Summary

As light vehicles face security challenges in urban environments, such as visibility issues, especially during adverse weather conditions, and the complexity of navigation in large cities, as well as interactions with other road users, the implementation of this embedded platform will help address and mitigate these issues effectively.

The project consists in the implementation of different sensors and actuators on a custom-designed PCB integrated with a NVIDIA Jetson Nano. Development environment will be Linux, with programming conducted in C/C++ and Python.

Detection and visualization of the surroundings will be performed using a LiDAR and a camera.

Localization and positioning of the vehicle will be determined by GPS, accelerometers and gyroscopes, along with additional information obtained by proximity and touch sensors.

Accident prevention will be facilitated by the use of a brake and motor manipulation, supplemented by alerting the user with lights and sound emitted from LEDs, displays, and alarms.

Safety enhancement features include obstacle detection of various kinds, such as pedestrians, cars, or urban objects, and bike lane departure warning. Additionally, the project has huge scalability, as it can be implemented in any kind of light vehicle, such as bikes, motorbikes, or electric scooters, with minimal adjustments required.

Potential future developments, such as the addition of more advanced sensors, integration of machine learning algorithms for predictive analytics, establishment of communication protocols between vehicles, or integration with an IoT application, could further enhance the capabilities and adaptability of the embedded platform.

Descriptors

* Embedded Linux
* IoT
* PCB Design
* Embedded system
* Edge Computing

Index

[1. Introduction 11](#_Toc169028460)

[1.1 Justification of the project 11](#_Toc169028461)

[1.2 Reach of the project 12](#_Toc169028462)

[1.3 State of Art 14](#_Toc169028463)

[1.3.1 Edge computing and Cloud computing 14](#_Toc169028464)

[1.3.2 Internet of Things 15](#_Toc169028465)

[1.3.3 Computation on vehicles 17](#_Toc169028466)

[1.3.4 Current state 18](#_Toc169028467)

[2. Objectives 19](#_Toc169028468)

[2.1 Main objective 19](#_Toc169028469)

[2.2 Secondary objectives 19](#_Toc169028470)

[3. Work Breakdown Structure 21](#_Toc169028471)

[4. Project Design 22](#_Toc169028472)

[4.1 Embedded platform 22](#_Toc169028473)

[4.1.1 STM32 F041RE Series 22](#_Toc169028474)

[4.1.2 Raspberry Pi 4 23](#_Toc169028475)

[4.1.3 NVIDIA Jetson Nano 2GB Development Kit 24](#_Toc169028476)

[4.2 LiDAR 25](#_Toc169028477)

[4.2.1 Cygbot CygLIDAR D1 25](#_Toc169028478)

[4.2.2 Lightware SF45/B 26](#_Toc169028479)

[4.3 Camera 26](#_Toc169028480)

[4.3.1 Arducam Synchronized Dual Camera 26](#_Toc169028481)

[4.3.2 ADLINK NEON-202A-JNX Kit 27](#_Toc169028482)

[4.3.3 Logitech C170 28](#_Toc169028483)

[4.4 GPS 28](#_Toc169028484)

[4.4.1 Parallax GPS Receiver Module 29](#_Toc169028485)

[4.5 Light Mobility Vehicle 29](#_Toc169028486)

[4.6 Software selection 30](#_Toc169028487)

[4.6.1 Kubernetes 30](#_Toc169028488)

[4.6.2 Docker 30](#_Toc169028489)

[4.6.3 Node-Red 31](#_Toc169028490)

[4.6.4 InfluxDB 31](#_Toc169028491)

[4.6.5 Mosquitto 32](#_Toc169028492)

[4.6.6 Kafka 33](#_Toc169028493)

[4.6.7 Grafana 33](#_Toc169028494)

[4.6.8 NVIDIA DeepStream SDK 34](#_Toc169028495)

[5. Project Development 37](#_Toc169028496)

[5.1 Software Development 37](#_Toc169028497)

[5.1.1 GPS Programming 37](#_Toc169028498)

[5.1.2 AI Vision Programming 40](#_Toc169028499)

[5.1.3 LiDAR Programming 42](#_Toc169028500)

[5.1.4 I2C Communication Protocol 46](#_Toc169028501)

[5.2 Hardware Development 51](#_Toc169028502)

[5.2.1 PCB Design 51](#_Toc169028503)

[5.2.2 3D Pieces Design 60](#_Toc169028504)

[5.2.3 Prototype Fabrication 65](#_Toc169028505)

[6. Validation 67](#_Toc169028506)

[7. Budget 68](#_Toc169028507)

[7.1 Hardware Estimated Costs 68](#_Toc169028508)

[7.2 Software Estimated Costs 69](#_Toc169028509)

[7.3 Human Resources Estimated Costs 69](#_Toc169028510)

[8. Ethical Validation 71](#_Toc169028511)

[9. Closure and Future Work 73](#_Toc169028512)

[9.1 Closure 73](#_Toc169028513)

[9.2 Future Work 74](#_Toc169028514)

[10. Bibliografía 76](#_Toc169028515)

Table Index

[Table 5.1: Raspberry Pi Pico Pinout 55](#_Toc169028516)

[Table 5.2: RGB LED Colour guide. 56](#_Toc169028517)

[Table 5.3: Buzzer frequencies. 57](#_Toc169028518)

[Table 5.4: Switch Functionality Table. 58](#_Toc169028519)

[Table 5.5:Assembly Components 60](#_Toc169028520)

[Table 5.6: 3D Printing Parameters. 65](#_Toc169028521)

[Table 5.7: Laser Cutting Settings 65](#_Toc169028522)

[Table 7.1: Hardware Costs 68](#_Toc169028523)

[Table 7.2: Software Costs 69](#_Toc169028524)

[Table 7.3: Human Resources Costs 69](#_Toc169028525)

Figure Index

[Figure 1.1: Edge Computer Paradigm. Extracted from [5]. 14](#_Toc169028526)

[Figure 1.2: Cloud Computing Paradigm. Extracted from [5]. 15](#_Toc169028527)

[Figure 1.3: IoT device number estimation for 2030. Extracted from [11]. 16](#_Toc169028528)

[Figure 1.4: Working of Blockchain. Extracted from [13]. 16](#_Toc169028529)

[Figure 1.5: Structure of an autonomous vehicle that makes use of edge-computing devices. Extracted from [42]. 17](#_Toc169028530)

[Figure 3.1: Diagram of workflow. 21](#_Toc169028531)

[Figure 4.1. STM32 F041RE development board. 22](#_Toc169028532)

[Figure 4.2. Raspberry 4 Model B. 23](#_Toc169028533)

[Figure 4.3. Jetson Nano 2GB development board. 24](#_Toc169028534)

[Figure 4.4: Determination of Distances In LiDAR 25](#_Toc169028535)

[Figure 4.5: CygLiDAR D1. 25](#_Toc169028536)

[Figure 4.6: Lightware SF45/B. 26](#_Toc169028537)

[Figure 4.7: Arducam DualCamera Kit. 27](#_Toc169028538)

[Figure 4.8: ADLINK NEON 202A. 27](#_Toc169028539)

[Figure 4.9: Logitech C170 webcam. 28](#_Toc169028540)

[Figure 4.10: GPS Determination. 28](#_Toc169028541)

[Figure 4.11: parallax GPS Receiver Module. 29](#_Toc169028542)

[Figure 4.12: Ebike Conor Wrc Shake Fs Deore. 29](#_Toc169028543)

[Figure 4.13: Kubernetes Container Deployment Diagram. 30](#_Toc169028544)

[Figure 4.14: Docker Container Deployment Diagram. 31](#_Toc169028545)

[Figure 4.15: Node-Red Example Illustration. 31](#_Toc169028546)

[Figure 4.16: InfluxDB Logo. 32](#_Toc169028547)

[Figure 4.17: MQTT Working Diagram. 32](#_Toc169028548)

[Figure 4.18: Kafka Working Diagram. 33](#_Toc169028549)

[Figure 4.19: Grafana Example Dashboard. 33](#_Toc169028550)

[Figure 4.20: TAO Overview. Extracted from [36]. 34](#_Toc169028551)

[Figure 4.21: Pruning Efficiency. Extracted from [36]. 35](#_Toc169028552)

[Figure 4.22: DashCamNet Pruned Model Example of Use. Extracted from [36] [43]. 36](#_Toc169028553)

[Figure 5.1: NMEA0183 Format Structure. 37](#_Toc169028554)

[Figure 5.2: I2C Communication Protocol. 46](#_Toc169028555)

[Figure 5.3: i2c Custom Communication. 47](#_Toc169028556)

[Figure 5.4: Custom Communication Example. 47](#_Toc169028557)

[Figure 5.5: PCB Power Supply Schematic 51](#_Toc169028558)

[Figure 5.6: ULN2003 Darlington Transistor Array 52](#_Toc169028559)

[Figure 5.7: LED Matrix Schematic 53](#_Toc169028560)

[Figure 5.8: PCB connection Schematic. 54](#_Toc169028561)

[Figure 5.9: RGB LED Schematic. 56](#_Toc169028562)

[Figure 5.10: Buzzer Schematic. 57](#_Toc169028563)

[Figure 5.11: Switch Schematic 58](#_Toc169028564)

[Figure 5.12 I2C PCB Connection Schematic. 59](#_Toc169028565)

[Figure 5.13: Design of the PCB Box. 60](#_Toc169028566)

[Figure 5.14: Adapter design. 61](#_Toc169028567)

[Figure 5.15: Design of the Clamp. 61](#_Toc169028568)

[Figure 5.16: Handlebar Assembly 62](#_Toc169028569)

[Figure 5.17: Left Frame Wrap Design. 63](#_Toc169028570)

[Figure 5.18: Right Frame Wrap Design. 63](#_Toc169028571)

[Figure 5.19: Design of the Box for the provisioning of the Intelligent Systems. 64](#_Toc169028572)

[Figure 5.20: Assembly of the Box for the provisioning of intelligent systems. 64](#_Toc169028573)

Lists Index

[List 5.1: GPS Message Reading Function 38](#_Toc169028574)

[List 5.2: NMEA Message DEcoding Function 39](#_Toc169028575)

[List 5.3: Deepstream Configuration 41](#_Toc169028576)

[List 5.4: LiDAR Configuration. 43](#_Toc169028577)

[List 5.5: LiDAR Data Reception 44](#_Toc169028578)

[List 5.6: I2C Communication Settings. 48](#_Toc169028579)

[List 5.7: I2C Data Reception and Processing. 49](#_Toc169028580)

[List 5.8: Matrix Update Function 50](#_Toc169028581)

# Introduction

## Justification of the project

In the past years, the demand for innovative solutions to enhance the safety, efficiency, and user experience of light mobility vehicles, from now on referred as LMV, has skyrocketed. This has caused the need of an embedded platform capable of providing intelligent systems at the same time that is able to control different peripherals. This project aims to address that need of a sophisticated embedded platform, tailoring an embedded system with the latest advancements in artificial Intelligence (AI), edge computing and hardware design to create a safety device for LMV.

DeustoTech, the University of Deusto research centre, has been developing many research projects related to the use of AI or edge computing, between many others, on the last few years. Within DeustoTech, the Smart-Mobility research team is responsible of the development of projects related to societies and industries needs in the mobility and transportation areas, as stated in their website:

“DeustoTech Mobility tiene como cometido dar respuesta a las necesidades actuales y futuras de la sociedad y la industria en el sector de la movilidad y el transporte, desde una apuesta decidida por la investigación en el campo de las nuevas tecnologías.” [1]

(Traduced) DeustoTech Mobility's mission is to respond to the current and future needs of society and industry in the mobility and transport sector, with a firm commitment to research in the field of new technologies.

This project emerges as a new research area, as the response to European Union’s announcement about new research projects. This particular project lays the ground for future research projects, specialized on edge computing and accident prevention.

This document has been authored exclusively by the writer within the scope of their current thesis project. While the content draws from work undertaken during their internship at the research centre DeustoTech, DeustoTech has no involvement in the creation of this document and bears no responsibility for its content, accuracy, or veracity. Several activities outlined in this project are also integral to the Smart-Mobility project.

## Reach of the project

As stated in the project title, the expected scope of the project is to provide intelligent systems for light mobility vehicles, enhancing their security and providing users with valuable information while using the system.

As previously mentioned, these intelligent systems will be implemented using an embedded system, which will be connected to selected sensors. This approach is known as edge computing, distinguishing it from cloud computing because real-time data processing occurs within the embedded platform. The benefits of this approach are significant, as it enables the device to not only collect data, as IoT devices traditionally do, but also to respond to changes occurring in its presence.

