Politecnico di Torino

Master’s Degree in MECHATRONIC ENGINEERING
(INGEGNERIA MECCATRONICA)
A.a. 2022/2023
Sessione di Laurea Dicembre 2023

Development of an open, customizable communication box for vehicular applications

Relatori:
Prof. Claudio CASETTI
Prof. Francesco RAVIGLIONE
Prof. Marco RAPELLI

Candidato:
Marta DUTTO

Summary

The objective of this master thesis was to integrate and configure a special Linux-based operating system for vehicular communications, namely OpenWrt-V2X 21.02.1, on customizable embedded boards (PC Engines APU2), equipped with Sierra Wireless MC7455 mPCIe LTE modules and dedicated Wi-Fi modules for IEEE 802.11p communication. The main objective was to integrate into OpenWrt-V2X the Vanetza framework, an open-source framework for vehicular communications, based on the ETSI European Standards. The main aim of the project was to configure the APU boards to act as vehicular On-Board Units (OBUs), able to send both CAMs (Cooperative Awareness Messages) and DENMs (Decentralized Environmental Messages), integrating data from a precise GNSS.

The thesis starts by analyzing the accident rate in Italy in 2022 and what could be the benefit given by the autonomous driving. An Intelligent Transportation System (ITS) is defined by ETSI as a "system that adds information and communications technology to transport infrastructures and vehicles in an effort to improve their safety, reliability, efficiency and quality".

This means that the actual benefit of deploying such technology is not only aiming to reduce accidents but it can also be useful to keep a low carbon footprint. Nowadays the management of emission has become a critical issue for most of the companies due to the climate change. The idea of using this kind of technology to fully exploit the resources and at the same time reduce the impact on the environment is a challenge and a duty to preserve the planet. It is also presented a description of road users and how they can interact with each other in order to ensure an efficient management of the traffic and a safer road. In order to have a complete and exhaustive understanding of the environment is possible to consider the communication between vehicles (V2V) and with the infrastructure (V2X).

The thesis also analyzes the European standard ETSI and the American standard IEEE, highlighting their strengths and history. The most important result is that the development by different institutions of vehicular communication technology has been done in the name of compatibility.

Indeed, the development of the ETSI standard is based on the IEEE guidelines to ensure that the research in this field could be consistent and that the evolution of the technologies could be useful all over the world. The thesis presents a brief description of the IEEE WAVE standards, and then focuses on the ETSI ITS-G5 standard, i.e., the one implemented by Vanetza. It is highlighted how the use of the messages CAM and DENM has been introduced and the security protocols that are already available to ensure truthful communication.

The main features of the OpenWrt operating system (on which OpenWrt-V2X is based), and the ones of Vanetza, are showcased.

OpenWrt is a Linux operating system targeting embedded devices. It relies on a fully writable filesystem with package management instead of using a single static firmware. This choice was made to allow full customization of the system allowing the installation of desired applications through packages that can be both found already compiled in a library or can be compiled by the user. This flexibility allows the user to find the perfect fit for each particular application.

This operating system has the characteristic of being package-based. This means that all of the applications can be installed only if they are correctly compiled and built in compatible packages.

The other main software used in this thesis is Vanetza, an open-source implementation of the ETSI C-ITS protocol suite designed to send vehicular messages such as CAM and DENM in order to enable a standardized communication between vehicles. This project was designed to operate on ITS-G5 channels using IEEE 802.11p, a vehicular version of "Wi-Fi". The thesis also presents the project NAP-Vanetza as an enhancement of the Vanetza main project.

In the fourth chapter I described the methodology followed to develop the thesis work. I started with the installation of the OpenWrt operating system inside the OBU taking into account all its hardware characteristics in order to correctly configure all of the needed packages.

Then I proceeded by flashing the OS image on the OBU. In addition, the LTE module has been configured to enable cellular connectivity from the OBU. This step was not straightforward, as it required the integration and configuration of proper packages into the OpenWrt-V2X OS image, to enable seamless and automatically managed connectivity.

The next step was to link the OBU public key to allow remote access connection.

Once the OBU was ready and functioning I proceeded with the study of the dependencies of Vanetza. This project depends on three main external libraries: Boost, Crypto++ and GeographicLib.

At this point I cross compiled all of the libraries with the appropriate toolchain to ensure compatibility with the OBU hardware characteristics. I then proceeded with the cross compilation of Vanetza libraries and applications, specifying the appropriate compiled files for the dependencies.

As stated in the third chapter, OpenWrt is a system based on packages. Thus, a dedicated Vanetza package, that can be easily installed, was created.

The creation of a Vanetza package faces several challenges, as its core libraries and dependencies do not depend only from each other but also on other external libraries. I created a manifest file for the Vanetza package where I specified the characteristics such as name and version of the package. I also wrote a suitable Makefile in order to successfully compile and build a complete and working package of the Vanetza application.

Once the package was successfully compiled and build I proceeded to the installation of the package inside the OBU. The package with the Vanetza framework has been initially tested by means of the "socktap" example.

The SOCKTAP tests showed that CAMs were correctly sent every second and read by a second On-Board Unit, connected to the first via IEEE 802.11p.

The main result of this thesis is given by the successful creation of the vanetza pacakge that was correctly implemented on the OBU. Specifically, the OBU was able to properly send and receive CAM messages thanks to the Vanetza package.

The future development of this project will be the development of an integrated application sending and receiving both CAMs and DENMs (over LTE and IEEE 802.11p), interfacing with a GNSS receiver, thanks to Vanetza. This thesis thus provides an important starting point for the development of complex Vehicle-to-Everything (V2X) applications on customizable OBUs with the OpenWrt-V2X operating system.

Acknowledgements

Ringrazio di cuore i miei genitori per avermi sostenuto e supportato, soprattutto nei momenti più difficili, durante tutto il percorso di laurea e mia sorella Giorgia per il sostegno che mi ha offerto durante i momenti più belli e quelli più tristi di questi anni.

Ringrazio Nonna Lucia e Romina per tutto l'affetto e il sostegno che mi hanno dato in questi anni, per tutte le loro preghiere durante i miei esami e per tutto l'orgoglio che traspare dalle loro parole quando parliamo di questo importante traguardo. Ringrazio i nonni, le zie, gli zii e i cugini per il loro affetto e sostegno.

Ringrazio tutti i miei amici, in particolare i miei compagni di corso con cui ho condiviso buona parte delle sfide e degli esami in questi ultimi due anni. Un ringraziamento particolare va a Massimo ed Angelo che sono stati per me non solo amici ma anche confidenti nei momenti migliori e peggiori di questi due anni e che sono sempre stati presenti e d'aiuto in ogni situazione.

Ringrazio la community di OpenWrt, in particolare Henrik Ginstmark aka mrhaav e Raphael Riebl, che sono stati così gentili da darmi dei consigli per proseguire nei momenti più impegnativi della tesi.

Ringrazio il relatore e i correlatori.

" Ed ora dentista Croazia che fine avrai fatto
"Luci a San Siro" adesso te la canta qualcun altro
Ci hai insegnato che si vive solo di momenti
E che qualsiasi cosa passa se stringiamo i denti"
Dentista Croazia – Pinguini Tattici Nucleari

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Exploring the pattern of road accidents

In the news, almost everyday, we hear and read about fatal accidents involving vehicles that are caused mainly from driver's distraction or reckless driving style. An ISTAT (National Institute of Statistics of Italy) official document [1] that reports the 2022 Road accidents in Italy, shows that there were 165,889 road accidents in which are registered 3,159 deaths and 223,475 injured people. Sadly accident statistics for all of the road users, show an increase every year with the previous, 2022 it is not an exception with respect to 2021.
There are various group of users of the road and must be kept in mind that they are exposed differently to this dangerous situation. Road users are divided in:

- Occupants of cars: People that are inside the cars, both drivers and passengers.
- Truck occupants: People inside of trucks, drivers or passengers.
- Motorcyclist: People using any vehicle equipped with less than four wheels with a speed over 45km/h and a displacement of over 50 cc, drivers and passengers.
- Moped riders: people using a vehicle equipped with two wheels with a maximum speed of 45 km/h and with a displacement of less than 50 cc, drivers and passengers.
- Cyclist: People using electric bike (e-bike) or a bike without an electrical motor to give assistance, drivers and passengers.
- Pedestrians: People that are walking on the side of the road, on a sidewalk or using crosswalks.

People that are in cars have some sort of protection already integrated in the vehicle, such as airbags, while already motorcyclist do not have this kind of protection. The category at most risk is the pedestrian one, that have not any protection from the vehicle or any protecting devices such as a motorcycle helmet.

From ISTAT's data, it is possible to see all of the sanctioned behaviours and in particular, the most sanctioned behaviour in Italy is speeding. There are also high percentages of penalties imposed due to improper use of devices while driving and for driving under drugs and alcohol influence.
In the Figure 1.1 it is reported a bar graph that uses ISTAT's data to show the division of 2022 fatalities among road users. In particular, it is possible to compare 2022 data in blue with the respective data divided by users of the 2021 in red. It is possible to see a clear increase in all of the categories but the most significant one is the vehicle's occupants.

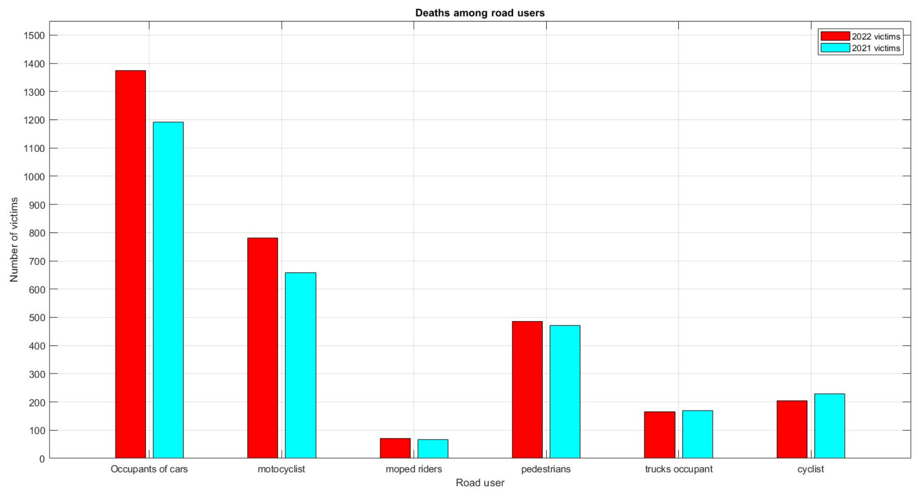


Figure 1.1: Victim distribution in fatal accidents among road users in 2022 (blue) and 2021 (red).

Considering the large implementation of vehicular automation successfully reached inside the factories, it comes almost natural to think about implementing some kind of automation in vehicles to limit or avoid accidents.
A big level of automation has already been deployed for our comfort in transportation such as for the metro system. In Turin, for example, the subway is completely autonomous. This system is anyway very simple and has huge limitations since it moves only on a predefined route with no interference such as pedestrians and traffic jam.

Another example of autonomous transportation in factories can be given by Autonomous Mobile Robot (AMR). In this case sophisticated automated guided vehicles are deployed in order to transport material around the factory to speed up the whole production chain.
There are many ways of indicating the route, for example the AMR can follow an electromagnetic signal emitted by wires embedded in the floor. Once again is possible to see that the huge limitation is due to the fact that they have no understanding of the environment around them and that many embedded wires inside the factory floor must be installed.
These solution are not feasible with autonomous vehicle moving along roads all over the world. In order to implement autonomous vehicles in our everyday life, it is important to understand the limitations given by these models and reflect on achievable solution to get around the problems.
The main challenges to face in order to be able in implementing autonomous driving are given by the appropriate estimation of the location of the vehicle in every instant, the perception and elaboration of the vehicle surroundings in order to make decision in real time. Additionally we must consider that there is not a predefined path to the destination and that the best path depends heavily on the environment. This is due to the fact that once the car is moving along the desired path, it is not alone in the road. It must be able to catch interference on the path such as unexpected obstacles, people crossing the road and other vehicles performing manoeuvres. In order to have a complete and exhaustive understanding of the environment is possible to consider the communication between vehicles (V2V) and with the infrastructure (V2X). This communication have been developed in order to successfully implement autonomous vehicles that can to avoid accidents. The use of this technology will have a very significant impact in many fields. For example, the time optimization in the highway through the platoon system can be used not only to ensure the fastest and the most secure path but can also lead to an important fuel consumption optimization. Many vehicles using this system can use a constant velocity and avoid to brake and accelerate ensuring a lower fuel consumption. This is a nice innovation for both the consumer and the environment reducing the carbon footprint.

1.2 Intelligent Transportation system

As defined by ETSI [2], the term Intelligent Transportation System (ITS) identifies all of the vehicles that are able to exchange additional information in the transportation infrastructure. The communication among vehicle is aimed to improve vehicle's safety, reliability, efficiency and quality. This definition highlight all of the advantages of ITS, accentuating that these functions can be exploited not only by

vehicles on the road, but by all means of transportation such as railways, aviation and nautical sector.
ETSI's Technical Committee for ITS is the European institution in charge of creating and maintaining a standard to integrate communication among vehicles. To guarantee global interoperability in the development of the technology, ETSI is cooperating with standardization bodies around the world such as International Standards Organisation (ISO), Institute of Electrical and Electronics Engineers (IEEE) and all of the major standardization institutions.
In Figure 1.2 it is proposed an ETSI representation of a connected world where are present all of means of transportation that can support this kind of communication and the road infrastructures to improve transportation efficiency and safety.

