OMI RON

Intelligent Road Asset Management Platform

Modular Robotic Platform Prototype

Deliverable D3.2

27 April 2023

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 955269

Technical References

Project acronym	OMICRON
Project full title	Towards a more automated and optimised maintenance, renewal and upgrade of roads by means of robotised technologies and intelligent decision support tools
Grant number	955269
Project website	https://omicronproject.eu/
Coordinator	CEMOSA

Deliverable No.	D3.2
Deliverable nature	R
Work Package (WP)	Work Package 3
Task	Task T3.2
Dissemination level [1]	PU
Number of pages	77
Keywords	Prototype, Hardware, Software, Robot Platform
Due date of deliverable	30.04.2023
Actual submission date	27.04.2023

[1] PU = Public

PP = Restricted to other programme participants (including the Commission Services)

RE = Restricted to a group specified by the consortium (including the Commission Services)

CO = Confidential, only for members of the consortium (including the Commission Services)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Authors

Name	Beneficiary	Details of contribution
Ander Ansuategi Cobo	TEK	Deliverable responsible and author
LMS Team	LMS	Deliverable reviewer
Jose Solís Hernández	CEM	Deliverable reviewer

Document history

V	Date	Beneficiary	Author
V0.1	13.04.2023 - Complete version for revision	TEK	Ander Ansuategi
V0.2	24.04.2023 - Technical revision	LMS	LMS
R1	27.04.2023 - Final revision	CEM	Jose Solís Hernández

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Executive Summary

The aim of this document is to provide a description of the implementation of the Modular Robotic Platform that is developed in OMICRON project.

The starting point for the implementation of the modular robotic platform has been the hardware and software design of the platform itself, which was worked on in task T3.1 and is described in deliverable D3.1.

This document describes the implementation of both hardware and software of the Modular Robotic Platform. The hardware includes the following points:

1. The basic solution, which consists of a customised container to house the necessary elements and a robot arm that performs the operations.
2. The basic elements that make up the platform, consisting mainly of the power supply and communication interfaces.
3. The camera-based perception system.
4. The tools necessary to handle the different elements in the operations to be carried out.

For the software, a four-layer architecture has been implemented on the ROS (Robot Operating System) framework, including the drives, the modules for robot manipulation, the perception modules and the specific applications for each operation.

Disclaimer

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table of tables

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table of figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

List of abbreviations

Abbreviation	Description
API	Application Programming Interface
BSD	Berkeley Software Distribution
LAN	Local Area Network
LED	Light Emitting Diode
PC	Personal Computer
PLC	Programmable Logic Controller
QoS	Quality of Service
RGB	Red, Green, Blue (color model)
RGB-D	Red, Green, Blue, Depth
ROS	Robot Operating System
SSD	Solid State Drive
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

1 Introduction

This document contains a description of the development of the modular robotic platform to perform multiple road maintenance operations. The document includes both hardware and software implementation descriptions. The implementation starts from the design done in WP3 task T3.1, which has the main objective to fulfil the following requirements:

- **Cope with multiple maintenance intervention actions.** The modular robotic platform will allow the following interventions to be covered: (1) assist in the installation of safety barriers; (2) installation of safety cones; (3) assist in the installation of signalling during construction works; (4) cleaning of vertical signals; (5) removal of horizontal marks; (6) sealing of pavement cracks.
- **Distributed computing**. The architecture will be designed with distributed computing in mind. The main benefits over a centralized system for the OMICRON solution are that (1) the several tasks can be computed in individual computers while still appearing to its users as a single coherent system and (2) the system can be easily expanded.
- **Safe operation of the overall system**. The system will be equipped with components enabling the robot to operate safely.
- **Ease of use.** The system will have mechanisms for easy and intuitive use (e.g., for teleoperation).

Purpose and scope

The purpose and scope of this document is to report on the results of the general robot platform architecture development for the OMICRON solution, a work carried out in Task 3.2 in WP3.

Document structure

The document is composed of the following main sections:

- Section 2 provides an introduction to the design and functionality of the platform.
- Section 3 describes the hardware implementation of the Modular Robotic Platform prototype, providing details about the main components of the system, as well as the power supply and communication interfaces.
- Section 4 describes the development of the platform's control software. This section includes information on the programming languages and tools used, as well as a detailed description of the software architecture.
- Section 5 explains how the hardware and software components of the platform are integrated to create a functioning system.

Finally, section 6 presents the conclusions drawn from the results of the task.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

2 Overview of the Modular Robotic Platform

The modular robotic platform developed at OMICRON consists of a container with an extended section at the back, where a robotic arm is located to perform various maintenance operations. The platform has been developed to be able to perform the following road maintenance interventions:

1. Support in the installation of safety barriers. Road maintenance workers need support in the safety barriers installation process, particularly in the elevation and support of the fence while it is being adjusted to the post. The robotic arm assumes the operation of positioning the metal safety barrier, minimizing manual handling of loads, and forced postures of the workers.
2. Cones placement & collection. Within the tasks of temporary signalling the installation of cones is one of the most common activities. In this type of operation, operators are exposed to the risk of being run over, as cones are placed and picked up while traffic is moving.
3. Cleaning of road signals. The cleaning of signals is currently performed manually by road workers using water spraying systems. A robotic arm assumes the operation of cleaning the road assets, minimizing the exposure of road workers to traffic.
4. Installation of signals during construction works. Before any road works, it is mandatory to place the required signalling on site in order to inform users and order traffic along the section under works. This operation is currently done manually by workers, handling heavy loads in forced postures.
5. Removal of horizontal marks. Currently available methods for road marking removal provide suboptimal results, in terms of surface state of the pavement (colour, texture, markings) or incomplete removal, which can cause confusion among drivers when passing through these places. OMICRON proposes the study and implementation of a new laser-based cleaning system to remove paint from the road.
6. Sealing of pavement cracks. The process of sealing cracks in surface pavement layers comprises operating pipes of a dedicated machinery by a worker to follow and seal the crack. The machine heats up bituminous mixes at high temperatures. The manipulation of the bituminous materials can be dangerous, as they are very sticky and at high temperature, so any single mistake can induce severe burns in the worker. The robotic modular platform supports road workers in the operation of sealing cracks in surface pavement layers.

This container contains sufficient space to house all the elements that make up the system and has been custom designed to meet the requirements of the operations. This container can be loaded onto a truck using a standard hook system, offering the advantage that the multi-purpose modular robotic platform solution is not tied to a specific truck, and the truck can be rented to perform the operations. The rear platform contains a vertical and horizontal linear motion systems to allow the robot to be positioned at different heights and move from side to side.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