The project will be divided into phases based on their end goal: software or hardware development.

The first phase of the project focus on software programming. The phase will begin by downloading the SDK and setting up a comfortable work environment in the board, Linux knowledge will be needed, as Linux-based systems are open-source and freely available with a wide range of configuration and customization options. Nvidia’s Linux based SDK is the best option as it comes ready to exploit the board’s computational power for an optimal performance.

As the camera used in the lacks video processing or AI software, the use of an AI or machine learning algorithm will be necessary. For this purpose, Nvidia provides a SDK that enables the use of Artificial Intelligence for image and video processing. After installing and properly configuring the necessary drivers and software, and given the training of AI is not the subject of this work, a pruned or pre-trained model will be used for the deployment of the application.

Then, in order to program al the sensors such as the LiDAR, GPS module, or camera, will be needed to obtain knowledge in python. Thanks to the simple syntax and easy readability python offers, the learning of this high-level programming language will be rapid. However, its interpreted nature can lead to slower execution speeds or performance issues, as it is not as fast as compiled languages like C or C++.

Now that all the sensors receive data, it needs to be processed and stored. For this purpose, a popular and open-source platform will be used: Docker. This platform allows the user to build, deploy and run software packets. This software packets, or containers, will act in a way similar to virtual machines, and interact with the local host and between each other.

This service will help to stablish communication protocols and services that allow the data flow and storage. To achieve this, a thorough research and determination of the optimal services will be necessary, with the aim of improving the efficiency of the system. Communication brokers like Mosquitto, Amazon SQS or Kafka are alternatives that are plausible, but a further study on the optimal ones will be needed. Similarly, for the data bases, it will also be needed a determination of the optimal choice between a wide range of options like MongoDB, MariaDB or InfluxDB, among others.

All the deployment of services will be made on a browser-based flow editing container, Node-Red. This will facilitate an easy flow programming, debugging and execution.

The second phase of the project will focus on the hardware design. Due to the fabrication process constrains and delivery time considerations, the PCB design and fabrication will be prioritized. For the design, some constraints and considerations must be taken into account to design a LED matrix that allows the communication with the other board.

Another aspect of hardware design involves the creatin of an structure that is able to securely hold the components, ensuring the sensors operation and protecting them from adverse meteorological conditions. With this in mind, a modelling of the previously selected bike will be done first, with the objective of easing the design of the structures. Two different mounts will be required: one for holding the PCB at the handlebar and other for holding the rest of components. Once the designs are finished, all the parts will be fabricated using 3D printing technology.

Once the two main phases have ended, a final phase will be initiated. This phase will start by mounting all the 3D-printed parts onto the bicycle, making the necessary adaptations for improving the stability. Subsequently, all the Hardware components will be mounted and connected, and their functionality will be tested for ensuring a correct working.

The device will be validated by doing real life tests in the streets, with adjustments made in function of the obtained results. These tests will be recorded using a point of view (POV) camera, with further review to identify possible weak points, issues, or bugs.

While this project is carried on, the documentation of the project will be completed, collecting thoroughly all the steps taken to obtain the final product.

The documentation will consist of:

* This document, as the memory of all the work done
* PCB schematics
* 3D Printed parts drawings
* LiDAR python code
* Deepstream configuration code
* Manual of use.

## State of Art

In urban mobility scenarios, light mobility vehicles (LMVs) have taken centre stage. Motorcycles, bicycles, e-scooters, and the more recent electric scooters are some examples found in urban cores, harmoniously coexisting within traffic flows. According to the Spanish Bicycle Barometer in 2022, 57.1% of the population (aged 14 to 70) admits to occasionally using bicycles [2]. Additionally, 30% of the population uses them on a daily basis, according to the same source. According to the European road safety observatory in 2021, concluded that over 80% of e-scooter rider deaths and 50% of trauma patients’ injuries resulted from a crash involving a heavier motor vehicle [3].

This state of the art will provide insights into the latest discoveries and improvements in embedded systems specialized in enhancing the functionality and security of LMVs, addressing the increasing need for embedded platforms to provide intelligent services on these vehicles. In this part, the development of edge computing devices will be described, alongside their implementation to the Internet of Things (IoT) and specifically to vehicles.

### Edge computing and Cloud computing

Edge computing differs totally from the concept of traditional computing. Its core idea is to make computing closer to the source of data [4]. Data is constantly produced at the edge of the network, hence the necessity of accelerating the process and optimising the efficiency of the system. [5]

Diagrama

Descripción generada automáticamenteAccording to Satyanarayanan, lecturer at Carnegie Mellon university in the United States: ‘‘Edge computing is a new computing model that deploys computing and storage resources (such as cloudlets, micro data centres, or fog nodes, etc.) at the edge of the network closer to mobile devices or sensors’’ [4]

Figure 1.1: Edge Computer Paradigm. Extracted from [5].

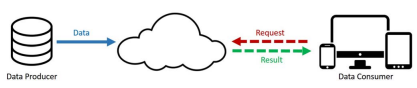
The cloud is just an allegory for the internet [6]. Cloud computing is an internet-based computing where processing resources and data are shared with other devices on demand [7]. This means that instead of storing information on the users storage unit, the information is storage in a specialised server, accessed by the internet.

Figure 1.2: Cloud Computing Paradigm. Extracted from [5].

But why is edge computing needed? A study in 2013 shows that in the future, autonomous vehicles will generate approximately 2 Petabytes of data in their entire life, an approximate of 1 Gigabyte per second [8]. To obtain an autonomous navigation, decisions must be made each millisecond, interpreting and reacting in real-time, to ensure that the correct decision is being taken. If instead of acting on the spot, the information was sent to the IoT cloud and waited for a response, much more time would elapse before any action was taken. This concept is called: *Push from the cloud services.*

On the other hand, almost all electrical devices will be part of the IoT. Take for example a humidity sensor. If the humidity sensor sent all the raw data to the cloud, and the other thousands of devices did the same, the cloud would become full of unusable raw data, extending its bandwidth. Instead, when the cloud decides to take information from the devices, another concept appears: *Pull from IoT*

The same way a device can push information, the information can be pulled. Or, in other words, is changed from consumer to producer.

### Internet of Things

The IoT refers to the net of tangible devices connected to the internet, that can collect and share information between them, eliminating the need of a human-to-human or human-to-machine communication [9]

A report by Cisco in 2015 estimated a total of 50 billion IoT devices for the start of 2020 [10], while in the actuality, in 2024, we count with just above 18 billion endpoints connected to the IoT [11]. TransformaInsights, the source of the actual IoT endpoints, also expected a total of 40 billion of IoT endpoints for 2030, with China being the country with most of them (8.9 billion) [11].

Gráfico, Gráfico de barras

Descripción generada automáticamenteThe huge versatility and scalability of IoT has had a huge impact on the society, transforming diverse areas as healthcare, manufacture, or logistics, for example [12]. However, IoT security might not be the best [13], as the centralized systems in which the IoT works is not the best security available. For example, an IoT device as the one that could be in an autonomous vehicle wouldn’t count with the same security as other endpoint devices as smartphones or tablets, being much more vulnerable to attacks [14].

Figure 1.3: IoT device number estimation for 2030. Extracted from [11].

With the purpose of creating a secure IoT, many alternatives have emerged [14]. The most known one, blockchain, is commonly used for online transactions, but can have an infinite variety of uses.

“Blockchain will provide enhanced device security through implementation of a specific authentication scheme by applying a Quantum Random Number to eliminate possibility of hacks occurring.” [15].

Diagrama

Descripción generada automáticamenteIt consists of sequence of blocks which is linked, holding the transaction that are time stamped secured by public-key cryptography and also verified by community of network [16].

Figure 1.4: Working of Blockchain. Extracted from [13].

### Computation on vehicles

According to reports by the World Health Organization in 2018, an estimate of 1.35 million people lose their life and other several millions of people are injured every year in traffic accidents, with material loses ranging from 2-4 trillion dollars [17].

This situation has inspired companies to develop edge-computing based safety systems, with companies like Tesla or Waymo going further developing “Full self-driving” technologies. Traffic is a complex, long stablished social domain, where human attitudes like “yielding” cannot be analysed by any means of computing [18].

Diagrama

Descripción generada automáticamenteThe addition of sensing and computing equipment to obtain an autonomous navigation system leads to the overall increase in the cost of the vehicle and its energy consumption [19].

Figure 1.5: Structure of an autonomous vehicle that makes use of edge-computing devices. Extracted from [42].

“Autopilot is an advanced driver assistance system that enhances safety and convenience behind the wheel. When used properly, Autopilot reduces your overall workload as a driver. Each new Tesla vehicle is equipped with multiple external cameras and powerful vision processing to provide an additional layer of safety. All vehicles built for the North American market now use our camera-based Tesla Vision to deliver Autopilot features, rather than radar.” [20]

As Tesla’s autopilot description points, its autopilot is a driver assistance system, rather than a full autonomous driving system. This belief has led to thousands of deaths, as drivers fully delegated their driving responsibilities to the copilot, as the system “discourages” drivers from staying involved [21].

### Current state

As technology advances, cities, vehicles and overall, the surrounding environment, keeps getting smarter. Edge computing techniques like point cloud compression for autonomous driving [22] and full autonomous driving provide vehicles like cars the necessary tools to become a much safer method of transportation.

Nowadays, some light mobility vehicles, like bikes or e-scooters, are IoT devices. They provide data for the user, like ambient temperature, speed, or even gases, as they incorporate specialised sensors. But the field of safety has not been explored, as none of the alternatives include any type of sensor that could help measuring potentially hazardous situations nor include any kind of alerting system for the user.

Currently, the safety on LMV field hasn’t been neither explored nor commercialized. This creates an opportunity for small and independent researchers to create their devices and present their alternatives to various administrations or companies. The tools have already been developed, but their implementation to LMV has not occurred yet due to the pricey alternatives edge computing and field sensors like LiDARs or smart cameras offer.

The lack of alternatives indicates a problematic situation in the development of alternatives for the security of light mobility vehicles, as the investment in this research field has not been adequate up to this moment. European Union’s announcement can be the starting point for the development of many of these devices, enhancing LMV’s security and helping to save thousands of lives.

# Objectives

The following part will discuss both the principal and secondary objectives of the project.

## Main objective

As previously mentioned, the lack of innovation and research in the field of the safety in Ligh Mobility Vehicles creates an opportunity for small research groups to create and develop their innovative ideas.

Single Board Computers or SBCs are a great alternative for robotics enthusiasts, as they provide a high software personalization level combined with the facility of connection with different peripherals, modules, and sensors. This allows the creation of IoT platforms capable of edge and cloud computing, with the enough computational power to process multiple data sources at the same time.

Moving forward, the project will focus on the potential applications of SBCs for enhancing the safety in Light Mobility Devices. Exploring the potential of cutting-edge technologies like LiDARs and smart cameras that can redefine the safety panorama in vehicles, with the selection and programming of them, and most importantly, designing and implementing a customized embedded platform that integrates all of the hardware and software.

Addressing all the potential SBCs hold and utilizing all the potential of new technologies, the aim of the project is to create a pioneer solution in safety and efficiency in Light Mobility vehicles, contributing to the evolution of the smart-mobility and creating a new path for more sustainable transportation methods.

## Secondary objectives

To achieve the main objective, several partial objectives have been set. These objectives are designed to steer the development of the project and complete the flaws of the main objective.

1. Study and selection of the hardware alternatives.

The development of this project will be marked by the evaluation of the technical characteristics, performance, and suitability of different hardware components, along with the evaluation of the costs associated with the implementation of the selected technologies.

1. Establishment of an internal server.

The establishment of a server will help to establish communication protocols like MQTT or Kafka between services and sensors.

1. Sensor and communications programming

The integration of the software will be stablished by the limitations of the System on Module (SOM) selected, along with the constrains of the selected hardware components. High level programming languages will be used for the convenience of the easiness of learning and coding, and low-level languages will be used to improve the response time and efficiency of the system.

1. PCB design and manufacturing

The creation of a PCB comes with the design, manufacturing and soldering of the different components required for a functional board. Necessity of a thought process for creating a information dashboard for the user.

1. Vehicle support design and manufacturing

Given that the hardware needs a strong and stable support structure to be mounted on, the design process must be aligned to the specifications of the selected vehicle. As this is a prototype, a cheap and quick way to manufacturing will be selected.

# Work Breakdown Structure

Diagrama

Descripción generada automáticamenteIn this section, the work plan is detailed in figure 2. This includes defining for each phase what will be done. It is divided in 4 phases, each of one subdivided into different steps that will need to be carried out in order to complete the project.