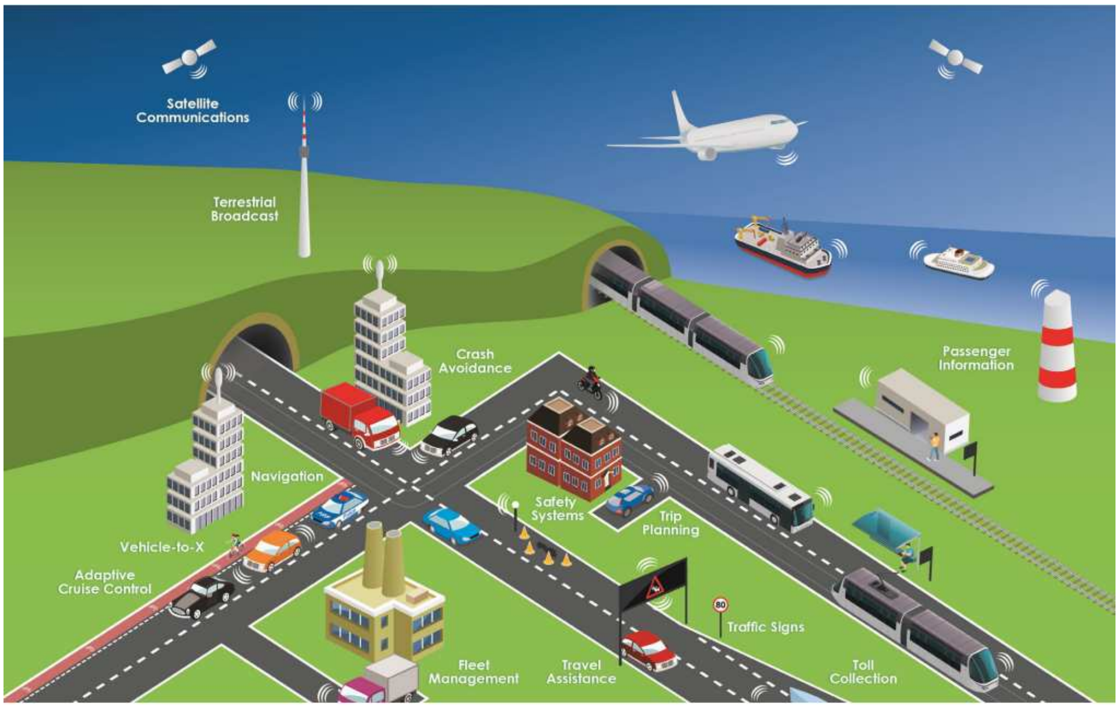
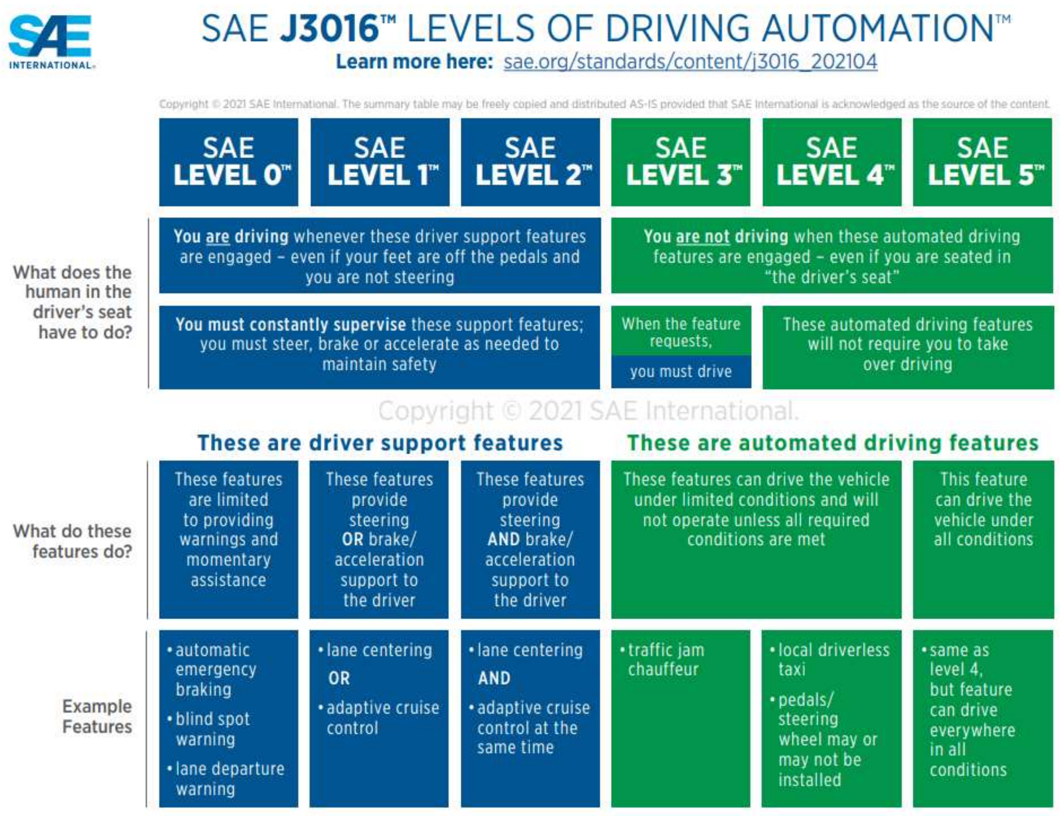


Figure 1.2: Communication among all possible vehicles and road side units [2].

The key to an efficient use of ITS rely on a great quantity of data that must be exchange. To ensure the correct and efficient level of communication, a wide use of wireless networks is needed. In particular the most popular ones are called Vehicular ad hoc networks (VANETs) and their characteristics will be fully analyzed in the second chapter.

As previously anticipated, vehicle's automation takes in consideration the most challenging scenario where the location must be estimated accurately and there

are not predefined paths to reach a destination. What makes it even more difficult is the heavy presence of interference due to the environment.
The autonomy of the ITS is addressed by considering different levels of automation in the vehicle. In particular for what concerns the definition of the different levels of automation, the SAE, Society of Automotive Engineers [3], established a new standard called J3016.
In Figure 1.3 it is shown the most updated J3016 standard visual chart related to April 2021.



SAE J3016™ LEVELS OF DRIVING AUTOMATION™

Learn more here: sae.org/standards/content/j3016_202104

	SAE LEVEL 0™	SAE LEVEL 1™	SAE LEVEL 2™	SAE LEVEL 3™	SAE LEVEL 4™	SAE LEVEL 5™
What does the human in the driver's seat have to do?	**You are driving** whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			**You are not driving** when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			**These are automated driving features**		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering **OR** brake/ acceleration support to the driver	These features provide steering **AND** brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering **OR** • adaptive cruise control	• lane centering **AND** • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

Figure 1.3: Visual chart provided by SAE for J3016 standard [3].

It is possible to see that the first three levels require the full attention and presence of the driver. They are considered to improve the driver's experience with some level of automation. Only from the SAE Level 3 it is possible to consider automated driving features, where the driver's responsibility is limited. In fact, the responsibility of the vehicle increases with the level of the automation peaking in the SAE Level 5, the complete automated vehicle. Here the vehicle is completely hands-off, so it is fully capable of moving autonomously in every environment and

under every circumstances.
The development of this technology is still a work in progress and at the moment, the highest level of automation reached is the SAE Level 3 automation. This automation level includes a Dynamic Driving Task (DDT) fallback of the driver in certain situations. The DDT fallback is a necessary implementation that consist in a warning to the driver that is again in charge of the whole vehicle. This happens when the road traffic or situation can not be managed autonomously by the vehicle. The relevance of this characteristic has been misinterpreted by many drivers that were using this kind of automation as a complete hands-off implementation of autonomous driving. The faith in technology and this misjudgment lead to many accidents due to the missed DDT fallback due to the fact that the drivers were asleep or completely distracted, ending up in not paying enough attention to the warning sent by the vehicle. Due to these events this level of automation has not been implemented anymore in vehicles.
In order to apply this kind of automation and grant an efficient level of communication, OBUs must be implemented inside of all the means of transport present on the road.
Another important aspect of the use of ITS is given by the possibility of optimizing the vehicle's fuel consumption. This can be reached by the use of the platooning technique, and allows the transportation to be more environmentally friendly. This will help to reduce emission that contributes to the global warming and for this reason many studies has been done about this topic.
In particular, in the article " Truck platooning reshapes greenhouse gas emissions of the integrated vehicle-road infrastructure system", a study regarding the reduction of greenhouse gas emission has been analyzed [4]. This study shows how truck platooning is a good strategy to lower the emissions from vehicle roads, decreasing the greenhouse gas emission and mitigate climate change. The overall truck platooning results in a 5.1% emission reduction of the integrated vehicle-road system. This practice, in contrast with the emission reduction, leads to an additional financial burden to car users and transportation agencies that must be took in consideration. In order to implement such a solution, it will be necessary to evaluate a trade-off between emissions and costs.
The platooning is a method applied to allow a group of vehicles to move in the same direction, especially on the highway, maintaining a short distance between them. This is meant to increase the capacity of the road via an automated highway system. The constant distance can be maintained through a continuous exchange of messages among vehicles in the entire platoon. It has been demonstrated that the shorter is the distance between vehicles, the better the result can be. In order to make it possible, the frequency of messages exchange vehicles increases. There are many methods and the efficiency increase with the decrease of the distance between vehicles.

1.3 Communication among vehicles

In order to support and improve a network in which the main focus is on vehicular safety and traffic efficiency applications, it is needed continuous information about the vehicles and the environment. In particular, in order to exchange this kind of information two types of messages have been standardized by the European Telecommunications Standards Institute (ETSI): the Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM). These standards specification contain both the packet format and dissemination rules.
The use of CAM and DENM messages can improve road safety in order to optimize the environment knowledge of the vehicle and take into account other vehicles. The use of these messages is a huge step for vehicular communication and it bring us closer to the realization of full autonomous vehicles capable of managing every situation on their own.
To ensure a reliable and efficient communication to implement autonomous driving, it is crucial to take in consideration that all of the road users should be able to communicate with each other. As shown in the article " Securing Vehicle-to-Everything (V2X) Communication Platforms" [5], this particular necessity raised the concept of vehicle-to-everything (V2X) communication that includes the most recent generation of networking technology. Communications among road users can be divided in:

- Vehicle-to-vehicle (V2V) communication
- Vehicle-to-infrastructure (V2I) communication
- Vehicle-to-pedestrian (V2P) communication
- Vehicle-to-cloud (V2C) communication

This will ensure a continuous exchange of information among all of the road users to improve traffic efficiency, road safety and road pollution.
In Figure 1.4 provided in the article " Securing Vehicle-to-Everything (V2X) Communication Platforms" [5], it is shown a high level illustration of vehicle-to-everything communication.

It is possible to immediately notice in this illustration that the V2X-enabled vehicle is able to communicate with other vehicles and infrastructures called road side units (RSU). The vehicle communication unit is called on board unit (OBU) and is in charge of messages exchange. This OBU is part of the vehicular control system and act as an external communication interface with other road users.
It must be took in consideration also the fact that communication among the whole environment should be truthful and safe. For this reason potential vulnerabilities

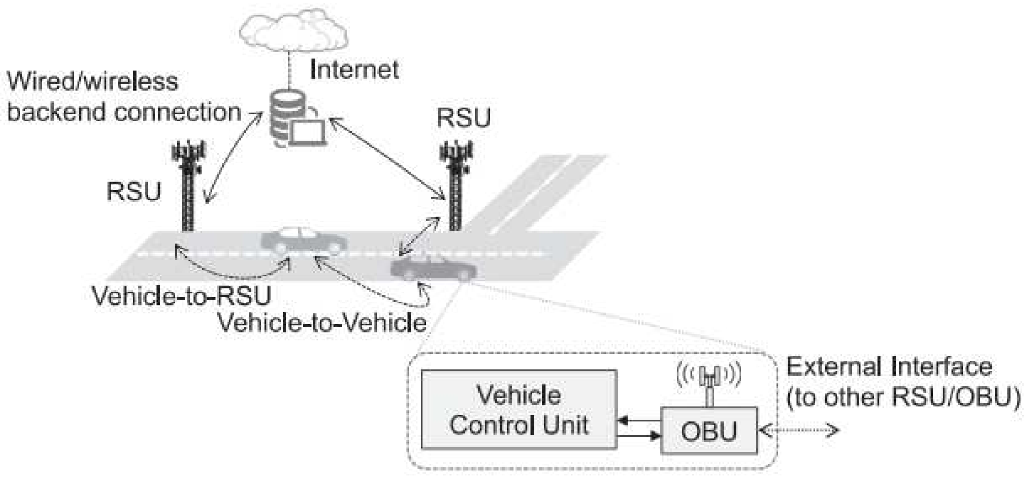

Internet
Wired/wireless backend connection
RSU
RSU
Vehicle-to-RSU
Vehicle-to-Vehicle
Vehicle Control Unit
OBU
External Interface (to other RSU/OBU)

Figure 1.4: High level illustration of V2X communication [5].

that can be exploited for future attacks are already been studied and corrected.
An example of these kind of attacks could be given by considering that a malicious vehicle can send false information about the road status, reporting an accident or traffic jam, and bias other vehicles forcing them to reroute and follow another path or by slowing them down. Attack detection and mitigation is essential to deploy a safe V2X system considering that the attack can be done through physical access to a subset of the system. This kind of attacks are dangerous since it can cause data loss, component failure and damage infrastructures.
For these reasons the development of the V2X communication technology must be done taking into account the security issues. The study of these vulnerabilities will not only avoid many possible attacks, but will also increase the user's degree of trust, a fundamental parameter for the inclusion of this technology in our everyday life.

Chapter 2

Vehicular communication Standards

In order to ensure a correct development of inter-vehicular communication, both the Institute of Electrical and Electronics Engineer (IEEE) and European Telecommunications Standards Institute (ETSI) have developed vehicular communication standard based on the IEEE 802.11 standard.

2.1 A Focus on IEEE and ETSI

The Institute of Electrical and Electronics Engineer known as IEEE is an American based international organization dedicated to technological development of facilities that can benefit the humanity. It is designed to standardize the most basic aspects of electrical, electronic, computer and telecommunication fields in order to guarantee a common ground for the developing of all of the projects around the world. The IEEE was born as the fusion of two previous institution: the Institute of Radio Engineers (IRE) and the American Institute of Electric Engineers (AIEE). The IEEE institution is responsible for almost the thirty per cent of world's literature in electrical, electronics and computer engineering fields. The development of a set of rules in order to approach and standardize a new technology happens way before the technology itself. In fact, already in November 2004 a task group was formed to develop the protocol 802.11p for vehicular communication. The drafts has been developed between 2005 and 2009, by April 2010 the amendment was approved and by July 2010 published with the title " Amendment 6: Wireless Access in Vehicular Environments".

The European Telecommunications Standards Institute (ETSI) is the European institution to rule standards about electronic communication networks and services among which there are telecommunications. The ETSI developed a standard based on the IEEE 1609.x IEEE 802.11p called ITS-G5, to ensure compatibility with the American standard. The IEEE 802.11 operates at 5.9GHz on a band that is uniquely used for inter-vehicle and infrastructure. This standard supports communication with data rate between 3 and 27 Mbps over a 10 MHz channel bandwidth and between 6 and 54 Mbps in a 20MHz channel bandwidth. The ITS-G5 can be exploited in a range up to 1000 m in many environments such as rural, urban, suburban and highways with a limit on the maximum relative speed set at 110 km/h. The bandwidth can be selected according to the need of the VANET.

2.2 VANETs

VANETs were first introduced in 2001 as “ car-to-car ad hoc mobile communication and networking" applications to have a reliable network on which was possible for vehicles to communicate. In Figure 2.1 a visual representation of a VANET provided by the CAR 2 CAR communication consortium [6].

