ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Enabling smart hybrid environments with cooperative perception

Miguel Ferreira Oliveira

Thesis Research Plan

Master's degree in Critical Computing Systems Engineering

Supervisor: Ricardo Augusto Rodrigues Da Silva Severino

Porto, February 13, 2023

Abstract

With the rapid development of Automated Vehicles (AV), the boundaries of their functionalities are being pushed and new challenges are being imposed. In increasingly complex and dynamic environments, it is fundamental to rely on more powerful onboard sensors and usually AI. However, there are limitations to this approach. As AVs are increasingly being integrated in several industries, expectations regarding their cooperation ability is growing, and vehicle-centric approaches to sensing and reasoning, become hard to integrate. The proposed approach is to extend perception to the environment, i.e. outside of the vehicle, by making it smarter, via the deployment of wireless sensors and actuators. This will vastly improve the perception capabilities in dynamic and unpredictable scenarios and often in a cheaper way, relying mostly in the use of lower cost sensors and embedded devices, which rely on their scale deployment instead of powerful sensing abilities. Consequently, to support the development and deployment of such cooperation actions in a seamless way, we require the usage of co-simulation frameworks, that can encompass multiple perspectives of control and communications for the AVs, the wireless sensors and actuators and other actors in the environment. In this work, we will be relying on ROS2 and micro-ROS as the underlying technologies for integrating several simulation tools, to construct a framework, capable of supporting the development, test and validation of such smart, cooperative environments. This will enable the demonstration of multiple cooperation scenarios and also ease the deployment phase by relying on the same software architecture.

Keywords: autonomous vehicle, cooperative perception, micro-ROS, simulation, dynamic environment

Resumo

Com o rápido desenvolvimento dos Veículos Autónomos (AV), os limites das suas funcionalidades estão a ser alcançados e novos desafios estão a surgir. Em ambientes complexos e dinâmicos, é fundamental a utilização de sensores de alta capacidade e, na maioria dos casos, inteligência artificial. Mas existem limitações nesta abordagem. Como os AVs estão a ser integrados em várias indústrias, as expectativas quanto à sua capacidade de cooperação estão a aumentar, e as abordagens de perceção e raciocínio centradas no veículo, tornam-se difíceis de integrar. A abordagem proposta consiste em extender a perceção para o ambiente, isto é, fora do veículo, tornando-a inteligente, através do uso de sensores e atuadores wireless. Isto irá melhorar as capacidades de perceção em cenários dinâmicos e imprevisíveis, reduzindo o custo, pois a abordagem será baseada no uso de sensores low-cost e sistemas embebidos, que dependem da sua implementação em grande escala em vez da capacidade de perceção mais potente. Consequentemente, para apoiar o desenvolvimento e implementação destas ações em cooperação, é necessária a utilização de frameworks de co-simulação, que abranjam múltiplas perspetivas de controlo e comunicação para os AVs, sensores e atuadores wireless, e outros atores no ambiente. Neste trabalho será utilizado ROS2 e micro-ROS como as tecnologias subjacentes para a integração das ferramentas de simulação, de modo a construir uma framework capaz de apoiar o desenvolvimento, teste e validação de ambientes inteligentes e cooperativos. Isto irá permitir a demonstração de múltiplos cenários de cooperação e também facilitar a fase de implementação, utilizando a mesma arquitetura de software.

Contents

Chapter 1

Introduction

1.1 Problem Statement

Automated Vehicles (AV) rapid development pace has been pushing the boundaries of their functionalities, and also imposing new challenges. AVs have been increasingly integrated into several industries such as retail, logistics, the military, and healthcare. As a consequence, expectations regarding their ability of cooperation with humans, continues to grow.

In a highly dynamic environment with both humans and AVs in play, the strategies planned to fulfil certain tasks keep changing, depending on the status of the environment, other actors involved, and the progress in the planned operation. For that, it is fundamental to rely on onboard sensing, usually relying on AI to learn and reason about the environment, but also sensors deployed in the environment (e.g. cameras, lidar and radar) as well as infrastructure-based computing and sensor fusion, to gather the information in real time and process it. These sensing and reasoning processes, both about the environment and vehicle perception, have proven to be hard to integrate, which has been becoming a major hurdle in deploying safe human machine cooperation. They are usually integrated with devices with high computational capacity, and thus are quite expensive and hard to gain access to. That is why the use of embedded devices which use microcontrollers is important since their use minimizes the cost and are easier to obtain.

Since the perception of the environment is a major requirement for the vehicle's ability to adapt and overcome dynamic and unpredictable scenarios, the motivation of this work is to improve the perception ability by enabling it closer to the environment while relying on microcontrollers.

1.2 Goal and Objectives of the Work

This thesis aims at integrating environmental and on-board sensors with an AV perception layer, focusing on embedded devices, where will be adopted recent ROS tool-sets for these type of devices eg. micro-ROS. This approach will be supported by relying on existent co-simulation tools such as COPADRIVE or AuNa, which will be leveraged to integrate these sensing interconnections. The main goal is to contribute to a co-simulation framework that can simulate complex highly dynamic environments, while relying on heightened perception via a smart sensing wireless infrastructure. The objectives of this thesis are the following:

- Understand the fundamentals of IoT communications, cooperation, cooperative perception, and smart hybrid environments;

- Create and present a consistent argumentation regarding the human-machine cooperation problem and challenges and current hurdles of cooperative perception;
- Survey current distributed cooperation perception techniques, IoT technologies, and relevant toolsets. Compare different approaches in solving the problem;
- Survey and compare existent and relevant IoT communication stacks and simulation frameworks to support the sensing infrastructure, as well as current robotic/perception frameworks;
- Design a system's architecture to carry out distributed sensing for enhanced and cooperative perception in smart hybrid environments;
- Implement over a simulation framework the above mentioned architecture;
- Carryout a simulation analysis and evaluate the performance and limitations of the proposed architecture under different network and control settings;
- Write and publish a research paper featuring part of the results.

Chapter 2

State of the Art

2.1 Concepts

The following concepts to be addressed in this section, were chosen due to their relation with the scope of this thesis, and therefore their common challenges and problems need to be taken into account, in order to apply these concepts in the context of this work.

2.1.1 Smart Environments

An environment is considered smart whenever it is composed by systems which employ sensors to perceive their environment, interpret the perception on an abstract level to yield some non-trivial ("intelligent") decisions beyond simple reactive sense-act rules, and are able to interact with the environment or alter it using some actors. In short, smart environments and systems acquire and apply knowledge in order to assist people (Wolter and Kirsch 2017).

From the viewpoint of human-machine interaction, smart environments present a multitude of new options for interacting with users and offering new services. While classical user interfaces rely on explicit input from users, smart environments can offer proactive services (Wolter and Kirsch 2017).

2.1.2 Cooperative Perception

Automated vehicles rely on sensors (eg. lidars, radars and cameras) for perception and localization of the driving environment. This environment is composed by static elements (eg. road, traffic signs, other objects) and dynamic elements (eg. pedestrians, other vehicles). However, there are still limitations regarding dynamic elements perception, and that is why it is necessary cooperative perception in order to improve AVs sensing and perception capabilities (Thandavarayan, Sepulcre, and Gozalvez 2020).

