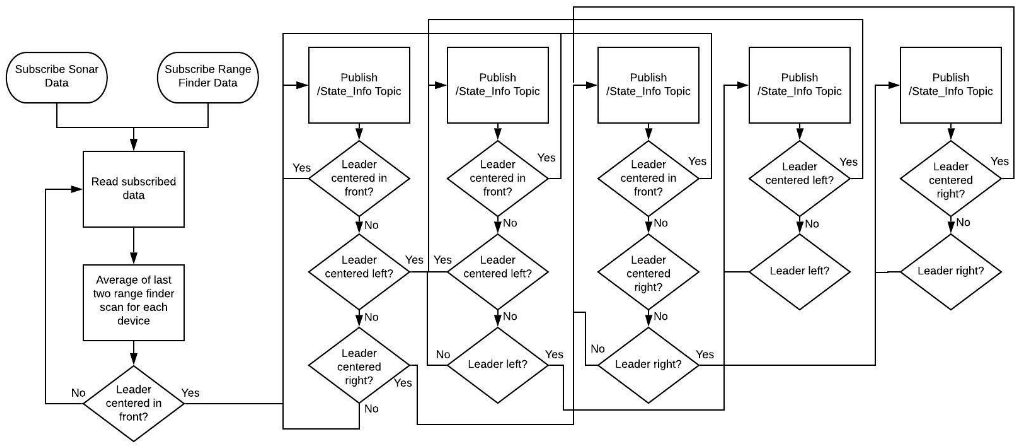

Subscribe Sonar Data
Subscribe Range Finder Data
Read subscribed data
Average of last two range finder scan for each device
Leader centered in front?
No
Yes
Publish /State_Info Topic
Leader centered in front?
Leader centered left?
Leader centered right?
Leader left?
Leader right?
Yes
No

**Figure 5.7:** *Car State Flowchart without camera*

If the two center range finders are detecting the car, within a maximum threshold between them and the center sonar, the steering applied will be 0°, denominated center state. If one of the center range finders is detecting the car and the other one is not, there will a verification of a certain threshold with the sonar in the center and if it's validated, there is a compensation of 10º to that side in the steering. These states are called center right and center left.

In the case that only the peripherals range finders are detecting the car, the steering is 15º to that direction after the same validation is done with the respective peripheral sonar and those are the right state and left state. In resume there are five states that we must consider in this case. For safety reason, the platoon can only start in the center state and the shift between states can only occur to an adjacent one, so that it won't confuse the car with other objects that might be in the track or even the walls. In Figure 5.8, we can verify the positioning of the range finders and sonars, for an easier perception of the algorithm applied.

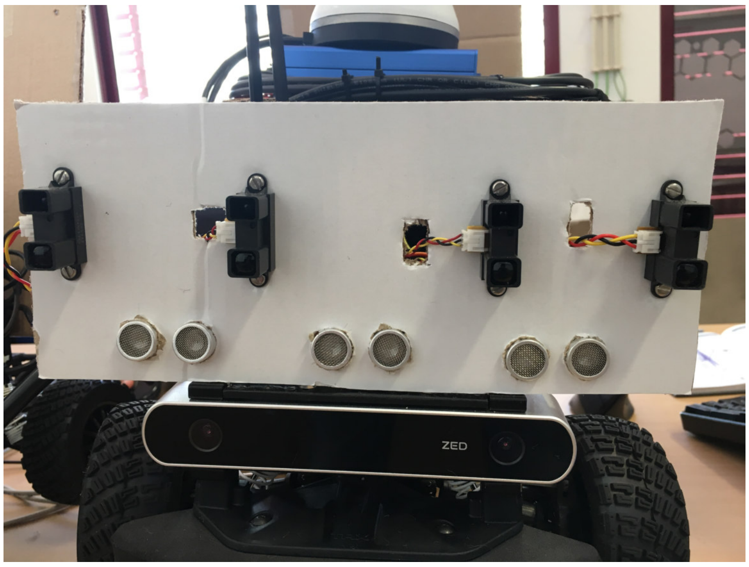


**Figure 5.8:** *Sensor Control Board*

### 5.1.2 Longitudinal Control

To control the vehicles speed, the car depends on the previous states. According to the sonar that was used to validate the steering, that same distance is utilized to calculate an error between the actual distance to the car in front and a safe distance established. This error is then applied in a PD control, that indicates if the car must accelerate or decelerate to maintain that same distance.

## 5.2 Camera-based Platooning

After the implementation of our baseline described earlier on, we proceeded to improve that scenario. For this purpose, the ZED camera was used to provide an image for further processing and object detection. In this specific case, it was used Yolo v3, a neural network capable of detecting certain objects in an image, producing a surrounding box, and Darknet, a framework used to train a neural network. These were chosen because there is support for the ZED camera, making it simpler to work with. A pre-trained Yolo v3 algorithm that can detect up to 80 different objects was selected, from which was chosen a stop sign to perform the detection of the car. This way, there was no time wasted in training a specific image for this use case. Therefore, there were stop signs attached at the rear end of the first and second car, as in 5.9.

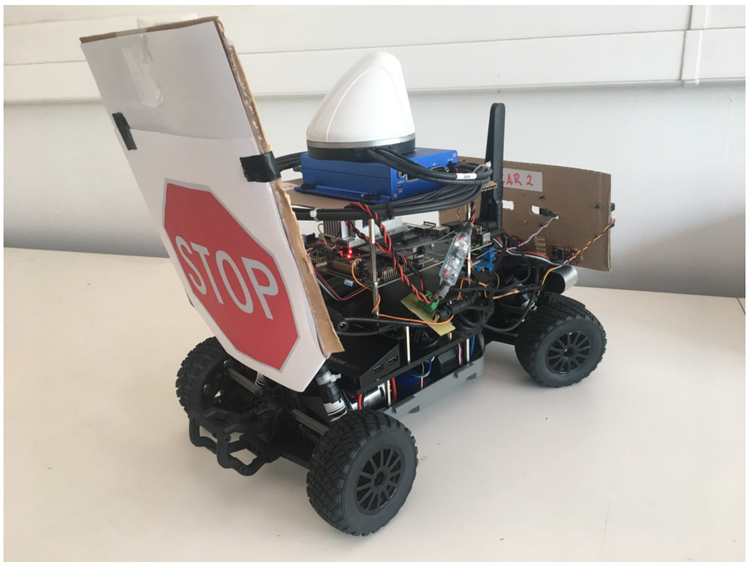


**Figure 5.9:** *Stop Sign*

This camera-based platoon has a similar way of functioning as our baseline. It also consists in a set of nodes, but some will have a slight change in their message comparing to the previous role, while others are completely new. These nodes are represented in Figure 5.10. /Sonars_Ranging is still an independent node, however there are two new independent nodes. These nodes are /CAMERA and /imu_node. The first one is responsible for detecting the stop sign, calculating it's x coordinate in the image and publishing it to the /X topic. The second node takes the heading values from the IMU and publishes them to a topic named /My_IMU.

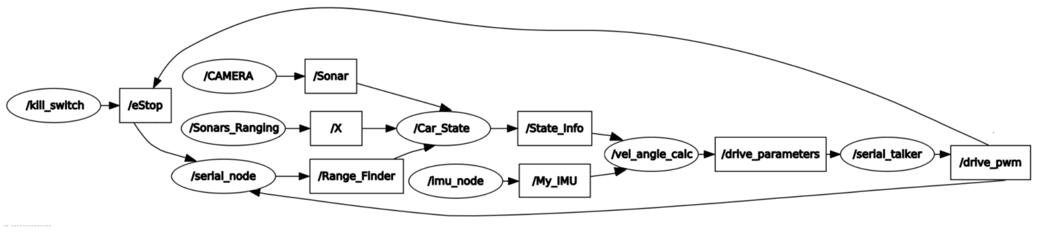


**Figure 5.10:** *Camera-based Rqt Graph*

As before, there is the /Car_State node, except that it, in this algorithm, subscribes three topics, /Sonar, /X and /Range_Finder. This last topic is already known. The /Car_State node determines the car's status, based on the new information that is being subscribed. The car's status is assessed differently and is explained in sub chapter 5.2.1. It publishes the topic /State_Info, containing more fields than before, with the information of the x coordinate, the distance to the car in front, it's status and the distance in the previous iteration. At that point, the node /vel_angle_calc subscribes this topic and the newly added topic /My_IMU. The /vel_angle_calc functions similar as before, with the addition of the camera detection. The camera detection influences the vehicle's state and this node applies a different method to determine the steering and speed, as we can observe in Figure 5.11. The /My_IMU topic is only used at the moment for results and conclusions. The other nodes functions remain the same.