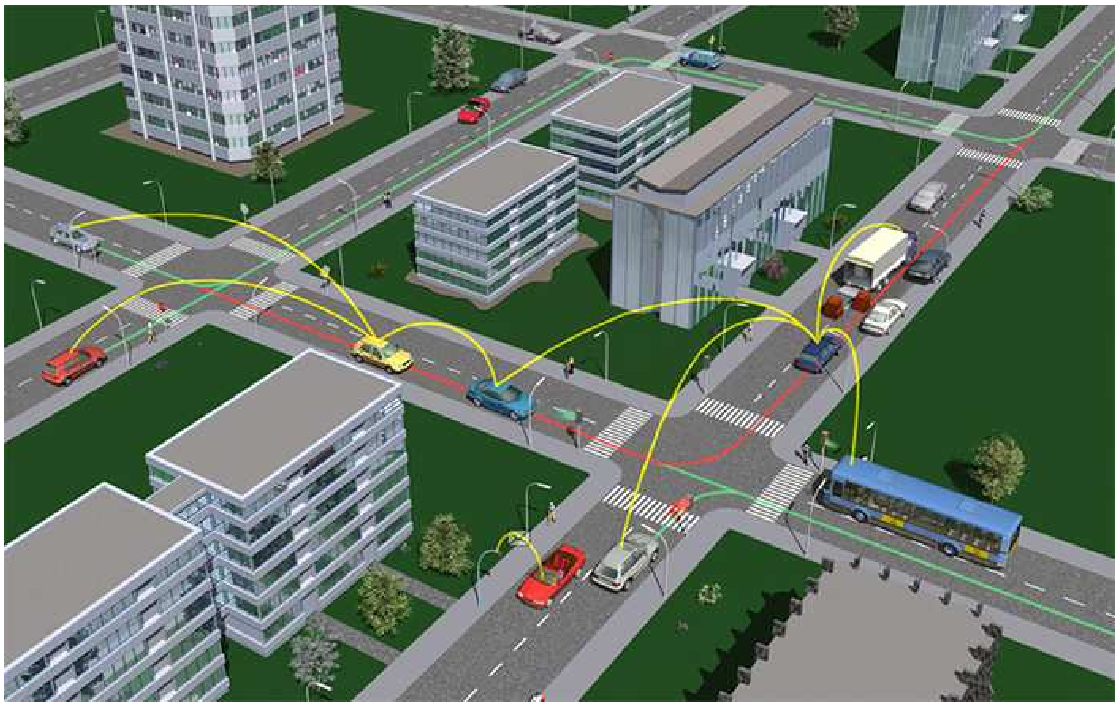


Figure 2.1: Visual representation of a VANET provied by CAR 2 CAR [6].

Even if at the beginning it was more oriented to a one-to-one communication, by 2015 the use of VANET was synonym to the communication among vehicles. The strength of VANETs networks is given by many fundamental aspects, in particular the ability of provide an effective wireless communication among moving vehicles equipped with GPS that, for this reason, represent a highly dynamic topology. Vehicular ad hoc networks (VANETs) are based on IEEE 802.11p. In order to support different cooperative ITS applications, other wireless carriers such as the 3G or LTE can be used.

The main difference between VANETs and cellular technology is the presence of the central controller in the former. The cellular technology relies on a central controller with full knowledge of the net. This characteristic ensures reliability of the network and allows the optimization of the resources to exploit all of the nodes at their best [7]. In cellular connection the presence of the central controller under the form of a base station is mandatory to make possible the communication between nodes. VANETs do not need the base station to ensure effective communication. In this style the communication is allowed directly between the two nodes in each other's range. The ad hoc solution though can arise the problem of scalability. The lack of a central control mechanism managing the communication could bring, in the worst case scenario, to all of the vehicles transmitting at the same time on the same channel, not allowing anyone to receive a meaningful message. The use of a common channel is mandatory in order to allow the vehicles to communicate since there isn't any central control that can direct communication on different channels. This frequency channel is called *control channel* is known a priori by all nodes. The control channel is the core of a VANET and allows to manage the road traffic safety application since on this channel the most important data will be transmitted. There are also more available channels that can be used to facilitate cooperative ITS application when there are higher bandwidth requirements.
The main building blocks of VANETs are:

- On Board Unit (OBU): communication facility on the vehicle equipped with a satellite positioning system such as GPS.
- Road Side Unit (RSU): fixed access point along the road side.

It is necessary to identify different priority for messages and a macro distinction can be made between safety and non-safety related issues and their time-scales. Time scale can vary also based on the priority of the message: there are many periodic messages such as road traffic or parking availability and many event based messages such as an accident notification. We need to consider also that they might have different latency needs and message length based on the content transmitted with the message.
Nowadays it has been demonstrated by researchers that the high density of vehicles and frequent change of network topology characteristic of the VANETs has become

a limit. In fact this is not supported by the traditional routing protocols [8]. New protocols to better fit VANETs characteristics have been developed and the main focus is to improve the network performances by reducing the overhead.
This is developed by allowing communication among all of the nodes that are considered reliable due to their geographical position and availability. The inclusion of a clustering technique in the routing process aims to reduce the unnecessary nodes and enhance a better communication with respect to the previous approaches.

2.3 The 802.11p protocol

With 802.11 it is represented a set of IEEE standards on wireless network technology (WLAN). This protocols sets not only the standard for the interface between a client and an access point and vice versa but also the standard for physical layer, Mac layer, and interconnection between devices and security. The 802.11 imposes a set of rules for the physical layer (PHY) and medium access layer (MAC) that belongs to the first two layers of the ISO-OSI layers, physical and data layer.
The IEEE 802.11p is a protocol for medium and access control layers of vehicular networks that includes an approved amendment to the 802.11 standard to regulate the wireless vehicular communication. The wireless communication among vehicles is usually referred to as WAVE that stands for wireless access in vehicular environments. This protocol defines the support required by ITS applications via 802.11 wireless protocol marketed as WiFi, including high speed exchange of data between vehicles and road side units [9].
Thanks to the VANETs, it is possible to ensure operation of many applications such as traffic safety, drivers assistance and entertainment for passengers. The motion of the vehicles at various speeds in various directions makes the topology of the network very difficult to predict and very volatile. What makes this situation even more tricky to manage is that the information about the vehicles has a limited duration in time. Since the vehicles are moving at a certain speed, the positioning information inside the message may expire very quickly based on the vehicle velocity. It is necessary to also implement protocols that take into account these characteristic to allow an efficient and meaningful message exchange between vehicles and infrastructures.

IEEE presented standards for vehicular network communications (WAVE): 802.11p for physical and mac layers and IEEE 1609 for security and network management of the VANETs. WAVE, by covering both the physical (PHY) and the MAC layers, defines an architecture and a complementary set of services and interfaces that enable secure vehicle-to-vehicle and vehicle-to-infrastructure wireless communications.

The main difference between the 802.11p and 802.11 is given by the fact that the second one uses frequencies that are not appropriate to the vehicular communication since they have a long association time and the high rate of data loss makes the communication unstable. In the academic paper " Performance comparison between 802.11 and 802.11p for high speed vehicle in VANET" [10], it is shown how 802.11p has efficiently enhanced the network performances. In particular the network throughput is increased, delay is decreased, and packet delivery ratio is increased as well.

2.4 WAVE standard

As previously anticipated, the IEEE 1609 Family of Standard for Wireless Access in Vehicular Environments (WAVE) provides a sufficient foundation for what concerns the management functions and modes of operation of the devices [11].

The WAVE provides a communication protocol stack optimized for the vehicular environment [12]. This objective is reached by employing both customized and general-purpose elements as shown in the Figure 2.2, provided by the IEEE official document " IEEE Standard for Wireless Access in Vehicular Environments—Security Services for Applications and Management Messages".

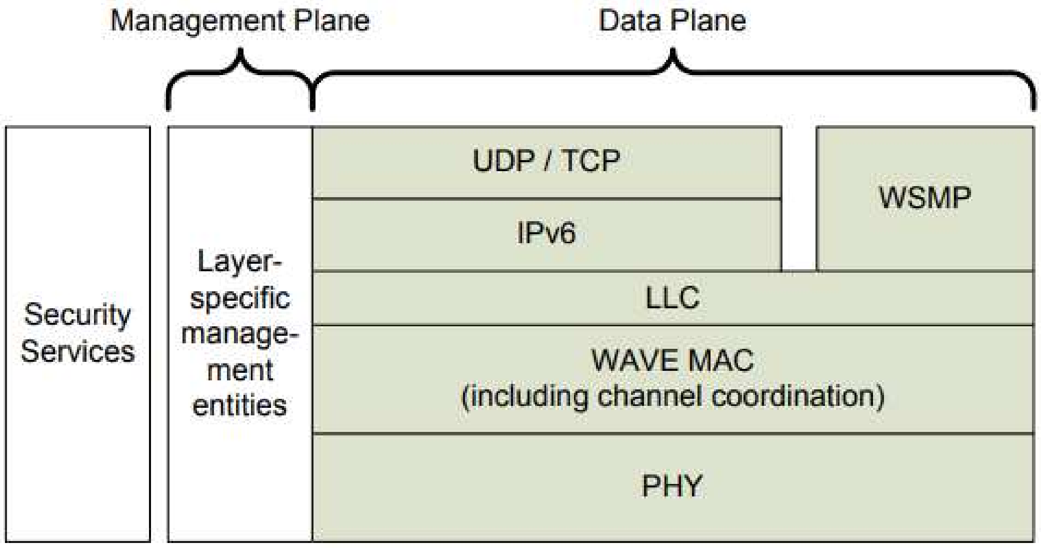


Figure 2.2: Official WAVE reference model provided by IEEE [12].

The IEEE 1609 family of standards define the architecture, communication model and management structure for high speed (up to 27 MB/s) short range (up to 1000 m) low latency communication in vehicular environment. It does also regulate the security mechanisms and the physical access of all the devices. The basic architecture component of these standards are the on board units (OBU), road side units (RSU) and the WAVE interface. Moreover, it must be considered that the standard includes all of the needed specifications that should be added to the IEEE 802.11 to provide an efficient wireless communication in a vehicular environment.

In particular, the protocols that are part of the WAVE stack are

- IEEE 802.11-2016, IEEE Std 802.11p: Wireless Access in Vehicular Environments (Amendment 6).
- IEEE 1609.0: IEEE Guide for Wireless Access in Vehicular Environments (WAVE) Architectur
- IEEE 1609.2: IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages
- IEEE 1609.3: IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services
- 1609.4-2016 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation
- IEEE 1609.11: IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Over-the-Air Electronic Payment Data, Exchange Protocol for Intelligent Transportation Systems (ITS)
- IEEE 1609.12: IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Identifier Allocations

WAVE supports IP and non-IP based data transfers to allow the use of a single networking stack, adapting to the individual characteristics of the device that is communicating. The IP based data transfer uses the IPv6 model while, the non-IP based data transfer are supported through the use of *WAVE Short Message Protocol (WSPM)* define in the IEEE Standard 1609.3.
There is also a collection of *WAVE Security Services* available, but not limited, to the WAVE devices to ensure a truthful and safe communication among devices. The WAVE Security services are formed by WAVE Internal Security Services and WAVE Higher Layer Security Services. The WAVE Internal Security Services are:

- Secure data service (SDS): manages the transformation of unsecured protocol data units into secured protocol data units during the communication and processing the incoming secured protocol data units. An entity using the secure data services is referred to as a secure data exchange entity.
- Security management: manages the information about certificates

The WAVE Higher Layer Security Services are:

- Certificate revocation list (CRL) verification entity (CRLVE): validate the incoming certificate revocation lists and provides the necessary information for storage.
- Peer-to-peer certificate distribution (P2PCD) entity (P2PCDE): enables the peer-to-peer certificates distribution.

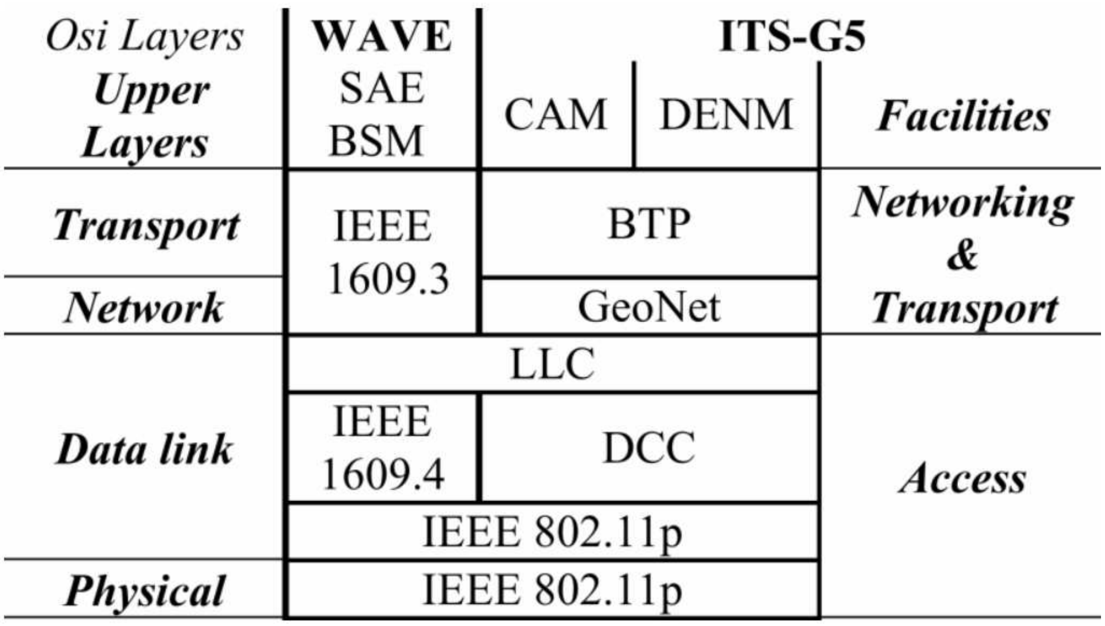
2.5 ETSI ITS G5

The ETSI ITS G5 protocol has been developed by the ETSI institute, based on the IEEE WAVE [13]. This protocol introduces some changes in the Network and Transport layers. This protocol also uses the Facilities level to minimize the probability of radio channel congestion. In Figure 2.3 it is shown a visual representation of WAVE and ITS-G5 protocols.