Figure 3.1: Diagram of workflow.

# Project Design

To develop this project, a collection of sensors and actuators must be connected to an embedded system. The board will act as the brain, connecting all the sensorics and processing the information in real time.

## Embedded platform

A wide variety of options are available to choose from, each with its own unique features, computational power, energy consumption, and connectivity options.

### STM32 F041RE Series

The STM32 F041RE Series is a Cortex-based 32-bit MCU renowned for its versatility in power, GPIO configurations, and reliability in embedded applications. With its wide range of models offering varying capabilities, STM32 provides a flexible solution for integrating sensors and actuators into the system. STM32 Cortex-M series cores provide a wide variety of processing power with balanced energy consumption and ease of use, ranging from Cortex-M0 to Cortex-M7.

The STM32 family ranges from entry-level microcontrollers to high-performance devices with advanced features, enabling users to choose from various configurations depending on the project's needs and the user's capabilities. Peripherals included in any of the microcontrollers encompass GPIOs, timers, communication interfaces such as SPI, I2C, or UART, digital-to-analog (DAC) or analog-to-digital (ADC) converters.

Some of the peripherals included in any of the microcontrollers include GPIOs, timers, communication interfaces such as SPI, I2C, or UART, digital-to-analog (DAC) or analog-to-digital (ADC) converters.

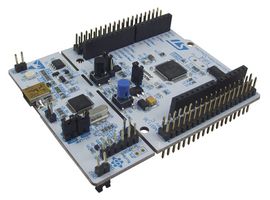
Overall, a STM32 F041RE Series would cover the communications and peripherals of this project, but the lack of computing power or specific technologies for artificial vision renders this alternative not suitable.

Figure 4.1. STM32 F041RE development board.

### Raspberry Pi 4

Raspberry Pi is a highly versatile single-board computer renowned for its widespread adoption, robust community support, and broad range of functionalities. It serves as an excellent platform for a variety of embedded applications, ranging from hobbyist projects to industrial solutions.

Raspberry Pi features a Broadcom system-on-chip (SoC) with ARM-based architecture, offering ample computational power and memory resources to handle diverse tasks. The latest models, such as Raspberry Pi 4, boast quad-core processors with enhanced performance and efficiency compared to earlier versions.

One of the key advantages of Raspberry Pi is its extensive connectivity options, including HDMI, USB, Ethernet, Wi-Fi, and Bluetooth, enabling an easy integration with peripherals and external devices. Additionally, it comes with a 40 pin GPIO header, with communication protocols such as SPI, I2C and UART.

The Raspberry Pi 4 uses a Debian based OS (Operating System), called Raspbian but also supports other Linux based distributions as Ubuntu. This flexibility allows users to choose the most suitable environment for their projects and leverage a vast ecosystem of software libraries and tools.

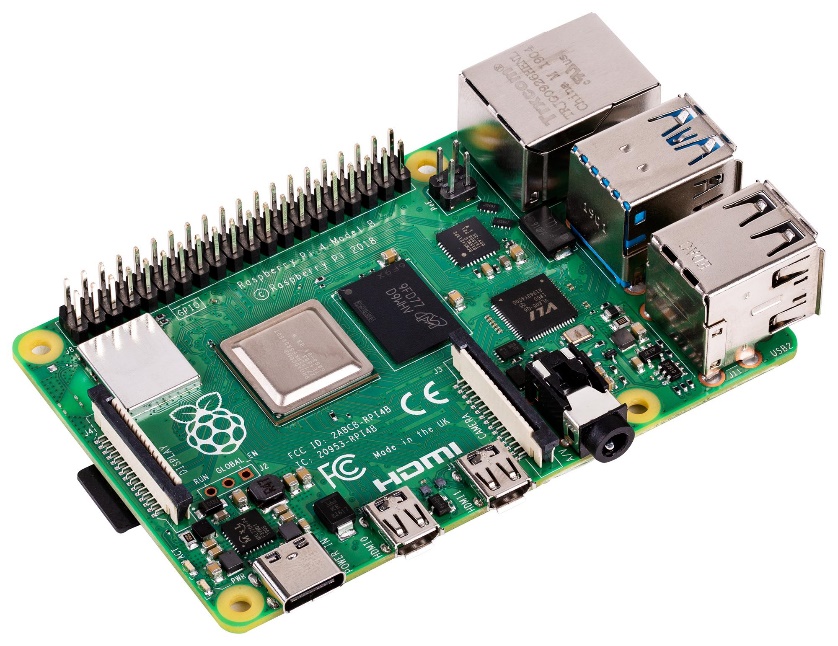
While Raspberry Pi 4 excels in many aspects, it may not be the ideal choice for applications requiring real-time processing or specialized hardware acceleration, such as artificial vision or AI tasks. This makes the Raspberry the perfect choice for network and application management.

Figure 4.2. Raspberry 4 Model B.

### NVIDIA Jetson Nano 2GB Development Kit

Nvidia Jetson Nano stands as a powerful embedded computing platform renowned for its specialized capabilities in AI and machine learning applications. Developed by Nvidia, Jetson Nano offers exceptional performance and efficiency specifically made for edge computing tasks.

This SBC is equipped with a quad-core ARM Cortex-A57 CPU (Central Processing Unit), providing a balance of general-purpose processing power and energy efficiency. Additionally, a NVIDIA Maxwell architecture with 128 NVIDIA CUDA cores GPU (Graphics Processing Unit) delivers the necessary power for edge computing tasks without the need of constant cloud connectivity.

This combination of CPU and GPU resources enables seamless integration of sensor data processing, AI algorithms, and control logic in real time, making it an ideal choice for applications requiring high-performance computing at the edge. Additionally, it comes with 2GB LPDDR4 memory, ensuring fast access to its internal memory for efficient data handling.

Jetson Nano offers a wide variety of connectivity options, including HDMI, 4 x USB 3.0, Ethernet, and GPIO pins, with SPI, I2C, I2S and UART communication, facilitating integration with a wide range of peripherals, displays, and sensors. This enables an easy integration of sensors and actuators with a fast processing with AI capabilities.

Additionally, Nvidia provides software support for Jetson Nano, including the JetPack SDK (Software Development Kit) and compatibility with popular AI frameworks such as TensorFlow, PyTorch or OpenCV. This integration allows the user to utilize Jetson Nanos capabilities right from the start.

With its impressive computational power, high connectivity options, and ready-to-use software framework, Jetson Nano will be selected as the board used to carry out the project.

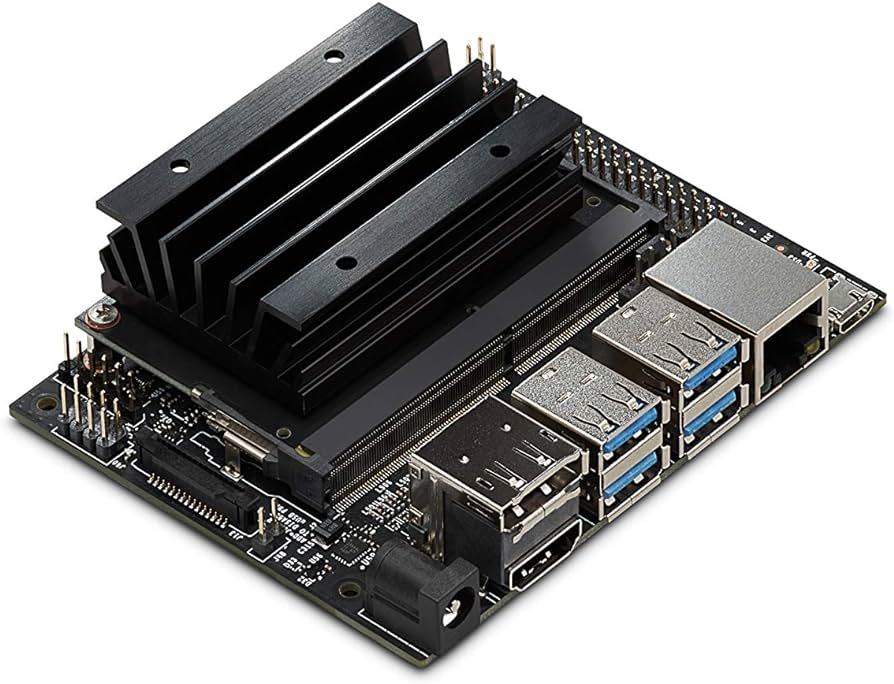


Figure 4.3. Jetson Nano 2GB development board.

## LiDAR

A Laser Imaging Detection and Ranging (LiDAR) device is capable of determining the distance to an object or surface by employing a laser. This laser emits a pulse of light which is directed towards the target surface. Once it hits the surface, the light is reflected back to the LiDAR sensor. By measuring the time it takes for the emitted light to return to the sensor, the LiDAR calculates the distance to the object or surface [23].

Un dibujo de una persona

Descripción generada automáticamente con confianza bajaIts distance is measured using the next formula:

Figure 4.4: Determination of Distances In LiDAR

It is commonly used in wide area detection and mapping, often installed on drones or cars for this purpose.

### Cygbot CygLIDAR D1

The Cygbot Lidar is a solid-state LIDAR, which uses optical and solid-state electrical components instead of mechanical components. Instead of generating the laser with mechanical components, SS (Solid State) lidars make use of laser diodes or optic fibre lasers.

The lack of mechanical components that can break or wear off usually make them a more precise, higher resolution and overall, a much more reliable choice than a regular LiDAR system.  
However, as for this application it is only needed for measuring distances and the complexity of integration of this device is much higher than a regular one, it deems unsuitable.

Figure 4.5: CygLiDAR D1.

### Lightware SF45/B

LiDAR SF45/B Scanning MicroLiDAR Sensor is a small and lightweight scanning sensor, tailored for small autonomous vehicles and drones. With its 60 grams weight and 50 m laser fire range, is controlled by an horizontal servomotor, making him obtain a precise FoV (Field of View) ranging from few degrees to 320 degrees to let the user adapt it to its preferences.

Figure 4.6: Lightware SF45/B.

Furthermore, the device offers customization options for scan speed and measurement rate, as well as an internal alarm feature. Its compact size, light weight, and adjustable settings make it highly suitable for this application, offering flexibility and reliability for integration into various autonomous systems.

## Camera

The integration of cameras in autonomous navigation systems has changed the way they perceive and react to their surroundings. Cameras provide detailed information of the objects surrounding the system, detect pedestrians and vehicles or even traffic signs, as well as calculate measurements and distances.

When combined with the use of computer vision algorithms and deep learning techniques or pre-trained models, can become a valuable tool for detecting real-time problems or potentially risky scenarios for the user.

### Arducam Synchronized Dual Camera

The Arducam 12MP Synchronized Stereo Camera Kit is specifically designed to run two 12MP cameras using a single MIPI CSI-2 camera slot connection from the Jetson Nano, instead of the 2 available ones. It makes use of a camera HAT to disguise the dual camera as a single camera setup, and still be able to rely on the system for depth-related vision systems.

However, despite the innovative design it presents, this camera kit presents a important drawback in terms of kernel and SDK compatibility. The kernel and exiting SDK versions don’t support the use of this kit, as its drivers haven’t received an update for a long time, leading to unmet requirements and unresolved issues coming from the kernel versions, JetPack versions and driver versions mismatch. Pitifully, this renders the use of this kit unsuitable for the development of this project.



Figure 4.7: Arducam DualCamera Kit.

### ADLINK NEON-202A-JNX Kit

The Neon 202A Camera kit is an integrated AI smart camera system featuring the use of a NVIDIA Jetson Xavier NX, an 8 MP sensor and bundled with vision software suites such as Adlink EVA.

Packaged as an all-in-one kit, the Neon 202A Kit reduces the need for extensive cabling and minimizes its footprint, simplifying installation and deployment. Mainly intended for industrial environments, it comes preinstalled with ready-to-use software and sample code, ensuring its ease of use.

Figure 4.8: ADLINK NEON 202A.

With dimensions of 21.4 x 15.6 x 10.3 cm and a weight of 1.1 kg, requires a sturdy and fixed support due to its considerable weight. While its software configuration and deployment advantages are significant, its high co bulk and weight make it unsuitable for lightweight transportation vehicles as bicycles or scooters.

### Logitech C170

Sometimes, simplicity is key, and the Logitech C270 webcam is the best at it. With its affordable price, USB-A connectivity, and a 1 MP sensor, delivers reliable capture video at 30 fps (frames per second) at 720 p resolution.

The webcam’s compact dimensions, simplicity and straightforward “plug and play”, make it the ideal choice for developing this project. To obtain real time detection, it will be needing video processing software and AI algorithms.