Cooperative perception enables vehicles to exchange their sensor data, which provides additional data about the driving environment, including data beyond their on-board sensors' field of view. This improves detection accuracy and increases the confidence about the detected objects. It relies on V2X (Vehicle-To-Everything) communications, which consists in communicating with external entities (eg. other vehicles, external sensors, etc.), in order to gather information about the environment to improve its accuracy (Thandavarayan, Sepulcre, and Gozalvez 2020).

2.1.3 Human-Machine Cooperation

The term "machine" regarding human-machine cooperation, refers to an intelligent system that can make decisions in an autonomous and (partially or fully) independent manner, which autonomy is realized through artificial intelligence (AI), deep learning, or other algorithms. These machines, with their increased computation power and advanced algorithms, are capable of catching up with or even surpassing human capabilities in various contexts. However, they are often developed for specific purposes and trained with limited data, which only allows them to perform well in a pre-defined scope, and thus their flexibility, adaptability, and accountability are lacking (Xiong et al. 2022).

Human-machine cooperation consists in having humans and machines working together to achieve a goal, for example in decisions involving high stakes such as human lives, it is necessary human supervision due to insufficient reasoning under moral dilemmas.

2.1.3.1 Human-Machine Cooperation Problems and Challenges

Communication and information sharing play a critical role in achieving an understanding of intentions and behaviors and creating an effective human–machine team. High uncertainty and task interdependence bring forward request to bi-directional transparency in real-time in human–machine teams to support effective communication and smooth task transition. However, human–machine teams do not have a common ground and linguistic interaction, thus problems would emerge when machines identify humans' explicit and implicit intentions in decision making as well as signal their intentions (Xiong et al. 2022).

In order to analyse human-machine cooperation, a recent study conducted by (Whiting et al. 2021) investigates human-human, robot-robot, and human-robot cooperation in a strategically rich resource-sharing scenario. The autors used a game called Block Dilemma to conduct a series of simulations and user studies regarding interaction between the two players. The scenarios studied consisted in human-human pairs, machine-machine pairs, and human-machine pairs. A scenario was also studied where no communication was allowed between the elements of the pair.

By analysing the results of the studies, the authors concluded that human-human pairs demonstrated an efficient cooperation when communication was allowed, and an aloof cooperation when communication was not allowed. Machine-machine pairs were efficient when communication was both allowed and not allowed, and human-machine were only efficient when communication was allowed, and in most of the cases, when communication was not allowed, there was a conflict in the behaviour of the pair. In short, without communication, human behavior and machine behavior are not aligned, as pairs of humans tend to converge to Aloof Cooperation, whereas pairs of machines converge to Efficient Cooperation. As a result, when people and these machines are paired together in the Block Dilemma, they often fail to cooperate when communication is restricted due to their strategic differences.

The results lead to the conclusion that the difference between human-machine behaviour and mindset is significant, which means that in order to ensure cooperation between the two entities it is required communication and understanding, which are the main barriers in human-machine cooperation.

2.1.4 Internet-of-Things (IoT)

Internet of Things (IoT) refers to the network of interconnected devices, such as sensors, that quickly exchange large amounts of data. The devices are connected through Machine-to-Machine communication (M2M) which is a standardized method for different machines to share information globally. IoT is commonly used for smart environments such as smart traffic management, healthcare and medicine, smart cities, self-driving vehicles, etc (Gazis 2021).

The number of devices connected to the internet is steadily multiplying, however global Internet coverage is still far from being achieved as more than half of the world's population lacks access to the Internet, even so, it could become a limitation for IoT. Another limitation is the availability of data, as IoT heavily relies on data by nature. The IoT sector is in high demand for scalable methods to handle, process, analyze and mine data in real-time as a large volume of data is exchanged daily through interconnected computing devices. Studies suggest that multiple fields of computer science such as cloud computing, cybersecurity, AI, Big Data, and data analytics can benefit from IoT (Gazis 2021).

2.2 Existent Approaches

2.2.1 OFERA Project

The OFERA (Open Framework for Embedded Robot Applications) project (OFERA n.d.) is an EU founded project whose objective is contribute to the faster growth of a competitive industry of small robots and robot components manufacturers, using highly resource-constrained devices (microcontrollers). OFERA purpose is to bridge the technological gap between the ROS platform for high-performance computational devices and the low-level libraries for microcontrollers, which will allow to microprocessors and microcontrollers to be mixed together seamlessly in any robotic system.

The name of the new robotic framework targeting embedded and deeply embedded robot components with extremely constrained computational resources is micro-ROS, which will be further explained in the next section.

2.2.1.1 Smart Warehouse micro-ROS Use-Case

The following work (Kołcon, Maciaś, and Malki 2019) depicts a use-case of a robot mobile platform that is capable of moving in a warehouse in order to perform scheduled tasks, this work is part of the OFERA project. While the actual use-case of this work might slightly differ from the scope of this thesis, the technologies used are quite relevant, and thus its analysis is very important.

The purpose of this work was to demonstrate how micro-ROS can be used in a low power wireless network environment, and checking how micro-ROS can operate in such environments, as providing good quality network connection over wireless in industrial application is very difficult. The robot was composed by the following components:

- Robot base;
  - PIAP Scout base;
- Autonomy module;

  - GNSS - satellite localisation system for outdoor use;
  - UWB - radio localisation system based on beacons, intended to use indoors.
- Robot control unit & Context system - for providing tasks to the robot, also to provide required context (world model) to execute the task, and is also the main place for monitoring system performance and task execution status.
- Radio link (WiFi) - main communication is using WiFi to communicate with the internet and also to send commands to the robot, as well as receiving status.
- Sensors and actuators;
  - Door opener - actuator mounted close to the warehouse door, used to control its opening and closing;
  - Laser scanner module - situated next to the road to provide information to the platform whether the road is empty;
  - Final effector system - a amber warning light used for indication of scenario status;
  - Humidity sensor - used to measure the humidity and temperature in the warehouse.

The control unit used for each sensor and actuator was a Olimex STM32-E407 board, which is a microcontroller and a reference platform for using micro-ROS.

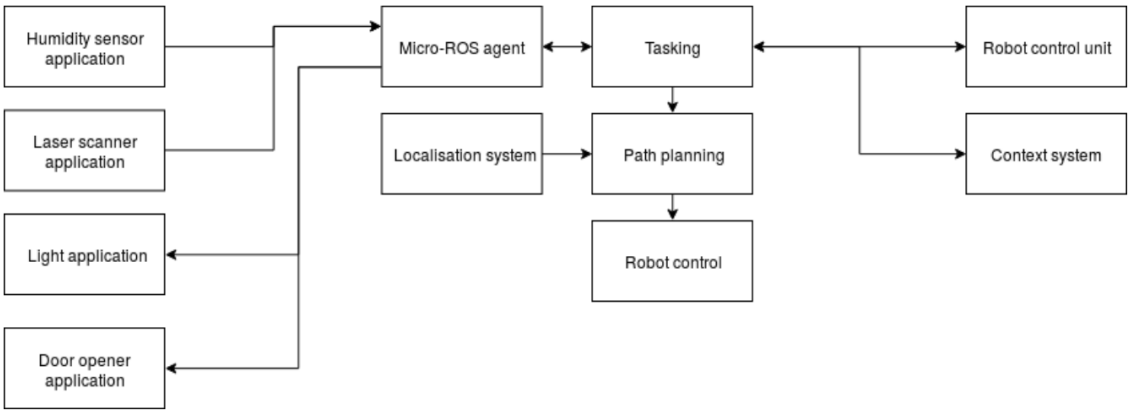

Figure 2.1: Software Architecture. (Kołcon, Maciaś, and Malki 2019)

In order to test micro-ROS features, the authors created missions for the robot to perform tasks according to the developed scenario. The scenario consists in a adapted laboratory hall which will simulate the warehouse. The robot mobile platform can go to the warehouse door in the meantime collecting data from the sensors like humidity sensor and lidar according to the planned tasks. After reaching the gate it can be opened using an actuator which uses 6LoWPAN and a micro-ROS stack for communication. To get to the final effector (light), the robot mobile base has to cross the road, so it must check whether crossing the road is safe.