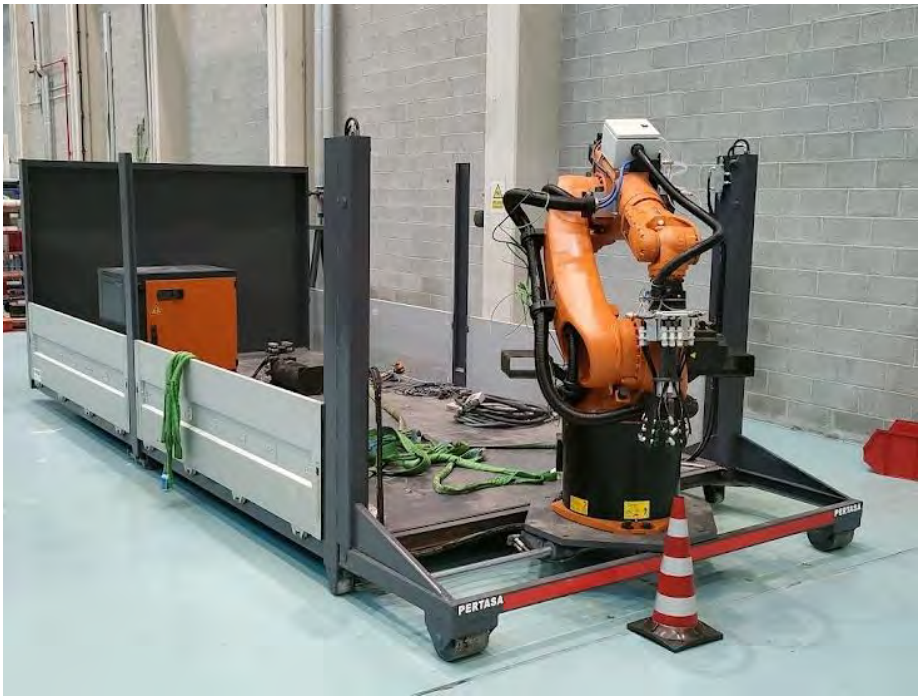


Figure 1. The Modular Robotic Platform prototype

The assistance of the robotic system in multiple operations is only possible through the combination of machine vision and AI tools. Perception systems have been developed for the automatic detection of cones, traffic signs and cracks in the pavement. In addition to this, tools have been designed and developed for the handling of elements by the robot. A software architecture has been designed and developed on top of ROS framework, which allows a manufacturer independent solution.

The following sections describe in more detail the elements that make up the platform.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

3 Hardware implementation

This section describes the hardware implementation of the modular robotic platform prototype. We provide details on the components used to build the platform, along with the power supply and communication interfaces.

3.1 Main hardware elements

3.1.1 KUKA KR60 industrial robot arm

The selected robot arm is a well-known KUKA KR60 industrial robot, a six-axis articulated robot. This robot has been selected because TEKNIKER already has one and it meets the requirements needed to perform the various road maintenance operations.

Figure 2. KUKA KR-60 robot

The following table contains the basic data of the arm:

Table 1. KUKA KR-60 robot specifications

Characteristic	Value
Number of axes	6
Weight	665 Kg
Payload	60 Kg

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Characteristic	Value
Reach	2033 mm
Repeatability	0.06 mm
Power requirements	480 volts 3 phase; 18 amps

The robot cabinet used for KUKA KR-60 is the KR C4.

Figure 3. KR C4 cabinet for KUKA robot

The following table contains the basic data of the robot cabinet.

Table 2. Robot cabinet specifications

Characteristic	Value
Model	KR C4
Weight	144 Kg
Dimensions LxWxH	792x558x960mm
Ingress Protection Code (IP)	50
Operating temperature range	+5ºC to +45ºC

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Characteristic	Value
Power requirements	AC3 x 380/400/440/480V

3.1.2 Main Computer

The computer selection process considered multiple factors including computing power, communication interfaces with various elements, and the ambient conditions of the OMICRON system. As AI implementation requires significant computational resources, a high-performance graphics card will be integrated into the computer. However, incorporating such a graphics card into industrial equipment is often impractical. Therefore, it was decided to use a conventional PC and apply certain measures, such as using an SSD disk and enclosing the PC within a cabinet, to make it more closely resemble an industrial PC.

Figure 4. Dell Precision 3660 computer

The following table contains the basic data of the PC:

Table 3. Dell PC specifications

Characteristic	Value
Manufacturer	Dell
Model	Precision 3660
CPU	Intel© Core™ i7-12700CPU @2.1 to 4.9Ghz x 12
Memory	32GB

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Characteristic	Value
GPU	Nvidia GeForce RTX3060, 3660T
I/O	2 x USB 3.2 Gen2 (Type-A) 2 x USB 3.2 Gen2 (Type-C) ComPort 2 LAN Ports
Dimensions (WxLxD) (mm)	170x330x300mm
Operating temperature range	+0ºC to +50ºC
Power requirements	230 VAC

3.1.3 PLC

The selected PLC is a PILZ PNOZmulti2. The PNOZmulti2 is a programmable logic controller (PLC) designed for safety-related applications. It is a compact device that integrates safety functions, such as emergency stop, light curtain, and guard door monitoring, with standard control functions. It supports a variety of communication protocols, including Ethernet, CANopen, and PROFIBUS, which allows it to communicate with other devices and systems.

Figure 5. PILZ PNOZmulti2 PLC.

3.1.4 Access Point

The Brightassistant Router WiFi MóDem 4G is a networking device that provides wireless internet connectivity through cellular networks. The device has built-in WiFi and 4 LAN ports that can be used to connect wired devices to the network. The Brightassistant Router WiFi MóDem 4G can be configured and managed using a web-based interface, which provides access to a variety of features, including network security settings, network management, and device status monitoring.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Figure 6. Brightassistant Router WiFi MóDem 4G

The following table contains the basic data of the access point:

Table 4. Brightassistant Router WiFi MóDem 4G specifications

Characteristic	Value
Manufacturer	Brightassistant
Model	500335411A2
Dimensions (WxLxD) (mm)	200 x 120 x 80
Weight (gr)	750
Standards and protocols	Wi-Fi 802.11g, Wi-Fi 802.11b, Wi-Fi 802.11n
Power requirements	10W/240VCA

3.1.5 Switch

Since the access point has only 4 LAN ports, and so that this is not a limitation in case we need to connect more devices, we have added a switch, to provide the possibility to connect multiple devices on the network.

The NETGEAR GS308E is a compact and affordable 8-port gigabit Ethernet switch, equipped with advanced networking features, including VLAN (Virtual Local Area Network) support, QoS (Quality of Service) for prioritizing network traffic, and loop detection to prevent network loops. It also has a web-based management interface, which allows you to configure and monitor the switch from a computer or mobile device.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Figure 7. NETGEAR GS308E switch

3.1.6 RGB-D cameras

In the selection process of the RGB-D cameras to be used with the perception modules, three type of cameras have been considered and tested:

- Intel RealSense D435.
- OpenCV OAK-D-PoE.
- ZED 2i.

Figure 8 The three types of RGB-D cameras considered and tested. From left to right: Intel RealSense D435, OpenCV OAK-D-PoE and Zed 2i

The first camera considered was the Intel RealSense D435. The D435 features a high-resolution depth sensor that can capture 1280 x 720 depth frames, as well as an RGB camera that can capture 1902 x 1080 frames at up to 30 frames per second. It also includes an infrared projector and stereo infrared cameras that allow it to work in a variety of lighting conditions. Moreover, it has an official ROS driver that can be used to interface with the camera, allowing an easy incorporation of the camera in our software architecture. However, this camera is intended for indoor use, and while it can tolerate some level of moisture or dust, it is not recommended for use in harsh outdoor environments.

This led us to consider alternative cameras better suited to outdoor environments. The OpenCV OAK-D-PoE camera is a depth camera that has an official IP67 rating, which means that it is dust-tight and can withstand being immersed in water up to a depth of 1 meter for up to 30 minutes. This makes it suitable for use in a variety of harsh outdoor and industrial environments, where moisture, dust, and debris may be present. It features a high-resolution depth sensor that can capture 1280 x 720 depth frames at up to 30 frames per second, as well as a 12-megapixel RGB camera that can capture 4K video at up to 30 frames per second. Moreover, this camera has an ethernet interface, instead of USB. This feature is desirable, since we will need long cables from the camera to the location of the PC or switch, and Ethernet connections are generally better than USB connections for long cables, reducing the risk of signal loss and interference.