ETSI ITS-G5 standard adds a few interesting characteristics compared to the WAVE protocol. The ETSI ITS-G5 standard uses the Decentralized Congestion Control (DCC) protocol for the network load control. This protocol dynamically changes the channel access rules by means of transmitting power parametrization, modifying the minimum time interval transmission of periodic messages, changing the transfer rate and radio sensitivity. The real innovation in the ETSI ITS-G5 protocol is given by the introduction of new messages in the facility level, in particular:

- CAM: Cooperative Awareness Message
- DENM: Decentralized Environmental Message

These messages can be transmitted more than once and contribute to the load of the channel. There are many techniques to implement an efficient radio channel loading and efficient re-trasnmission of the packet.
The CAM messages are periodical messages exchanged in the ITS network to create and maintain awareness of the environment and each other and support cooperative performances of vehicles using the road network. In particular, CAMs are generated and broadcast by the Vehicle through the IEEE 802.11p wireless



Osi Layers	**WAVE**	**ITS-G5**		
Upper Layers	SAE BSM	CAM	DENM	***Facilities***
Transport	IEEE 1609.3	BTP		***Networking & Transport***
Network		GeoNet		
Data link	LLC			***Access***
	IEEE 1609.4	DCC		
	IEEE 802.11p			
Physical	IEEE 802.11p			

Figure 2.3: Visual comparison between WAVE and ITS-G5 protocols [13].

interface. Roadside units and vehicles within the coverage area will receive the CAM containing all of the status information about the vehicle.
CAMs are periodic messages that update the status data of the vehicle to the neighbouring nodes. The CAM message is structured as shown in Figure 2.4. The structure of CAM messages in Figure 2.4 does not contain extra optional fields for the sake of clarity. In the header is contained the protocol version, the message identifier or the generation time. The body and the reference position contain the data about the actual position.

The DENM messages are implemented to support the facility providing a notification service about the road status. The main characteristic of the DENM is that the message is sent when triggered, for example by traffic events or relevant modification in the vehicle's speed. This is due to the fact that using this kind of messages, all of the neighbour vehicles can be warned in real time about the sudden changes.

The DENM message is structured as shown in Figure 2.5. In the DENM structure it is possible to acknowledge that the header is identical to the CAM but the body does contain more fields. In particular there are three main categories:

- Decentralized Situation Management Group that include general information

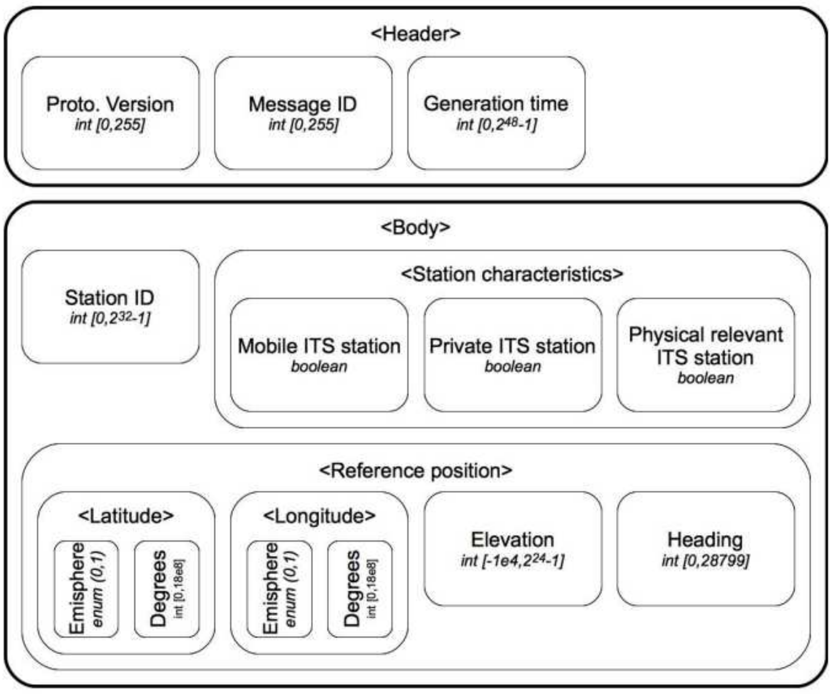


Figure 2.4: General structure of the CAM messages provided by the article " Experimental evaluation of CAM and DENM messaging services in vehicular communications" [14].

about the event

- Decentralized Situation that collect specific details about the event
- Decentralized Situation Location that specifies the event location

The CAM and DENM messages can be transmitted more than once and bring additional channel load. There are many techniques to implement both efficient radio channel loading and packet re-transmission, through the implementation of complementary protocols.
The Geo-Networking protocol manages the packet routing based on the geographic position of the node. The Contention Based Forwarding (CBF) is a multi-hop packet forwarding algorithm that is used in ad hoc networks. According to this algorithm, a node forwards a packet to all of its neighbor nodes. The receiving nodes compete to choose the only one that forwards the packet farther. This algorithm uses the geographic positioning information to select the node which has the grater distance from the sender node, to allow him to transmit the packet even farther. The Basic Transport Protocol (BTP) manages the point-to-point

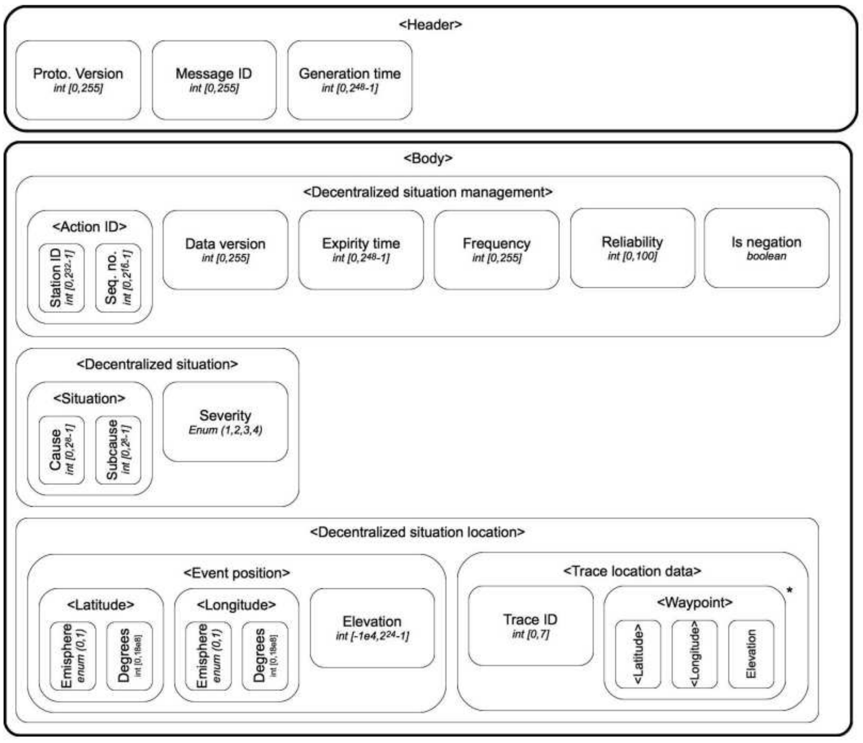


Figure 2.5: General structure of the DENM messages provided by the article “ Experimental evaluation of CAM and DENM messaging services in vehicular communications" [14].

connection less transport service inside the network. Its main goal is to multiplex and de-multiplex the messages in Facility level in order to be transmitted and received through the Geo-Networking protocol.
In the article “ Cooperative ITS security framework: Standards and implementations progress in Europe" [15] it is analyzed the actual security situation about the security in the Cooperative Intelligent Transportation System (C-ITS). The ETSI group that deals with the privacy, data protection and all of the security aspects of the ITS is called ETSI TC ITS WG5. Three steps compose the security process of this group, the first is to identify and catalogue ITS security risks. The second step is the definition of security requirements and definition of a list containing the potential countermeasures to these attacks. The third and last step is to specify an architecture and a standardized set of services and interfaces that enable secure V2X communications. In particular ETSI ITS Release 1 is a major achievement that has been the design of C-ITS security frame, including Public Key Infrastructure (PKI) implementation plans for digital certificate management. In table 2.1 are reported ETSI ITS security standards.

Standard reference	Title	Status
TR 102 893	Threat, Vulnerability and Risk Analysis (TVRA) technical report	v1.1.1 Published
TS 102 731	Security services and architecture	v1.1.1 Published
TS 103 097	Security header and certificate formats	v1.2.1 Published
TS 102 940	ITS communications security architecture and security management	v1.2.1 Under approval
TS 102 941	Trust and privacy management	v1.1.1 Published under revision
TS 102 942	Access Control	v1.1.1 Published
TS 102 943	Confidentiality services	v1.1.1 Published

Table 2.1: ETSI ITS security standards.

Chapter 3

Developing Tools

In order to develop this thesis project I exploit the functionalities of two already available software: OpenWrt and Vanetza. In the following chapter I will briefly illustrate the main characteristics of the two and how their features were exploited.

3.1 Motivations

The choice of using **OpenWrt** software on the APU board was based on the careful analysis of the software characteristics. The first advantage of using a Linux based operating system is the interoperability offered among devices. The Oxford's Dictionary define interoperability as “ the ability of computer systems or programs to exchange information." This means that using a Linux-based distribution the devices are always able to communicate between each other and exchange information. This is a crucial requirement for the vehicular communication technology that is based on the exchange of messages. Due to the fact that Linux is an open source operating system, there are not costs due to the proprietary code such as in Windows based systems or iOS systems. The most important aspect is that developing a solution based on Linux it is possible a continuous development and update of the system maintaining backwards compatibility with the previous versions. For an expanding technology such as the inter vehicular communication based on the exchange of positioning and movement messages rather than cameras, it is crucial that all of the versions can effectively communicate together to have a complete understanding of the environment.

The choice of the software **Vanetza** has been made in order to exploit an already developed and working tool able to integrate all of the standard requirements to be used for vehicular communication. This software offers some examples such as *socktap*, *benchmark* and *certify* to showcase the functions already integrated. This could be considered a game changer when developing an on board unit since the parameters of the standard are already available and always updated to the most recent standard. This project in fact, as for OpenWrt, is constantly updated to offer the best solution on the market.

3.2 OpenWrt

The OpenWrt project is defined as a Linux operating system targeting embedded devices. It relies on a fully writable filesystem with package management instead of using a single static firmware. This choice was made to allow full customization of the system allowing the installation of desired applications through packages that can be both found already compiled in a library or can be compiled by the developer. This flexibility allows the user to find the perfect fit for each particular application [16]. The use of OpenWrt offers a lot of features, among which the extensibility that allows to replicate the same setup, including more than three-thousand standardized applications, on any supported device. It does also support security since the software's components are kept up-to-date by closing vulnerabilities shortly after they are discovered. The standardization of the firmware's modules used in all the supported device allow the continuous improve and bug fixing ensuring the best performances and stability of the entire operating system.
The name OpenWrt showcases the intention of this project of being part of the open wireless router movement, starting with the first White Russian release for WRT54G router. This marked the beginning of the wireless router firmware development.

3.2.1 OpenWrt history

The OpenWrt project has been developed since 2004 by a team of volunteers that actively improve and maintain the whole project. The team of volunteers manage to maintain open source and updated the whole system, making it the best choice for developers. The first OpenWrt version was based on Linksys GPL sources for WRT54G and a buildroot from uClibc project. This was the first stable version of OpenWrt and became of wide use. In 2005 new developers joined the team and developed the first experimental version of OpenWrt. This version was depending on a heavily customized build system based on buildroot2 from the uClibc project. OpenWrt uses official GNU/Linux kernel sources and is able to manage system on chip and drivers for network interfaces through the addition of patches. The developer team has an active role in re-implement most of proprietary code inside the GPL tarballs of the different vendors. In 2016 a the LEDE (Linux Embedded Development Environment) project was founded and shared many goals with the OpenWrt project. This project aimed to build and embedded Linux distribution supporting developers to build and customize software for embedded

devices, especially wireless routers. Many members of the OpenWrt project worked also on the LEDE project and in January 2018 the OpenWrt and LEDE project agreed to merge under the name OpenWrt, working towards the shared goals. One of these goals was to offer a stable release of the system. This was achieved keeping in mind LEDE's code quality requirements for the repository.
The official stable releases of OpenWrt are the following:

- White Russian (2007)
- Kamikaze (2008)
- Backfire (2010-2011)
- Attitude Adjustment (2013)
- Barrier Breaker (2014)
- Chaos Calmer (2015 - 2016)
- LEDE 17.01 (2017 - 2018)
- OpenWrt 18.06 (2018 - 2020)
- OpenWrt 19.07 (2020 - 2022)
- OpenWrt 21.02 (2021 - 2023)
- OpenWrt 22.03 (2022)
- OpenWrt 23.05 (2023)

3.2.2 OpenWrt structure and features

As previously anticipated, OpenWrt hosts a writable root file system, as opposed to many other firmware that are based on read-only file systems, to allow changes and addition of installed software without the problem of rebuilding and flashing the complete firmware image. The main file present in OpenWrt is a heavily customized Buildroot system, a set of Makefiles and patches to allow the automation of building a complete Linux-based operating system for an embedded device through the use of appropriate cross-compilation toolchain. In order to allow consistency with the file compiled on the computer and the one on the embedded device. it is required a cross-compilation through the correct toolchain. For each device there is a specific toolchain to employ, in order to take into account the processor model and the characteristics of the device. The compilation through the specific toolchain generates compatible code for the specific device and ist processor's instruction set

(ISA).
Once the main options are implemented on the OpenWrt system through the issue of the command *makemenuconfig*, many other packages can be installed to fully customize the device. OpenWrt offers several thousands of packages to extend the functionality of the device that can be installed trough the issue of the *opkg* package manager. The package list is available on the official OpenWrt website contains a browsable list with more than 27000 packages ready to be installed. Additionally there is also the possibility of compiling and deploying specific applications needed that are not ready to be installed. This operation bring the customization of the specific board on the highest level possible.
The installed packages can be manage from a command line interface on the terminal or through a web interface called LuCI [17], an open and independent project. The LuCI WebUI is a graphic interface that can make many administration tasks easier. In the most recent releases, it does not need installation since it has been integrated by default on the OpenWrt image itself. The LuCI interface includes security features and encryption in order to allow a reliable communication between the developer and the board. LuCI was founded in March 2008 as Freifunk-Firmware LuCI named “ FFLuCI" to support OpenWrt from the Whiterussian release to Kamikaze. This project was born due to the need of a free, clean, extensible and easily maintainable web user interface for embedded devices. Unlike other interfaces that make a heavy use of shell-scripting language, LuCI uses Lua [18] programming language to ensure higher performances, lightweight installation size, faster run time and, most importantly, better maintainability.
The main characteristic of Lua is that combines simple procedural syntax with powerful data description construct with extensible semantics. Lua implements a lot of functions among which automation in memory management that makes it the ideal language for configuration. Lua supports procedural programming, object-oriented programming, functional programming, data-driven programming and data description.