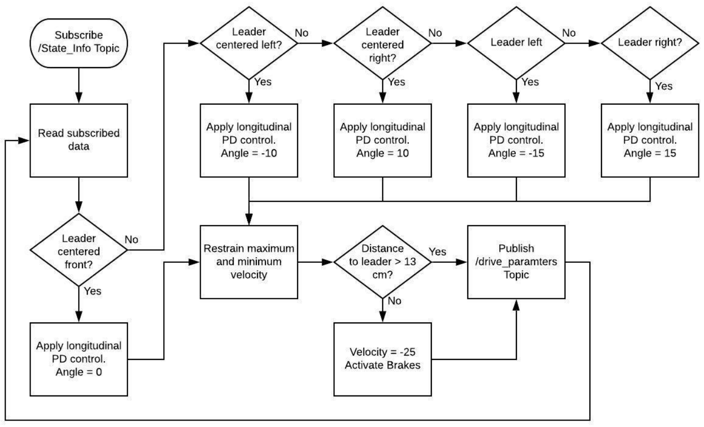

Subscribe /State_Info Topic
Read subscribed data
Leader centered front?
No
Yes
Apply longitudinal PD control. Angle = 0
Leader centered left?
No
Yes
Leader centered right?
No
Yes
Leader left
No
Yes
Leader right?
Yes
Apply longitudinal PD control. Angle = -10
Apply longitudinal PD control. Angle = 10
Apply longitudinal PD control. Angle = -15
Apply longitudinal PD control. Angle = 15
Restrain maximum and minimum velocity
Distance to leader > 13 cm?
Yes
No
Publish /drive_paramters Topic
Velocity = -25 Activate Brakes

**Figure 5.11:** */vel_angle_calc Flowchart*

### 5.2.1 Lateral Control

The original code [37] was adapted to only consider stop signs and to work with ROS. In ROS, the /CAMERA node publishes it's medium width point through a topic called /X. This value is also displayed on the screen of the detection with the respective rectangle surrounding the stop sign, as shown in Figure 5.12.

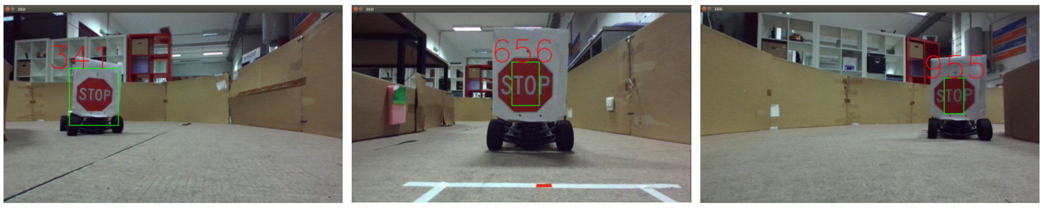


**Figure 5.12:** *Stop Sign Detection Scenarios*

The width value will then be subscribed through ROS by /Car_State node that decides, based on the range finders, the sonars and this value, the vehicle's state. This node's flowchart is demonstrated in Figure 5.13.

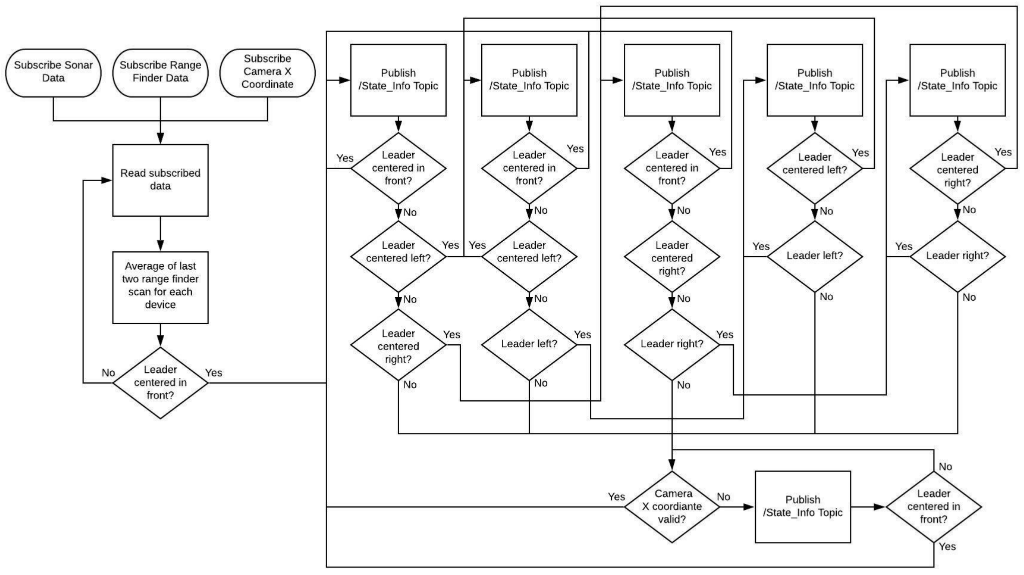

Subscribe Sonar Data
Subscribe Range Finder Data
Subscribe Camera X Coordinate
Read subscribed data
Average of last two range finder scan for each device
Leader centered in front?
No
Yes
Publish /State_Info Topic
Leader centered in front?
Leader centered left?
Leader centered right?
Leader left?
Leader right?
Camera X coordiante valid?
Yes
No

**Figure 5.13:** *Car State Flowchart*

The vehicle's state is determined a slightly different than in our baseline. The previous deterministic algorithm still applies if the follower is less than 30 cm apart from the leader, due to the camera control in this conditions not being reliable. However, if this distance is surpassed, the vehicle's state is named "Camera Mode", whereas it will publish the x coordinate of the stop sign detected, as well as, the actual distance to the leader, the distance in the previous iteration and the car's state. If the vehicle publishes the vehicle's state as "Camera Mode", the node /vel_angle_calc applies a PD control based on the deviation of the midpoint of the image, that represents the midpoint of the car, calculating the steering angle. The steering angle value is then published to /State_Info topic. Subsequently, there is a node /serial_talker that subscribes the previous topic and converts this steering angle to a number that will be interpreted as a PWM and is published to a topic, called /drive_parameters. Teensy subscribes this topic and sends the PWM to the servo.

### 5.2.2 Longitudinal Control

The deviation value calculated for the lateral control is also used to select which sonar is used to perform a longitudinal PD control, that's simultaneously based on a reference distance to the front car, as in our baseline, and on it's acceleration, that is calculated through the distance to the front vehicle in consecutive iterations. This means that if the distance to the leader has increased, the follower is driving slower than the front car and must accelerate. There is also the opposite case, where the distance is decreasing and so the follower starts decelerating or stops, if it's too close.

## 5.3 Results

In this section there will be a presentation of the results obtained for the lateral and longitudinal control applied to the platoon.

### 5.3.1 Lateral Control Results

These results are obtained at constant velocities so that it is possible to observe it's impact in the lateral control. The results are achieved through the comparison of the heading of the vehicles when passing the same coordinates. These coordinates are calculated knowing the vehicles speed in meters per second and the time that has been passed between iterations. This is done to have a correct comparison between the metrics that we are using so we can calculate the deviation to the expected result.

In the first platooning algorithm, our baseline explained in section 5.1, we realized tests at a constant speed of 0.8 m/s, where the path is an approximation of an oval track. This is shown in Figure 5.14, as well as the car positioning when starting an experiment.

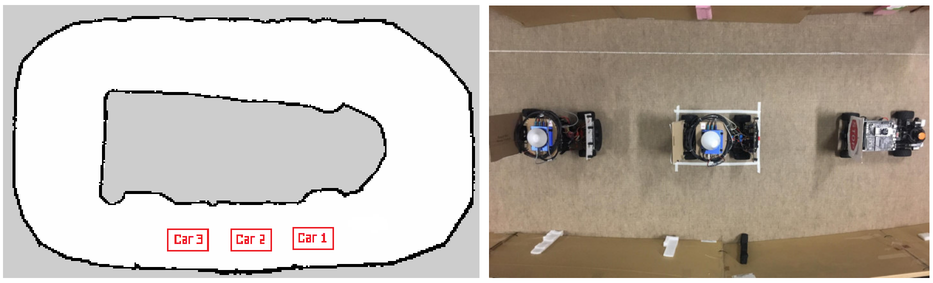


**Figure 5.14:** *Track and Car Positioning*

In Figure 5.15, we can observe the real time headings of each member of the platoon while performing a lap around the oval track demonstrated before. The moment where there is a sudden drop of the heading, represents the instant in which the vehicle's are turning and surpass southwest to southeast coordinates.

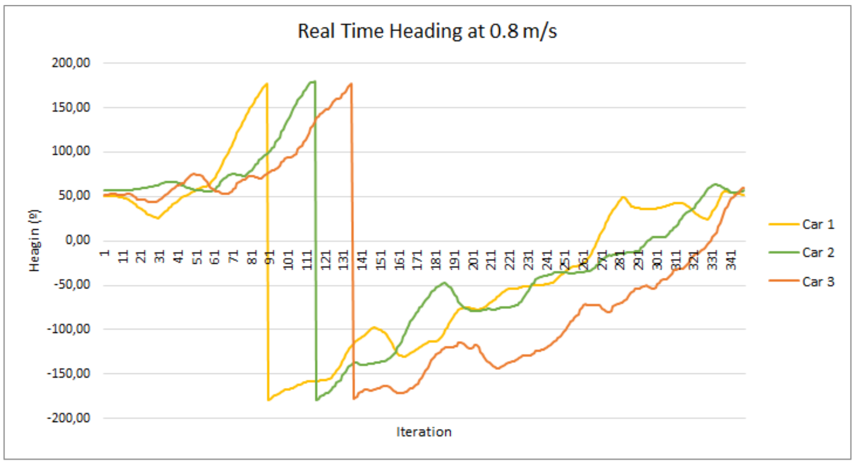


**Figure 5.15:** *Real Time Headings at 0.8 m/s without camera detection*

As our purpose is to be able to compare each vehicle path throughout a full lap around the track and afterward held a comparison between our algorithms, Figure 5.16 represents each car's heading at the same positional coordinates on the track.