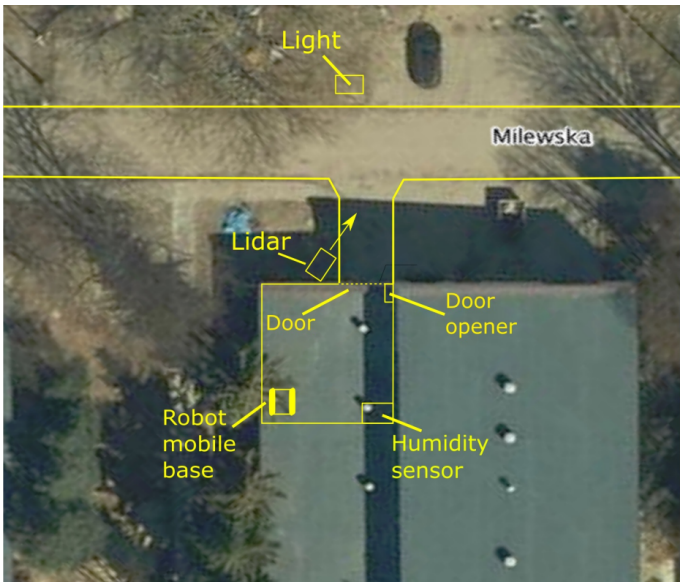

Figure 2.2: Warehouse Plan. (Kołcon, Maciaś, and Malki 2019)

According to the authors, this work showcased the possibilities of using micro-ROS tools effectively, and how simple it is to to combine basic elements such as sensors and actuators based on microcontrollers to a larger system such as the ROS/ROS2 environment (Kołcon, Maciaś, and Malki 2019).

Regarding this thesis, the analysis of this work proved to be very important due to the technologies and methods used to develop a smart environment with microcontrollers, the use of micro-ROS and the scenario used for testing demonstrated its potential.

2.2.1.2 micro-ROS default simulation environment release

The following work (Flores Muñoz and Muguruza Goenaga 2019), also belonging to the OFERA project, studied how could micro-ROS be simulated in a regular computer. Several tools were analysed in order to check their suitability for micro-ROS execution. The main objective was to execute a simulation which includes the complete set of micro-ROS layers, including NuttX RTOS, Micro XRCE-DDS middleware, RMW and the micro-ROS client library.

After some studies, the authors concluded that the simulator needed to fulfil the following requirements:

- UART communication with two available ports, where one is used for interfacing and the other one for micro-ROS Agent communication: Use of serial, 6LOWPAN or UDP/TCP through Ethernet to communicate with the micro-ROS Agent;
- RTC and Timer support: Basic functions of the RTOS, requires timing tools to work;
- Polling support: The communication on micro-ROS is asynchronous, so it is required a tool to check an incoming message;
- C++ support: It is heavily used by ROS2;

- 36 KB of RAM and 383 KB of flash: This requirement is based on the memory analysis made on the Olimex-STM32 E407 board and it shows the minimum memory size that is required for micro-ROS execution.

After some research, the authors found two options that could meet the requirements, NuttX official simulator and QEMU. NuttX simulator doesn't simulate any architecture, it only simulates the NuttX RTOS ifself. QEMU however, is capable of simulating CPU and microcontroller architectures.

The authors further analysed the NuttX official simulator and found the following limitations:

- Different RTOS behaviour: it does not simulate the MCU architecture;
- Limited support for ucLibc++ library: the library does not compile on X86/X64 architecture;
- Networking: the authors were able to achieve a local connection between the host PC and the simulator, but lost the internet connection.
- External Serial: It is a requirement for micro-ROS, but the authors were unable to use it, even though NuttX stated that it is possible to run a virtual serial port.

Given the previous limitations, the authors concluded that the NuttX official simulator was not suited for micro-ROS, and focused instead in QEMU. The master branch of QEMU's repository includes several MCU architectures, and also some other forks contained support for other MCU vendors. The fork that the authors considered to be most relevant was QEMU_STM32, which contains the development of the Cortex-M3 cores for STM32 MCU family.

Having chosen QEMU_STM32, the authors were able to run NuttX for the STM32F103 part number inside the simulator, and were also able to run the micro-ROS client. After some tests regarding communication with micro-ROS agent, it was concluded that it was unstable, and the authors found the following limitations:

- Memory management is not done correctly by the simulation;
- UART buffer management is not done correctly, workaround is to decrease the baudrate to the minimum;
- The option to connect a serial port of the MCU to a virtual port risks causing a overflow in the buffer when it is necessary to work with the system console. The alternative found be the authors was redirecting the serial port to a TCP server, which accepts external connections and provides stability to the system.

In a later deliverable (Flores Muñoz and Outerelo Gamarra 2019) the authors revisited the NuttX simulator and noticed some improvements regarding networking, but the lack of support for the C++ library still persists, which will make impossible to run micro-ROS applications using that API. The QEMU_STM32 did not receive significant improvements, but the authors tried to optimize micro-ROS usage and were able to obtain a slightly stability improvement, but there were still an unacceptable number of random crash situations.

Finally the authors abandoned the development of a simulator for micro-ROS, and concluded that despite QEMU_STM32 being the most promising one, the performance was weak and its development seemed abandoned because there was no relevant commits since

May of 2019. So at that moment, there were no valid options that could fit the authors requirements.

2.2.2 AutoC2X: Open-source software to realize V2X cooperative perception among autonomous vehicles

The following work (Tsukada et al. 2020) consists in the development of a software named AutoC2X which enables cooperative perception by using OpenC2X for Autoware based autonomous vehicles.

Cooperative perception is achieved by sharing information about detected objects, such as vehicles and pedestrians, among neighboring nodes using the Autoware software and following ETSI ITS standards. The system is designed by combining two open-source software, Autoware and OpenC2X, to enable cooperative perception, with the host node responsible for autonomous driving and the router handling the cooperative ITS function.