The OpenCV OAK-D-PoE camera that we tested, however, does not provide an IR dot pattern projection, which helps to improve the accuracy and reliability of the depth sensing. Therefore, further tests showed that the camera was not able to provide good depth information over uniform traffic signals surfaces.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Finally, we tested a Zed 2i camera. This camera, manufactured by Stereolabs has an official IP66 rating. A recent scientific paper [1] led by PhDs in Image Processing and Robotics from the University of Pecs, Hungary, compared the performances of the Zed cameras with Intel's Realsense. According to their experiments, ZED 2i cameras generated a depth map of very good quality (which is not the case for the images captured with the RealSense cameras).

Considering all the above, Zed 2i cameras are integrated as the first option. Depending on future tests during the development of operations, however, a change in the type of cameras is not discarded. In fact, the implementation of the perception modules described in section 4.4 of this document have been implemented so that they can be used independently of the RGB-D camera being used. This is due to the fact that the perception modules receive as input data from the cameras via ROS topics using standard ROS messages. This means that any RGB-D camera can be used, as long as there is a driver that publishes the camera information via ROS.

3.1.7 Air compressor

The function of the compressor is to provide pressurised air to those elements that need it, such as the suction cup tools, as well as the crack sealing tool, for its blowing operation.

Figure 9. Air compressor

The main characteristics of the air compressor are:

Table 5. Air compressor specifications.

Characteristic	Value
Max Pressure	8 Bar
Tank Capacity	125 L/min
Flow rate	125 L/min
Power Consumption	750W/230 VCA
Weight	20Kg
Dimensions	640 x 650 x 340 mm

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

For the initial tests, the elements requiring compressed air will be connected to the compressed air system in TEKNIKER's workshop. The compressor will be used for the operation on the truck.

3.1.8 Sealing machine

The sealing machine that has been integrated into the modular robotic platform is a machine supplied by PAVASAL. This sealing machine has had to be modified to reduce its size and to be able to hold it in the container.

The adaptation of the sealing machine has been done in task T4.5 of WP4 of the OMICRON project, and the detailed description will be included in deliverable D4.7. The following figure shows the machine once it has been adapted.

Figure 10. The adapted sealing machine

3.1.9 Pressure washer

The function of the pressure washer is to apply a strong, concentrated stream of water to the vertical traffic signals to clean them effectively.

Figure 11. Pressure washer.

The main characteristics of the pressure washer are:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 6. Pressure washer specifications.

Characteristic	Value
Manufacturer	Cecotec
Flow rate	540 L/h
Max pressure	225 bar
Power Consumption	3200W
Weight	31.85 Kg
Dimensions	1000 x 460 x 430 mm

3.1.10 Remote emergency stop button

An industrial enhanced safety radio remote control has been incorporated into the system (see Figure 12). The emergency button of this remote control ensures safe operation. This allows the operator to safely stop the robot's movement at any time.

Figure 12. Jay Electronique UR Series industrial radio remote control system.

3.1.11 Lighting system

The function of the lighting system is to provide sufficient illumination to the area of the pavement where the operations are to be carried out, to obtain images with the cameras for the subsequent identification of assets using the perception modules.

The selected light sources are LEDs, due to their good ratio between illumination level and consumption. Tests have been carried out with different types of light, finally selecting diffuse light sources, which prevent the appearance of focused light rings in the camera images (which negatively affects the perception modules).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Figure 13. Selected LED light source.

Characteristic	Value
Manufacturer	JBM
Type	Diffused light
Brigthness	1450 lumens
Voltage	10-40V
Power Consumption	9 leds 3W
Dimensions	140 x 45 x 123 diam. mm

3.2 Power supply and communication interfaces

The following diagram shows the communication interfaces between the main hardware components of the Modular Robotic Platform:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

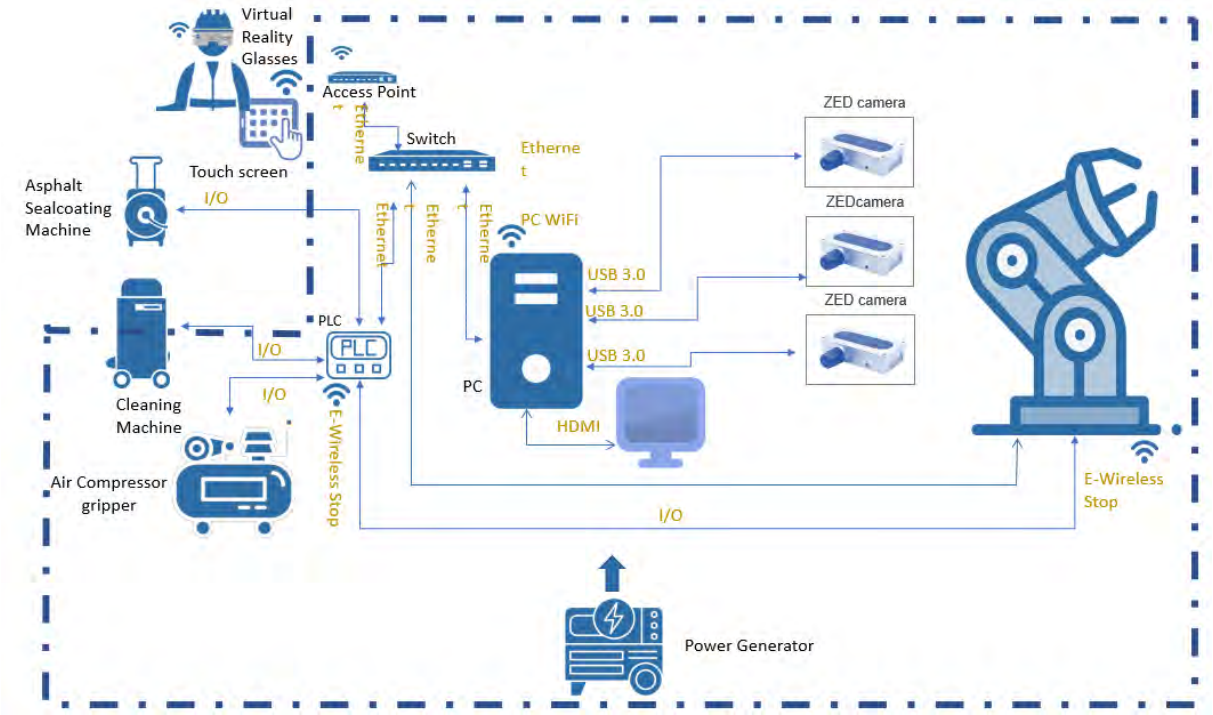


Figure 14. Communication interfaces between the main hardware components

The consumption of each of the elements that make up the system has been identified. The following table shows the consumption:

Table 7. Consumption of the elements of the system

Element	Power consumption (W)
Robot (cabinet)	8500
Main Computer	500
PLC	70
Access Point	10
Switch	6
Cameras	Powered through USB
Lights	180
Air compressor	750

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Element	Power consumption (W)
Sealing Machine	3000
Pressure washer	3000
Remote emergency stop button	12

Based on this data, a maximum consumption exercise has been carried out for each of the operations for which the system will be used. This has allowed us to size the power generator that we will need.

During the static tests in TEKNIKER's shopfloor, the elements will be directly plugged into the mains. A power generator will be rented during the testing period at TEKNIKER, as well as for the technical demonstrations and the final demonstration.

An analysis has been made of available generators for rental that meet the consumption requirements of our system. This analysis has taken into account the dimensions of this equipment. This equipment usually has a large size, which is an important restriction in terms of the location of the elements in the container.