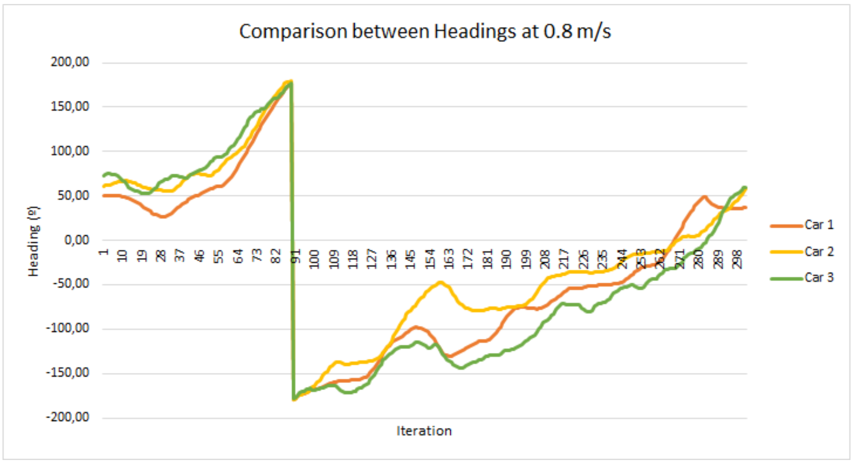


**Figure 5.16:** *Comparison between Headings at 0.8 m/s without camera detection*

We can observe that they address the desired path alike, despite having some oscillations. The second vehicle's oscillations tend to be on the right side of the desired path, having an average of -16.65º heading deviation. This is a slight deviation, however the maximum and minimum deviation are elevated. These results are 38.37º and -79.73º.

However, the third vehicle has an average oscillation of 21.28º, when comparing to the second vehicle. This demonstrates that the third car tends to be on the left side of the desired path. It's maximum and minimum deviation are 82.87º and -22.18º, elevated as the second vehicle. These values are observable in Figure 5.17.

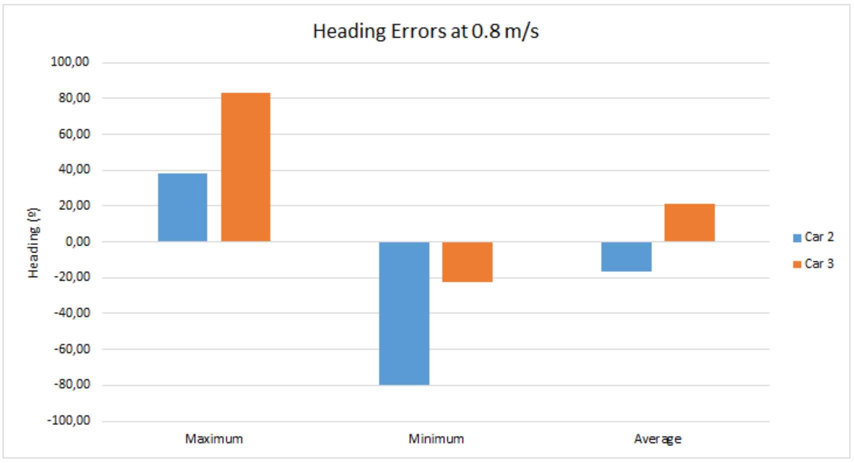


**Figure 5.17:** *Heading Errors at 0.8 m/s without camera detection*

For the first tests with the second platooning algorithm, described in section 5.2, the speed of the leader is 0.8 m/s.

In Figure 5.18, it is observable that the car start the track headed to 50° from north and end the track with an approximate heading, demonstrating a full lap of the track. As explained before, the moment there is a drop in the heading represents that the vehicles are pointing to south.

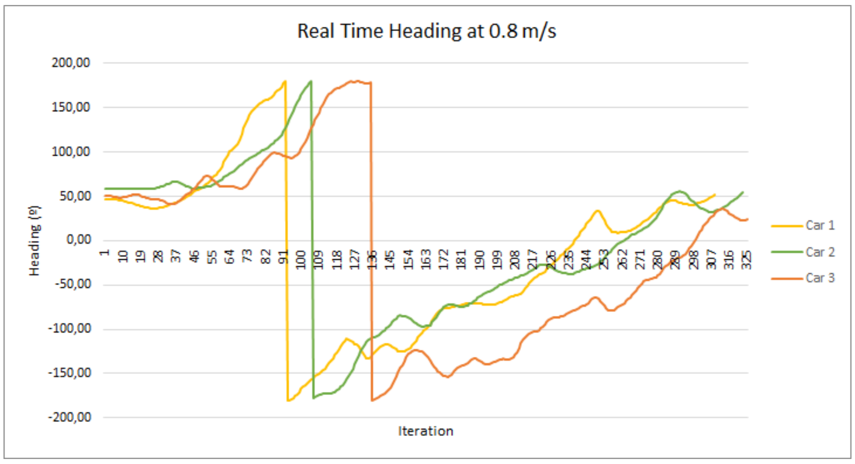


**Figure 5.18:** *Real Time Headings at 0.8 m/s*

As a result of the previous graphic, Figure 5.18, we created a new one, to analyze the vehicle's lateral control, that is presented in Figure 5.19.

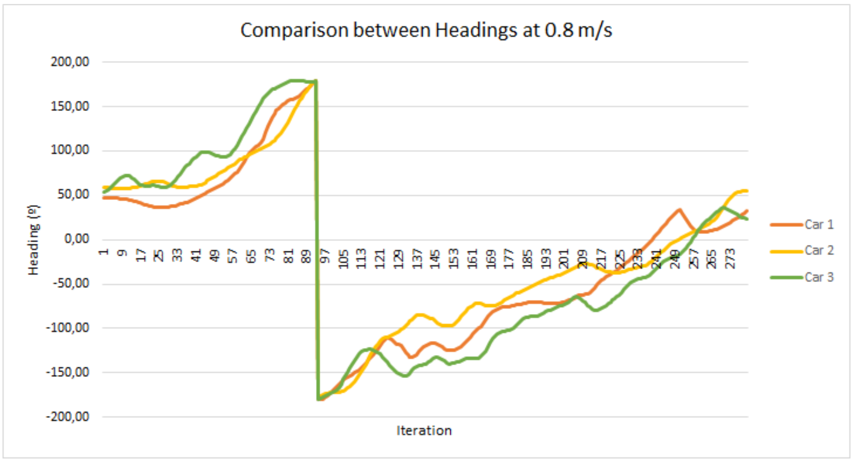


**Figure 5.19:** *Comparison between Headings at 0.8 m/s*

As stated Figure 5.20, the maximum and minimum error between the two first vehicle's is respectively, 33.47° and -43.97°. Although these errors, the average deviation between these two is -9.22°, that represents a satisfying result. The third platoon element, presented errors of 62.15º and -59.54º, as maximum and minimum. Despite these elevated errors, it's average remains at 10.20º, being able to complete the platoon with success.

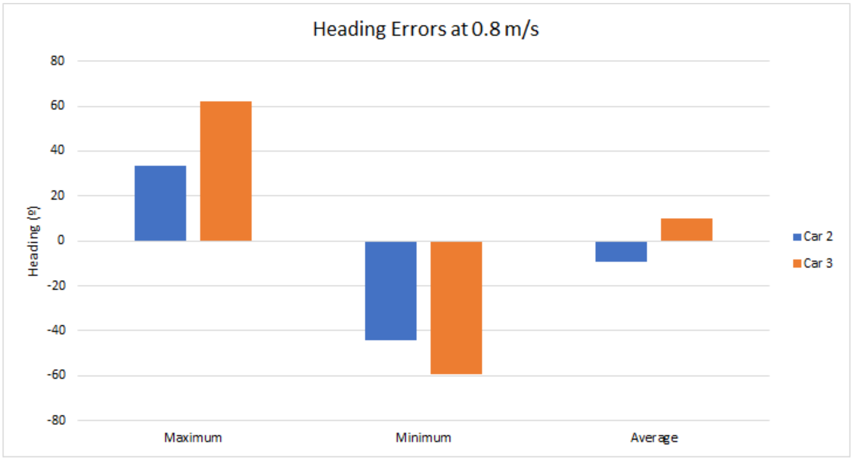


**Figure 5.20:** *Heading Errors at 0.8 m/s*

As the objective is to evaluate how the vehicle perform on different velocities, there were performed another tests at a higher speed as 0.9 m/s. In Figure 5.21, we can observe the heading of each element of the platoon during one lap, as before.

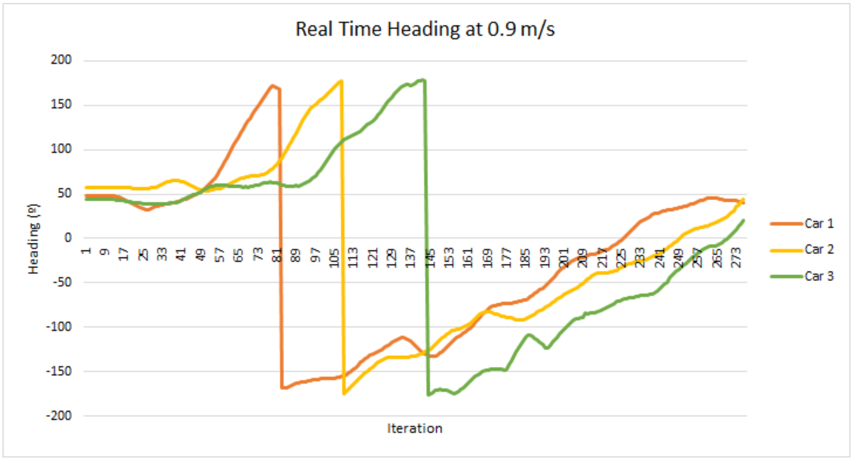


**Figure 5.21:** *Real Time Headings at 0.9 m/s*

By comparing the heading of the vehicle in Figure 5.22, it is observable that they have a similar path.