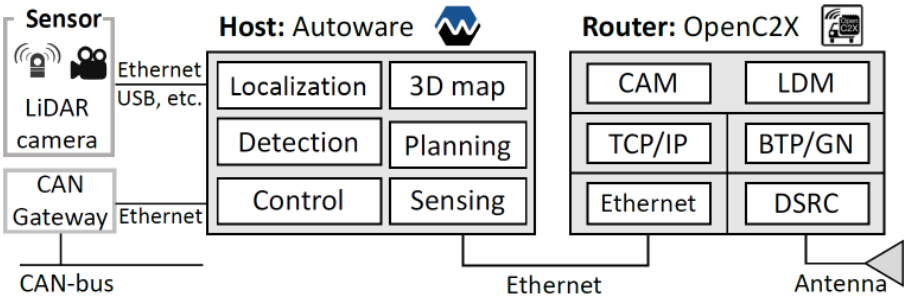

Figure 2.3: System model to integrate Autoware and OpenC2X. (Tsukada et al. 2020)

Autoware is responsible for sensing using 360-degree LiDAR scanners and cameras, as well as localization through scan matching between 3D point cloud maps and point cloud data from LiDAR scanners. The autonomous vehicle's actions such as acceleration, braking and steering is controlled via controller area network (CAN). OpenC2X is responsible for managing communications and handles almost the entire protocol stack in the ITS station architecture. Cooperative perception is enabled by sharing information through the use of cooperative awareness messages (CAM) which are basic safety-related V2V (Vehicle-to-Vehicle) messages. The received presence information is stored in a local database called local dynamic map (LDM) which provides support for various ITS applications .

AutoC2X is developed using C++ by extending both Autoware (version 1.11.1) and OpenC2X (standalone version 1.5). When the system is initiated, Autoware reads the 3D maps and begins receiving sensor data. The point cloud data from the LiDAR is used to localize the ego vehicle and detect neighboring objects. The position of the ego vehicle and the objects it detects are converted from a local coordinate system to a global coordinate system and transmitted to the OpenC2X router via TCP/IP over Ethernet.

For cooperative perception, Autoware detects the surrounding objects, and after the sensor fusion, the information of the detected objects is published in the topic of "/detection/objects". The coordinates are then transformed and sent to OpenC2X who adds the received detected object information to a queue and encodes the latest information in the queue into a proxy CAM. This information includes timestamp, latitude, longitude, speed, and ITS

station ID. The data of the objects derived from the V2X communication is visualized, after the coordinate transformation, in RViz which is a 3D visualization tool.

2.3 Simulation Ecosystem

2.3.1 COPADRIVe

COPADRIVe (Vieira et al. 2019) is a realistic co-simulation framework that integrates Gazebo, which is an advanced robotics simulator, with the OMNeT++ network simulator, over the ROS framework, while supporting the simulation of advanced cooperative applications such as platooning in realistic scenarios.

COPADRIVe simulation framework was built over the Veins simulator, a framework for running vehicular network simulations, and the Vanetza communications stack, which is a implementation of ETSI C-ITS protocol suite. For the integration of OMNeT++ with Gazebo, it is used ROS publish/subscribe mechanisms.

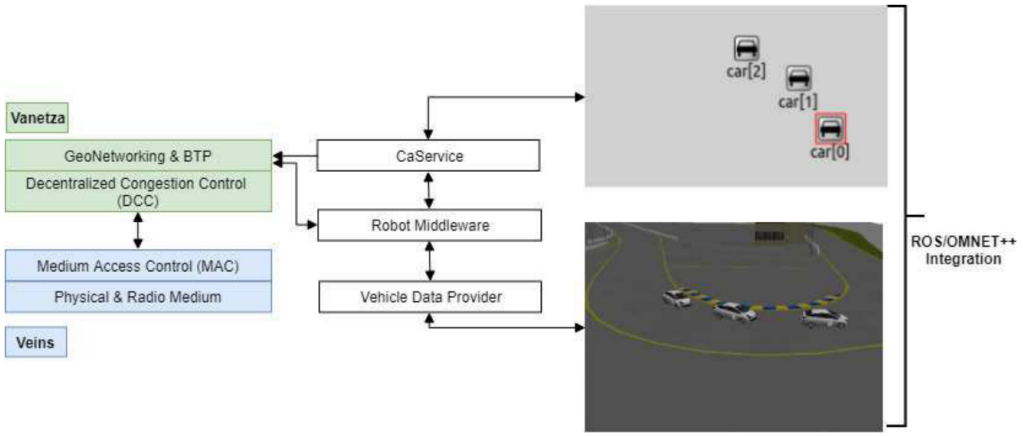

Figure 2.4: COPADRIVe Framework Architecture (Vieira et al. 2019)

2.3.2 AuNa

AuNa is a modularly integrated simulation framework for cooperative autonomous navigation. The main objective of the project that created AuNa was to integrate the simulation tools of robotics, communication and control namely ROS2, OMNeT++, and MATLAB to simulate cooperative driving scenarios (Teper et al. 2022). AuNA is based in COPADRIVe, as its framework is very similar, but with some improvements. Its framework is composed as follows:

- Robot Simulation: the simulation environment should allow multiple robots to be spawned dynamically, with each robot having a unique name or index. For each robot, a connection is established via ROS2 nodes, which allows the control of the robots and publishes sensor data. For the robot simulation, Gazebo provides plugins for implementing sensors and robot controls. By adding a namespace to the Gazebo plugins, each robot is given a unique name or index, so that each node and topic corresponds to its own robot.
- Communication Simulation: for the communication architecture, the framework uses OMNeT++ and Artery. In order to synchronize OMNeT++ and the other simulations

using ROS2 nodes, it was implemented a custom scheduler within OMNeT++ that runs the execution of OMNeT++ events, the synchronization between OMNeT++ and Gazebo, as well as the updates between the navigation system in ROS2 and communication module in OMNeT++ of each robot.

- Control Simulation: MATLAB and Simulink provide a complete feature set for designing control systems of any complexity. The 2022 release of MATLAB and Simulink provides ROS2 support with the ROS Toolbox, and thus, instead of implementing the controllers directly into the code, the authors opted to integrated MATLAB and Simulink into the framework.
- Navigation Systems: it is required a navigation system in order to the robots autonomously reach a target destination. For this purpose the authors considered using the Nav2 package in ROS2 which provides a collection of algorithms for localization, mapping, and navigation that are designed for dynamic environments (Teper et al. 2022).

In comparison with COPADRIVe, AuNa addresses some of its issues such as the efficiency in the synchronization of the robot simulation and the communication simulation, which addresses flexibility and event management and synchronization. Also unlike COPADRIVe, AuNa is based on ROS2, and the architecture of Artery is not adjusted to only support the platooning scenario.

2.4 Critique Analysis

In this section, an critique analysis will be carried out regarding the existing approaches mentioned previously. This analysis consists in reviewing which of the existing approaches are related and better suited for this thesis.

Starting with the OFERA project (OFERA n.d.), this approach is the most related one in terms of goals, as it aims to contribute to bridge the gap between high-performance devices and microcontrollers, in order to allow cooperation between them. The robotic framework developed in this work, micro-ROS, will be the main component for the development of this thesis. The smart warehouse use-case (Kołcon, Maciaś, and Malki 2019) is a fine example of how micro-ROS can be used in the development of a smart environment using microcontrollers as the main component. The technologies used could be a good example to follow for the approach of this thesis, although some small changes might be necessary to fit the scope. Since one of the objectives of this thesis is the simulation of the environment of the proposed use-case, the OFERA work regarding micro-ROS default simulation environment (Flores Muñoz and Muguruza Goenaga 2019) (Flores Muñoz and Outerelo Gamarra 2019) is very important as it is something that will be necessary in the implementation phase. Unfortunately, the authors were unable to obtain a stable simulation environment and abandoned this approach, but the tools they used could still serve as a basis later for the implementation.