Figure 4.9: Logitech C170 webcam.

## GPS

GPS, or Global Positioning System, is a worldwide satellite navigation system that provides the user information related to positioning, direction and time. This system can be found in almost any electronic system such as cars, planes, smartphones or smartwatches, and became an indispensable tool for navigation.

Diagrama

Descripción generada automáticamenteGPS operates by utilizing receiver devices and internal algorithms to communicate with a network of 24 satellites arranged in six orbital planes, each with four satellites, orbiting at an altitude of approximately 13,000 kilometres and a speed of 14,000 km/h [24].

The GPS system employs a technique known by the “trilateration” name where each satellite is sending a signal of its exact position at any given moment. The device, once received, interprets it and calculates the distance to that satellite. However, with just one satellite it is not possible to calculate the position; A minimum of 3 satellites are needed to determine it. With connections to three satellites, the receiver’s position is determined by the convergence of the 3 radii to each satellite. If a fourth satellite is connected, the altitude can be determined [25].

Figure 4.10: GPS Determination.

### Parallax GPS Receiver Module

The Parallax GPS Receiver Module is a fully integrated and cost-effective unit that comes equipped with an onboard patch antenna. Built around the Polstar PMB-248 chipset, this module offers a complete GPS solution within a remarkably small footprint.

Its compact size, availability and cheapness make this alternative a fit for the project.

Figure 4.11: parallax GPS Receiver Module.

## Bicicleta recargada en la pared Descripción generada automáticamenteLight Mobility Vehicle

As for the Light Mobility Vehicle, and with the objective of creating a first prototype capable of being adjusted to any kind of LMV, a robust and stable vehicle must be selected. For this purpose, the best selection will be a bicycle. Additionally, if we select an e-bike, issues regarding power management could be solved by connecting all the embedded platform to the bike’s battery.

The selected bicycle will be the “Ebike Conor Wrc Shake Fs Deore 11s E7000 29 2022”.

Figure 4.12: Ebike Conor Wrc Shake Fs Deore.



## Software selection

Now, with the objective of implementing all the necessary functions in the selected hardware, the justification and selection of the software used will be discussed.

This part will determine the software used for virtual machine containerization, AI deployment and communication protocols.

### Kubernetes

Tabla

Descripción generada automáticamenteAs the main page of Kubernetes explains: “Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services” [26]. These containers act in a similar way to Virtual Machines, but they hey have relaxed isolation properties to share the Operating System (OS) among the applications [26].

Figure 4.13: Kubernetes Container Deployment Diagram.

These containers have its own memory, own their share of the hosts CPU and memory. They are great because they provide huge flexibility, fast creation and deployment, and resource isolation.

This alternative suits perfectly the project, addressing the need of deploying different services using containers. However, the use of Kubernetes is not free, as it costs 0.60$ per container deployment on Google Cloud [27] or 12$/month on DigitalOcean [28]. Due to this, the use is considered not suitable for our project, as extra costs will be inquired.

### Docker

Docker is a similar platform to Kubernetes, as “Docker is an open platform for developing, shipping, and running applications” [29]. Both work on a similar way, but the main difference is that, as Josh Campbell explains, Docker is a containerization platform and container runtime, while Kubernetes is a platform for running and managing containers from numerous container runtimes [30].

As Docker’s main page states: “Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers” [29].

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente con confianza mediaThe efficiency, extensive imagery available and cost-free use of Docker makes it a suitable software solution for the project.

### Node-Red

Figure 4.14: Docker Container Deployment Diagram.

Diagrama

Descripción generada automáticamenteNode-Red is a programming tool used for the connection of hardware devices, APIs and online services. It comes as a browser-based flow editor where flows can be connected using the numerous nodes from the palette [31].

Figure 4.15: Node-Red Example Illustration.

The low level of programming required, and the quickness of deployment make this tool suitable for the use in the project. It will be deployed inside a Docker container.

### InfluxDB

InfluxDB stands out as a time-series database specifically created for the storage of sensor data in IoT applications. Its open-source nature and specialized design make it an optimal choice for projects alike this one, where the need for robust data storage and retrieval capabilities are sought.

The unique characteristics of InfluxDB, such as its ability to efficiently manage time-stamped data, ensure that our project can seamlessly integrate sensor data streams and derive valuable insights from them. Additionally, its scalability and performance make it well-suited for handling the large volumes of data generated by IoT devices in real-time [32].

By the use of InfluxDB, we can not only store sensor data effectively but also enhance the data analysis processes, enhancing the overall functionality and performance of our project. It will also be deployed inside a docker container.

Figure 4.16: InfluxDB Logo.

### Mosquitto

Eclipse Mosquitto is an open-source message broker that implements different MQTT versions. MQTT protocol is a lightweight way of messaging using the publisher/subscriber model [33]. Thanks to the community, it counts with supporting libraries for most programming languages.

The wide support it offers, ease of use and open-source make it a good fit for the project. It will be included inside a docker container and used to communicate the docker container with our host machine.

Figure 4.17: MQTT Working Diagram.

### Kafka

Apache Kafka is an open-source distributed event streaming platform [34]. Its high availability, high throughput, scalability, and permanent storage make Kafka a widely used resource.

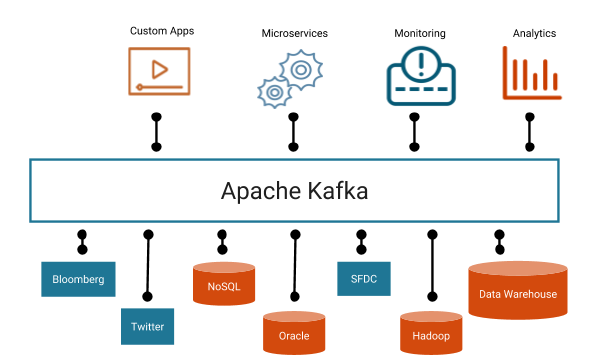
It is mainly used for real time data streaming, and it fits for the project perfectly. Additionally, as Deepstream can’t handle Mosquitto at the used version, Kafka will be selected to carry out the data streaming tasks. As the Deepstream application will be installed on the host machine, so will need to be the Kafka services.

Figure 4.18: Kafka Working Diagram.

### Grafana

Interfaz de usuario gráfica

Descripción generada automáticamenteGrafana is an open-source software that allows data visualization along with the creation of custom control panels. It is easily accessible from services like Doker, Kubernetes or via traditional Databases like MongoDB or InfluxDB.

Figure 4.19: Grafana Example Dashboard.

This resource will allow the creation of custom data graphics for the visualization of tendencies. The connection with InfluxDB is recommended, as the time-series based database enhances the software capabilities.

### NVIDIA DeepStream SDK

NVIDIA’s DeepStream software development kit (SDK) is an AI-powered software for video analytics and edge IoT applications. It provides a flexible framework for real-time video process and analysis, which enables the developers to create intelligent systems that can detect and respond to events in different environments [35].

Some of the key benefits of the development kit include:

* Deep Learning inference:

The SDK includes pre-trained models already optimized for NVIDIA’s GPUs, allowing an easy and fast deployment without the needs of training and extensive knowledge in machine learning.

* High Performance and efficiency:

Deepstream uses the power of the GPU to enable real-time processing of multiple video streams, along with object detection, classification, and tracking. Deepstream is also designed to efficiently utilize GPU resources, enabling a low latency with a high throughput.

* Customizable Pipeline:

Developers have the option to create custom pipelines, allowing flexibility and customization. Additionally, it supports integration with different input and output streaming protocols .

#### NVIDIA TAO Toolkit

Diagrama

Descripción generada automáticamente con confianza mediaAs explained on NVIDIA’s official site, “NVIDIA [**TAO Toolkit**](https://developer.nvidia.com/transfer-learning-toolkit) is a low-code AI toolkit built on TensorFlow and PyTorch, which simplifies and accelerates the model training process by abstracting away the complexity of AI models and the deep learning framework” [36]. With this toolkit, users can select from the 100+ pre-trained vision AI models available and customize their own dataset.

Figure 4.20: TAO Overview. Extracted from [36].

Model pruning iss the key differenciator of NVIDIA’s TAO Toolkit. The provided explanation on their official website explains that pruning involves removing from the neural network nodes that contribute less to the overall accuracy of the model, reducing the overall size of the model, significantly reducing the memory footprint, and increasing inference throughput–all factors that are very important for edge deployment [36].

Gráfico, Gráfico de barras

Descripción generada automáticamenteFor example, the Frames per Second (FPS) obtained using the TrafficCamNet, DashCamNet, and PeopleNet pruned models are almost the double of the obtained in the unpruned models [36].

Figure 4.21: Pruning Efficiency. Extracted from [36].

#### DashCamNet

The DashCamNet model is included in the TAO Toolkit, with the intention of simulating the effects of a car Dashcam. This can be used for the implementation in the project due to the similarities between a Dashcam and the necessities of the project.

The models are based on NVIDIA DetectNet\_v2 [[1]](#footnote-1) detector with ResNet 18[[2]](#footnote-2) as a feature extractor. This model detects objects from the next categories and returns a labelled box around the object.

* car
* persons
* road signs
* bicycles

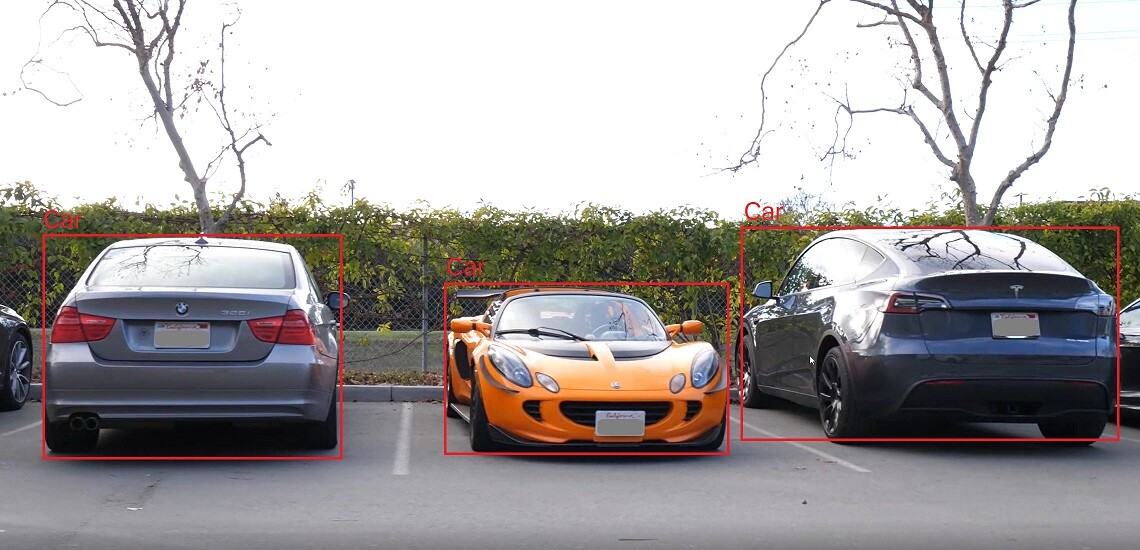


Figure 4.22: DashCamNet Pruned Model Example of Use. Extracted from [36] [43].

# Project Development

Once all the components and tools are selected, in this chapter, the implementation of the different hardware components along with the software tools will be described.

## Software Development

To start, the set-up and programming of the Jetson Nano and Raspberry Pi 4 will be done.

Then, the programming of the different modules will be done. For this, a set of the previously mentioned tools will be used, in order to create connections, send and receive data. For this, the ‘main.py’ archive will be created, including all the code to control the peripherals.

Additionally, the programming of the Raspberry Pi Pico and its peripherals will be done.

Lastly, the set-up of the Docker and programming of it’s containers with all the needed applications will be done.

All the code will be available in a public git repository, and fragments will be used to demonstrate the points made. [37]

### GPS Programming

The previously selected GPS module is the Parallax GPS receiver module.

The GPS module comes with 4 accessible pins. GND (Ground), VCC (Voltage at Continuous Current), SIO (Serial Input Output) and RAW (Raw data). When the RAW pin is left unconnected (internally pulled high), the smart mode is activated. This mode allows the user, via SIO pins, to request data using commands and receiving the selected data.

However, the smart mode does not properly work on the GPS module, so the only option is to receive the data in NMEA0183 (National Marine Electronics Association) format. For this, the RAW pin will need to be connected to the GND pin.

The structure of the received data in this format is the next one:

Escala de tiempo

Descripción generada automáticamente

Figure 5.1: NMEA0183 Format Structure.