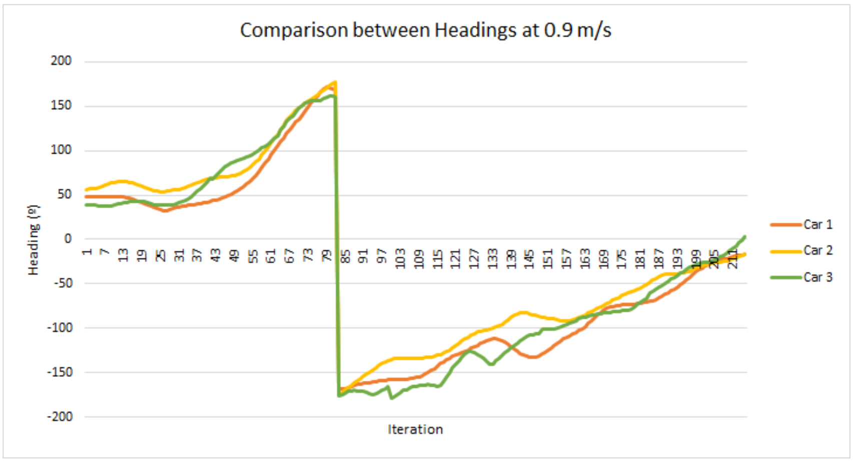


**Figure 5.22:** *Comparison between Headings at 0.9 m/s*

When analyzing their data, the second vehicle has an average deviation of -15.58º to the leader, while having a maximum error of 7.78º and a minimum error of -49.67º. The last element of the platoon had an average of 12.47º. This value is symmetrically identical to the second vehicle's value, meaning that the deviation is to an opposite side. The third vehicle's maximum heading is 44.39º, while it's minimum is -19.39º, as plotted in Figure 5.23.

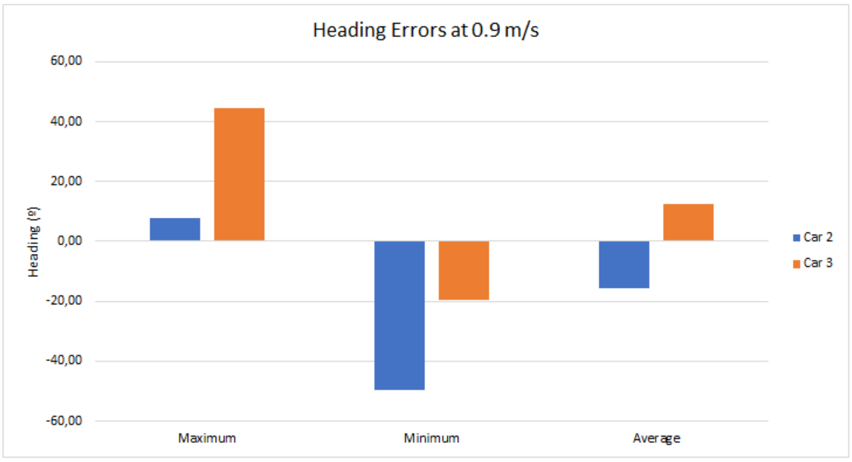


**Figure 5.23:** *Heading Errors at 0.9 m/s*

Comparing to the experiment in which the speed was lower, it is assumable that the impact on the heading is higher, having a higher average deviation.

The heading values at 1.0 m/s are shown in Figure 5.24. This information is calculated in real time, for further comparison.

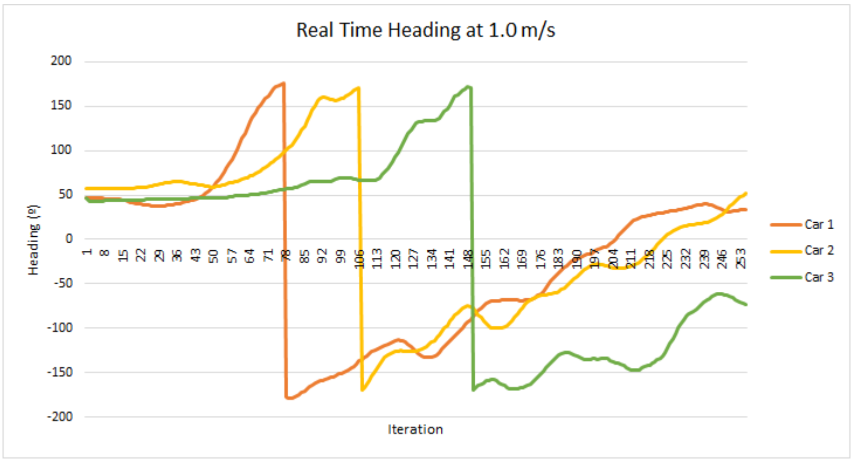


**Figure 5.24:** *Real Time Headings at 1.0 m/s*

To have a better method to compare, Figure 5.25 demonstrates the heading values from each vehicle in the same position.

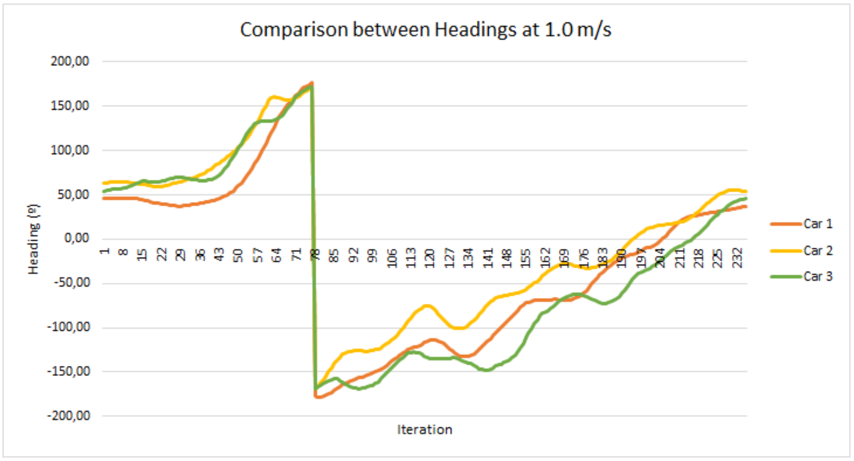


**Figure 5.25:** *Comparison between Headings at 1.0 m/s*

The second vehicle has an average deviation of -24.18º, whereas it's maximum heading is 6.22º and minimum is -44.37º. The third vehicle has an average of 27.67º. The maximum and minimum heading values of this element are 77.26 and -6.13º. These results are demonstrated in Figure 5.26.

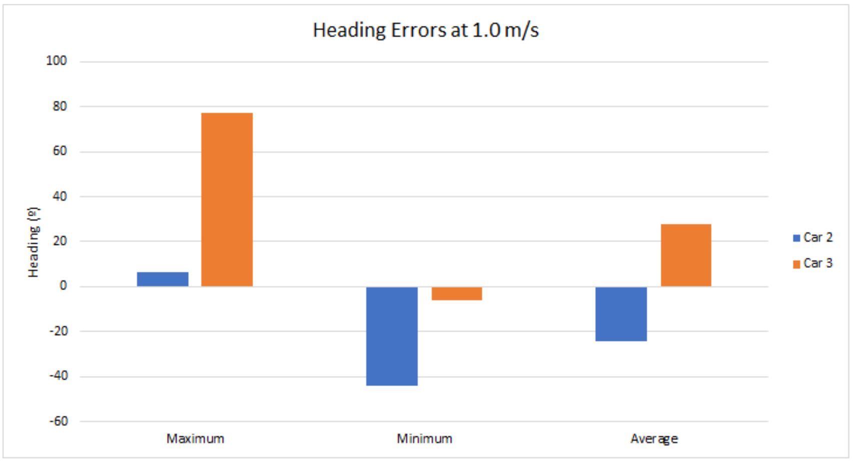


**Figure 5.26:** *Heading Errors at 1.0 m/s*

### 5.3.2 Longitudinal Control Results

The values used to validate our longitudinal control are the nearest and furthest distance retrieved in real time, the medium distance that the followers keep from the vehicle in front and the respective deviation error.

For our first longitudinal algorithm, explained in 5.1.2, the test was done with leader's speed constant at 0.8 m/s to observe the followers reaction, analyzing the values described before. In figure 5.27, the orange line represents the real time distance from the second vehicle to the leader, while the blue line represents the third vehicle. These results were obtained from an average of three experiments.

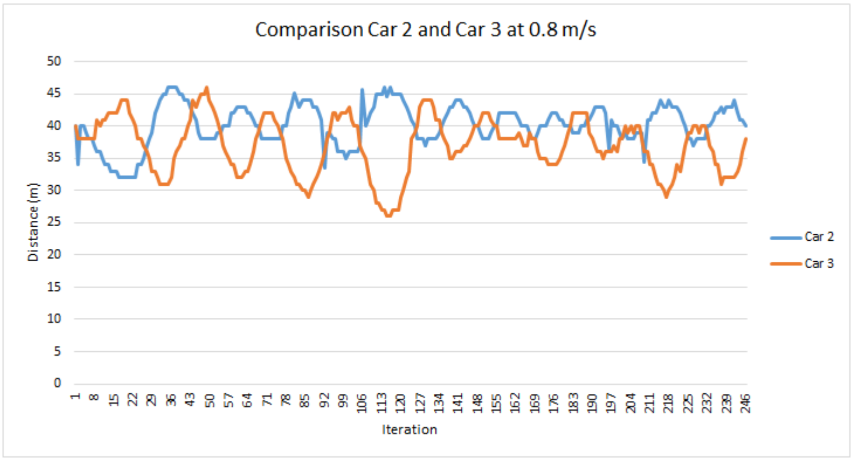


**Figure 5.27:** *Comparison between Distances from Car 2 and Car 3 at 0.8 m/s without camera detection*

The goal is to have a constant safety distance of 40 cm and, by evaluating the results, we can observe some oscillations around that value. The average deviation results in an error of 0.49%, that represents 40.19 cm, while it's maximum and minimum error translate to 15% and -20% respectively. These two values stand for 46 cm and 32 cm.