The work regarding AutoC2X (Tsukada et al. 2020) despite being related to cooperative perception among autonomous vehicles, its approach is the most different from the desired in this thesis. Although AutoC2X was not developed with microcontrollers in its scope, the communications model used for the exchange of data could serve as an example, but might be difficult to replicate in a simulated environment.

The simulation approach introduced by the authors in the AuNa project (Teper et al. 2022) is mostly related to the objectives of this thesis regarding the development of a simulation environment. The tools used in this project coincide with most of the needs for simulation regarding this thesis, and thus, the framework introduced by AuNa will be the main reference for the simulation framework of this work, as it just lacks integration with micro-ROS, which is will be one of the goals to be implemented.

Chapter 3

Technologies

3.1 Robotics Framework

3.1.1 micro-ROS

micro-ROS is a framework for enabling ROS2 in microcontrollers. Microcontrollers are important because they are used in almost every robotic product. Typical reasons are:

- Hardware access;
- Hard, low-latency real-time;
- Power saving.

Some of its key features are, supporting all major ROS concepts, following the ROS2 architecture, and multi-RTOS (Real-Time Operating System) support (micro-ROS 2022).

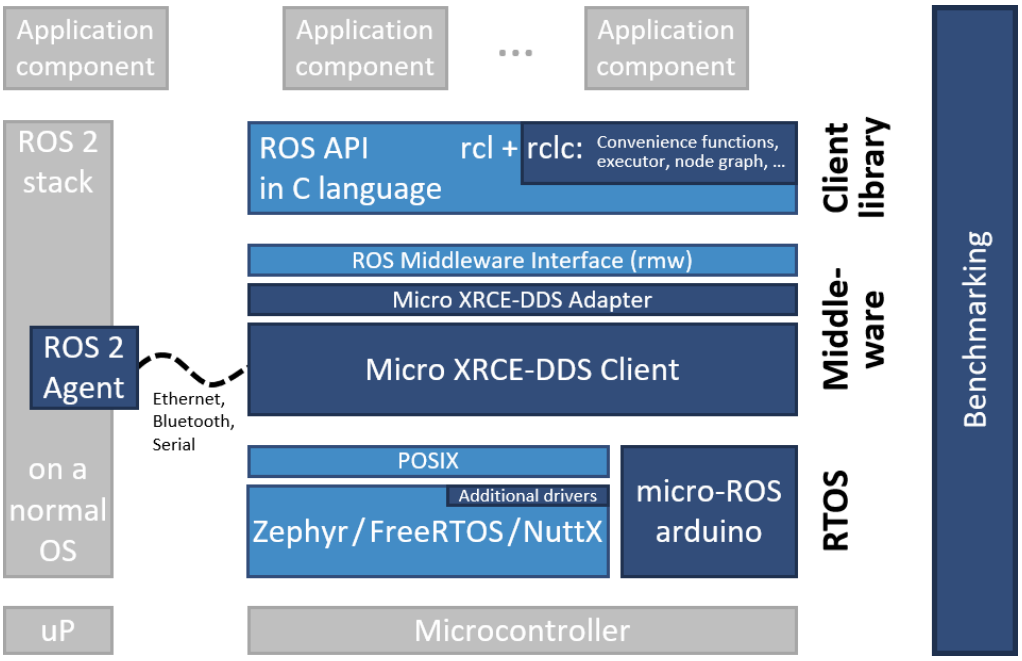

Figure 3.1: micro-ROS architecture. (micro-ROS 2022)

3.1.2 ROS2

ROS2 is the latest version of ROS, it is an open source software development kit for robotics applications (Macenski et al. 2022). At its core, ROS provides a message-passing system, often called "middleware" or "plumbing". Communication is one of the first needs to arise when implementing a new robot application, or really any software system that will interact with hardware. ROS's built-in and well-tested messaging system saves time by managing the details of communication between distributed nodes via an anonymous publish/subscribe pattern (openrobotics 2021).

ROS breaks complex systems down into many modular nodes, each node should be responsible for a single, module purpose (e.g. one node for controlling wheel motors, one node for controlling a laser range-finder, etc). Each node can send and receive data to other nodes via topics, services, actions, or parameters (openrobotics 2022).

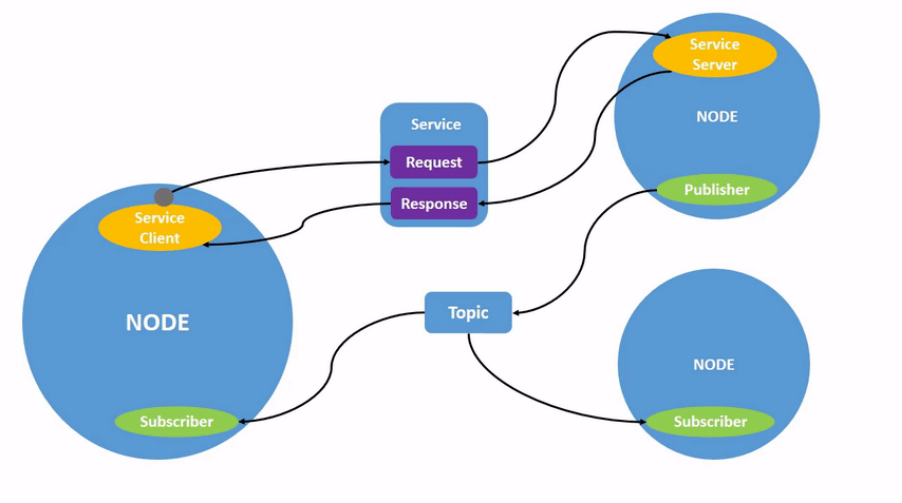

Figure 3.2: ROS2 nodes, topics and services. (openrobotics 2022)

3.1.3 Autoware

Autoware is an open-source software project for autonomous driving, it is built on ROS and enables commercial deployment of autonomous driving in a broad range of vehicles and applications. Autoware consists of all the functionality required for autonomous driving (i.e. perception, planning, control) in a modular architecture with crisply defined interfaces and APIs (The Autoware Foundation 2021).