In order to correctly receive all the sentences, each individual byte will need to be processed. With this purpose, the following function was created, along with the UART connection:

List 5.1: GPS Message Reading Function

ser = serial.Serial("/dev/ttyTHS1", baudrate=4800, bytesize=serial.EIGHTBITS, parity=serial.PARITY\_NONE, stopbits=serial.STOPBITS\_ONE, timeout=0.5)

def recibir\_info():

    """

    Reads NMEA sentences from the GPS module and processes them.

    """

    sentencias\_nmea = []  # List for storing NMEA sentences

    while True:

        mensaje\_bytes = ser.read()  # Read a byte

        if mensaje\_bytes == b'$':  # If start byte

            sentencia = b'$'

            while True:

                byte = ser.read()  # Read next byte

                if byte == b'\r':  # If end byte

                    sentencia += byte

                    byte = ser.read()  # Read next byte

                    if byte == b'\n':  # If skip byte

                        sentencia += byte

                        sentencias\_nmea.append(sentencia)  # Add to list

                        if sentencia.startswith(b'$GPRMC'):  # If control sentence

                            for s in sentencias\_nmea:

                                datos = decode\_nmea0183(s.decode('ascii').strip())  # Decode all sentences

                                if datos:

                                    imprimir\_datos(datos)

                            sentencias\_nmea = []  # Clear list

                        break

                else:

                    sentencia += byte  # Add byte to sentence

Once the GPS is connected and the received data is interpreted, the only thing left is to communicate. The data will be sent using Mosquitto to the InfluxDB database, where it will be visualised.

Following, the GPS code for obtaining and communicating the required parameters, decodifying NMEA sentences:

List 5.2: NMEA Message DEcoding Function

def decode\_nmea0183(message):

    parts = message.split(',')  # Split message into parts

    if len(parts) >= 6 and parts[0][0] == '$':  # Verify message

        message\_type = parts[0][1:]

        if message\_type == 'GPGGA':

            time\_utc = parts[1]  # Time in UTC format

            latitude = convertir\_coordenadas\_grados(parts[2], parts[3])  # Latitude in degrees

            longitude = convertir\_coordenadas\_grados(parts[4], parts[5])  # Longitude in degrees

            altitude = parts[9]  # Altitude in meters

            return {

                'message\_type': message\_type,

                'time\_utc': time\_utc,

                'latitude': latitude,

                'longitude': longitude,

                'altitude': altitude

            }

        elif message\_type == 'GPRMC':

            # Example: $GPRMC,220516,A,5133.82,N,00042.24,W,173.8,231.8,130694,004.2,W\*70

            time\_utc = parts[1]  # Time in UTC format

            latitude = convertir\_coordenadas\_grados(parts[3], parts[4])  # Latitude in degrees

            longitude = convertir\_coordenadas\_grados(parts[5], parts[6])  # Longitude in degrees

            speed = parts[7]  # Speed in knots

            track\_angle = parts[8]  # Track angle

            return {

                'message\_type': message\_type,

                'time\_utc': time\_utc,

                'latitude': latitude,

                'longitude': longitude,

                'speed': speed,

                'track\_angle': track\_angle

            }

    return None

### AI Vision Programming

As previously explained, to set up the Artificial Intelligence vision, a webcam and a NVIDIA software will be used.

**Webcam Setup**

The main advantage of using a USB Webcam comes from its connectivity and straight plug-and-play. Once connected the USB on one of the Jetson Nano’s USB ports, we can run the following command to verify that the camera is working.

**Deepstream setup**

Now that the webcam is correctly stablished, it is needed to install and update all the Deepstream related software. It can be downloaded from NVIDIA’s webpage [35] or pulled from the git page. TensorRT, CUDA, Gstreamer and some additional software are required, too.

Once Deepstream is correctly set up on the host machine, we can access the downloaded files on:

Additionally, sample apps and sample source code can be obtained on :

On the project design it was decided that the pre-trained and pruned model DashCamNet was going to be used for this artificial vision task. This model included the detection of cars, persons and bicycles

Before launching the application, some parameters must be set. This include the source of the video, the model selection and output adjustments. The ‘source’ and ‘streammux’ options will be related to the source, the Logitech webcam, and the video input and output.  
The sink1 will be handling all the Kafka communications, and the ‘primary-gie’ specifies the motors for the DashCamNet pruned model, along with other different options.  
  
The following code specifies all the settings:

[source0]

enable=1

type=1

camera-width=640

camera-height=480

camera-v4l2-dev-node=0

camera-fps-n=25

camera-fps-d=1

gpu-id=0

cudadec-memtype=0

[streammux]

gpu-id=0

batch-size=1

batched-push-timeout=4000

width=1920

height=1080

live-source=1

[sink1]

enable=1

#Type - 6=MsgConvBroker

type=6

msg-conv-config=dstest5\_msgconv\_sample\_config.txt

msg-conv-payload-type=0

msg-broker-proto-lib=/opt/nvidia/deepstream/deepstream-6.0/lib/libnvds\_kafka\_proto.so

msg-broker-conn-str=192.168.0.23;9092;my-topic #Raspberry IP

topic=my-topic

[primary-gie]

enable=1

gpu-id=0

plugin-type=0

batch-size=1

gie-unique-id=1

config-file=/opt/nvidia/deepstream/deepstream/samples/configs/tao\_pretrained\_models/nvinfer/config\_infer\_primary\_dashcamnet.txt

List 5.3: Deepstream Configuration

**Kafka and Zookeeper**

As an output, Deepstream will send a JSON string using the configured method. For the Deepstream version used, 5.02, the best option is to handle the messages using Kafka.   
As previously mentioned, the Jetson Nano isn’t able to handle all functions, and the Kafka and Zookeeper servers will be hosted on the Raspberry Pi 4.

Zookeeper is a tool needed for the handling of the Kafka partitions, and both will be needed to be set up at the same time. After installing all the necessary software, the commands for setting up both servers on the Raspberry will be the next ones:

*/home/smart-mobility/Descargas/kafka\_2.13-3.7.0/bin/kafka-console-producer.sh --bootstrap-server localhost:9092 --topic my-topic*

*/home/smart-mobility/Descargas/kafka\_2.13-3.7.0/bin/kafka-console-consumer.sh --bootstrap-server 0.0.0.0:9092 --topic my-topic*

This way, the JSON string messages can be received both on the docker and on the python script.

### LiDAR Programming

Making use of the LightWare SF45/B and the Jetson Nano, the first objective was to connect and receive data from the device, so that later can be processed at the edge and sent to the IoT.

The SF45/B comes with 3 connection methods: via Header Pins UART, Header Pins I2C and USB UART. LightWare provides some example code for different languages like C++ or Python, which will be he ultimately selected programming language.

After many tests, it was determined that only the connection via USB UART was possible, as the device couldn’t stream data using the other two methods.

The LiDAR Programming will be done using the LWNX library functions, provided by the manufacturer. They include functions to create the CRC, build the packet, parse it and most importantly, sending it and waiting for the response. This way, it is only needed to send specific commands with specific values (Available at the manual or manufacturer webpage) [38] to configure the LiDAR to the specifications of the project.

The settings used for the UART communication, along with the sent commands where the following:

As for the data, the LiDAR is constantly streaming data including distance and the angle it was taken at. This is done in an synchronous way, so that the other processes in the program don’t delay the data collection. It will also be sent via MQTT.

List 5.4: LiDAR Configuration.

def executeCommand(port, command, write, data=[], timeout=1):

    packet = buildPacket(command, write, data)

    retries = 4

    while retries > 0:

        retries -= 1

        port.write(packet)

        response = waitForPacket(port, command, timeout)

        if response is not None:

            return response

    raise Exception('LWNX command failed to receive a response.')

serialPortName = '/dev/ttyACM0'

serialPortBaudRate = 921600

port = serial.Serial(serialPortName, serialPortBaudRate, timeout=0.1)

print\_product\_information(port)

executeCommand(port, 66, 1, [4]) #Sets update rate to 400 samples/s

executeCommand(port, 85, 1, [5, 0, 0, 0]) #Sets scan speed to 200

executeCommand(port, 98, 1, [12, 0, 0, 0])#Sets scan-low angle to -10,5

executeCommand(port, 99, 1, [12, 0, 0, 0]) #Sets scan-high angle to 10,5

set\_default\_distance\_output(port)

executeCommand(port, 30, 1, [5, 0, 0, 0]) #Enables data streaming

The MQTT configuration will be done using python libraries ‘paho.mqtt.client’ , and will allow to send data from the LiDAR to the docker for further analysis. It will be explained with more detail on the following pages.

The next list details the python code from the repository [37] with the functions for data streaming treatment.

List 5.5: LiDAR Data Reception

def wait\_for\_reading(port, timeout=1):

    response = waitForPacket(port, 44, timeout)

    if response is None:

        return -1, 0

    distance = (response[4] << 0 | response[5] << 8) / 100.0

    yaw\_angle = response[6] << 0 | response[7] << 8

    if yaw\_angle > 32000:

        yaw\_angle = yaw\_angle - 65535

    yaw\_angle /= 100.0

    return distance, yaw\_angle

async def procesar\_datos\_lidar():

    while True:

        distance, yaw\_angle = wait\_for\_reading(port)

        if distance != -1 and distance <= 30:

            client.publish("lidar/output/distancia", distance, 0)

            client.publish("lidar/output/angulo", yaw\_angle, 0)

            with objetos\_lock:

                for obj\_id, obj\_data in objetos\_detectados.items():

                    if abs(obj\_data['angulo\_x'] - yaw\_angle) <= 0.5:

                        obj\_data['distancia'] = distance

            distancia\_media = calcular\_distancia\_media()

            if distancia\_media and abs(anguloObj - yaw\_angle) <= 0.5 and cola >= 1:

                if distancia\_media <= 20:

                    client.publish("lidar/output/distanciaMedia", distancia\_media, 0)

                    cola -= 1

            print(f"Distancia: {distance}, Ángulo: {yaw\_angle}, Distancia media: {distancia\_media}, Cola: {cola}")

        await asyncio.sleep(0.1)

### Docker

As previously explained, Docker will handle all the IoT connections using different tools. The Due to the Jetson Nano’s capabilities being limited, Docker will be installed in the Raspberry Pi 4, with the purpose of offloading some of the workload for the Jetson Nano.

The tools being used fulfil different functionalities and complement each other’s strength and weaknesses. The deployment will be done using docker-compose, and the archive for the volume containing the settings will be upload in the GitHub repository [37]. These tools are often used together, and as previously explained the selected tools are:

* Portainer
* Mosquitto
* InfluxDB
* Grafana
* Node-Red

#### Portainer

Portainer is your container management software to deploy, troubleshoot, and secure applications across cloud, datacenter, and Industrial IoT use cases [39]. This tool will enable the visualization of all the containers and its statuses, logs and connections, among other configuration utilities for containers, networks and volumes, using a user-friendly web interface.

In the context of this project, Portainer is not strictly essential but highly beneficial. It will be implemented to streamline the management of Docker containers running on the Raspberry Pi 4. By providing a clear overview and easy access to configuration settings and logs, Portainer will significantly reduce the time required for troubleshooting and maintenance tasks. This efficiency is particularly valuable in a development environment, where quick iteration and problem resolution are critical.

#### Mosquitto

As previously explained, Mosquitto is a message broker that allows the different data collected from the devices on the scripts to be sent to the IoT. For its deployment, it is necessary that its connection is stablished both on the script and in the reception node, in the Node-Red container.

The connection on the script was done using “paho-mqtt” libraries on python, as it was previously explained. The set-up code used is the following one:

The Node-Red connection node will need to include the same parameters of Host Ip, port and topic for the connection to be stablished. This way, whenever the Jetson Nano and the Raspberry Pi are connected to the internet, the data will be sent using the broker.

List 5.6: Paho-Mqtt setup code.

mqtt\_broker = "192.168.0.23"  # IP of the Host

mqtt\_port = 1883  # Port for MQTT connection

topic\_input = "lidar/input"  # Topic to subscribe to

# Create an MQTT client instance

client = mqtt.Client()

# Connect to the MQTT broker

client.connect(mqtt\_broker, mqtt\_port, 60)

client.subscribe(topic\_input)  # Subscribe to the specified topic

client.loop\_start()  # Start the MQTT client loop to process network traffic and dispatch callbacks

#### InfluxDB

InfluxDB is a time-series based database that is a perfect fit for the context of this project due to the sensors constantly sending information between each other. InfluxDB excels at handling high-write and query loads, making it ideal for IoT applications where data is generated continuously.