The distance values of the third member indicate a higher fluctuation around the desired safety distance, resulting in a maximum error of 15%, translating into 46 cm, and a minimum error of -35%, that represents 26 cm. As the distance tends to be under the desired value, we can expect a lower average distance, that is translated to an error of -7.56%, representing 36.98 cm. The obtained errors are plotted in Figure 5.28 for a better comprehension.

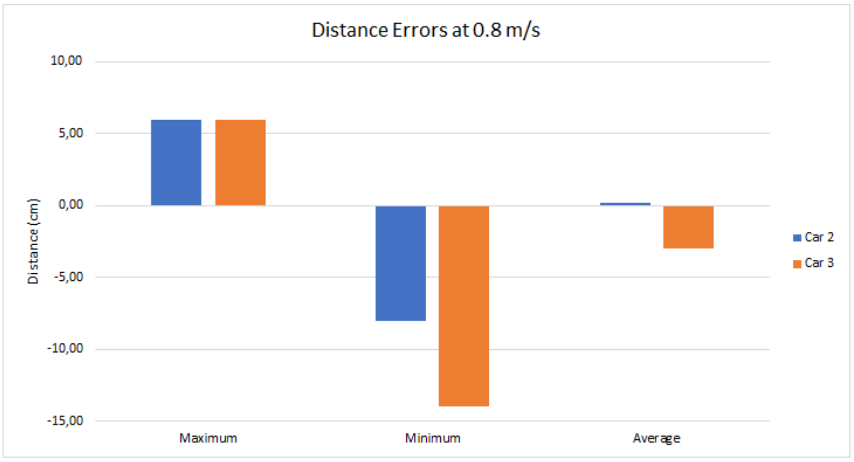


**Figure 5.28:** *Distance Errors at 0.8 m/s without camera detection*

We can observe a real time comparison between both safety distance. It is possible to constate that they have symmetrical reactions. This is due to the fact that whenever the second vehicle starts falling behind, it accelerates coming closer to the leader while the third element, that is currently close, recedes because he's still maintaining it's speed.

For the second algorithm, described in 5.2.2, the tests were performed three times, for each trial. The difference between trials is the constant speed of the leader that is increasing. It was able to test up to three trials with the following velocities: 0.8 m/s, 0.9 m/s and 1.0 m/s.

In Figure 5.29, it is possible to observe the followers response to the leader's constant speed of 0.8 m/s. The second vehicle is displayed in orange, while the third element in blue.

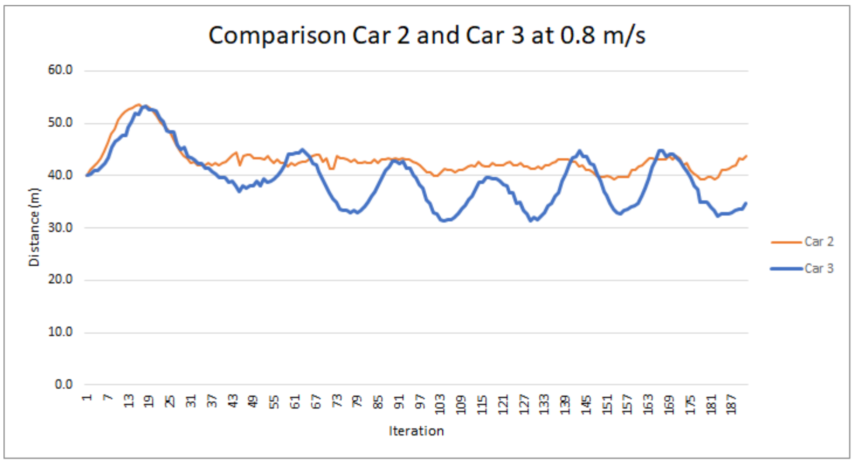


**Figure 5.29:** *Comparison between Distances from Car 2 and Car 3 at 0.8 m/s*

It is observable that in the first acceleration until the leader reaches it's speed, the follower is not able to keep the distance of 40 cm as supposed. The follower's maximum distance to the front car is 57 cm, that results in an error of 42.5%, while it's minimum is 39 cm, representing an error of -2.50%. Regarding the previous errors, the second car is able to follow the leader at a 43,03 cm average distance with an error of 7.58%.

These tests also contemplated the distance of the third car to the car in front. This car has a similar approach as the second vehicle, whereas in the initial acceleration, it also can't keep the distance.

However, it's maximum distance to the front vehicle is lower then in the first case, as this one is 54 cm, with the corresponding error of 35%.On the other hand, it's minimum distance is 31 cm, an error of -22.5%. As the second vehicle has it's oscillations, the third has more difficulty in maintaining the speed constant, resulting in aggravated oscillations, as observable. Despite this fact, the car averages a distance of 39,28 cm that represents an error of 1.8%. The distance errors obtained from both vehicles to the car in front are demonstrated in Figure 5.30.

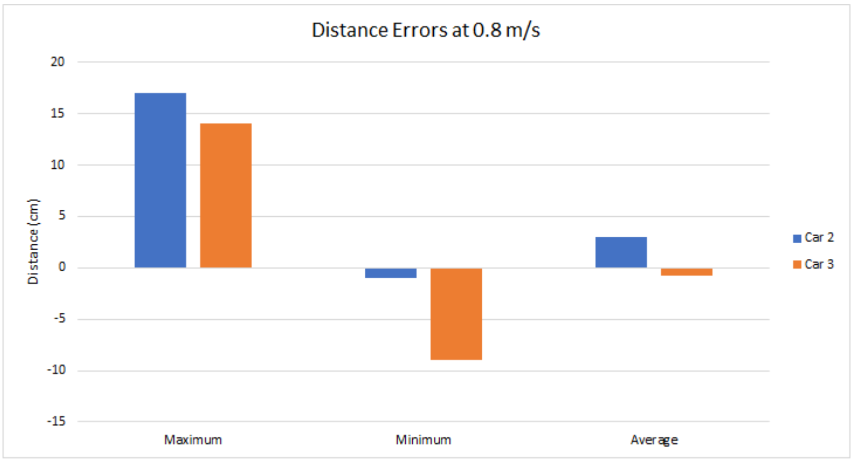


**Figure 5.30:** *Distance Errors at 0.8 m/s*

Following the previous experiment, the next test was done with a higher constant speed of 0.9 m/s, so that it is possible to analyze differences in the platoon's longitudinal control, related to the speed of the leader.

In Figure 5.31 is presented the distance of the second vehicle to the leader at 0.9 m/s in orange and the last element of the platoon in blue.

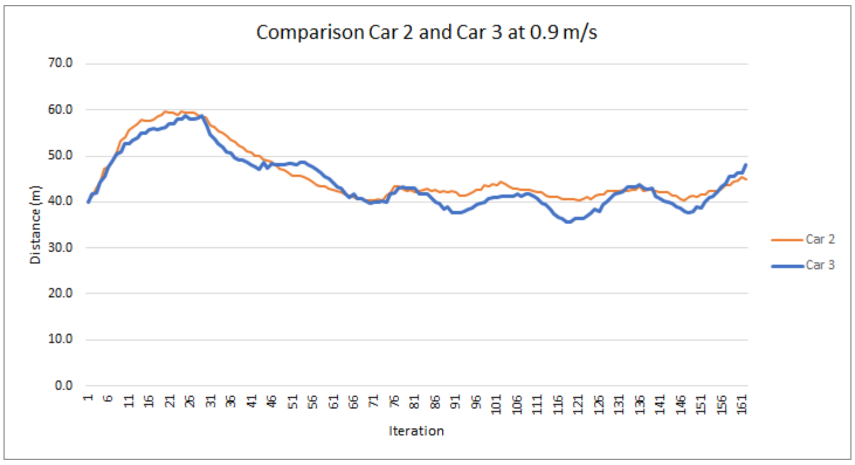


**Figure 5.31:** *Comparison between Distances from Car 2 and Car 3 at 0.9 m/s*

As before, there is an overshoot at the beginning that reaches 60 cm, resulting in an error of 50%. Yet the car is able to stabilize and averages 45.72 cm to the front car, translated into an error of 14.3%. Also, it's minimum distance to the front car is 40 cm.

The distance of the third car in this scenario is similar to the previous test at a lower speed. Firstly, in the acceleration, it is not able to accompany the second vehicle and the maximum distance to the vehicle in front is 59 cm, that represents an error of 47.5%. It also has more oscillations than the second car, but manages to have a steady state, averaging 44.05 cm. This metric is traduced to an error of 10.13%. On the other side, it's minimum distance is 36 cm, that means an error of 10%.

Both vehicle's distance errors are displayed in Figure 5.32, for a improved understanding of the metrics.