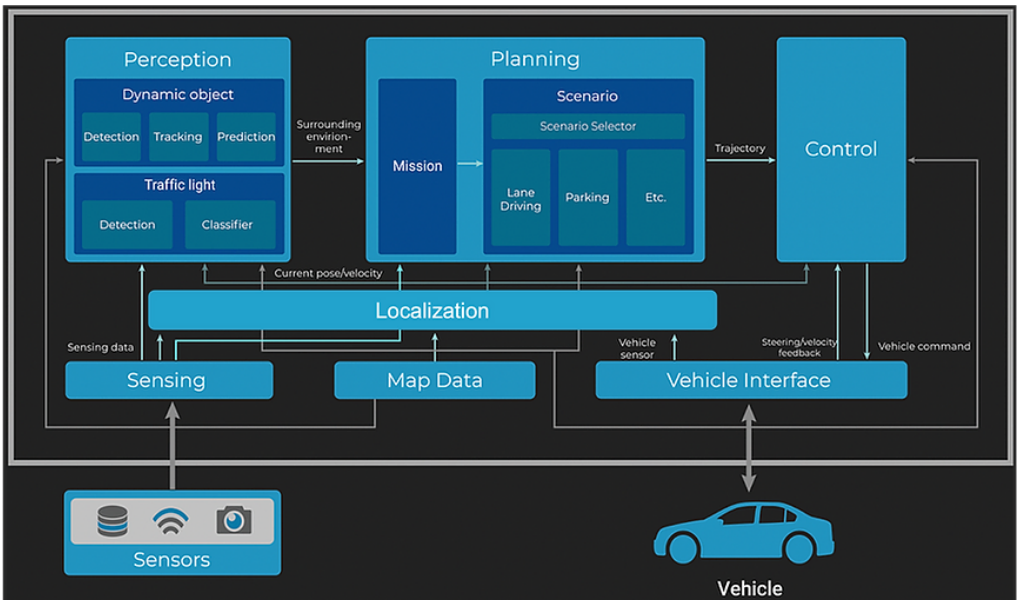

Figure 3.3: Autoware architecture. (The Autoware Foundation 2021)

3.2 Communications technologies

3.2.1 6LoWPAN

6LoWPAN, which was used in the previously mentioned work by (Kołcon, Maciaś, and Malki 2019), is an open standard defined in RFC 6282 by the Internet Engineering Task Force (IETF). It enables the use of IPv6 over IEEE 802.15.4 low-power wireless networks, which means that low-power devices with limited processing capabilities have their own IPv6 address, and are able to connect directly with the Internet using open standards. 6LowPAN characteristics make this technology ideal for markets such as home automation with sensors and actuators, street light monitoring and control, residential lighting, smart metering and generic IoT applications with Internet connected devices (Olsson 2014).

3.3 Simulation tools

3.3.1 QEMU

QEMU is a generic and open source machine emulator and virtualizer. Its most common use is for system emulation where it provides a virtual model of an entire machine (CPU, memory and emulated devices) to run a guest OS (Developers 2022). As previously seen in the work by (Flores Muñoz and Muguruza Goenaga 2019), QEMU is capable of simulating MCU architectures such as the STM32 ARM architecture.

3.3.2 OMNeT++

OMNeT++ is a simulation library and framework, primarily for building network simulators, it includes wired and wireless communication networks, on-chip networks, queueing networks, etc. Its main components are the following:

- Simulation kernel library (C++);

- The NED topology description language;
- Simulation IDE based on the Eclipse platform;
- Interactive simulation runtime GUI (Qtenv);
- Command-line interface for simulation execution (Cmdenv);
- Utilities (makefile creation tool, etc.);
- Documentation, sample simulations, etc (OpenSim Ltd. 2019).

3.3.3 Gazebo

Gazebo is an open-source 3D robotics simulator. By using multiple high-performance physics engines, it is capable of creating dynamic simulation replicating gravity, friction, torques, and any other real life conditions that could affect a simulation's success. Gazebo is a good choice when there is no access to actual robotic hardware or we want to test multiple robots simultaneously (FS Studio 2021).

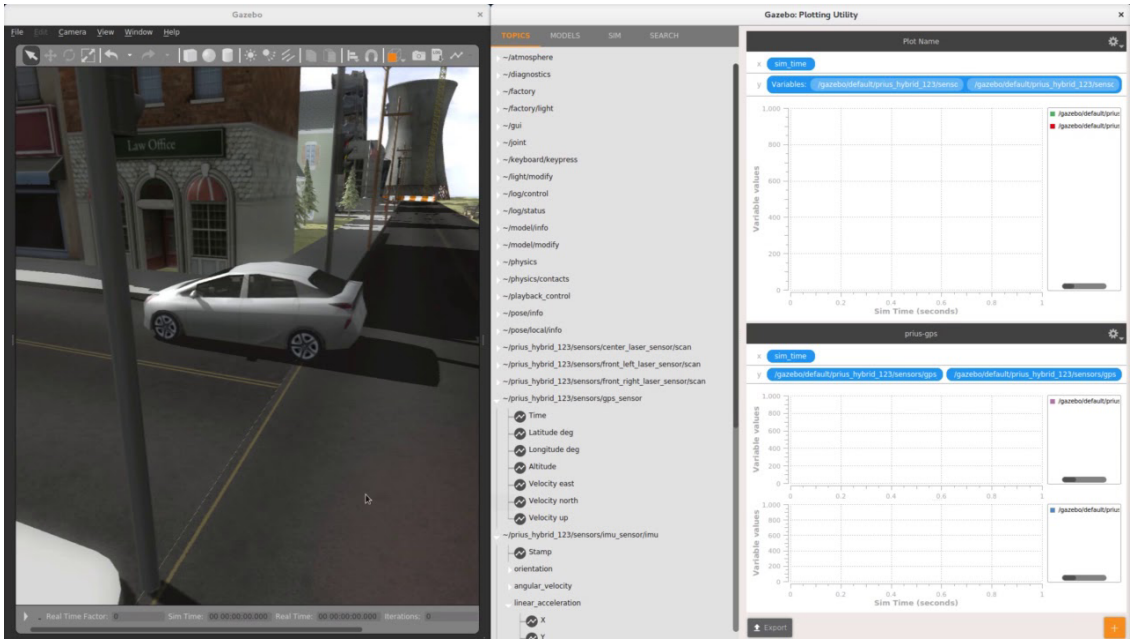

Figure 3.4: Gazebo vehicle and city simulation. (Gazebo 2017)

Chapter 4

Proposed Approach

4.1 Concept

As stated before, this thesis focuses on the investigation and development of a proof-of-concept co-simulation framework that can simulate dynamic environments which use embedded devices for perception. For that purpose, the use of micro-ROS will be used, since it is capable of integrating recent ROS tool-sets in embedded systems.

The concept will consist in following the approach introduced by AuNa, and adapt it to meet the needs for the goals of this thesis. Most of the tools used by AuNa such as ROS2, Gazebo and OMNet++, will also be used for the framework of this work with the following roles:

- ROS2: The core of the system, responsible for the integration and communication with all subsystems using the node, topic and service architecture. It is also responsible for the communication between the micro-ROS agent in the microcontroller and a normal OS using ROS2;
- Gazebo: Responsible for the simulation of the desired environment, vehicles and sensors;
- OMNeT++: Responsible for simulating the communications network. It will simulate the communication between the microcontrollers and the vehicle using 6LoWPAN or another communication technology.

In addition to the tools used by AuNa, it is planned to use QEMU for the simulation of the microcontroller, but is still subject to change as the authors (Flores Muñoz and Muguruza Goenaga 2019) were unable to achieve a stable simulation, and therefore it may be necessary to search for an alternative.