3.3 Customized container

A custom container has been built for the modular robotic platform, tailored to fit its specific requirements. The container features the following characteristics:

- Dimensions: 6000 mm (L) x 2550 mm (W) x 1850 mm (H)
- Rear lifting and lateral movement system: The container is equipped with a rear lifting system and lateral movement capability to facilitate transportation and positioning of the robotic system.
- Robot securing piece: The container includes a custom-designed piece to securely hold the robotic arm in place.
- Foldable walls: The container's walls are hinged to allow for easy access to the platform when elements need to be loaded / unloaded, during the platform configuration process to perform a specific operation.
- Support pillars and metal profiles: The container is reinforced with sturdy support pillars and metal profiles to ensure the safety and stability of the robotised system.
- Element placement zones: The container has designated areas with pre-drilled holes for the placement of static elements, as well as specific elements required for each operation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Figure 15. Customized container built for the modular robotic platform.

3.4 Robot tools for the operations

This section presents the tools designed and manufactured to enable the robot to manipulate various assets different operations. The specifications and features of the tools are described.

3.4.1 Cone handling tool

The cone handling tool will be provided by a three-finger gripper to hold the cone from the top. In order to avoid slippage of the cone, each of the fingers will have three venturi activated suction cups.

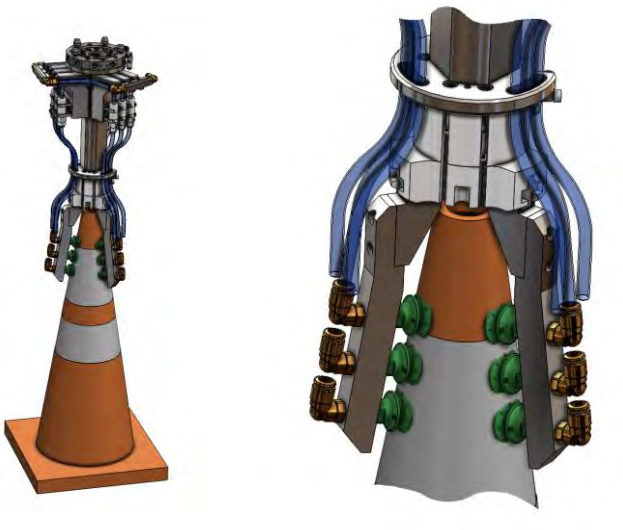


Figure 16. 3D model of the cone handling tool

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

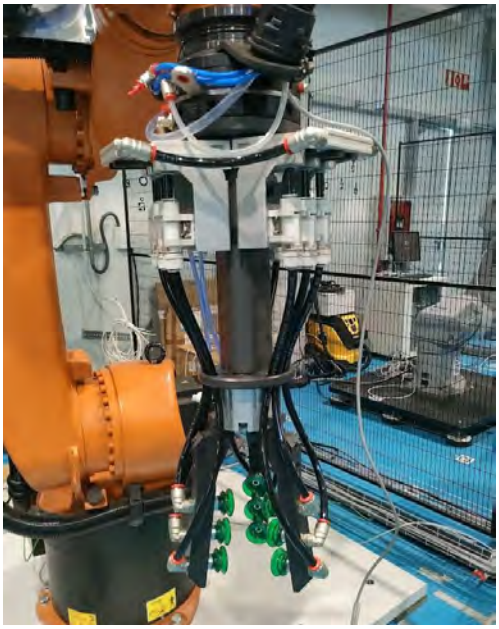


Figure 17. The manufactured cone handling tool

For cone collection, a tool has been designed to retain the road cone in a certain position, allowing the robot arm to go to that position.

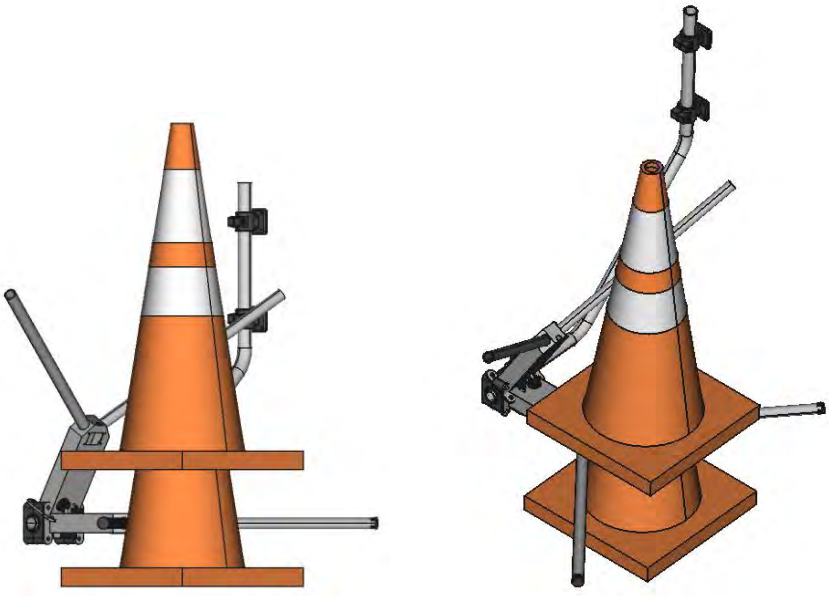


Figure 18. 3D model of the cone retaining tool

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

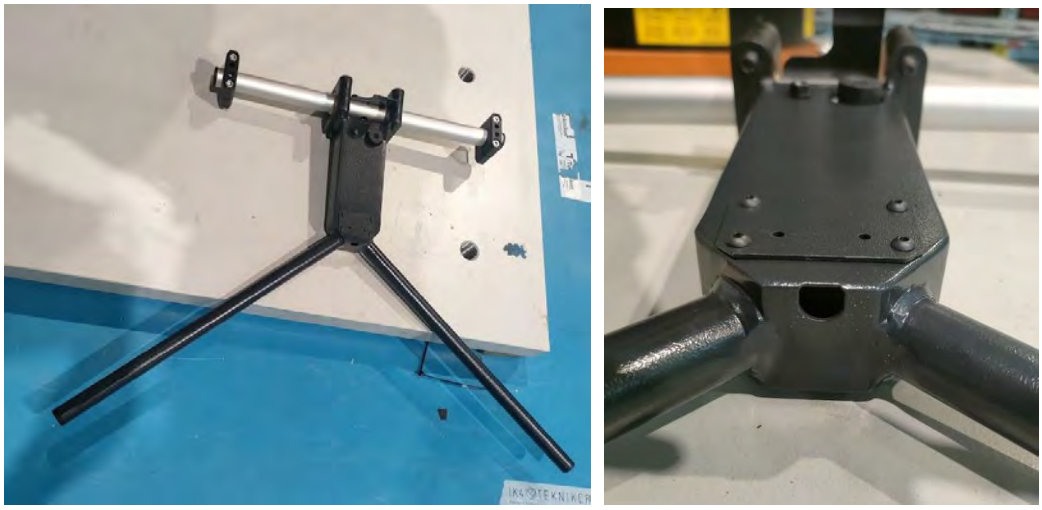
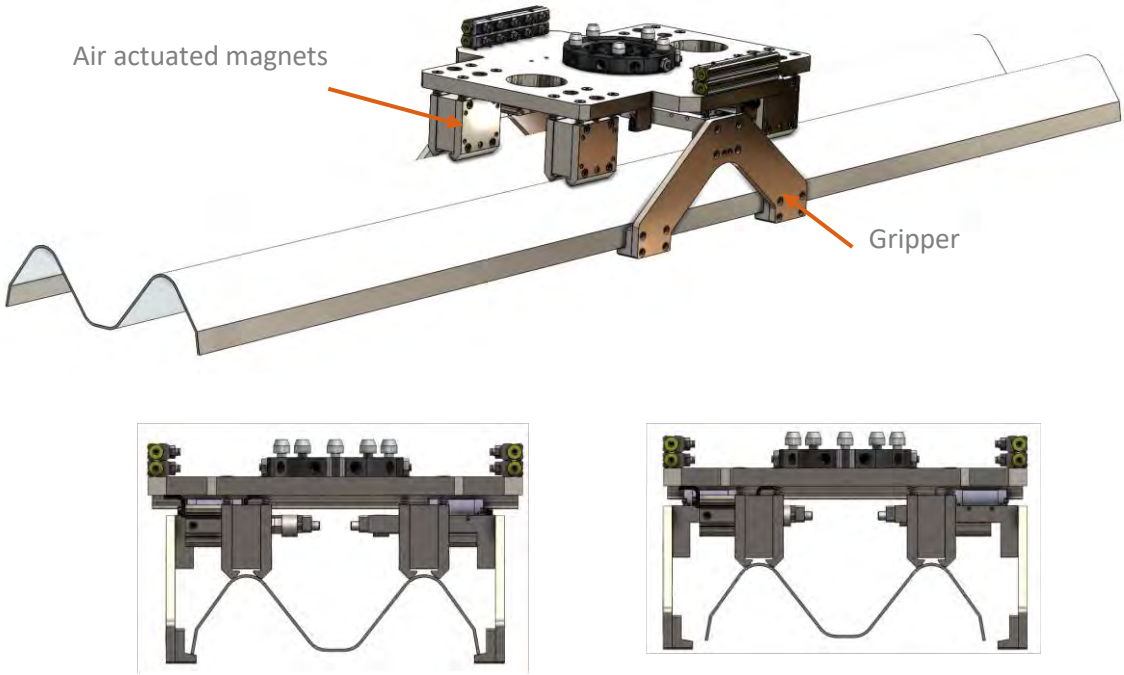