InfluxDB will be used to store data collected by the devices. This data includes measurements from the LiDAR like distance and angle, position and speed from the GPS, objects detected from the Deepstream algorithm and acceleration related information form the accelerometer.

In order to connect the information in Node-Red, the host’s IP and port will be needed to be set up. The port used is the 8086.

The database's high efficiency in managing time-stamped data ensures that the system can handle the large volumes of data generated without performance degradation. Additionally, the efficient storage and availability InfluxDB offers is greatly complemented by Grafana, covered in the next section.

#### Grafana

Grafana is an open-source platform for monitoring and observability, which allows you to visualize, analyze, and understand your data through customizable dashboards. It is highly compatible with InfluxDB, making it an ideal choice for this project to visualize the data collected from various sensors.

Grafana will be used to create real-time dashboards that display data. This visualization helps in monitoring the system’s performance and detecting any anomalies promptly. To setup the data transfer between InfluxDB and Grafana, the configuration needed will be:

* Type: InfluxDB
* URL: http://influxdb:8086
* Database: smart
* User/ Password: smart/ smart
* Port: 3000

Once the connection is stablished, the panel must be created. The data received on the database will automatically be transferred to Grafana, and the configurated dashboard will display according to its configuration. It looks like the next image:

#### Node-Red

Node-Red is a flow-based development tool for visual programming, designed to integrate hardware devices, APIs, and online services. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette, connecting devices and services for the Internet of Things.

Node-Red will be used in this project to manage the data flow between the various sensors and devices, handle IoT protocols, and integrate with other software tools like InfluxDB and Grafana. It acts as the central hub for data processing and communication, ensuring that data is routed correctly and efficiently between components. The port set for this configuration is the port 3000.

The settings used for the node connections have already been explained, and the node configuration is visualized in the next image:

As a brief descrition, the MQTT nodes will receive data based on its topic, and send it directly to the InfluxDB database. The Kafka nodes will act similarly, however, it is mandatory to extract the information first, as the payload is in JSON format.

### I2C Communication Protocol

Inter-Integrated Circuit, or I2C, has become a popular communication protocol for connecting multiple devices. Combining the simplicity of UART's asynchronous serial communication and SPI's high-speed synchronous data transfer, I2C allows users to control multiple slave devices with a single master, though configurations with multiple masters or multiple slaves can also be implemented.

**Overview of I2C Protocol**

I2C uses two wires to transmit data between devices:

SDA (Serial Data Line): Carries the data.

SCL (Serial Clock Line): Carries the clock signal.

This two-wire setup reduces the number of connections required. The communication makes use of an addressing system, where a direction of 7 or 10 bits is sent to the devices.

**Data Transmission**

Data is transmitted bit by bit along the SDA wire, with the SCL clock synchronizing the sampling of the bits. The transmission is always controlled by the master device (Blue) and the slave device will always be listening and acknowledging (Orange). The structure followed in I2C is the following:

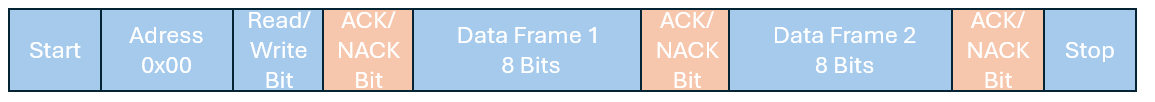
The start is marked by pulling the SDA line from high to low while the SCL line remains high. Then, the address frame is sent, along with the read or write bit. Once the slave has received and compared their address, they will respond with a one bit pull down on the SDAA line to acknowledge its existence and will keep acknowledging with the reception of each data frame until the stop bit. The master ends the transmission by releasing the SCL line to high before releasing the SDA line to high.

Figure 5.2: I2C Communication Protocol.

An additional bit is sent to indicate whether the master intends to read from (1) or write to (0) the slave.

To implement different slaves to the jetson nano, their directions must be stablished. Using the following command, the addresses can be determined:

This command scans the Nano’s I2C 1 bus in its entirety, and returns the following addresses for the devices:

* OLED ssd1306 : 0x3c
* Raspberry Pi Pico : 0x28
* Accelerometer : 0x

Escala de tiempo

Descripción generada automáticamenteFor both the accelerometer and the OLED display custom made libraries exist, with the intention of making easier for the user to control the devices. However, for the RPI Pico, a custom message must be created, with the intention of controlling the LED Matrix and the RGB LED.

Figure 5.3: i2c Custom Communication.

Escala de tiempo

Descripción generada automáticamenteThe message implemented will consist of one address byte, 4 Led matrix control bytes, and 3 RGB LED control bytes. To control the matrix, 25 individual LEDs need to be managed, which will be stored in the 4 bytes along with 7 ceros arranged in a way where the first LED is on the MSB, but after the unneeded ceros. For example, to turn all the LEDs on the matrix on, along with the three RGB pins:

Figure 5.4: Custom Communication Example.

As previously mentioned, this will only work for the Raspberry Pi Pico, as the other devices use specific python libraries for the communication.

#### I2C Master and Slave Connection

To send the data from the master to the slave functions in the main.py will be used [37]. These functions will receive a distance and an angle, and will make the necessary calculations for displaying the position in the cone shaped LED matrix. Then, the functions will structure

The Raspberry Pi Pico programming was done using the Arduino IDE because of the simplicity of connection it offers. It was decided to program the microcontroller using c language based on the impossibility of programming the controller as a slave in upython. The following code included in the setup() allows to do such action [37]:

    // Configure internall pull-up resistances

    pinMode(SDA\_PIN, INPUT\_PULLUP);

    pinMode(SCL\_PIN, INPUT\_PULLUP);

    // Configure pins in wire.h library

    Wire.setSDA(SDA\_PIN);

    Wire.setSCL(SCL\_PIN);

    // Start i2c communication

    Wire.begin(I2C\_SLAVE\_ADDRESS);

    Wire.onReceive(receiveEvent);

    Wire.onRequest(requestEvent);

List 5.7: I2C Communication Settings.

The code below is extracted from the main.c of the Pico in the GitHub repository [37]. The function reads the 8 received bytes, and if the address coincides the data is stored for updating the LED matrix and RGB LED.

List 5.8: I2C Data Reception and Processing.

// This function is called when the master sends data to the slave.

void receiveEvent(int howMany) {

    Serial.print("Number of bytes received: ");

    Serial.println(howMany);

    digitalWrite(GP25\_PIN, HIGH);  // Turn on the GP25 pin

    if (howMany == 8 && switchState == 0b0000) {  // Verify that 8 bytes are received and that it's in default mode

        matrizData = Wire.read() | (Wire.read() << 8) | (Wire.read() << 16) | (Wire.read() << 24);  // Read the 4 matrix data bytes

        redValue = Wire.read();  // Read the red color value

        greenValue = Wire.read();  // Read the green color value

        blueValue = Wire.read();  // Read the blue color value

        buzzer\_on();  // Turn on the buzzer

        delay(100);   // Keep the buzzer on for 100 ms

        buzzer\_off(); // Turn off the buzzer

        actualizar\_matriz\_desde\_datos();  // Update the LED matrix

        actualizar\_led\_rgb();  // Immediately update the RGB LED values

        debug\_print\_data();  // Print the received data for debugging

    } else {

        Serial.print("Unexpected number of bytes received: ");

        Serial.println(howMany);

        while (Wire.available()) {

            Wire.read();  // Discard the bytes if not exactly 8 are received

        }

    delay(10);

    digitalWrite(GP25\_PIN, LOW);  // Turn off the GP25 pin

    }

}

Other relevant function of the code for the correct working is how the led matrix is updated.   
The function, extracted from the source code [37], describes the following:

List 5.9: Matrix Update Function

void actualizar\_matriz\_desde\_datos() {

    apagar\_matriz();

    for (int row = 0; row < 5; row++) {

        digitalWrite(rowPins[row], LOW);

        for (int col = 0; col < 5; col++) {

            int bitIndex = row \* 5 + col;

            int bitValue = (matrizData >> bitIndex) & 1;

            digitalWrite(columnPins[col], bitValue);

        }

        delay(2);

        apagar\_matriz();

}

## Hardware Development

The hardware development includes the design of a Printed Circuit Board (PCB) for the display of information and the design and fabrication of the different components that will be used for mounting all the devices in the bicycle.

### PCB Design

During the following sections, the design of a PCB will be described and explained. The objective is to facilitate the visualization of obstacles to the user, and as a result avoiding possible accidents.

The development of the PCB will start with the creation of a power supply, followed by the implementation of an LED matrix and control board, and concluding with the addition of the remaining components.

The design of the PCB was carried out using KiCad software, as it provides a wide variety of components and footprints.

#### PCB Power Supply

Diagrama, Esquemático

Descripción generada automáticamenteTo power the PCB, a standard USB-C port will be used. This port offers the necessary durability to hold the cable securely and the advantage of quick disconnection options, allowing the device to be quickly installed in other light mobility vehicles.

Figure 5.5: PCB Power Supply Schematic

#### LED Matrix

An LED matrix is an array of LEDs arranged in rows and columns, used to display information visually. Since each LED can be turned on and off individually, by using high-frequency switching in combination with the matrix design, images and text can be created.

This particular matrix is a 5x5 configuration, meaning it includes 5 rows and 5 columns for a total of 25 LEDs. Each row and column will be controlled by the Raspberry Pi Pico. Instead of controlling each LED individually, this setup allows many LEDs to be manipulated simultaneously. This technique, called multiplexing, uses rapid switching so that to the human eye it appears as though every LED is on at the same time.

However, the multiplexing technique means that each pin of the Raspberry Pi Pico must control up to 5 LEDs simultaneously, which can exceed the current capacity that the board can provide. The solution to this problem is to use LED drivers.

##### ULN2003A

Diagrama, Esquemático

Descripción generada automáticamenteThe ULN2003A is a high-voltage, high-current Darlington transistor array. It is commonly used to drive loads such as relays, lamps, LEDs, and stepping motors. Each ULN2003A contains seven open-collector Darlington pairs, each capable of sinking 500 mA and withstanding up to 50 V.

In this project, the ULN2003A will be integrated to handle the current demands of the LED matrix. Each output pin of the Raspberry Pi Pico will be connected to the inputs of the ULN2003A, and the corresponding outputs of the ULN2003A will be connected to the rows and columns of the LED matrix. This setup ensures that the LEDs receive adequate current, maintaining the brightness and reliability of the display.

Figure 5.6: ULN2003 Darlington Transistor Array

##### Resistances

To control the current flowing through both the input and output of the Darlington arrays, some resistances will be used. To calculate the value of each resistance, Ohm’s law will be used. Some additional data is needed, like the output current of the RPI Pico is and output voltaje . Taking this into account, the resistance for the ULN2003 entry is calculated.

The resistance required will be 275 Ohms, but as a resistance with that value does not exist and with the intentions of simplifying the circuit, a resistance with value will be used.

Now, to calculate the resistances between the ULN2003A and the 5 LED column, the following data is required: and . Thanks to this the resistance can be calculated.

Diagrama, Esquemático

Descripción generada automáticamenteThe resistance required will be 24 Ohms, but as a resistance with that value does not exist and with the intentions of simplifying the circuit, a resistance with value will be used. The power dissipation is negligible as almost any resistor will be able to dissipate the small current. Finally, the schematic for the LED matrix will be the following:

Figure 5.7: LED Matrix Schematic

#### Raspberry Pi Pico

The Raspberry Pi Pico is a fast, flexible and cheap microcontroller board, based on the RP2040. This board counts with 2 – 20 pin headers, that allow different connections like GPIO, I2C, SPI or UART. It can be programmed in MicroPython or C/C++, and thanks to this millions of projects explore its versatility.

The RPI Pico will be in charge of receiving data from the Jetson, interpreting it and acting according to the receiving data. The communication will be done using I2C, and the data received will include information about object detection and distances, relevant for the display on the LED matrix, the control of the RGB LED and the buzzer control.

Following, the schematic for the control of the different components using the PCB and the list of components connected on each pin:

Diagrama, Esquemático

Descripción generada automáticamente

Figure 5.8: PCB connection Schematic.