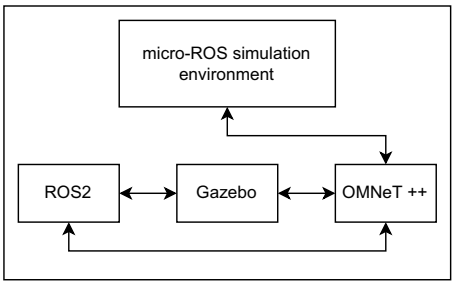

Figure 4.1: Concept architecture.

4.1.1 Use-Case/Scenario

To support the development of the proposed co-simulation framework, we are proposing a preliminary use-case scenario. The proposed scenario consists in the cooperation between an autonomous vehicle with a predefined route, and a microcontroller running micro-ROS. A common real world scenario is the blocking of a road due to road works or a natural phenomenon such as a fallen tree blocking the road. These type of situations cannot be predicted by a vehicle with a predetermined route, so the purpose of this use-case is to warn with precedence the vehicles approaching the road block so they can take the detour.

In practise, this will consist in using the microcontroller next to the road block to send a message to the approaching vehicles. The vehicles will then receive the message and obtain the information that the road is blocked and the current trajectory cannot be used, which will force them to use another route. This approach will still be further studied in the development phase to answer some doubts such as whether it should be the microcontroller to send the detour route, or should the vehicle be responsible for calculating it, as well as other questions.

The following figure (4.2) represents the use-case scenario described previously. In it, we can see a vehicle with a predefined route approaching a road that is blocked, which is then warned by the microcontroller to take the detour route represented in green.

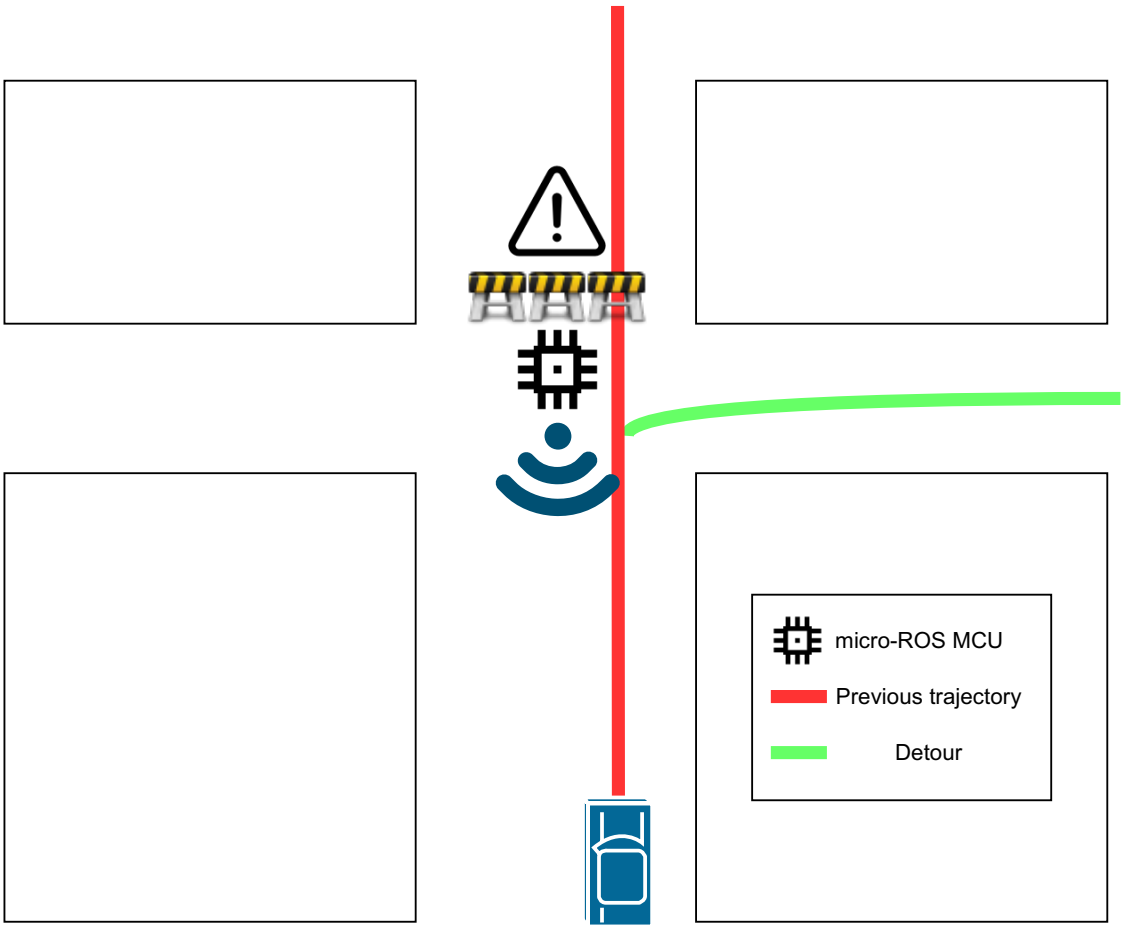

Figure 4.2: Use-Case scenario.

As stated previously, this a preliminary scenario which purpose is to serve as a first contact for the integration of the framework and technologies of the proposed approach. After its replication, more complex scenarios will be implemented in order to further explore human-machine cooperation.

4.2 Methodology

The implementation phase will be guided by the following methodology:

- micro-ROS simulation: investigating a simulation environment for micro-ROS, preferably, capable of simulating a microcontroller architecture. If QEMU does not meet the requirements, it will be necessary to search an alternative, even if it only simulates a RTOS capable of running micro-ROS. Hardware-in-the-loop could also be considered as a last resort;
- micro-ROS integration with AuNa simulation environment: integrating the micro-ROS simulation with AuNa, this is, establishing communications between the ROS2 nodes present in AuNA and the micro-ROS nodes. For this purpose, it will be used OMNeT++ for simulating the network. This also includes the investigation about the communication technology that will be used;
- Implementing the use-case scenario: Achieving the desired outcome of the preliminary scenario. This includes all the work necessary for the cooperation between the micro-ROS nodes and the vehicle (algorithms, sensors, etc.). Having the preliminary scenario replicated, more complex scenarios will be explored.

During the implementation, the state of art will still be explored, as well as the technologies related. Therefore, the proposed approach concept is still subject to change, which could also lead to changes in the methodology.

Chapter 5

Development Plan

5.1 Methodology

Regarding the development of this thesis as a whole, the methodology is organized as follows:

- Problem Statement and motivation: definition of the problem being addressed and the reasoning behind the importance of finding a solution within the context of this study;
- Goals and objectives definition: this involves outlining the goals related to the problem at hand and the objectives associated with the learning outcomes;
- State of the art and existent approaches research: consists in researching current methods and techniques in the scope of this thesis;
- Analysis and design: the design is developed through the examination of the current state of the field, including the frameworks, patterns, techniques, and technologies. This analysis is used to create a proof of concept application that embodies the chosen solution and achieves the desired objectives;
- Development and implementation: all the work involved in the development of framework and the execution of the use-case scenarios;
- Evaluation and experimentation: testing of developed outcome and evaluation in order to analyse if the initial objectives have been met.