Figure 19. The manufactured cone retaining tool. The figure on the right shows the hole that houses the optical presence sensor.

This tool contains an optical presence sensor to allow the system to know the presence of the cone, and thus automate the movement of the robot arm.

3.4.2 Safety barrier handling tool

This handling tool will be provided by four magnets and a griper. When the barriers are stacked on top of each other, the manipulator will pick up and separate the top barrier from the others thanks to the magnets. These magnets will only apply a magnetic force when pneumatically activated. Once this operation has been carried out, the gripper will close and hold the barrier firmly so that it can be handled safely. A photoelectric detector will warn of the presence of the part.



Griper closed	Griper opened

Figure 20. 3D model of the safety barrier handler

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

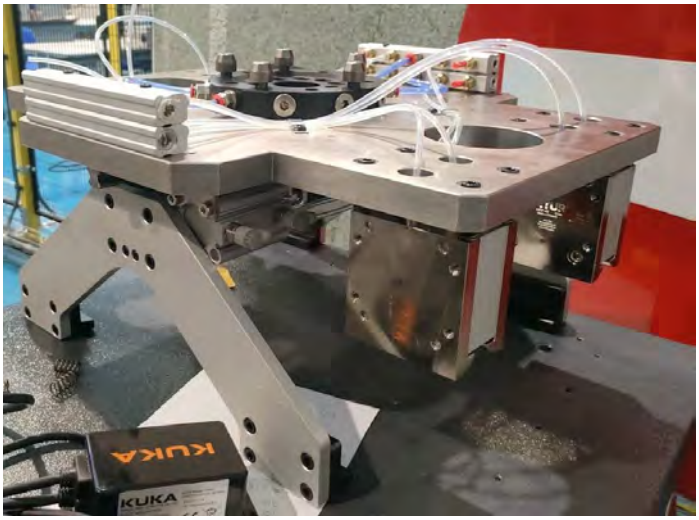


Figure 21. The manufactured safety barrier handler

3.4.3 Tool for handling signalling elements

For the operation of road works signalling, the transport vehicle will be provided of the following three buffers:

1- Traffic Signals
2- Signal Bases
3- Sandbags

The buffers have been designed to house the signs and other elements according to the signalling regulations of Italy, where the final demonstration will take place.

Because there are other elements, the space available on the container is limited, so these buffers must occupy limited floor space.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

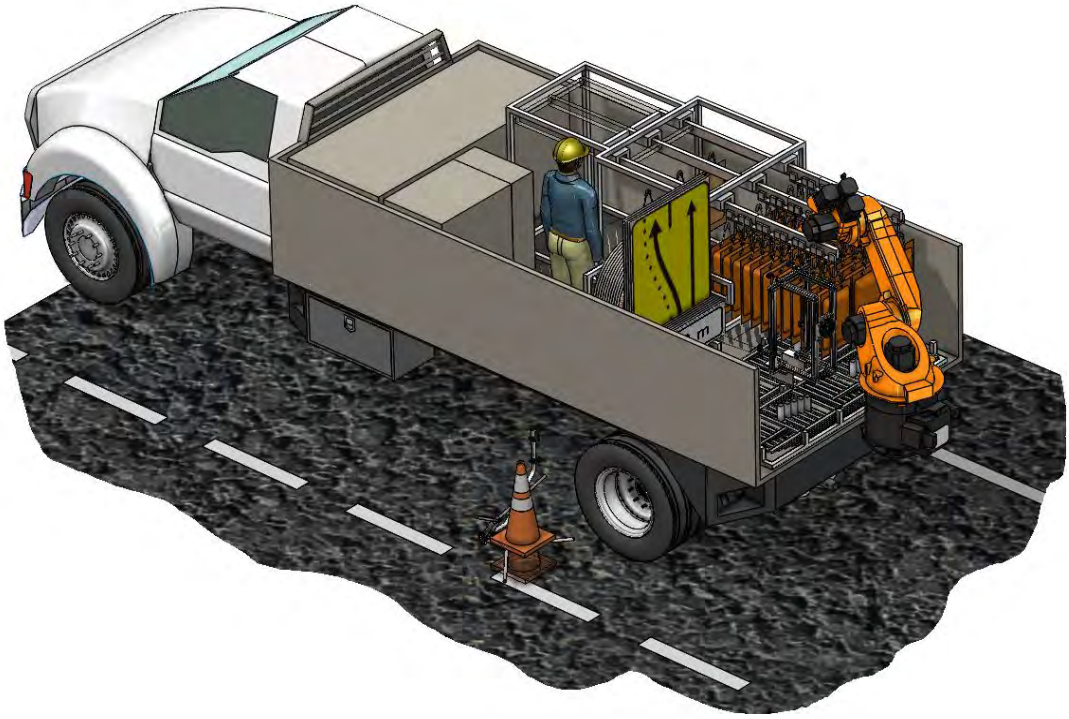


Figure 22. Transport Vehicle occupied by peripherals and signal buffers

Description of the three main buffers:

Traffic Signal Buffer:

This buffer will hold three types of signals: four rectangular signals (1355x902mm), two triangular (1110x1110x1110mm) signals and twelve circular signals (Ø895mm).

This buffer will also give a specific position to each signal to simplify the robot manipulation sequence.