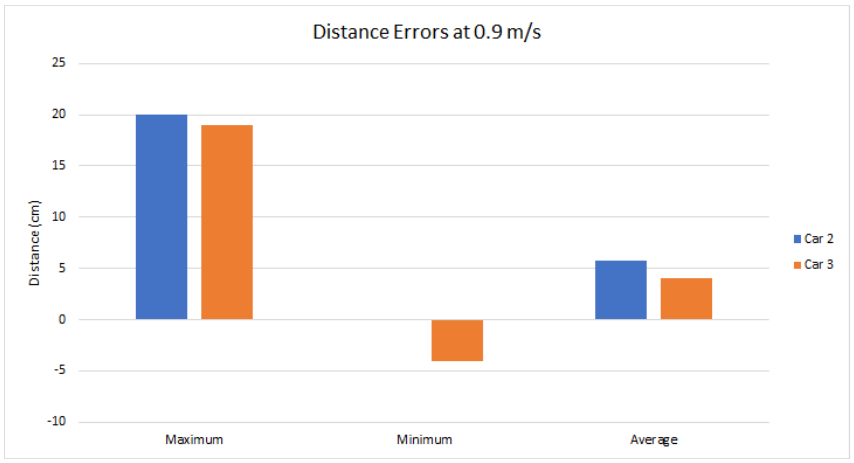


**Figure 5.32:** *Distance Errors at 0.9 m/s*

In the last test, Figure 5.33, the speed was increased up to 1.0 m/s. As expected, the second vehicle has a substantial overshoot, reaching a distance of 74 cm, that corresponds to an error of 85%. It's minimum distance is 40 cm at the beginning, while further on, the minimum is 43 cm. The average distance has an error of 36.6%, that corresponds to 54.64 cm.

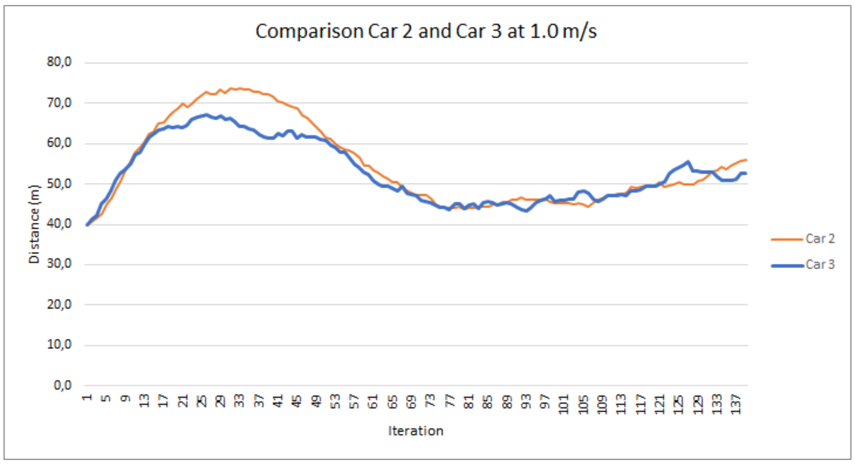


**Figure 5.33:** *Comparison between Distances from Car 2 and Car 3 at 1.0 m/s*

Despite the second vehicle having a high initial overshoot, the third elements overshoot isn't so elevated, only reaching 67 cm, the equivalent to an error of 67.5%, as shown in blue. It's minimum distance, just as the second vehicle is 40 cm at the initial state, but throughout the platoon test, it is 43 cm. The average distance has an error of 32.43%, that means the distance is 52.97 cm. The error obtained through the distance metric from each car are displayed in Figure 5.34.

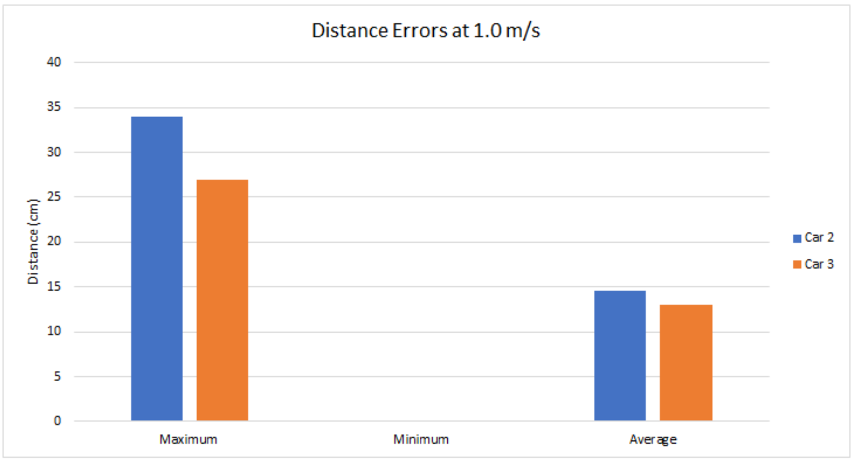
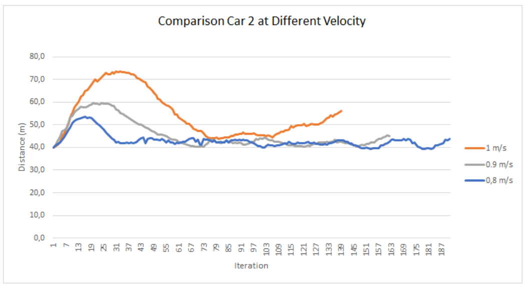


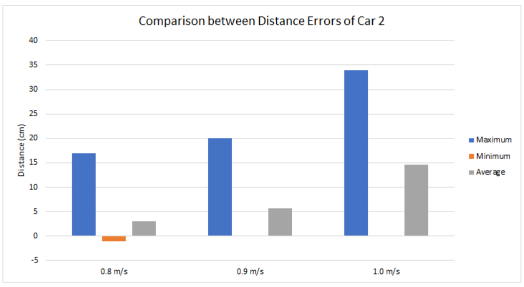
**Figure 5.34:** *Distance Errors at 1.0 m/s*

It is possible to observe the differences in the overshoot mentioned before and the further stability of the platoon that has a tendency of rising the distances between vehicles at this speed.

In Figure 5.35, it is evident the increase of the initial overshoot while the speed also increments. The settle time until there is stability increases likewise. It is also possible to verify an increase in the average distance to the car in front, as well as the maximum distance. Although these variations, the car still manages to be quite stable at all velocities. The reason that there are fewer iterations in the tests where the speed is higher is the fact that the distance is always the same, so the vehicles take less time to travel it.



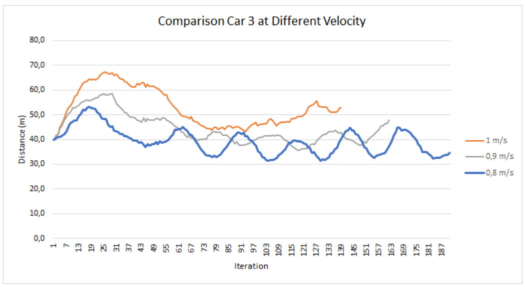
**(a)** *Comparison between Distances of Car 2 at Different Velocities*



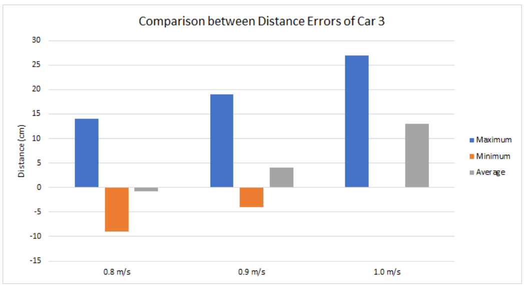
**(b)** *Comparison between Distance Errors of Car 2 at Different Velocities*

**Figure 5.35:** *Comparison of Car 2 metrics*

In Figure 5.36, it is observable that the overshoot and settle time to stability are affected the same way as the second car. However, the third car presents less stability at lower velocities. It also demonstrates the escalation of the average distance to the front vehicle at higher velocities, just as the minimum and maximum errors.



**(a)** *Comparison between Distances of Car 3 at Different Velocities*



**(b)** *Comparison between Distance Errors of Car 3 at Different Velocities*

**Figure 5.36:** *Comparison of Car 3 metrics*

From these results, additionally, it was concluded that the second vehicle has always a superior overshoot than the third vehicle, due to the fact that the acceleration of the first car is higher then the second one. This can be justified, since the first car aims directly to the constant speed of each test, while the second and third car are controlled by a PD control.

### 5.3.3 Camera Detection Result

We felt the need to demonstrate the quality of our detection with the ZED camera while accelerating the leader. With this, we assured a correlation between the quality of the detection presenting the confidence of itself and the distance to the leader. Throughout three experiments, this test is done with the follower stopped. We also made these tests in different lighting conditions. In a first case scenario, we use natural lighting during the day, while in the second experiment we use artificial lighting whilst night.

In Figure 5.37, where the test was performed with natural lighting, we can observe that, as expected, while the distance to the leader is increasing, the confidence of the detection is diminishing. At a approximate distance of 92 cm, there is a sudden drop of the confidence, making the detection unstable. At approximately 100 cm, the confidence drops below 50%. Therefore, it is not advisable working at higher distances than this breaking point. Despite this, the stop signal detection can work up to a maximum distance of 117 cm, in this condition.

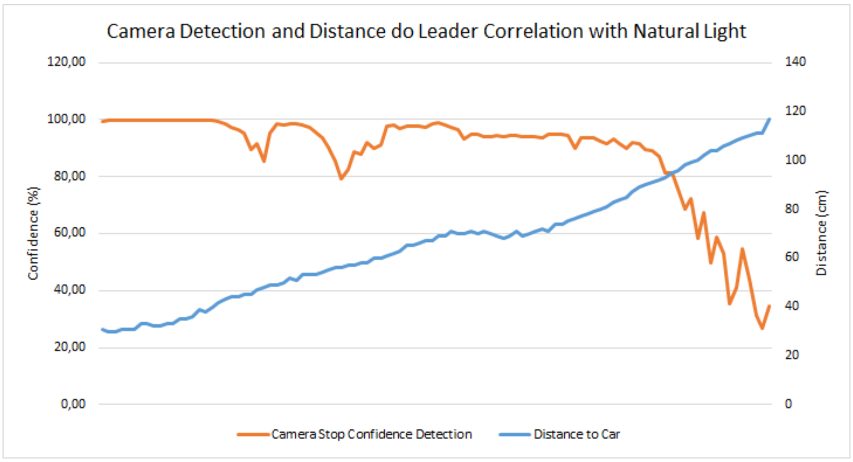


**Figure 5.37:** *Correlation between Camera Detection and Distance to Leader with Natural Lighting*

The Figure 5.38 demonstrates our experiment with artificial lighting at night. As in the first case, we see the same behaviour of inverted proportionality between the distance and the confidence. Although this resemblance, at 60 cm, the confidence starts to drop steeper, maintaining the confidence around 85% until 84 cm, where it start declining harshly and at 90 cm falls under 50% confidence. It reaches a maximum distance of 93 cm.