Table 5.1: Raspberry Pi Pico Pinout

|  |  |  |  |
| --- | --- | --- | --- |
| Pin Number | Connection | Pin Number | Connection |
| 1 | ULN2003A\_1\_1 | 40 | - |
| 2 | ULN2003A\_1\_2 | 39 | +5V |
| 3 | - | 38 | GND |
| 4 | ULN2003A\_1\_3 | 37 | - |
| 5 | ULN2003A\_1\_4 | 36 | +3,3V |
| 6 | ULN2003A\_1\_5 | 35 | - |
| 7 | ULN2003A\_2\_1 | 34 | SW1 |
| 8 | - | 33 | - |
| 9 | ULN2003A\_2\_2 | 32 | SW2 |
| 10 | ULN2003A\_2\_3 | 31 | SW3 |
| 11 | - | 30 | - |
| 12 | - | 29 | SW4 |
| 13 | - | 28 | RGB\_1 |
| 14 | ULN2003A\_2\_4 | 27 | RGB\_GND |
| 15 | ULN2003A\_2\_5 | 26 | RGB\_2 |
| 16 | I2C\_0\_SDA | 25 | RGB\_3 |
| 17 | I2C\_0\_SCL | 24 | - |
| 18 | - | 23 | - |
| 19 | - | 22 | Buzzer R |
| 20 | - | 21 | - |

#### RGB LED

An RGB LED can emit a wide range of colours, from primary colours like Red, Green and Blue, and secondary colours like Yellow, Cyan and Magenta to intermediate tones. The use given to the RGB LED in this PCB is as a status light, indicating the functioning of the system according to the available resources.

The following list indicates the code representing the functioning of the system or error:

Table 5.2: RGB LED Colour guide.

|  |  |
| --- | --- |
| Colour | Meaning |
| Green | Correct functioning of the system |
| Red | Error in the i2c communication |
| Yellow | Error in the data receiving |
| White | SW1 active |
| Cyan | SW2active |
| Magenta | SW3 active |
| Blue | SW4 active |

Diagrama

Descripción generada automáticamenteAnd the schematic of the RGB LED:

Figure 5.9: RGB LED Schematic.

#### Buzzer

Whenever on the bike, it is essential to keep the attention on the road. While the addition of an LED Matrix allows visualization, it also diverts the user's view away from the road. With the intention of warning the user without distraction, a buzzer will be implemented into the system. The buzzer will beep according to the distance to the obstacles, with higher or lower frequency. The design also includes a variable resistance of 1k7Ω that regulates the sound intensity, allowing the user to customize it.

The following table shows the frequency according to the distance:

Table 5.3: Buzzer frequencies.

|  |  |
| --- | --- |
| Distance | Frequency |
| Greater than 10 m | None |
| Between 10 and 8 m | None |
| Between 8 and 5 m | 1 Hz |
| Between 5 and 3 m | 5 Hz |
| Between 3 and 1 m | 20 Hz |
| Less than 1 m | Continuous |

Diagrama, Esquemático

Descripción generada automáticamenteThe buzzer is a component which drains more current than the available from one of the Pico’s pins. Because of that, a NPN BJT Transistor will be used to control the flow of current.

Figure 5.10: Buzzer Schematic.

The transistor acts as a switch, allowing a higher current to pass through the buzzer without overloading the microcontroller pin. The specific model used is the 2N2222A from Texas Instruments. The corresponding schematic is shown along:

#### Switch

With the flexibility that the Raspberry Pi Pico offers, it was decided to install a switch with the intention of expanding the boards functionalities. The switch consists on 2 rows of header pins, and the switching will be made with the addition of jumper cables.

Following, the table of each switch with its functionality and its schematic:

Table 5.4: Switch Functionality Table.

|  |  |
| --- | --- |
| Switch | Functionality |
| SW1 |  |
| SW2 |  |
| SW3 |  |
| SW4 |  |

Diagrama, Esquemático

Descripción generada automáticamente

Figure 5.11: Switch Schematic

#### I2C Connections

As previously mentioned, the communication between devices will be made using I2C buses. This allows the user to transmit data at high speed, targeting it to the desired device.

To make the communication work, pull-up resistances must be set so that the high level is ensured whenever the bus is not connected to any device. Although the Pico, OLED and accelerometer include internal pull-up resistances, it was decided to include resistances in case they fail. The typical value is of 4,7k Ω.

Diagrama, Esquemático

Descripción generada automáticamenteThe schematic:

Figure 5.12 I2C PCB Connection Schematic.

### 3D Pieces Design

The complex shape of the bicycle’s frame hinders the installation of all the components. Two main assemblies will be designed, consisting of different pieces. One will be located in the handlebar of the bicycle, with the components needed for the visualization of objects and distances. The other assembly will be located in the front part of the frame, above the front tire. The parts that compose each assembly will be detailed in the next table, and each individual part and the assemblies can be found on the GitHub repository [37].

All the pieces were created for this project using SolidWorks.

Table 5.5:Assembly Components

|  |  |
| --- | --- |
| Handlebar Assembly | Frame Assembly |
| Clamp | Right Adapter |
| Box for the PCB | Left Adapter |
| Adapter | Box for the Components |
| Methacrylate Screen | 2x Methacrylate Panels |

#### Handlebar Assembly

The handlebar assembly will be made with the objective of encapsulating the custom-made PCB. The design considerations taken are the bike’s geometry and the PCB’s geometry.

Imagen que contiene Dibujo de ingeniería

Descripción generada automáticamenteAll the used screws will be M3 and individually threaded with a specialised tool. The order followed for the creation of the assembly started by creating the top part based on the PCB, following by the creation of the adapter and the clamp tailored to the bike’s specifications.

Figure 5.13: Design of the PCB Box.

Dibujo de una persona

Descripción generada automáticamente con confianza mediaThe box’s design encapsulates the PCB leaving room for the LED matrix visualization, along with the RGB LED and Buzzer. The OLED display has 4 holes for installing directly on the box.

Una caricatura de una persona

Descripción generada automáticamente con confianza bajaThe adapter was paired with the dimensions of the bike’s handle and the designed clamp’s specifications. The M3 holes are specifically placed for ensuring the union between the different parts of the assembly.

Figure 5.14: Adapter design.

Figure 5.15: Design of the Clamp.

The final assembly of the handlebar components will include all the components previously mentioned, with the final result being the following:

#### Imagen que contiene lego Descripción generada automáticamenteFrame Assembly

Figure 5.16: Handlebar Assembly

The Connor E-Bike’s frame presents a difficult geometry, as there is no space where a box could be hung. For this purpose, two almost symmetrical pieces were created, wrapping the frame front and generating an anchor point for the box to be hung.

Both parts present 4 M3 holes for the coupling between both wraps, and 2 holes each for the wrapping with the box’s adapter.

Icono

Descripción generada automáticamenteImagen que contiene Icono

Descripción generada automáticamente

Figure 5.17: Left Frame Wrap Design.

The right frame presents the same structure except for a few channels introduced for the insertion of bike’s cables for the brakes and transmission.

Figure 5.18: Right Frame Wrap Design.

Imagen que contiene cesto, tabla

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente con confianza media

The box will be created for holding all the necessary electronic components for the provisioning of intelligent systems. This includes the Jetson Nano, the Raspberry Pi 4, the LiDAR, the GPS and the webcam. They will need to be hold securely, using M3 screws and making the necessary changes in the prototype.

Imagen que contiene caja

Descripción generada automáticamenteImagen que contiene caja, computadora

Descripción generada automáticamentehe assembly will include two methacrylate panels, hold securely using screws on each corner of the box.

Figure 5.19: Design of the Box for the provisioning of the Intelligent Systems.

Imagen que contiene refrigerador, pantalla, parado, computadora

Descripción generada automáticamenteThe final assembly will look similar to the next image:

Figure 5.20: Assembly of the Box for the provisioning of intelligent systems.

### Prototype Fabrication

All the components were fabricated using 3D printing technology. This technology allows fast, easy and cheap prototyping for experimental projects, with low production times and a high product quality.

The 3D printing was done using the Ultimaker 2+ and the Prusa i3 MK3/MK3s. The settings used on the project are detailed on the next table:

Table 5.6: 3D Printing Parameters.

|  |  |  |
| --- | --- | --- |
|  | Ultimaker 2+ | Prusa MK3 |
| Layer Height | 0.32 | 0.2 |
| Walls/ Wall Count | 0.8 mm/ 3 | 0.8 mm/ 2 |
| Top/Bottom Thickness | 0.8 mm | 0.8 mm |
| Infill/ Infill type | 15% / Lines | 30% / Grid |
| Print Tempreature/Plate Temperature | 205º/ 65º | 200º/ 60º |
| Print Speed | 30 mm/s | 60 mm/s |
| Support | Yes | Yes |

As for the methacrylate, it was cut using laser technology. The following table details the parameters used. S stands for speed and P for power.

Table 5.7: Laser Cutting Settings

|  |  |  |
| --- | --- | --- |
|  | Cutting | Etching |
| Mode | Rasterize | Vectorization |
| Speed/ Power | S40/ P60 | S10/ P60 |
| Frequency | - | 85 |

# Validation

With the purpose of ensuring the correct working of the safety device, validation tests must be done before the end of the project. The objective of these validation tests will be to ensure the correct working of all the hardware pieces, the correct detection of objects and measurements to them and display of the possible dangers.

The test must consist of a series of activities that confirm the correct measurement of the project, with measurable ways of validating. For this, various tests of car, bicycle and person detection will be completed, measuring the success rate. The distance measurement will also be included in all the tests.

To complete this test, a simulation will be done in the streets of Bilbao, riding through bike lanes, roads and pavements with cars, bikes, scooters and persons around us. Thanks to the data obtained and stored in the database, we will be able to determine the correct working of the system.

Additionally, the tests will be recorded using a Point of View (POV) camera, with the end of demonstrating how the tests were developed and provide an insight of the functioning of the prototype.

# Budget

In order to achieve optimal results within the allocated budget, it's imperative to begin with a thorough estimation. Throughout the entirety of the project, it's essential to adhere strictly to the budget. To do this, this part will be divided into the next subsections:

* Hardware estimated costs.
* Software estimated costs.
* Human resources estimated costs.

## Hardware Estimated Costs

The lack of materials has caused a shortage and in consequence, an exponential rise in the prices of electronic components. This shortage, mainly caused by the lack of semiconductors, which are used in any kind of electronic device like smartphones, cars, etc…, affects the whole industry and the customers. The COVID-19 pandemic, which led to a stop in the production of electronic wafers and semiconductor production lines [37] and the Russian Ukrainian war, with both of them being critical for the production of Palladium and Neon [38], has caused the skyrocketing of electronic components prices and the lack of stock of them.

The next table illustrates the prices of the electronic components used in the project:

Table 7.1: Hardware Costs

|  |  |
| --- | --- |
| Hardware Resource | Price (€) |
| NVIDIA Jetson Nano | 260 |
| LightWare SF45/B | 505 |
| Logitech C170 Webcam | 25 |
| Parallax GPS Module | 25 |
| Bicycle | 3267 |
| PCB and Components | 80 |
| TOTAL: | 4162 |

## Software Estimated Costs

The development of the project was done trying to use free access software resources. This includes software for 3D design and printing, PCB design and of course, the project development on the Jetson Nano. Even though the license for SolidWorks costs around 6.300 € annually [39], the university provides free access to it.

Table 7.2: Software Costs

|  |  |
| --- | --- |
| Software Resource | Price (€) |
| JetPack SDK | 0 |
| Docker | 0 |
| Deepstream | 0 |
| KiCAD | 0 |
| SolidWorks | 0 |
| Ultimaker Cura | 0 |

## Human Resources Estimated Costs

When carrying out this project, the staff was made of two persons: An Electronic Engineer and the Director. Considering the degree’s medium hourly rate [40], and the time dedicated to the project that rounds to 400 hours, an estimated is obtained.

Table 7.3: Human Resources Costs

|  |  |  |  |
| --- | --- | --- | --- |
| Human Resources | Hourly Rate (€) | Number of Hours | Total Cost (€) |
| Director | 50 | 20 | 1000 |
| Electronic Engineer | 15.6 | 400 | 6240 |
| TOTAL | - | - | 7240 |

# Ethical Validation

The present project, focused on the design and implementation of an embedded platform for light mobility vehicles (LMVs), addresses multiple ethical issues related to safety, equitable access to technologies, and environmental sustainability.

The first ethical consideration pertains to the use of artificial intelligence (AI). Recent advancements in this field have placed AI at the forefront of technological discussions. This project incorporates several ethical principles in the application of AI.

The usage of AI in this project aligns with the principles of beneficence and non-maleficence. The project has the potential to significantly improve the safety of LMVs, preventing accidents and enhancing the overall user experience. However, since the camera will be scanning and recording in public spaces, it is crucial that data transmission is securely handled to ensure the privacy and security of both the user’s and the public’s data.

The potential future inclusion of a similar system in the Bilbao Town Hall's bicycle program, BilbaoBici, exemplifies the principle of justice, as it will enable all citizens to equally benefit from the accessibility provided by AI technologies.