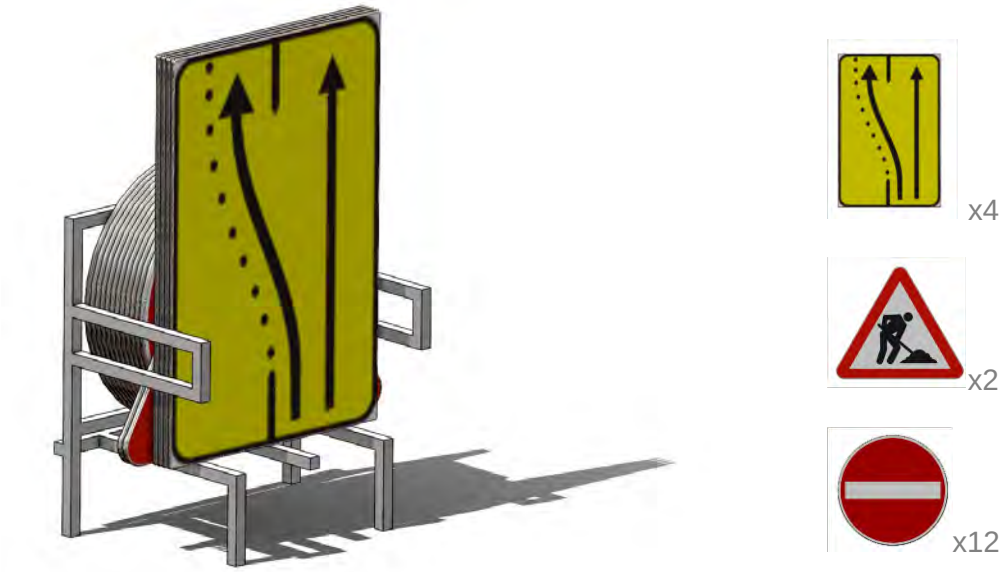


Figure 23 Traffic signals buffer

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Signal bases:

The traffic sign bases can be stacked, but no more than 5 can be stacked. This buffer will allow 4 groups of five stacked signs to be held and positioned in a fixed place.

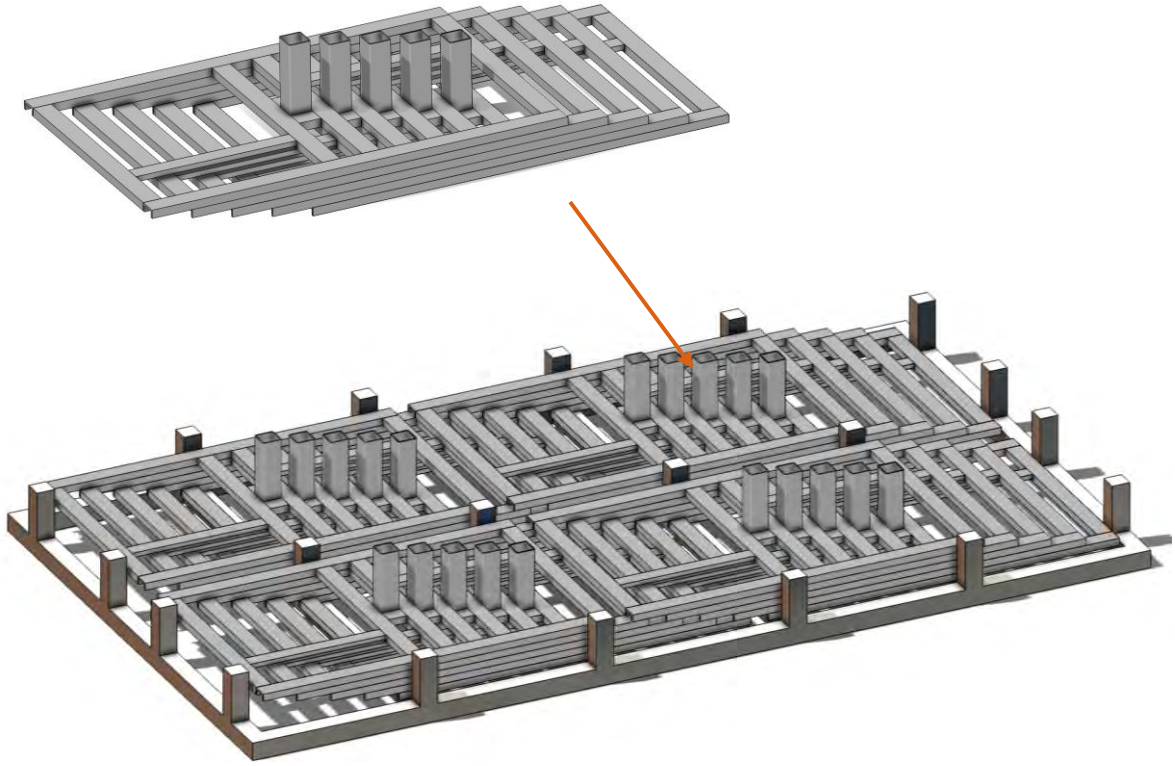


Figure 24 Signals bases buffer

Sandbags Buffer:

This buffer will stock and position 3 rows of 12 bags, thus 36 bags. Each bag is hanged by a hook fixed on a slider.

When the robot needs to pick up a bag, the three rows will lengthen to get closer to the robot's position. The robot will take the first bag from one of the three rows, after this operation, another bag will move forward by gravity and will be positioned waiting to be taken by the robot.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

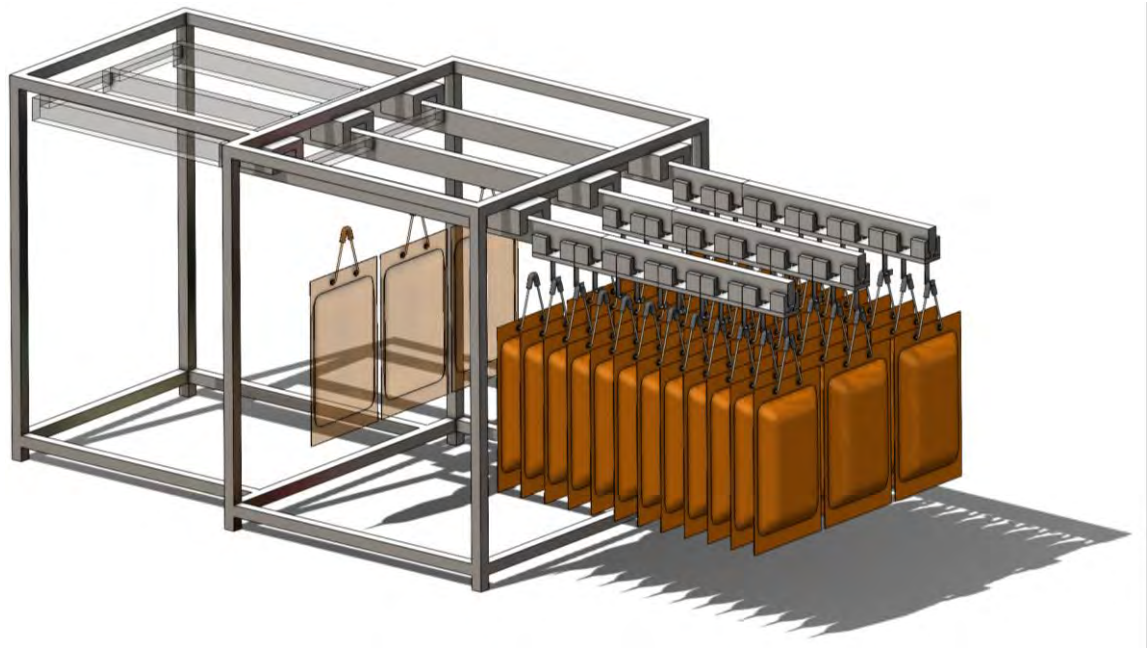


Figure 25 Sandbags buffer

Description handling tool:

This tool will be able to handle the following elements:

- Three types of signals: rectangular signals (1355x902mm), triangular (1110x1110x1110mm) signals and circular signals (Ø895mm).
- Signal Base
- Sandbag