3.3 Vanetza

Vanetza is an open-source implementation of the ETSI C-ITS protocol suite designed to send inter-vehicular messages such as CAM and DENM in order to allow the inter vehicular communication. This project was designed to operate on ITS-G5 channels and ad hoc Network called VANET using IEEE 802.11p, wireless access in vehicular environment [19].

3.3.1 Vanetza history

Vanetza is a project developed by the CARISSIMA Automotive safety research in Technische Hochschule Ingolstadt University. CARISSIMA [20] is a Center of automotive research inside the Technische Hochschule Ingolstadt University. Its important structure host eight different laboratories to allow vehicle safety researches. CARISSIMA's core area is the vehicle safety research and it develops different projects that allow an interdisciplinary approach. This is a very complex topic since it that takes into account not only technology development but also traffic psychology, traffic pedagogy and traffic economy. The use of this interdisciplinary approach allows the analysis of the situation from different angles in order to reach the optimal solution [21]. The technical objective of this test center, as stated on their official web page, is ' the realization of a „Global Safety System "that reproduces the human senses: to see, to hear, to taste, and to communicate using the bionic principles.'. CARISSIMA's interdisciplinary approach is not based primarily on the commercial aspect but aims to the general safety of the road user that is the main focus for the research. This is an innovative approach compared to the actual methods in the field of vehicle safety research.
Inside the CARISSIMA center, the Car2X Laboratory is the laboratory in which the Vanetza project has been developed [22]. The Car2X Laboratory provided the facilities to develop the research related to the communication among their own vehicles, both cars and motorcycles, and with their own traffic infrastructure such as traffic lights and intelligent road signs. This communication among the road users is also taking into account the presence of the whole environment and all of road users, including pedestrians and cyclist.
In this laboratory the Car2X protocols and applications are developed from the initial idea to the actual implementation inside the vehicle, ending with the test and evaluation of the whole application. In this projects many sensors are involved to have the most accurate perception of the environment possible. In particular, the vehicles are equipped with on board units (OBUs) that can communicate with the road side units (RSUs) present in the infrastructures.
In order to study the possible outcome of the application, simulations are used, in particular the Car2X Laboratory deploys *Artery* [23] to implement Vanetza.

The team that developed the project Vanetza is in charge of the continuous development and research about Vanetza's project. The team is composed by:

- Prof. Dr. rer. nat. Christian Facchi - Head of the Graduate Center
- Prof. Dr.-Ing. Andreas Festag - Business Start-Up Mentor of the Faculty
- Anupama Hegde - Research Assistant CARISSMA

- Silas Correia Lobo, M. Sc. - Laboratory Engineer and Research Assistant
- Daniel Maksimovski, M.Eng.
- Christina Obermaier, M.Sc. - Research assistant CARISSMA

Vanetza's GitHub repository can be found in as a Raphael Riebl pinned project. He is also the responsible for the manage of the issues on this repository.

3.3.2 Vanetza's structure and features

As previously stated Vanetza is an open-source implementation do the ETSI C-ITS protocol suite. Vanetza is a set of C++ libraries, some of which have dependencies on external libraries. Vanetza comprise also the following protocols and features:

- GeoNetworking (GN)
- Basic Transport Protocol (BTP)
- Decentralized Congestion Control (DCC)
- Security
- Support for ASN.1 messages (Facilities) such as CAM and DENM

Even if it was originally intended to be used on ITS-G5 channels in VANETs using IEEE 802.11p, Vanetza can also be combined with other communication technologies such as GeoNetworking over IP multicast.

In order to correctly use Vanetza is necessary to take into account all of its dependencies and check the installation of these application before the integration of this application. In the table 3.1, took by the official Vanetza documentation, there is a short representation of the Vanetza's component dependencies on external libraries.

The first step to fully exploit Vanetza is to install all of these dependencies on the board. After that it is possible to compile and build the application. In Vanetza are available also many unit tests that can be enabled through the activation of *BUILD_TESTS* CMake option. Additional tools can be enabled through the enable of the corresponding CMake option. The tool **benchmark** can be enabled by the issue of *BUILD_BENCHMARK* CMake option. Benchmark is a tool with security features, able to sign and validate packets.
Another additional tool is **certify**, it can be enabled through the issue of the *BUILD_CERTIFY* CMake option. This tool is created to set up a test PKI for secured V2X communication. It is also responsible to manage and handle security

Component	Depends on	Feature
access	net	Access layer, helpers for IEEE 802.11 PHY and MAC
asn1	-	Generated code and wrappers for ASN.1 based messages, e.g. CAM and DENM
btp	geonet	Headers and interfaces for BTP transport layer
common	-	General purpose classes used across Vanetza components, including serialization and timing
dcc	access, net	Algorithms for DCC cross-layer
facilities	asn1, geonet, security	Helpers to generate and evaluate ITS messages
geonet	dcc, net, security	GeoNetworking layer featuring geographical routing
gnss	-	Satellite navigation integration for positioning
net	common	Utilities for socket API and packet handling
security	common, net	Security entity to sign and verify packets

Table 3.1: Table of dependencies of Vanetza [19] .

certificates and authorization tickets.
The tool **socktap** can be enabled by the issue of the *BUILD_SOCKTAP* CMake option. This is an example application to showcase Vanetza's functions. This application demonstrates the API usage of Vanetza's libraries and it is thought to be the starting point for a custom application. *SOCKTAP* is thought for Cohda LLC sockets and I adapted it to the use on the developing hardware.

In order to exploit the full potential of this application on the hardware is necessary to take into account Vanetza dependencies. In particular it is necessary integrate the libraries Boost, Crypto++ and GeographicLib. Boost [24] is a free

peer-reviewed set of C++ libraries that are intended to be widely useful in managing tasks and structures for all the applications. The Boost license encourages the use of these libraries for all users with minimal restrictions.
Crypto++ [25] is a C++ security library intended to resist side channel attacks. It exploit constant-time, cache-aware algorithms and access patterns to minimize leakage.
GeographicLib [26] is a C++ library that offers interfaces to a set of geographic transformations. It was developed in order to improve the *geotrans* package transformation between geographic and MGRS (Military Grid Reference System) coordinates. It can now provide UTM, UPS, MGRS, geocentric, and local Cartesian projections, and classes for geodesic calculations.

At the moment Vanetza has been developed for network simulations and testing on embedded devices at Technische Hochschule Ingolstadt. It has also been used in other projects such as in Artery, where Vanetza is a network simulation of ITS-G5 protocols for a V2X simulation framework based on OMNeT++. It has been implemented also in motorcycles operated by the Connected Motorcycle Consortium [27] for the evaluation of new ITS application to increase the rider safety on the road. The last important project including Vanetza is the cube developed by nfiniity [28], one of the world's smallest V2X module that supports LTE-V2X/C-V2X and DSRC/ITS-G5 communication in one module.

3.3.3 NAP-Vanetza

NAP-Vanetza [29] is an extension of the Vanetza project that aims to the integration of the MQTT and JSON capabilities and additional types of ETSI C-ITS messages. NAP-Vanetza does support a larger number of messages compared to Vanetza, in particular it supports:

- Cooperative Awareness Messages (CAM)
- Decentralized Environmental Notification Message (DENM)
- Collective Perception Message (CPM)
- Vulnerable Road User Awareness Message (VAM)
- Signal Phase And Timing Extended Message (SPATEM)
- MAP (topology) Extended Message (MAPEM)
- Maneuver Coordination Message (MCM)
- Signal Status Extended Message (SSEM)

- Signal Request Extended Message (SREM)
- RTCM Extended Message (RTCMEM)
- Infrastructure to Vehicle Information Message (IVIM)
- Electric Vehicle Charging Spot Notification (EVCSN)
- Electric Vehicle Recharging Spot Reservation (EVRSR)
- Interference Management Zone Message (IMZM)
- Tyre Information System and Tyre Pressure Gauge (TISTPG)
- GeoNetworking Beacons

NAP-Vanetza's purpose is 'to manage the encoding, decoding, sending, and receiving of ETSI C-ITS messages, thus abstracting those layers from VANET application developers' as declared on their official documentation. The main difference between Vanetza and NAP-Vanetza is that in the lastes one, the applications send ETSI C-ITS messages to the service by building a JSON representation of the message. This is then published in a specific MQTT topic, to which Vanetza is subscribed.
In the same way, applications that wants to receive incoming messages published by Vanetza must subscribe to the MQTT to receive the JSON.
In Figure 3.1, provided by the official NAP-Vanetza's documentation, is represented the communication between the OBU and RSU through the MQTT topic subscription.

The development of NAP-Vanetza is part of an ongoing research work at Instituto de Telecomunicações, Network Architectures and Protocols Group (NAP) in Aveiro (Portugal) [30].
This group was launched in January 2011 to target proposal, analysis, evaluation and testing of new network architectures and protocols for the Future Internet. As stated on their official page, the main activity of the group are 'mainly concentrated in new architectures for mobility of users in inter-technology environments (integrating security), vehicular-to-vehicular communications, context-aware networks, self-control and self-management of networks, network and resources virtualization, and novel addressing paradigms for the Internet.'

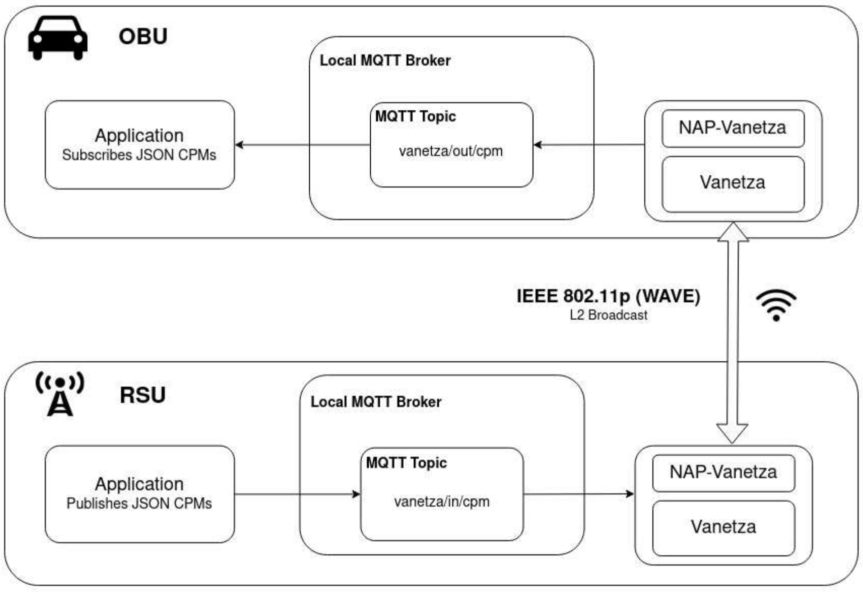


Figure 3.1: Official image representing the communication through MQTT topic subscription [29].

Chapter 4

Methodology

In the following chapter there will be a description of the steps that I took in order to install Vanetza libraries on the On Board Unit. For the sake of readability in this part no code will be shown.
I started by installing on the computer I worked on for the whole thesis, a virtual machine of the Ubuntu 22.04 operating system. For what concerns the hardware, I developed my thesis by configuring a Sierra Wireless MC7455 mPCIe module to work with OpenWrt 21.02.1 on a PC Engines x86_64 APU2 board (equipped with two mPCIe slot and one SIM slot, connected to one of the mPCIe slots). In the Figure 4.1 it is shown the APU board with all of the antennas connected and the GNSS module ArduSimple con U-blox ZED-F9R to integrate the GPS data as well in the CAM messages sent with SOCKTAP.

4.1 Installation of OpenWrt on the OBU

The first step to take in order to fully exploit the On Board Unit is given by the installation of an operating system on the board. There are many available operating systems and, to maintain the openness of the project, I decided to install on the OBU the Linux-based operating system developed by OpenWrt. The OpenWrt Project allows the installation of a Linux-based operating system in embedded devices. OpenWrt provides a fully writable file-system with package management, and allows the developers to exploit the framework to build an application without the commitment of building a complete firmware around it. It is also advantageous for the final user since it allows the full customization of the device.
I started by checking that all of the prerequisites needed by the OpenWrt system were satisfied and also that all of the dependencies packages were installed. After that, I proceeded with downloading a copy of the repository of OpenWrt version

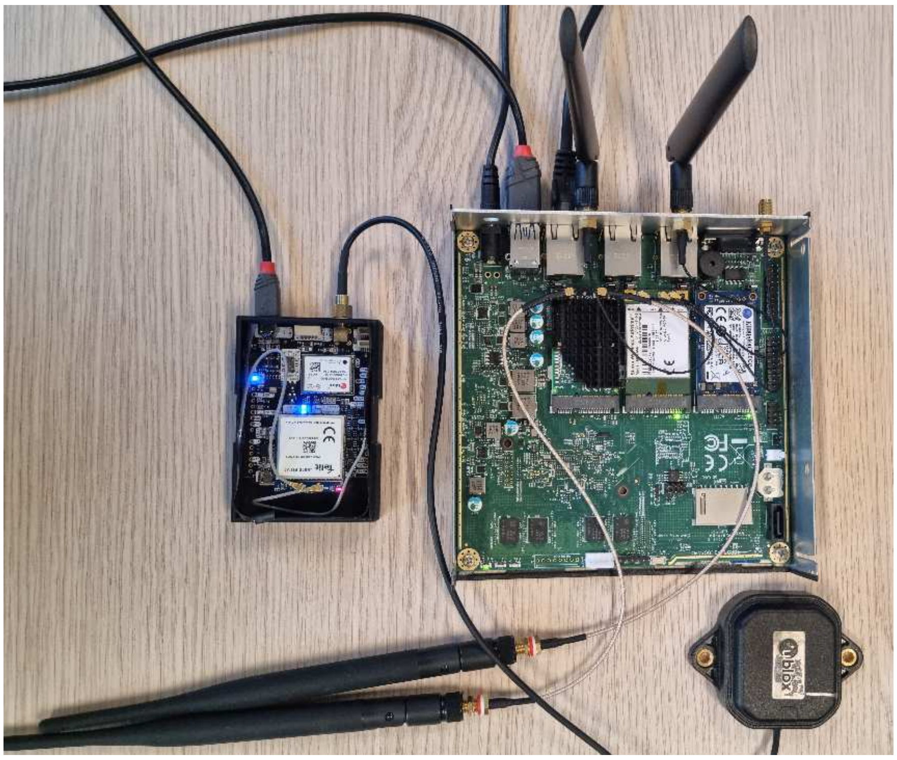


Figure 4.1: In this picture is shown all of the hardware used for the development of the project.