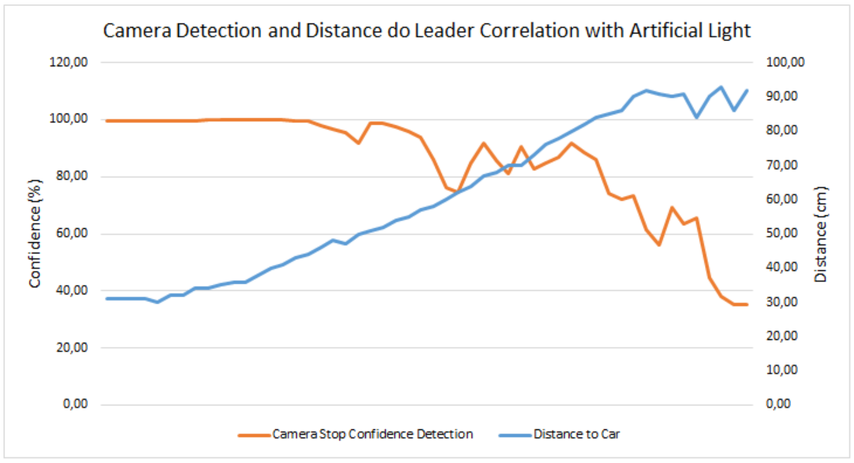


**Figure 5.38:** *Correlation between Camera Detection and Distance to Leader with Artificial Lighting*

# 6 OBU Cooperation Platooning

One of this Thesis objectives is to validate a safety mechanism on a platooning scenario. The testbeds were used as before, with cooperative ITS-G5 OBUs as additional components.

## 6.1 System

The presented system architecture, in Figure 6.1, refers to the fully implemented cooperative testbed, where the robots plan and run a platoon with V2V communications supported by the OBUs, and where the CLW and ETSI modules guarantee safety and security compliant with the defined standards. The OBU interface was achieved via a bridge, developed in partnership with GMV Skysoft, SA, through TCP sockets.

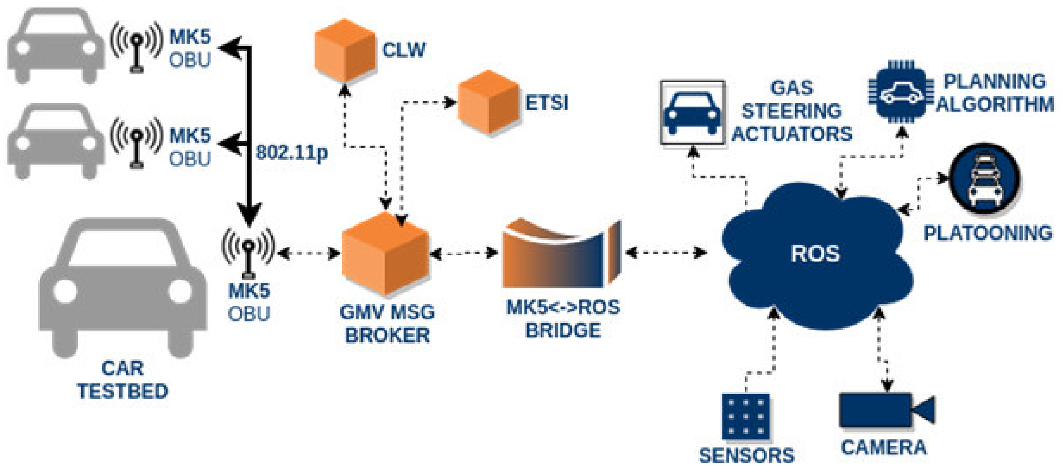


**Figure 6.1:** *Cooperative System Architecture*

On the OBUs side, the received messages are processed by the message broker and used by the other modules, as CLW and ETSI. On the robot side, the ROS-based system, running on the Jetson board, with all the above modules, uses the information received through the bridge in topics, carINFO and RXNetwork, shown in Figure 4.9, to get a good awareness of its surrounding environment and cars involved in the platoon. The ROS-based system is responsible to perceive the surroundings with the information received from the sensors and camera, plan according to the algorithm being applied and control the gas and steering actuators.

## 6.2 CLW integration

Control Loss Warning (CLW) systems, Figure 6.2, are designed to detect and alert about control loss situations. A CLW system has the ability to monitor systemś status, and trigger alerts if a control loss situation is detected. CLW mechanisms are particularly useful to monitor individual nodes that work together to create a complex autonomous system. As these system are appropriate to monitor the status of individual nodes, they can be applied to a platoon of vehicles travelling along a motorway. As alerts triggered by one node of the system may influence the behaviour of other nodes, these systems are identified as safety-critical ones. This use case aims at demonstrating how a safety assurance framework can be applied to automotive cooperative V2X-based systems.

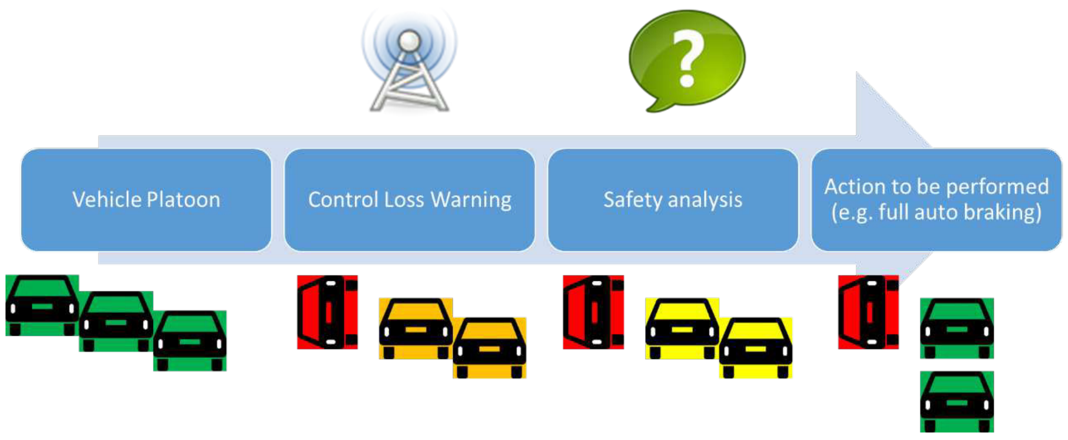


**Figure 6.2:** *CLW Overview*

Considering a scenario where a platoon of vehicles is travelling along a motorway, the CLW system should be able to detect a control loss warning situation, for example, the brakes on one of the cars fails. The CLW system detects the failure and sends a CLW alert to the other elements involved in the process. Those elements can be the rest of the cars in the platoon, police, emergency services, etc.

The CLW mechanism used in the platooning testbed is able to detect when a vehicle in a platoon loses control, and its platooning ability is thus affected. When this is detected all the vehicles on the platoon and the road infrastructure concessionary are notified of a control loss situation. If in any situation a CLW alert is activated, a safety analysis must take place before any action of other vehicles. Safety-assurance methods and tools, applicable to V2x systems, developed in the project, will be used in the demonstrator. These methods and tools include design and runtime mechanisms for safe V2X communications between cooperative vehicles, in order to ensure overall system safety.

In the platooning robotic testbed platform's experiments, we validated a CLW mechanism as Loss of Direction (LOD). The LOD is detected when the steering and the heading of the elements of the platoon are not within the desired parameters and triggers this safety mechanism, that results in the emergency breaking of the members of the platoon. In Figure 6.3, we can observe the robotic testbeds implemented with the OBU MK5, able to trigger a LOD mechanism if necessary.

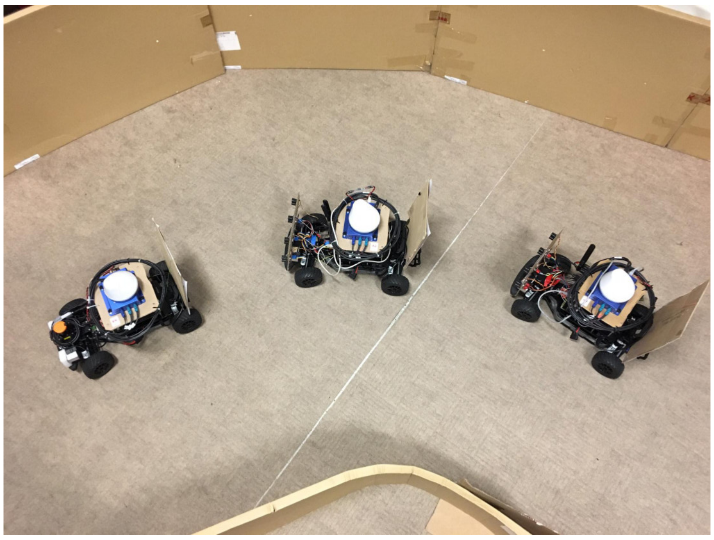


**Figure 6.3:** *Platooning with Safety Mechanism*

# 7 Conclusions and Future Work

## 7.1 Conclusions

This Thesis main objective was the creation and development of a low-cost, platooning robotic testbed that relied on the Robot Operating System. This was done by following the F1 tenth build, while adapting and improving their testbed. As this objective was achieved early on, there were assembled three examples with the hardware components that are described throughout this Thesis, in order to form a platoon.