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

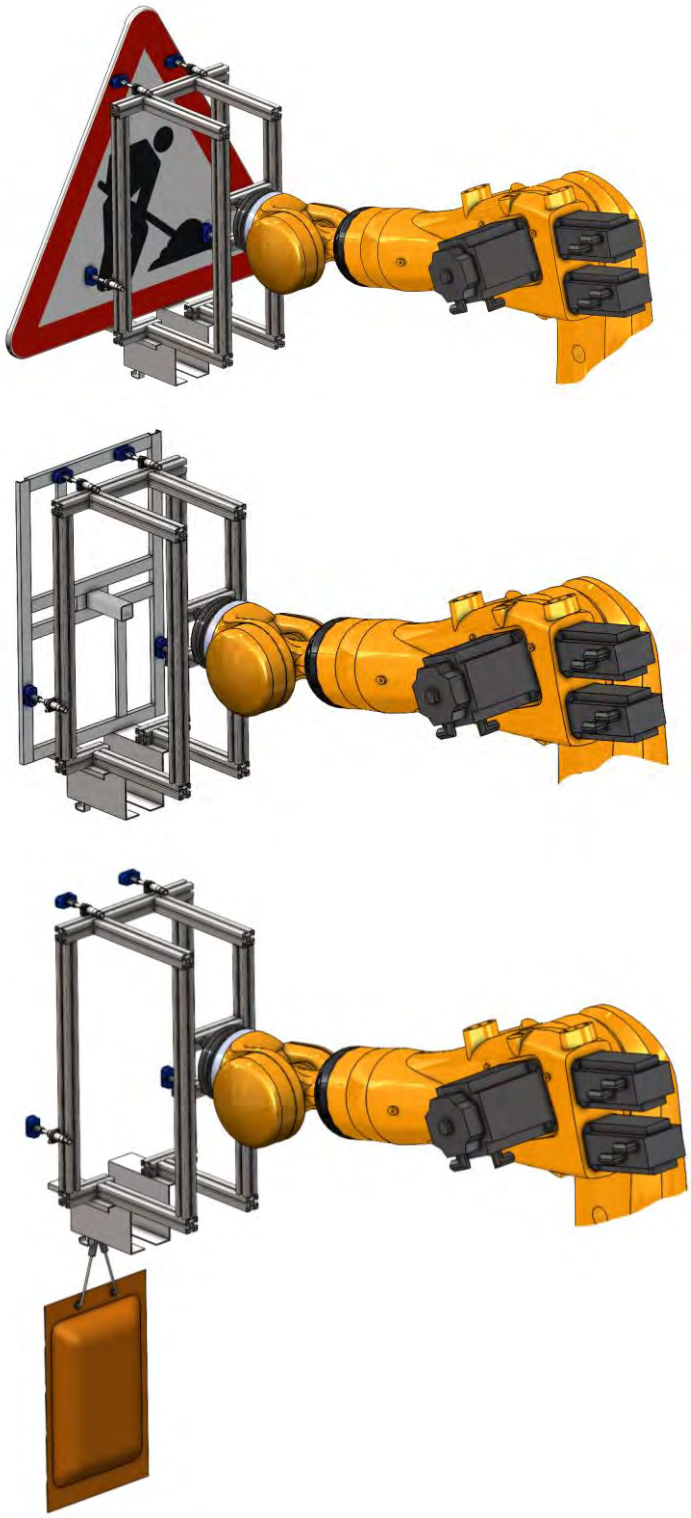


Figure 26 Handling tool.

Four venturi-actuated, rectangular-shaped suction cups are used to hold the traffic signals and the signal bases. In addition to these suction cups, a sheet metal plate placed on the bottom part of the manipulator will support the weight of the signs, so the suction cups will not be subjected to shear forces.

In order to handle the bag, the handler will have a hook at the bottom part.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

3.4.4 Clamping of the tool for sealing operations

The sealing tool is a custom-made tool for the robot, which is being developed in the context of WP4 task T4.5. The sealing tool shall be fixed to a base plate attached to the robot. This plate will have the possibility to move freely vertically thanks to a sliding system of two guided rods. This will help to absorb deviations and misalignments of the road. In the middle of the guide rods, a gas-activated spring will apply a certain force (about 10 kg) to the tool against the road.

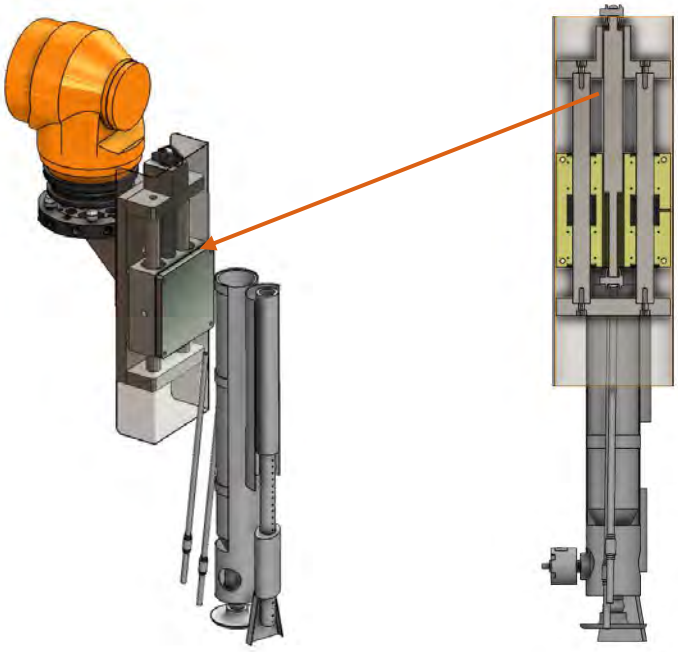


Figure 27 Sealing tool head.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

4 Software implementation

This section describes the implementation of the software necessary for the operation of the modular robotic platform.

4.1 Four-layer architecture

A four-layer architecture that was proposed in D3.1 has been implemented. It consists of four layers: the Drivers layer for interfacing with sensors and actuators, the Abilities layer for providing robot capabilities, the Application layer for implementing robot applications, and the User Interface layer for interaction between the user and the system. Additionally, there is a logging module that tracks the processes being performed and detects any possible issues in the system.

displays the overall framework of the software architecture at a high level.

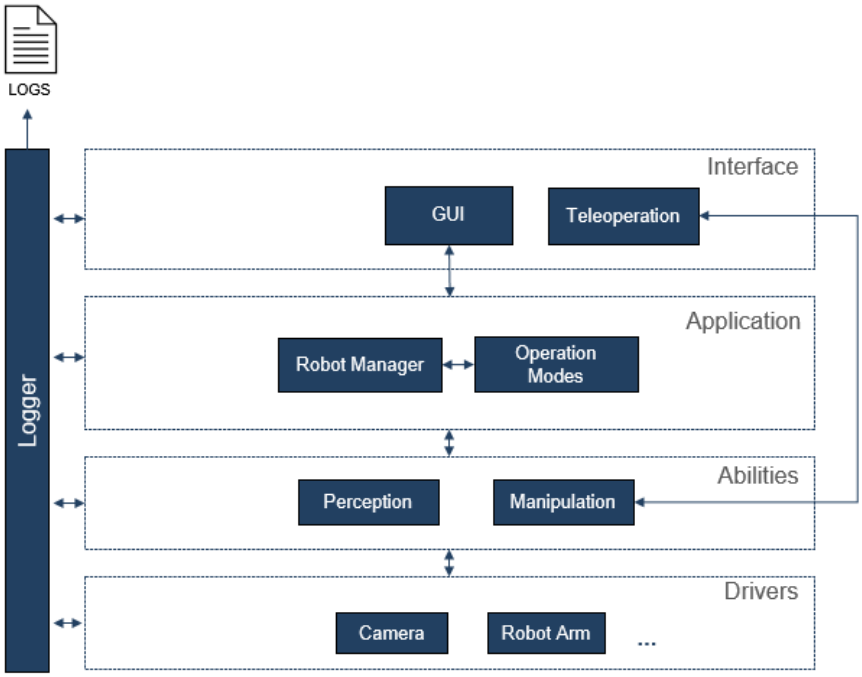


Figure 28. Four-layer software architecture.