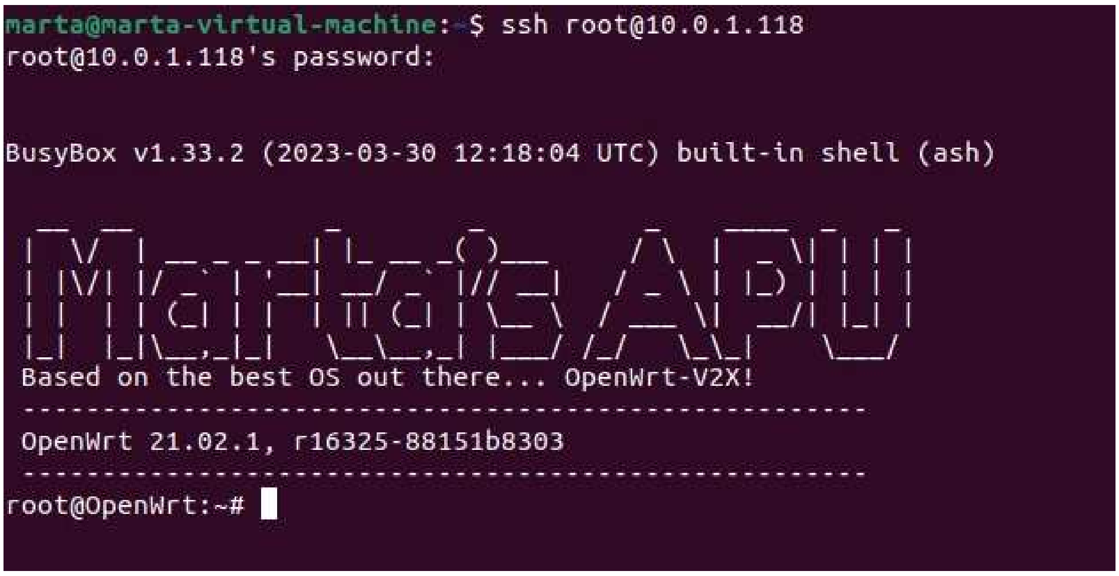
OpenWrt-V2X-21.02.1 on my developing virtual machine. I moved on with the installation, update and patch of the feeds and by selecting the tested configuration compatible with the APU. In the predefined configuration there are many useful packages already included such as CAN bus and GNSS support.
I opened the menu configuration and selected the target consistent with the board, in this case x86/x86_64 and set the previously downloaded default configuration file. Once everything was set I downloaded the dependency source files and build it. After the process was successfully completed I found the compiled image in the predefined folder. Then, I downloaded the system, thanks to the image just built, inside the board using TinyCore.
TinyCore Linux is a Linux distribution quickly bootable that easily allow the flashing process of an image inside an SSD through the use of an empty USB stick of maximum capacity of 8GB. The procedure with TinyCore on Windows based system is really quick and requires only one step while for Linux based systems, there are more operations to do. In order to exploit the simplicity and speed of TinyCore for Windows based systems, I downloaded a copy of the image built in the Linux based virtual machine in the Windows based PC. I downloaded the

appropriate file from TinyCore website, opened it and copied the image on the USB stick in the main directory. The OpenWrt system demands the copy of the gzipped file, in my case the name of the image was *openwrt-x86-64-generic-ext4-combined.img.gz*.
I moved on by inserting the USB stick in the board and connecting it via a serial cable to the developing computer. After that I connected the board to a 12V power supply I was able to find from the terminal the APU using the command screen and to interact with TinyCore. After imposing the value of the baud rate to 115200, I rebooted the board in order to use the connection SSH and safely remove the USB stick from the hardware. After the reboot, I connected the board to a modem by mean of Ethernet port 0. In this way the board was connected to the Ethernet of the laboratory. To fully exploit this connection, I edited the configuration network file commenting out the Ethernet bridge section and I selected in the LAN section the option device as eth1 and the static protocol, imposing the IP address as *10.0.2.118*. In the LAN1 section, where option device is set as eth2, i set an interface with the same characteristics of the previous configuration with IP address *10.0.1.118*.
These are the address that are used to issue the SSH command while sharing between the developing PC and the board the same Ethernet connection. By setting the SSH connection, the serial cable connected previously was not needed anymore and could be detached from both PC and board. I proceeded with enabling the WiFi and adding some DNS servers in order to let the board resolve IP addresses when connected to the Internet. In particular I added the DNS servers of Google with IP address *8.8.8.8* and *8.8.4.4*. I updated the *iw_startup* file by setting the setting the Wi-Fi IP address as *10.10.6.118* and the netmask *255.255.0.0* for the wlan0. I then automated the initial setup by adding the execution of the *iw_startup* in the *rc.local* file and a sound to know when the board was ready to be used. Once the board was ready, it could be accessed by the use of the password and the terminal appeared to be the one shown in Figure 4.2.

Now the board is all set with an operating system and connected to the Internet via Ethernet, it is possible to proceed with the LTE configuration in order to allow a wireless Internet connection via a SIM card.

4.2 LTE Connection

In order to setup an LTE connection on the APU2 board I proceeded with the following steps. The first thing to do was to update all of the already installed packages inside the board and check if all of the mandatory packages are present and up to date. To ensure that all of the updates were installed it is best to proceed with a reboot of the board.

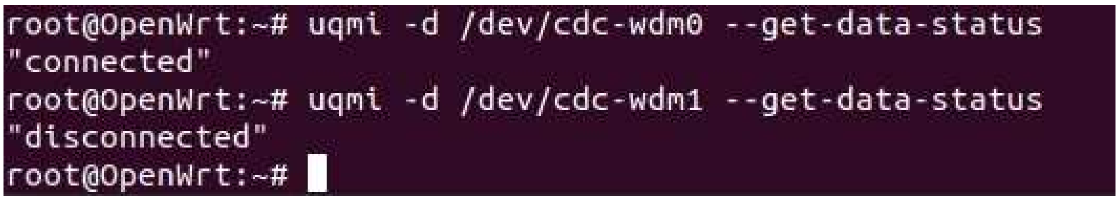


```
marta@marta-virtual-machine:~$ ssh root@10.0.1.118
root@10.0.1.118's password:


BusyBox v1.33.2 (2023-03-30 12:18:04 UTC) built-in shell (ash)

  __  __            _        _           _    ____  _   _
 |  \/  | __ _ _ __| |_ __ _( )___      / \  |  _ \| | | |
 | |\/| |/ _` | '__| __/ _` |// __|    / _ \ | |_) | | | |
 | |  | | (_| | |  | || (_| | \__ \   / ___ \|  __/| |_| |
 |_|  |_|\__,_|_|   \__\__,_| |___/  /_/   \_\_|    \___/
 Based on the best OS out there... OpenWrt-V2X!
 -----------------------------------------------------
 OpenWrt 21.02.1, r16325-88151b8303
 -----------------------------------------------------
root@OpenWrt:~#
```

Figure 4.2: Terminal's output once connected to the board.

I proceeded with the definition of the LTE interface inside the configuration network file of the APU. In the configuration I specified the QMI Cellular protocol to be used and that the authentication should not be required since the SIM's pin is deactivated. I also defined the packet data protocol (PDP) consistently with the service offered by the TIM operator at the moment that is IPv4 only.When IPv6 will be available, this option could be modified to ipv4v6 to allow the use of both the PDP. Since I used a TIM SIM card to connect the board to the Internet, I set the apn as *wap.tim.it*. The apn must be chosen accordingly to the the SIM's operator indication.
To verify that the SIM card was working and correctly read by the board, I issued some AT commands at the appropriate port and asked for the IMSI (International Mobile Subscriber Identity) of the SIM card. The SIM card, when read correctly prints on the monitor the IMSI number followed by *OK*, otherwise the output is *ERROR*. After that I opened the graphic interface of the board by using as address the APU IP, in my case *10.0.1.118*, logged in and edited the LTE interface.
I added the LTE to the same firewall zone as wan and wan6 to allow the connection to the Internet and proceeded with a power reboot. On the APU2 board there are two cdc-wdm interfaces: cdc-wdm0 and cdc-wdm1. After running many tests I noticed that the connection was randomly given between the two. Therefore, I decided to change the configuration to one active interface only *(RMNET0)* in order to have a stable and predictable outcome about the working interface.
This will ensure the correct initialization of the interface and the actual connection to the network. The most important part is that if any error occurs it is real and

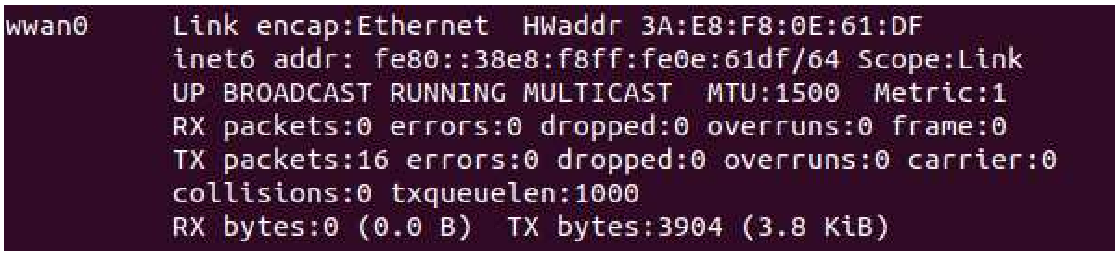
correctable, not given by the randomness in the initial choice of the interface. I proceeded then with the issue of the command to activate the Internet connection on the interface cdc-wdm0 and the deactivation of cdc-wdm1, as shown in the screenshot 4.3.



```
root@OpenWrt:~# uqmi -d /dev/cdc-wdm0 --get-data-status
"connected"
root@OpenWrt:~# uqmi -d /dev/cdc-wdm1 --get-data-status
"disconnected"
root@OpenWrt:~#
```

Figure 4.3: Screenshot of the terminal showing the state of the two internet interfaces.

I verified the the connection through the issue of the command *ifconfig*. This command will show every connection, among these there is one called wwan0 representing the LTE interface. As shown in the following Figure the connection is active since in the last row the amount of received and transmitted packets is different from zero.



```
wwan0     Link encap:Ethernet  HWaddr 3A:E8:F8:0E:61:DF
          inet6 addr: fe80::38e8:f8ff:fe0e:61df/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:3904 (3.8 KiB)
```

Figure 4.4: The wwan0 interface shown on the terminal through the issue of ifconfig command.

Then I checked that the connection was effectively working through the issue of a ping to a Google DNS that was successful.

4.3 Enabling remote access to the OBU

Once the APU was connected to the Internet through a stable wireless connection, I enabled the remote access to work on the APU also outside of the laboratory. Once the APU was connected to internet, I had the permission from the authorized people, to exploit the FULL server to implement this feature.
The remote access is the combination of software, hardware and network connectivity. The network connectivity is crucial to communicate with the facility

and the Internet. The software is needed to connect in a secure way the users to the facility for example trough a Virtual Private Network (VPN). The software encrypts the traffic before transmitting it over the Internet and is also responsible to decrypt the data sent to the VPN. The hardware is the set of all the hard-wired network interfaces and WiFi network interfaced to connect hosts and allow the communication.
The remote access to facilities had a strong development during the last pandemic. This was due to allow people to work from home, avoiding the complete paralysis of many systems without spreading the virus. This new work from home modality has become part of our life and unlocked a new standard. Many companies are in fact considering about keeping this possibility for their employees also after the pandemic in order to be able to hire the most qualified person with no limitation about the location. This massive development has been also beneficial to the research about the Internet of Things (IoT) that requires a constant exchange of data among all sort of devices. In these few years of pandemic, the remote access to a device has become a basic feature in the popular culture.
In order to be able to work remotely on the APU, even when I couldn't reach the laboratory, I used the following procedure. I enabled the APU communication to the server at a specific port, that was possible by using the server's password to authenticate. After using the password, I could communicate with the server and I added the APU to the list of known connection. I added the OBU public key to the list of the known devices already saved inside the FULL server, so that for all future connection attempts the device is immediately recognized. Now it is possible to remotely connect to the APU through the server by a double authentication. First of all it is required a password to access the server, once inside the server it is possible to issue the SSH command to connect to the APU. After the issue of the command a wireless connection from the server to the APU is created and the APU's password is required to access to the files. After inserting the password it is possible to work on the APU remotely. To enhance security and avoid pending connections, the APU connection is automatically closed after five minutes of inactivity.

4.4 Cross compilation

The cross compilation represented the most challenging part of my thesis. The cross compilation is a fundamental step in order to fully exploit libraries and application in many different possible platform. It must be done when the project is developed on a machine different from the host. This situation happens for the majority of the devices since the compilation on the host system is most of the time infeasible. This is due to the fact that the computing power required for the compilation would not be satisfied by the limited hardware sources.

The cross compilation can be done by the use of a tool called **compiler**, able to translate the code form the language compatible with the developing machine into the target source language. The compiler creates binary executable files compatible with the target system that allow the use of libraries and application on the target device.

One of the free source tools that can be used for the cross compilation is the GNU Compiler Collection (GCC). This tool supports many programming languages, hardware architectures and operating systems. This compiler has been adopted as the standard compiler for many Unix-like computer operating system and most of the Linux distributions.

The OpenWrt operation system has a defined set of packages already compiled and ready to use. Many packages that were needed as prerequisite for Vanetza and Vanetza itself were not already available. I cross compiled them and added them on the on board unit. Two items of the prerequisite did not need cross compilation: the GCC compiler and cmake of version 3.12 or higher, already available and installed in the APU.

4.4.1 Cross compilation of dependencies

I started by cross compiling the libraries on which Vanetza depend: boost, crypto++ and Geographic library.

Boost is a set of libraries for C++ that support different task and structures. In particular it is responsible for handling large numbers, linear algebra, regular expression, unit testing and pseudo random number generation.

The Boost library is required to be at version 1.58 or higher. To be consistent with the requirements I decided to download and use the latest update available, version 1.82.0 released on 14th April 2023. I downloaded and extracted the boost library in the download folder. I wrote a configuration file in the home folder and here I specified the architecture of the APU and the toolchain to be used to compile the executable file. As specified in OpenWrt documentation, to ensure compatibility, I used the musl compiler. As suggested in Vanetza documentation I edited some lines to override a bug in the installation of the library. After this small modification I installed the files inside the *Vanetza_deps* folder, the one containing all the dependencies files already cross compiled.

I moved on to the Crypto++ library, a free C++ library used for cryptographic schemes and able to create and handle cipphers, message authentication codes, one way hash functions and many other functions. The main reason for which this library is included is to guarantee security by crypting the messages and also support all the security functions for the authentication.

The Crypto++ library is required to be at version 5.6.1 or higher. In order to provide the compatibility with the most recent update of cryptography algorithms, I decided to implement the latest available update of this library, version 8.8.0 released on the 25th of June 2023. After downloading the library, I extracted the files to work on them and then move inside the directory. To cross compile the library is possible to use the GNUmakefile specifying which cross compiler to use and the path to follow to save the binary files. Once the cross compilation has successfully ended the compiled files appeared in the folder *Vanetza_deps*.