Lastly, the ability for users to control and customize the AI systems according to their needs and preferences aligns with the principle of autonomy. This empowers users to take control of the AI, understand how their data is used, and make informed decisions about their interaction with the system.

Cycling has a much lower environmental impact compared to other modes of transport, especially motorized and fuel-consuming vehicles. Aligned with the Sustainable Development Goals (SDGs), and through the goal of enhancing the safety of light mobility vehicles, this project contributes to the achievement of the following SDGs:

* Goal 3: Good Health and Well-being: Promoting safer light mobility vehicles can potentially decrease the number of traffic accidents involving LMVs, particularly bicycles. This reduces the high mortality and injury rates associated with these vehicles, thereby improving public health and well-being.
* Goal 7: Affordable and Clean Energy: Cycling is an energy-efficient mode of transportation that does not rely on fossil fuels. Other LMVs, such as e-scooters and e-bicycles, use electric energy sourced from clean energy providers, supporting the goal of promoting clean energy usage.
* Goal 9: Industry, Innovation, and Infrastructure: The development of this project contributes to innovation and advances research in new transportation technologies.
* Goal 11: Sustainable Cities and Communities: Encouraging the use of LMVs as a mode

of transport helps reduce traffic congestion, pollution, and carbon emissions, contributing to more sustainable urban environments.

* Goal 13: Climate Action: Cycling promotes the reduction of greenhouse gas emissions and the use of clean energy, thereby mitigating climate change and supporting sustainable transportation methods.

By focusing on improving the safety of light mobility vehicles, this project supports the achievement of a more sustainable future through the Sustainable Development Goals.

# Closure and Future Work

## Closure

The development of this project has created the opportunity for a new research line in the field of embedded systems aimed at preventing road accidents and enhancing overall security for light mobility vehicles (LMVs). It has been immensely rewarding to undertake this project using cutting-edge technologies like AI vision algorithms, 3D printing, and PCB fabrication, and combining all these elements to create a stable and functional prototype.

Once the project has been finished, it is safe to say that all the objectives have been successfully met. Although the project is still on a prototyping phase, it is fully functional and entirely fulfils its duty as an intelligent services provider.

The environmental impact, a crucial aspect when evaluating any engineering project, has been positively addressed. The security measures developed can significantly promote bike usage, directly reducing greenhouse gas emissions and improving the general well-being of the population, in addition to decreasing traffic accidents. The deployment of the prototype in the city’s public bicycle system would amplify these benefits, helping democratize technology by promoting the use of cutting-edge innovations and facilitating their control. This approach not only enhances public safety but also fosters greater acceptance and utilization of advanced technological solutions in everyday urban mobility.

Lastly, thanks to meticulous planning and a thoughtful design process, the project has established a solid foundation for future enhancements and developments. The comprehensive approach taken during the initial phases ensured that each component, from the AI vision algorithms to the 3D-printed parts and the PCB design, was carefully integrated and optimized for performance and reliability. This strong groundwork not only facilitated the creation of a functional prototype but also identified key areas for potential improvement and innovation.

As a result, the project is well-positioned to adapt to emerging technologies and incorporate additional features, such as Solid State LiDARs, improved communication protocols, or more sophisticated data analytics. Furthermore, the project's documentation and modular design enable seamless updates and scalability, allowing it to evolve alongside advancements in the field. This strong foundation ensures that future iterations can build upon the current success, driving continuous improvement in safety and functionality for LMVs, and contributing to the broader goal of creating smarter, more sustainable urban environments.

## Future Work

Although the project development could have been more comprehensive, many planned features were not implemented due to time constraints. Among these, the development of an IoT web application stands out as a crucial next step. This web application would enable users to visualize and interpret collected data, such as distance travelled, average speed, and other saved parameters, on their smartphone, tablet, or laptop. The implementation would involve using a Docker container with a React image.

Another potential upgrade is the redesign and fabrication of all the 3D-printed components using technologies that produce stronger and lighter pieces. Various alternatives could be explored, such as printing with different materials like ABS or other advanced composites, or using injection moulding. Redesigning these pieces would allow for better tolerances and an overall improved fit to the bicycle.

Regarding the PCB, significant improvements can be made. In this project, the creation of the LED matrix was done similarly to a Raspberry Pi Pico HAT, facilitating the change of microprocessor through socket pins. Instead, implementing an embedded SMD microchip like the RP2040 or alternatives such as the ESP32 or STM32 Cortex M0+ would reduce PCB costs and simplify the process. Additionally, incorporating a Wi-Fi module or similar technologies like LoRa or ZigBee could eliminate the need for a physical connection between the two assemblies, enabling independent functioning. Moreover, integrating a small lithium battery would allow for autonomous use.

Further enhancements could involve adding more advanced sensors to provide additional information, improving obstacle detection and reliability in various environmental conditions. Suitable alternatives include additional 2D or 3D LiDARs, thermal cameras, or webcams. The addition of a more precise GPS could allow, combined with an open map software, for exerting more control over the bike.

Establishing communication protocols between LMVs and other vehicles or infrastructure can also enhance overall traffic safety and efficiency. Finally, integrating the platform with IoT applications can facilitate real-time data sharing and remote monitoring, offering valuable insights for urban planning and traffic management.

These advancements will not only enhance the safety and functionality of LMVs but also contribute to broader efforts in creating smarter, more sustainable urban environments. Future iterations of the project should continue to prioritize ethical considerations and sustainability, ensuring that technological progress benefits all members of society equitably.

# Bibliography

|  |  |
| --- | --- |
| [1] | Deustotech Smart-Mobility Research Team , Breve descripción de los objetivos de investigación, 2024. |
| [2] | Gobierno de España, Ministerio de Transportes y movilidad Sostenible, «Barometro de la bicicleta,» 2022. |
| [3] | European Road Safety Observatory, «Road Safety Thematic Report, Personal Mobility Devides,» 2021. |
| [4] | M. Satyanarayanan, «‘‘The emergence of edge computing,’’,» de *Computer, vol 50*, Jan 2017, pp. 30-39. |
| [5] | W. Shi, Edge Computing: Vision and Challenges, IEEE INTERNET OF THINGS JOURNAL, VOL. 3, NO. 5, Oct. 2016. |
| [6] | Mandeep Kaur, Manish Mahajan, «Using encryption Algorithms to Enhance the data security in Cloud Computing,» de *International journal of communication and Computer technology, vol 1*, Jan 2013. |
| [7] | S. S, «Cloud computing an overview,» de *International Journal of Engineering & Technology, 7 (4) (2018) 2743-2746*. |
| [8] | D. M. V. Rieman, "Self-Driving Cars Will Create 2 Petabytes of Data,What are the oportunities for the car Industry?," 2013. |
| [9] | L. M. M. A. N. A. A. gerondimos, «IoT: Communication protocols and security threats,» 2023. |
| [10] | Cisco, «Fog Computing and the Internet of Things: Extend the Cloud to Where the Things are, White Paper,» Accesed on Abr. 1 2024, Available: https://www.cisco.com/c/dam/en\_us/solutions/trends/iot/docs/, 2015. |
| [11] | Transformainsights, «Current IoT Forecast Highlights,» Available: https://transformainsights.com/research/forecast/highlights, 2023. |
| [12] | V. d. A. A. K. L. M. D. Gupta, «Smart Sensors for Industrial Internet of Things: Challenges, Solutions, and Aplications,» 2021. |
| [13] | A. Shandaliya, «Role and Applications of IoT in Online Transactions using,» 2018. |
| [14] | K. S. M. Ahmad Khan, «IoT security: Review, blockchain solutions, and open challenges,» May 2018. |
| [15] | A. &. K. S. &. J. R. Dorri, «Toward an Optimized BlockChain for IoT,» 2017. |
| [16] | S. S. a. B. Bhushan, «A Comprehensive survey on Blockchain: Working, security analysis, privacy threats and potential applications,» de *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Kannur, India, 2019, pp. 922-926, doi: 10.1109/ICICICT46008.2019.8993210.. |
|  |  |
| [17] | W. H. Organization, «Global status report on road safety 2018,» Geneva, 2018. |
| [18] | M. B. Barry Brown, «The Halting problem: Video analysis of self-driving cars in traffic,» *ACM Digital Library,* 2023. |
| [19] | I. S. Cristian Sandu, «Edge computing for autonomous vehicles. A scoping review.,» 2021. |
| [20] | Tesla , «Autopilot and Full Self-Driving CapabilityDescription,» extracted from: https://www.tesla.com/support/autopilot. |
| [21] | A. J. Hawkin, «Tesla’s Autopilot and Full Self-Driving linked to hundreds of crashes, dozens of deaths,» The Verge , 2024. |
| [22] | R. Abbasi, A. K. Bashir, H. J. Alyamani, F. Amin, J. Doh y J. Chen, «Lidar Point Cloud Compression, Processing and Learning for Autonomous Driving,» 2014. |
| [23] | J. X. Nanxi Li, «A Progress Review on Solid-State LiDAR andNanophotonics-Based LiDAR Sensors,» *lpr-journal,* 2022. |
| [24] | GeoTab, «What Is GPS and how do global positioning systems work?,» March 2024. |
| [25] | C. BasuMallick, «What Is GPS (Global Positioning System)? Meaning, Types, Working, Examples, and Applications,» 2022. |
| [26] | Kubernetes, «Overview,» [En línea]. Available: https://kubernetes.io/docs/concepts/overview/. [Último acceso: 06 05 2024]. |
| [27] | Google Cloud, «Kubernetes pricing,» [En línea]. Available: https://cloud.google.com/kubernetes-engine/pricing?hl=es#:~:text=Engine%20(GKE).-,Standard%20edition,%240.10%20per%20cluster%20per%20hour.. [Último acceso: 06 05 2024]. |
| [28] | DigitalOcean, «Kubernetes Pricing,» [En línea]. Available: https://www.digitalocean.com/pricing/kubernetes. [Último acceso: 06 05 2024]. |
| [29] | Docker, «Docker Overview,» [En línea]. Available: https://docs.docker.com/get-started/overview/. [Último acceso: 06 05 2024]. |
| [30] | J. Campbell, «Comparación entre Kubernetes y Docker,» [En línea]. Available: https://www.atlassian.com/es/microservices/microservices-architecture/kubernetes-vs-docker#:~:text=Docker%20es%20una%20plataforma%20de,tiempos%20de%20ejecución%20de%20contenedores.. |
| [31] | Node-Red, «Node-Red Low-code programming for event-driven applications,» [En línea]. Available: https://nodered.org. [Último acceso: 07 05 2024]. |
| [32] | J. Martinez, «Qué es InfluxDB y primeros pasos». |
| [33] | Eclipse Foundation, «Eclipse Mosquitto™,» [En línea]. Available: https://mosquitto.org. [Último acceso: 07 05 2024]. |
| [34] | Apache Foundation, «Apache Kafka,» [En línea]. Available: https://kafka.apache.org. [Último acceso: 07 05 2024]. |
| [35] | NVIDIA, «NVIDIA DeepStream SDK,» [En línea]. Available: https://developer.nvidia.com/deepstream-sdk. [Último acceso: 07 05 2024]. |
| [36] | NVIDIA, «TAO Toolkit Overview,» [En línea]. Available: https://docs.nvidia.com/tao/tao-toolkit/text/overview.html. |
| [37] | Proto-Electronics, «Cómo afrontar el impacto en el PCB de la escasez de componentes,» 2023. |
| [38] | M. Gimenez, «La guerra de Ucrania golpea más aún a la industria de los chips,» CincoDias, 2022. |
| [39] | Solidworks, «SOLIDWORKS Licensing,» [En línea]. Available: https://www.solidworks.com/how-to-buy/solidworks-licensing. [Último acceso: 07 05 2024]. |
| [40] | talent.com, «Salario medio para Ingeniero Electronico en España, 2024,» [En línea]. Available: https://es.talent.com/salary?job=ingeniero+electronico. |
| [41] | United Nations, «The 17 Goals». |
| [42] | A. C. e. al., «Key considerations in the development of,» de *24’th Enhanced safety vehicles*, Gothemburg, Sweden, 2015. |
| [43] | NVIDIA, «DashCamnNet,» [En línea]. Available: https://docs.nvidia.com/tao/tao-toolkit/text/model\_zoo/cv\_models/dashcamnet.html. [Último acceso: 07 05 2024]. |

1. DetectNet\_v2 is an NVIDIA-developed object-detection model that is included in the TAO Toolkit [↑](#footnote-ref-1)
2. ResNet-18 is a convolutional neural network (CNN) architecture that belongs to the ResNet (Residual Network) family. [↑](#footnote-ref-2)