Secondly, as platooning is an emerging concept and becoming a trend, we proposed ourselves to develop a stable platoon merely based on sonars and range finders as our data input. This platooning baseline was achieved, but it's only stable for velocities lower then 0.8 m/s.

In the next objective we proceeded to improve this baseline with an additional camera with object detection. This enhancement resulted in reaching higher velocities without compromising the vehicle's stability.

In our lateral results, we are clearly able to conclude that our second algorithm in an obvious improvement on our baseline as we can achieve lower average deviation error even at higher velocities when comparing to our baseline that travelled at a maximum of 0.8 m/s. Only at 1.0 m/s of our improved platoon, there are superior average error, however the maximum and minimum error are still lower in the enhanced algorithm.

The fact that our baseline platoon presents better results in our longitudinal control throughout the tests, doesn't indicate, in this specific case, that it is a superior platoon model. These results are all influenced by the initial delay that the second platoon model presents. This delay impacts hugely on the maximum distance that the vehicles achieve. Consequently, it impacts the average distance by increasing it, despite being far more stable than our baseline. Truth of the matter that the camera algorithm method works on higher velocities proves our previous affirmation.

We also decided to held a comparison of the image detection algorithm when exposed to different lighting. For this purpose, by analyzing the confidence in the stop signal detection and the distance to the leader, at natural and artificial lighting, we were able to conclude that the detection results better with natural light. This is expected as the ZED camera has a greater performance outdoor rather than indoor.

Lastly, the bridge between the OBU and Jetson was accomplished with success, whilst working with GMV Skysoft, SA. This was validated successfully demonstrating the CLW implementation over the specific use case of loss of direction.

## 7.2 Future Work

We are currently looking into the possibility to advance to a cooperative platooning control model, based on communications and sensory input from the vehicle's already existing detection devices with the purpose of achieving a better and more stable platoon than the developed for this Thesis.

The aim is to implement a fusion between communications, that receive data from the sensors and transmit through a platoon network, and the sensors aboard the vehicles. This means that our purpose is that each car has it's own control model, aside from receiving information about the other vehicles from the platoon and then decides his actions, based on this data. With this fusion, our objective is to have a more stable platoon, considering our lateral control and longitudinal control, and also to reach superior velocities.

Another point would be to increase the number of vehicles of the platoon. Also, we are examining the potential in applying our platoon knowledge acquired from ground vehicles to airborne platforms such as drones. Also drones pose significant and similar challenges in terms of cooperation and could benefit from safety mechanisms similar to the one presented in this Thesis.

# Bibliography

[1] “Various-driving-patterns-in-Highway-scenario.png (734Ã305),”, https://www.researchgate.net/profile/Dongyaotony_Jia/publication/273770053/figure/fig1/AS:614368203005967@1523488202397/Various-driving-patterns-in-Highway-scenario.png, (Accessed on 09/10/2019).

[2] “IEEE Xplore Full-Text PDF:,”, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8240910, (Accessed on 09/13/2019).

[3] D. Lu, Z. Li, D. Huang, X. Lu, Y. Deng, A. Chowdhary, and B. Li, “VC-bots: A Vehicular Cloud Computing Testbed with Mobile Robots,” In *Proceedings of the First International Workshop on Internet of Vehicles and Vehicles of Internet*, IoV-VoI ’16 pp. 31–36 (ACM, New York, NY, USA, 2016).

[4] “tra74054-4_1.jpg (1200Ã960),”, https://images.amain.com/images/large/tra/tra74054-4_1.jpg, (Accessed on 09/20/2019).

[5] “MK5 OBU - Cohda Wireless,”, https://cohdawireless.com/solutions/hardware/mk5-obu/, (Accessed on 09/10/2019).

[6] “NVIDIA Jetson TX2 J21 Header Pinout - JetsonHacks,”, https://www.jetsonhacks.com/nvidia-jetson-tx2-j21-header-pinout/, (Accessed on 09/18/2019).

[7] “SRF08 Ultra sonic range finder,”, https://www.robot-electronics.co.uk/htm/srf08tech.html, (Accessed on 09/17/2019).

[8] “Safecop â Safe Cooperating Cyber-Physical Systems using Wireless Communication,”, http://www.safecop.eu/, (Accessed on 10/09/2019).

[9] R. Smith, “Directive 2010/41/EU of the European Parliament and of the Council of 7 July 2010,” in *Core EU Legislation* (Macmillan Education UK, London, 2015), pp. 352–355.

[10] J. Wan, D. Zhang, S. Zhao, L. T. Yang, and J. Lloret, “Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions,” IEEE Communications Magazine **52,** 106–113 (2014).

[11] S. Gong and L. Du, “Cooperative platoon control for a mixed traffic flow including human drive vehicles and connected and autonomous vehicles,” Transportation Research Part B: Methodological **116,** 25–61 (2018).

[12] E. Larsson, G. Sennton, and J. Larson, "The vehicle platooning problem: Computational complexity and heuristics," Transportation Research Part C: Emerging Technologies **60,** 258–277 (2015).

[13] S. Tsugawa and S. Kato, "Energy ITS: another application of vehicular communications," IEEE Communications Magazine **48,** 120–126 (2010).

[14] Y. Zhang and G. Cao, "V-PADA: Vehicle-Platoon-Aware Data Access in VANETs," IEEE Transactions on Vehicular Technology **60,** 2326–2339 (2011).

[15] F. Dressler, F. Klingler, M. Segata, and R. L. Cigno, "Cooperative Driving and the Tactile Internet," Proceedings of the IEEE **107,** 436–446 (2019).

[16] D. Eckhoff, N. Sofra, and R. German, "A performance study of cooperative awareness in ETSI ITS G5 and IEEE WAVE," In *2013 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 196–200 (2013).

[17] European Telecommunications Standards Institute, "ETSI EN 302 637-2 V1.4.0," Technical Report No. V1.4.0, ETSI (2018) .

[18] C. Wang, S. Gong, A. Zhou, T. Li, and S. Peeta, "Cooperative adaptive cruise control for connected autonomous vehicles by factoring communication-related constraints," Transportation Research Part C: Emerging Technologies (2019).

[19] I. M. Delimpaltadakis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Decentralized Platooning With Obstacle Avoidance for Car-Like Vehicles With Limited Sensing," IEEE Robotics and Automation Letters **3,** 835–840 (2018).

[20] K. Tammi and V. HyvÃrinen, "Lateral and longitudinal control of bus platoon," In *2018 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles International Transportation Electrification Conference (ESARS-ITEC)*, pp. 1–6 (2018).

[21] H. Cao, S. Gangakhedkar, A. R. Ali, M. Gharba, and J. Eichinger, "A Testbed for Experimenting 5G-V2X Requiring Ultra Reliability and Low-Latency," In *WSA 2017; 21th International ITG Workshop on Smart Antennas*, pp. 1–4 (2017).

[22] "Traxxas Ford Fiesta ST Rally — RC Rally Car,", `https://traxxas.com/products/models/electric/ford-fiesta-st-rally`, (Accessed on 09/08/2019).

[23] "JetPack 3.3 Release Notes — NVIDIA Developer,", `https://developer.nvidia.com/embedded/jetpack-3_3`, (Accessed on 09/10/2019).

[24] "ROS/Concepts - ROS Wiki,", `http://wiki.ros.org/ROS/Concepts`, (Accessed on 09/10/2019).

[25] "Master - ROS Wiki,", http://wiki.ros.org/Master, (Accessed on 09/10/2019).

[26] "Nodes - ROS Wiki,", http://wiki.ros.org/Nodes, (Accessed on 09/10/2019).

[27] "Messages - ROS Wiki,", http://wiki.ros.org/Messages, (Accessed on 09/10/2019).

[28] "Topics - ROS Wiki,", http://wiki.ros.org/Topics, (Accessed on 09/10/2019).

[29] "Bags - ROS Wiki,", http://wiki.ros.org/Bags, (Accessed on 09/10/2019).

[30] "Build,", http://f1tenth.org/build.html, (Accessed on 10/09/2019).

[31] "kinetic/Installation/Ubuntu - ROS Wiki,", http://wiki.ros.org/kinetic/Installation/Ubuntu, (Accessed on 09/19/2019).

[32] "GitHub - mlab-upenn/f1tenthpublic,", https://github.com/mlab-upenn/f1tenthpublic/, (Accessed on 09/17/2019).

[33] "GitHub - jetsonhacks/installACMModule: Install the CDC ACM and USB to Serial Modules for the Jetson TX1 or Jetson TX2 Development Kit,", https://github.com/jetsonhacks/installACMModule, (Accessed on 09/17/2019).

[34] "Stereolabs - Capture the World in 3D,", https://www.stereolabs.com/, (Accessed on 09/17/2019).

[35] "GitHub - stereolabs/zed-python-api: Python API for the ZED SDK,", https://github.com/stereolabs/zed-python-api, (Accessed on 09/17/2019).

[36] "Ubuntu 16.04: How to install OpenCV - PyImageSearch,", https://www.pyimagesearch.com/2016/10/24/ubuntu-16-04-how-to-install-opencv/, (Accessed on 09/17/2019).

[37] "GitHub - stereolabs/zed-yolo: 3D Object detection using Yolo and the ZED in Python and C++,", https://github.com/stereolabs/zed-yolo, (Accessed on 07/05/2019).