Finally I worked on with the Geographic library, a set of C++ libraries responsible for geographic transformation. Its main function is to improve the geotrans packages transforming between geographic and MGRS coordinates. The main characteristic of this set of libraries is the accuracy, with largest approximation error equal to 1 cm, with the willingness of reducing this error even more.
The Geographic library is required to be at version 1.37 or higher. In order to take full advantage of the library I decided to install the most recent available version, version 2.2 released on 7th March 2023. To cross-compile the library I extracted the files in the download folder and moved inside the directory to work on them. I wrote a suitable cmake file to specify the cross compiler to be used and the characteristics of the system for which I was cross compiling the library. I specified the path to follow in order to reach the folder and made the compilation begin. Once it successfully ended, the files were saved once again in the folder *Vanetza_deps* as specified.

4.4.2 Cross compilation of Vanetza

Now that all the external libraries on which Vanetza relies are cross-compiled, it is possible to move to the cross compilation of Vanetza. The first step for the compilation of Vanetza is to download a copy of the GitHub repository on the developing PC. I copied the repository in my home directory.
Once the repository was successfully cloned on my device, I moved in Vanetza directory and enable all of the available tools. The main tools are:

- Benchmark: a tool to benchmark some components of Vanetza, at the moment is used for signing and validating packets exist.
- Socktap: an experimental application to showcase Vanetza's basic feature.
- Certify: a tool to create and view certificates that can be used to set up tests for secured V2X communication.

Then I compiled Vanetza on the developing PC through the issue of a make command. Now a complete and compiled version is available on the device. To cross compile Vanetza and make it compatible with the operating system on the OBU, I created the folder *Vanetza_build* that will contain all of the necessary cross compiled files. I used the same musl compiler used to cross compile the dependencies and I specified, in the issuing of the command, to look for dependencies inside the *Vanetza_deps* folder.

After the completion of the compilation, a set of binary files are inside the *Vanetza_build* folder. These are all the compiled binary files needed to successfully install Vanetza library on the OBU. As previously stated, in order to use the cross compiled files it is requested to create a package containing the compiled binary files of the application or library.

4.5 Creation of an OpenWrt package

The OpenWrt operating system is based on the use of packages to integrate applications and libraries. Some packages has already been created and can be downloaded compiled to be easily installed on the board. Vanetza was not among the pre-compiled packages so I created a package for this library.
To be sure about the basic procedure to create compatible packages, I studied the guide *" Hello World!" for OpenWrt*, where a single executable file is created, compiled and put into a package. The situation I had to manage was quite different from the one described in this guide since I had libraries and files not only dependent form each other but also depending on external libraries. The first step in order to create a package was to create a package feed for the developed application. As explained in the guide, I decided to develop my package and save the folder *Vanetza* inside the the *mypackages* folder. This folder is assumed to be the one containing all of the developed packages.
For each package is necessary to create a package manifest file that describes the package and its functionalities. In this file it is possible to find specifications about the name, version and release number of the package, source settings, package definition and all of the install instructions. In the package manifest file I had the opportunity to choose these information and I decided to call this package *Vanetza*, release 1 and version 1.0. The OpenWrt build system uses a file called *feeds.conf* containing all of the packages available during the configuration stage.
This file is not created by default so, if new packages are created, this file has to be created as well in the OpenWrt main folder. Once the file is created, there must be included the path to follow in order to reach the folder *mypackages*, containing all of the customized packages. This will allow the system to know which packages are available to be implemented. Once the feed is defined, it is necessary to update

its package index so that all of the information index can be available in the configuration menu.
By opening the *make menuconfig* and opening the customized packages menu it is possible to chose which package include by typing “ y" for yes or “ n" for no. Once a package is choose it appears a “ *" symbol next to it. In this way it is always possible to control the number of packages considering the available memory of the device in order to optimize the performances. If the last step is completed successfully, it will appear on the terminal a success message saying that all of the packages have been installed from the folder *mypackages*.

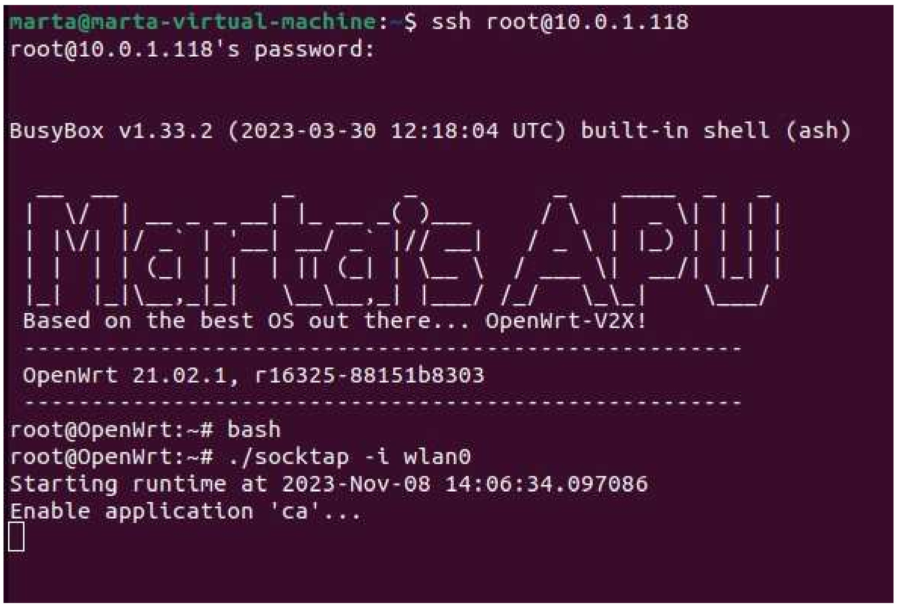
Once the package has been successfully installed from the feed, it is possible to proceed with the building of the package. This operation can be done by calling a *make* command specifying the name of the package and followed by the command *compile*. This operation will compile and create a brand new package in a *.ipk* file. This file will be situated in the folder containing all of the binary files, inside the folder *mypackages*. This file will report the name of the application followed by the version number and the toolchain name used to compiled it. In my case the full name of the package was *Vanetza_1.0-1_x86_64.ipk*.

4.6 Integration of the package

Once the package is compiled and available in the binary folder, it is possible to integrate the package on the OBU. Once the available version of the application is a package it is possible to transfer it on the board using WinSCP. Since this application is available on Windows based operating systems I proceeded in transferring the file from the Linux based virtual machine to the Windows based PC. I proceeded then connecting remotely to the server and then to the on board unit copying the package on the OBU.
Once the copy was available on the hardware it was possible to simply install it by issuing the *opkg* command. After the installation was successfully done, I copied also the compiled binary files for the test applications and made them executable in the OBU modifying their permission using *chmod 777* followed by the name of the file. Once the binary files could be executable I launched them and verified the correct functioning of the files.

4.7 Test with Socktap

The *SOCKTAP* function in particular was able to correctly send CAM messages respecting all of the requisites, one every second since the OBU was on a table. The messages were correctly received by the other OBUs inside the laboratory as shown in the following images. In the image 4.5 it is shown the terminal sending CAMs through the issue of the command *./socktap -i wlan0* where the messages are sent through the wlan0 interface of the OBU.



```
marta@marta-virtual-machine:~$ ssh root@10.0.1.118
root@10.0.1.118's password:


BusyBox v1.33.2 (2023-03-30 12:18:04 UTC) built-in shell (ash)

  __  __            _        _          _    ____  _   _
 |  \/  | __ _ _ __| |_ __ _( )___     / \  |  _ \| | | |
 | |\/| |/ _` | '__| __/ _` |// __|   / _ \ | |_) | | | |
 | |  | | (_| | |  | || (_| | \__ \  / ___ \|  __/| |_| |
 |_|  |_|\__,_|_|   \__\__,_| |___/ /_/   \_\_|    \___/
 Based on the best OS out there... OpenWrt-V2X!
 -----------------------------------------------------
 OpenWrt 21.02.1, r16325-88151b8303
 -----------------------------------------------------
root@OpenWrt:~# bash
root@OpenWrt:~# ./socktap -i wlan0
Starting runtime at 2023-Nov-08 14:06:34.097086
Enable application 'ca'...
```

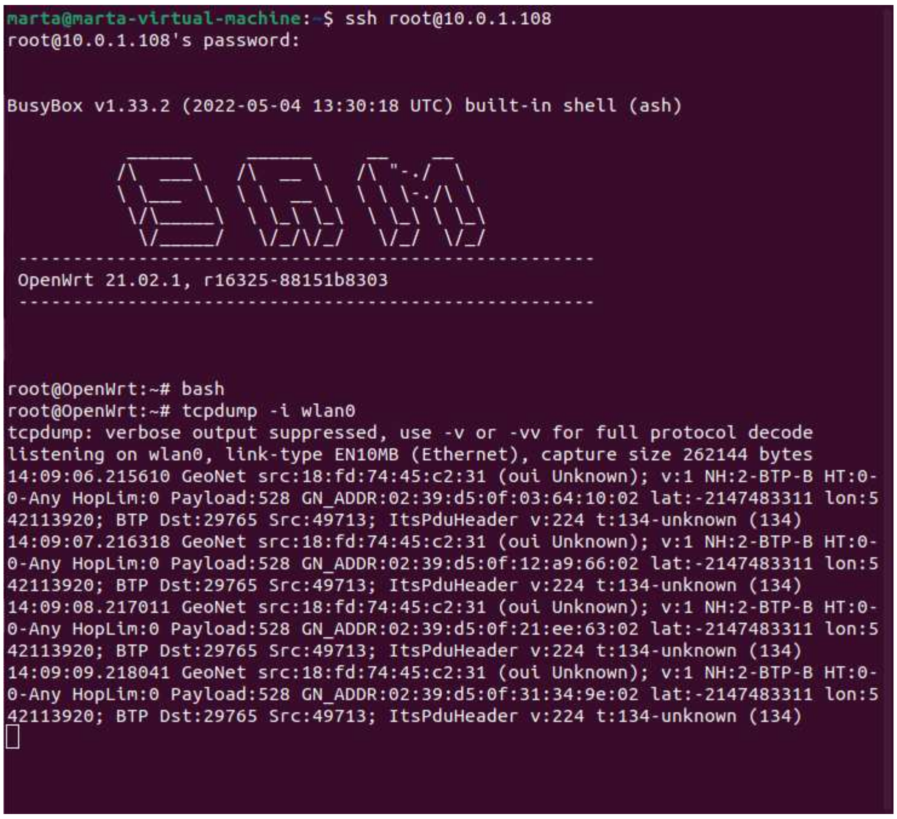
Figure 4.5: Screenshot of the terminal when issuing of the command on the OBU to send CAM messages through the SOCKTAP function.

In the image 4.6 it is shown the terminal of another OBU inside the laboratory successfully receiving the CAM messages sent by the OBU on which the entire thesis project was developed.

4.8 Integration of GNSS module

In order to send consistent data about the position of the OBU, it is necessary to integrate a GNSS module [31]. GNSS stands for Global Navigation Satellite Systems and has been developed for the aerospace domain. With the due changes this positioning model can be applied also to the land positioning. The main difference between the aerospace domain and the urban domain is high data redundancy, under the hypothesis that only one failure can occur at a time. The main data transmitted by the GNSS module are

- Latitude: it measures the distance north or south of the equator which is 0°.
- Longitude: it measures distance east or west of the prime meridian which is 0°.



```
marta@marta-virtual-machine:~$ ssh root@10.0.1.108
root@10.0.1.108's password:


BusyBox v1.33.2 (2022-05-04 13:30:18 UTC) built-in shell (ash)

           ______     ______     __    __
          /\  ___\   /\  __ \   /\ "-./  \
          \ \___  \  \ \  __ \  \ \ \-./\ \
           \/\_____\  \ \_\ \_\  \ \_\ \ \_\
            \/_____/   \/_/\/_/   \/_/  \/_/
 ---------------------------------------------------
 OpenWrt 21.02.1, r16325-88151b8303
 ---------------------------------------------------



root@OpenWrt:~# bash
root@OpenWrt:~# tcpdump -i wlan0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
14:09:06.215610 GeoNet src:18:fd:74:45:c2:31 (oui Unknown); v:1 NH:2-BTP-B HT:0-
0-Any HopLim:0 Payload:528 GN_ADDR:02:39:d5:0f:03:64:10:02 lat:-2147483311 lon:5
42113920; BTP Dst:29765 Src:49713; ItsPduHeader v:224 t:134-unknown (134)
14:09:07.216318 GeoNet src:18:fd:74:45:c2:31 (oui Unknown); v:1 NH:2-BTP-B HT:0-
0-Any HopLim:0 Payload:528 GN_ADDR:02:39:d5:0f:12:a9:66:02 lat:-2147483311 lon:5
42113920; BTP Dst:29765 Src:49713; ItsPduHeader v:224 t:134-unknown (134)
14:09:08.217011 GeoNet src:18:fd:74:45:c2:31 (oui Unknown); v:1 NH:2-BTP-B HT:0-
0-Any HopLim:0 Payload:528 GN_ADDR:02:39:d5:0f:21:ee:63:02 lat:-2147483311 lon:5
42113920; BTP Dst:29765 Src:49713; ItsPduHeader v:224 t:134-unknown (134)
14:09:09.218041 GeoNet src:18:fd:74:45:c2:31 (oui Unknown); v:1 NH:2-BTP-B HT:0-
0-Any HopLim:0 Payload:528 GN_ADDR:02:39:d5:0f:31:34:9e:02 lat:-2147483311 lon:5
42113920; BTP Dst:29765 Src:49713; ItsPduHeader v:224 t:134-unknown (134)
```

Figure 4.6: Screenshot of the terminal successfully receiving packets.

- Elevation: it measures distance in height from the level of the sea that is considered 0.