5.1.1 Evaluation

In order to analyse whether the work was somehow successful, it is necessary to define how its evaluation will be carried out. The objective of this section is to explain the criteria for success and the standards of evidence that will be used to evaluate the results of the study. The methodology for the evaluation is the following:

micro-ROS simulation

The micro-ROS simulation will be considered successful if it is able to run the micro-ROS client stably after multiple tests. Actual emulation of a microcontroller architecture is not a critical requirement, and thus it will not alter the end result of the evaluation. Also, as mentioned previously, in case the micro-ROS simulation is considered unsuccessful due to its instability or other factors, Hardware-in-the-loop will be used as a last resort, which will also be acceptable for the simulation environment.

micro-ROS integration with AuNa environment

Regarding the integration with the AuNa environment, its success depends whether the micro-ROS simulation is able to communicate successfully with the ROS2 nodes present in the AuNa environment, and some sort of interaction with the two is achieved, such as a basic message for sharing information.

Use-case scenario simulation

The simulation environment will be tested using the use-case described previously and other complex scenarios, which will be considered successful if, after several tests, the expected outcome success rate is acceptable. This topic will be later revised due to the use-case scenario still being subject to change during the development phase.

5.2 Timeline

The methodology referred previously, will be guided by the following work plan. First is the problem statement and motivation, which is required for understanding the objectives and the thesis purpose. This task will start mid December 2022 and end in early January 2023.

During the definition of the problem statement, the definition of the goals and objectives will also be carried out as these tasks are related with each other. The duration will be the same as the previous task.

Having the goals and motivation defined, it will the be possible to start the research of the state of the art which will contemplate the research and comparison on the potential technologies, architectures and techniques that can be used to achieve the desired results. It will be carried out from early January 2023 to early February 2023.

The analysis and design will be based on the state of the art, and thus it will be developed alongside this task. It will begin mid January 2023 and end late February 2023.

The development and implementation is the most extensive task as it includes all the work necessary for the development of the framework in order to achieve the goals and objectives. This task will start during the design task in order to complement each other, and end late May or early June 2023, depending on the difficulties faced.

Lastly, the evaluation and experimentation will be carried out during the end of the previous task in order to analyse if the goals were met, and fix the encountered problems if possible.

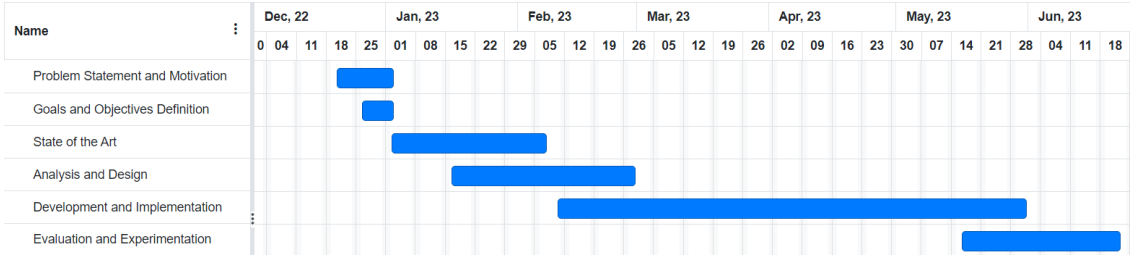

Figure 5.1: Project timeline Gantt chart.

Bibliography

Developers, The QEMU Project (2022). *About QEMU*. url: https://www.qemu.org/docs/master/about/index.html.

Flores Muñoz, Juan and Iñigo Muguruza Goenaga (June 2019). *micro-ROS default simulation environment release Initial Y1*. url: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5c54fa2ce&appId=PPGMS.

Flores Muñoz, Juan and Borja Outerelo Gamarra (Dec. 2019). *micro-ROS default simulation environment release Revised*. url: http://www.ofera.eu/storage/deliverables/M24/OFERA_13_D26_Micro-ROS_default_simulation_environments_release_revised__PU.pdf.

FS Studio (Jan. 2021). *Ros Gazebo: Everything you need to know*. url: https://roboticsimulationservices.com/ros-gazebo-everything-you-need-to-know/.

Gazebo (Oct. 2017). *Blog : Vehicle and city simulation*. url: https://classic.gazebosim.org/blog/car_sim.

Gazis, A (2021). "What is IoT? The Internet of Things explained". In: *Academia Letters*, p. 2.

Kołcon, Tomasz, Mateusz Maciaś, and Alexandre Malki (June 2019). *Smart warehouse micro-ROS use-case - initial*. url: http://www.ofera.eu/storage/deliverables/M18/OFERA_64_D67_Smart_warehouse_micro-ROS_use-case_release_Initial.pdf.

Macenski, Steven et al. (2022). "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66, eabm6074. doi: 10.1126/scirobotics.abm6074. url: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

micro-ROS (2022). url: https://micro.ros.org/.

OFERA (n.d.). *OFERA project: Open Framework for Embedded Robot Applications*. url: http://www.ofera.eu/.

Olsson, Jonas (2014). "6LoWPAN demystified". In: *Texas Instruments* 13, pp. 1–13.

openrobotics (2021). *The ros ecosystem*. url: https://www.ros.org/blog/ecosystem/.

– (2022). *Understanding nodes*. url: https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html.

OpenSim Ltd. (2019). *What is OMNeT++?* url: https://omnetpp.org/intro/.

Teper, Harun et al. (2022). "AuNa: Modularly Integrated Simulation Framework for Cooperative Autonomous Navigation". In: *arXiv preprint arXiv:2207.05544*.

Thandavarayan, Gokulnath, Miguel Sepulcre, and Javier Gozalvez (2020). "Cooperative Perception for Connected and Automated Vehicles: Evaluation and Impact of Congestion Control". In: *IEEE Access* 8, pp. 197665–197683. doi: 10.1109/ACCESS.2020.3035119.

The Autoware Foundation (2021). *Autoware Overview*. url: https://www.autoware.org/autoware.

Tsukada, Manabu et al. (2020). "AutoC2X: Open-source software to realize V2X cooperative perception among autonomous vehicles". In: *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pp. 1–6. doi: 10.1109/VTC2020-Fall49728.2020.9348525.

Vieira, Bruno et al. (2019). "COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications". In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 1–6. doi: 10.1109/ICCVE45908.2019.8965161.

Whiting, Tim et al. (2021). "Confronting barriers to human-robot cooperation: balancing efficiency and risk in machine behavior". In: *iScience* 24.1, p. 101963. issn: 2589-0042. doi: https://doi.org/10.1016/j.isci.2020.101963. url: https://www.sciencedirect.com/science/article/pii/S2589004220311603.

Wolter, Diedrich and Alexandra Kirsch (Aug. 2017). "Smart Environments: What is it and Why Should We Care?" In: *KI - Künstliche Intelligenz* 31.3, pp. 231–237. issn: 1610-1987. doi: 10.1007/s13218-017-0498-4. url: https://doi.org/10.1007/s13218-017-0498-4.

Xiong, Wei et al. (Mar. 2022). "Challenges of human—machine collaboration in risky decision-making". In: *Frontiers of Engineering Management* 9.1, pp. 89–103. issn: 2096-0255. doi: 10.1007/s42524-021-0182-0. url: https://doi.org/10.1007/s42524-021-0182-0.