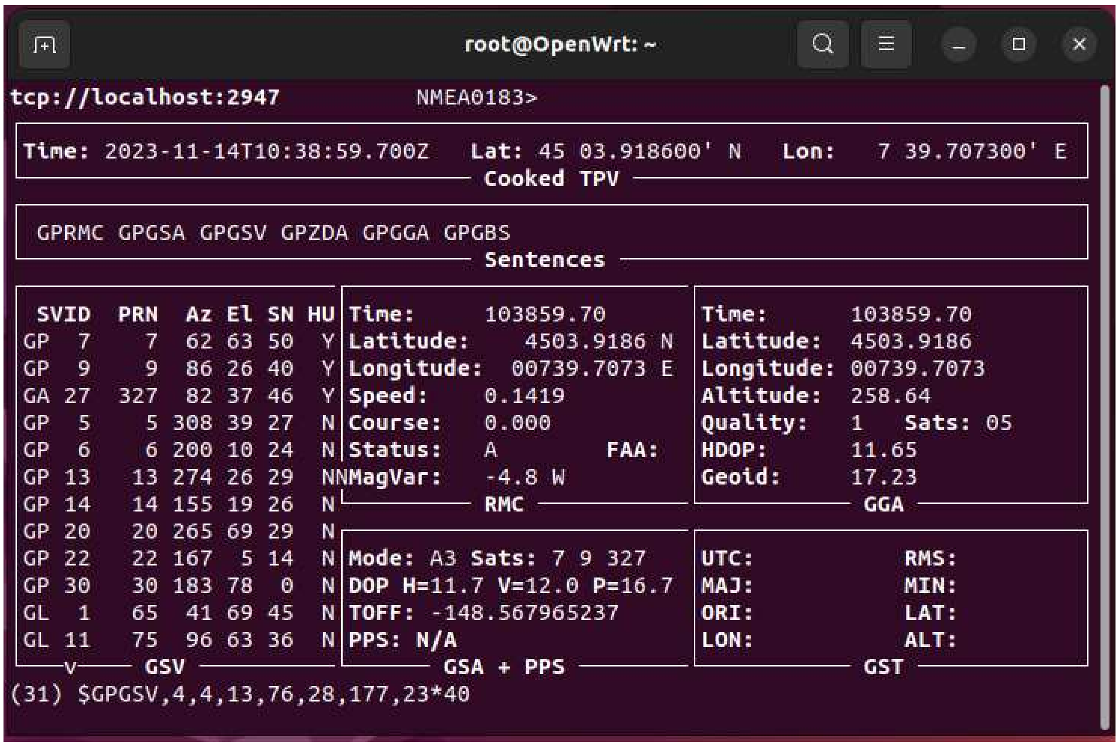
The time is also a fundamental variable contained in GPS data since it states how long does it take for the signal to travel from the GPS satellite to the GPS receiver on Earth.
The precision of the GPS data is given by the exploitation of different satellites through the technique called trilateration. This technique determines the position by knowing the distance of the GPS receiver from at least three known points. The three known points are the satellites themselves.

In my thesis project, I connected the module to the board and installed the *gpsd* [32] and *gpsd-clients* [33] packages.
The *gpsd* package installed on the OBU is version 3.23-1, the latest available version up to this moment. As described in the official documentation about the packages on OpenWrt website, the *gpsd* package is 'a userland daemon acting as a translator between GPS and AIS receivers and their clients'. This means that the the gpsd

is in charge of translating GPS data using some pre-defined protocols to allow the clients to understand it and use it. The gpsd listen on port 2947 for client requesting positioning, time and velocity information. The receivers must generate information according to the format: NMEA-0183 sentences, SiRF binary, Rockwell binary, Garmin binary format or other vendor binary protocols.
The *gpsd-clients* package installed on the OBU is version 3.23.1, the latest available version up to this moment. The *gpsd-clients* package contains auxiliary tools to be used in combination with the package *gpsd* in order to make the use of the positioning data even more simple for the clients.

The application *SOCKTAP*, developed by the Vanetza project, exploit the information that are translated by the daemon of *gpsd* package. In order to work on this application, even in absence of the GNSS module, *SOCKTAP* allows the use of a fictitious position. This data are fixed and correspond to latitude: 48° 46' 57,34" N and longitude: 11° 25' 55,44" E which corresponds to Nordost, 85055 Ingolstadt, Germany. The application *SOCKTAP* has a special variable called *SOCKTAP_WITH_GPSD* that is automatically activated to use real positioning data when the GNSS module is detected. To integrate the use of GPS data given by the GNSS module, I started by connecting the module to the OBU and positioning the antenna to be visible to the satellites. I proceeded searching on which port GPS data was present, in my case the port is */dev/ttyACM0* and set the port for the gpsd services. After this I proceeded with the check the correct visibility of data using the command *gpsmon -n localhost*. The data was correctly shown on the terminal as in Figure 4.7.

In order to activate the daemon and use this data with *SOCKTAP* I proceeded by editing the gpsd configuration file to add the serial port device to which the GPS is connected. This operation enabled the daemon and ideally managed the integration of the correct GPS data in the CAM message sent by the *SOCKTAP* application. From the testing I acknowledged that the data was not integrated. I proceeded to contact the responsible of Vanetza's code to ask more information about the connection between the application and the GNSS module. Unfortunately due to a bug discovered in Vanetza's code, the implementation of the data was not possible, but it will be fixed soon.



```
tcp://localhost:2947          NMEA0183>
┌──────────────────────────────────────────────────────────────────────────────┐
│Time: 2023-11-14T10:38:59.700Z   Lat: 45 03.918600' N   Lon:   7 39.707300' E │
└───────────────────────────────── Cooked TPV ─────────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────────────┐
│ GPRMC GPGSA GPGSV GPZDA GPGGA GPGBS                                          │
└───────────────────────────────── Sentences ──────────────────────────────────┘
┌───────────────────────┐┌─────────────────────────┐┌────────────────────────────┐
│ SVID  PRN  Az El SN HU│Time:      103859.70     │Time:       103859.70        │
│GP  7    7  62 63 50  Y│Latitude:     4503.9186 N│Latitude:   4503.9186        │
│GP  9    9  86 26 40  Y│Longitude:  00739.7073 E │Longitude:  00739.7073       │
│GA 27  327  82 37 46  Y│Speed:      0.1419       │Altitude:   258.64           │
│GP  5    5 308 39 27  N│Course:     0.000        │Quality:    1   Sats: 05     │
│GP  6    6 200 10 24  N│Status:     A        FAA:│HDOP:       11.65            │
│GP 13   13 274 26 29  NNMagVar:     -4.8 W       │Geoid:      17.23            │
│GP 14   14 155 19 26  N└────────── RMC ──────────┘└─────────── GGA ────────────┘
│GP 20   20 265 69 29  N┌─────────────────────────┐┌────────────────────────────┐
│GP 22   22 167  5 14  N│Mode: A3 Sats: 7 9 327   │UTC:            RMS:         │
│GP 30   30 183 78  0  N│DOP H=11.7 V=12.0 P=16.7 │MAJ:            MIN:         │
│GL  1   65  41 69 45  N│TOFF: -148.567965237     │ORI:            LAT:         │
│GL 11   75  96 63 36  N│PPS: N/A                 │LON:            ALT:         │
└──v──── GSV ───────────┘└──────── GSA + PPS ──────┘└─────────── GST ────────────┘
(31) $GPGSV,4,4,13,76,28,177,23*40
```

Figure 4.7: Terminal showing GPS data read from the GNSS module.

Chapter 5

Results

5.1 Results and conclusions

For this thesis project the objective was to implement the Linux based operating system OpenWrt 21.02.1 on a Sierra Wireless MC7455 mPCIe module working on a PC Engines x86_64 APU2 board and install the application Vanetza. The project was meant to send CAMs with consistent and real GPS data detected by the antenna.

The first part of the project has been successfully implemented since the board is now running on OpenWrt's operating system and exploits an LTE connection to access the internet, making the board wireless. This is a fundamental requisite to implement it in a vehicle. The package Vanetza, which was the most challenging part of the thesis, was created through the creation of Makefiles, cmake files and appropriate cross compilation of the dependencies and all of the libraries contained in Vanetza. The board contains the package Vanetza in its fullness and it is working correctly. However, due to problems in communication between Vanetza's code and the gpsd daemon, it was not possible to implement real GPS data.

5.2 Future development

This is only a small first step towards a future that aims to integrate autonomous driving in our everyday life. There will be room for investigate better this solution and compare this application with other applications available on the market.

Due to the nature of the Vanetza project, there is a constant innovation about the application and these upgrades must be actively integrated in all of the already existing projects.

An important future development is to find a solution about Vanetza's code to correctly integrate the GPS data inside CAMs messages exploiting the gpsd daemon. Moreover it could be possible to activate the DENMs messages, already supported in Vanetza, in the *SOCKTAP* application. Eventually, the integration of the on board unit inside a vehicle will be the last step before proceeding with real on-road tests to evaluate the communication performances with other vehicles.

Bibliography

[1] ISTAT. «Road Accidents. Year 2022». In: *Road accidents. Year 2022 - Istat.it* (2023) (cit. on p. 1).

[2] Definition of the ITS by ETSI `https://www.etsi.org/images/files/ETSITechnologyLeaflets/IntelligentTransportSystems.pdf` Last accessed on October 2023 (cit. on pp. 3, 4).

[3] SAE offcial website `https://www.sae.org/` (cit. on p. 5).

[4] Huailei Cheng, Yuhong Wang, Dan Chong, Chao Xia, Lijun Sun, Jenny Liu, Kun Gao, Ruikang Yang, and Tian Jin. «Truck platooning reshapes greenhouse gas emissions of the integrated vehicle-road infrastructure system». In: *Nature Communications* 14.1 (2023), p. 4495 (cit. on p. 6).

[5] Monowar Hasan, Sibin Mohan, Takayuki Shimizu, and Hongsheng Lu. «Securing Vehicle-to-Everything (V2X) Communication Platforms». In: *IEEE Transactions on Intelligent Vehicles* 5.4 (2020), pp. 693–713. DOI: `10.1109/TIV.2020.2987430` (cit. on pp. 7, 8).

[6] CAR 2 CAR offcial website `https://www.car-2-car.org/` (cit. on p. 10).

[7] European Telecommunications Standards Institute. «Intelligent Transport Systems (ITS); Performance Evaluation of Self-Organizing TDMA as Medium Access Control Method Applied to ITS; Access Layer Part». In: (2011 - 2012) (cit. on p. 11).

[8] Abdul Karim Kazi, Shariq Mahmood Khan, and Najmi Ghani Haider. «Reliable Group of Vehicles (RGoV) in VANET». In: *IEEE Access* 9 (2021), pp. 111407–111416. DOI: `10.1109/ACCESS.2021.3102216` (cit. on p. 12).

[9] Fernando A. Teixeira, Vinicius F. e Silva, Jesse L. Leoni, Daniel F. Macedo, and José M.S. Nogueira. «Vehicular networks using the IEEE 802.11p standard: An experimental analysis». In: *Vehicular Communications* 1.2 (2014), pp. 91–96. ISSN: 2214-2096. DOI: `https://doi.org/10.1016/j.vehcom.2014.04.001`. URL: `https://www.sciencedirect.com/science/article/pii/S2214209614000151` (cit. on p. 12).

[10] Mohammed Hasan Alwan, Khairun N Ramli, Yasir Amer Al-Jawher, Aws Zuhair Sameen, and Hussain Falih Mahdi. «Performance comparison between 802.11 and 802.11 p for high speed vehicle in VANET». In: *International Journal of Electrical and Computer Engineering* 9.5 (2019), p. 3687 (cit. on p. 13).

[11] «IEEE 1609 - Family of Standards for Wireless Access in Vehicular Environments (WAVE) `https://www.standards.its.dot.gov/factsheets/factsheet/80`». In: (2009) (cit. on p. 13).

[12] «IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages». In: *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)* (2016), pp. 1–240. DOI: `10.1109/IEEESTD.2016.7426684` (cit. on p. 13).

[13] Adrian Abunei, Ciprian-Romeo Comşa, and Ion Bogdan. «Implementation of ETSI ITS-G5 based inter-vehicle communication embedded system». In: *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*. 2017, pp. 1–4. DOI: `10.1109/ISSCS.2017.8034921` (cit. on pp. 15, 16).

[14] José Santa, Fernando Pereniguez-Garcia, Antonio Moragón, and Antonio Skarmeta. «Experimental evaluation of CAM and DENM messaging services in vehicular communications». In: *Transportation Research Part C: Emerging Technologies* 46 (Sept. 2014), pp. 98–120. DOI: `10.1016/j.trc.2014.05.006` (cit. on pp. 17, 18).

[15] Brigitte Lonc and Pierpaolo Cincilla. «Cooperative ITS security framework: Standards and implementations progress in Europe». In: *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 2016, pp. 1–6. DOI: `10.1109/WoWMoM.2016.7523576` (cit. on p. 18).

[16] OpenWrt offcial website `https://openwrt.org/` Last accessed September 2023 (cit. on p. 21).

[17] OpenWrt official website - LuCI essentials `https://openwrt.org/docs/guide-user/luci/luci.essentials` Last accessed October 2023 (cit. on p. 23).

[18] Lua official webpage `https://www.lua.org/home.html` Last accessed on November 2023 (cit. on p. 23).

[19] Vanetza official website `https://www.vanetza.org/` Last accessed November 2023 (cit. on pp. 23, 26).

[20] CARISSIMA official webpage `https://www.thi.de/en/research/carissma/` (cit. on p. 24).

[21] CARISSIMA philosophy statement `https://www.thi.de/en/research/carissma/philosophy/` Last accessed November 2023 (cit. on p. 24).

[22] Car2X Laboratory official webpage `https://www.thi.de/en/research/carissma/laboratories/car2x-laboratory/` (cit. on p. 24).

[23] Artery GitHub repository `https://github.com/riebl/artery` (cit. on p. 24).

[24] Official Boost website `https://www.boost.org/` Last accessed July 2023 (cit. on p. 26).

[25] Official Crypto ++ website `https://www.cryptopp.com/` Last acessed July 2023 (cit. on p. 27).

[26] Official GeographicLib website `https://geographiclib.sourceforge.io/#` Last accessed July 2023 (cit. on p. 27).

[27] Official Connected Motorcycle Consortium website `https://www.cmc-info.net/` Last accessed November 2023 (cit. on p. 27).

[28] cube by nfiniity offcial webpage `https://www.nfiniity.com/` Last Accessed November 2023 (cit. on p. 27).

[29] Official NAP-Vanetza GitLab webpage `https://code.nap.av.it.pt/mobility-networks/vanetza` Last accessed November 2023 (cit. on pp. 27, 29).

[30] Network Architectures and Protocols (NAP) group official webpage `https://www.it.pt/Groups/Index/36?groupLetter=NAP-Av` Last accessed November 2023 (cit. on p. 28).

[31] Ni Zhu, Juliette Marais, David Bétaille, and Marion Berbineau. «GNSS Position Integrity in Urban Environments: A Review of Literature». In: *IEEE Transactions on Intelligent Transportation Systems* 19.9 (2018), pp. 2762–2778. DOI: `10.1109/TITS.2017.2766768` (cit. on p. 40).

[32] Official documentation on OpenWrt package gpsd `https://openwrt.org/packages/pkgdata/gpsd` (cit. on p. 41).

[33] Official documentation on OpenWrt package gpsd-clients `https://openwrt.org/packages/pkgdata/gpsd-clients` (cit. on p. 41).