The Robot Operating System (ROS) is used as the core framework for the development of the robotic platform software system. ROS is an open-source framework that offers a collection of tools, libraries, and conventions to simplify the creation of complex and robust robot behavior across a wide variety of robotic platforms. ROS is designed to be thin and language-independent, making it easy to implement in any modern programming language. It also has a built-in unit/integration test framework and is suitable for large runtime systems and development processes.

4.2 Drivers' implementation

4.2.1 KUKA KR60 driver

ROS Industrial is an open-source project that emerges with the aim of combining the multiple advanced capabilities provided by ROS with existing industrial technologies, allowing to provide robust, flexible and more complex solutions currently required by automation and manufacturing sectors.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

It is a BSD-licensed software that provides libraries, tools and drivers for industrial hardware. It aims to standardize device communication, enabling hardware-agnostic software development and ensuring interoperability between them through ROS. Many industrial manipulators are supported by ROS Industrial, including KUKA. Therefore, for the context of the OMICRON project, we have decided to use this driver.

This driver has two main components. On the one hand is the application that runs on the robot controller, written in KRL, KUKA's proprietary programming language. On the other hand, there is a ROS node, which establishes the robot's connection via UDP and updates the robot's status. This node is also in charge of sending commands to the robot. This driver operates in streaming mode, which means that information is sent and received continuously.

Some of the robot operations require manual control of the robot. This has required us to modify the driver to add a service to enable/disable ROS control.

In addition, the driver has also been modified to be able to manage the input and output signals through a ROS service, in order to control the different tools. The ROS message defined for this purpose is:

Table 8. ROS message definition to control the different tools

```
# Note: 'fun' is short for 'function' (ie: the function the service should perform).
int8 FUN_SET_DIGITAL_OUT = 1
int8 FUN_SET_ANALOG_OUT = 3
int8 FUN_SET_TOOL_VOLTAGE = 4

# valid values for 'state' when setting digital IO or flags
int8 STATE_OFF = 0
int8 STATE_ON = 1

# valid 'state' values when setting tool voltage
int8 STATE_TOOL_VOLTAGE_0V = 0
int8 STATE_TOOL_VOLTAGE_12V = 12
int8 STATE_TOOL_VOLTAGE_24V = 24

# request fields
int8 fun
kuka_eki_hw_interface/OutputData output
---
bool success
```

4.2.2 Camera's driver

ROS framework provides support for several camera drivers, that allow to connect and perform captures automatically, through standard ROS communication mechanisms (topics and services). Each device has its own API (encapsulated as a library) to access and control it, so it is necessary to have a driver per camera.

In the context of OMICRON, three types of cameras have been tested (previously mentioned in section 3.1.6): Realsense D435, POE OAK-D and Zed 2i. ROS drivers are available for each of these cameras.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

These drivers are realsense2_camera [1], depthai_ros_driver [2] and zed_wrapper [3], respectively. These drivers have been downloaded from the repositories available on Github and deployed for use.

4.3 Manipulation implementation

This section describes the implementation carried out in each of the sub-modules that compose the handling architecture, following the design described in D3.1.

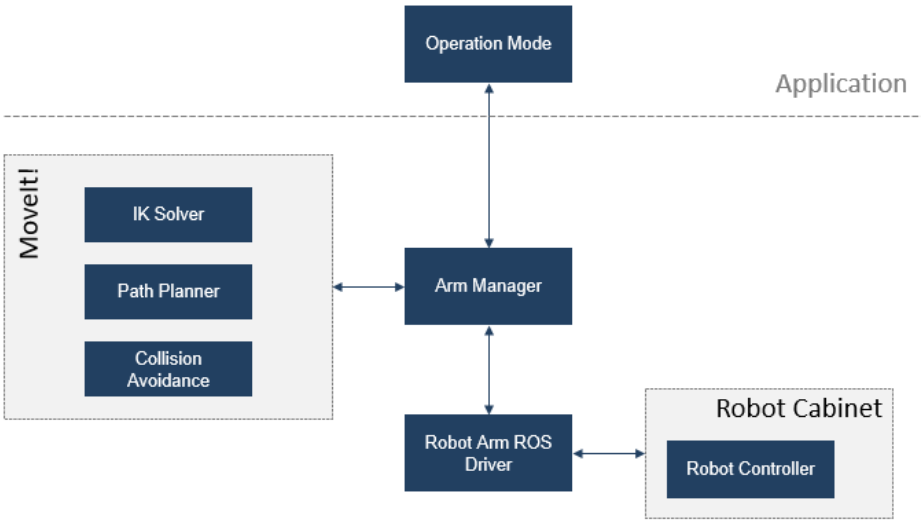


Figure 29 Manipulation software architecture diagram.

4.3.1 Arm manager

The arm_manager is a TEKNIKER's proprietary library which uses the Moveit! framework as a base. It is designed with the goal of providing an interface similar to the one offered by industrial robotics, incorporating the latest advances in motion planning. Industrial robotics is characterized by the robustness and precision it provides to automated solutions. However, it is not very flexible. The modularity and scalability of Moveit! design allows to test and validate different algorithms for trajectory generation and execution and choose the one that best suits the environment in each use case.

Therefore, in order to include all this added value to the manipulation operations, arm_manager has been integrated in OMICRON prototype, as a middleware between Moveit, whose API is quite complex, and the user level operation modes.

4.3.2 IK solver

An anthropomorphic robot, which is a set of links joined by axes, requires an inverse kinematics (IK) solver to determine the joint angles necessary to achieve a desired end-effector position in space. The IK solver is used to calculate the joint angles required to achieve a specific task, such as picking up an object or reaching for a target, based on the position and orientation of the end-effector.

By default, Moveit! uses KDL [4] (Kinematics and Dynamics Library) as default IK Solver, although it is possible to use any other algorithm whenever it is developed as pluginlib and uses Moveit!'s KinematicBase interface as base. In addition to KDL, two more solvers have been tested in the context of the project, specifically IK-Fast [5] and Trac-IK [6].

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

KDL is a numerical and model-agnostic solver. It is very versatile and flexible, being able to generate solutions with a wide variety of configurations. It is based on Newton's method, which does not offer good quality results in the presence of joint limits, a common condition in robotic arms. Moreover, it is an iterative method, which starting from an initial solution, applies a refinement through successive trials. The step size of each iteration is not configurable and assigned default value, returns often too much rotated solutions.

In order to obtain a better performance, it was decided to test IK-Fast. This is a robot model dependant solver, computing highly optimized solutions for a particular robot. This algorithm is much faster and efficient than generic numerical solvers such as KDL, but it is really complex to set up. The results obtained in the shopfloor tests were satisfactory, however, sometimes there were changes in the configuration of the robot, which can generate risky situations, since they lead to large rotations in some axes. Trying to avoid those configuration changes Track-IK was tested.

Trac-IK employs the Jacobian method to generate inverse kinematics solutions through a sequence of iterative computations that are based on the previous solutions, leading to a faster and more effective resolution than alternative iterative approaches. Internally it executes to IK implementations overcoming joint limits issue. In addition, Track-IK also provides secondary constraints of distance, manipulability and collision avoidance to ensure the best physically feasible solution.

Considering all the above, Track-Ik is the selected IK solver. However, depending on the future tests during operation modes validation, a change in the kinematics solver is not discarded.

4.3.3 Path planner

The path planner module contains the algorithms to find valid path for the robot to reach the target position. It is designed to work with different planner, being OMPL [7] the default option in Moveit!. This planner is a powerful collection of state-of-the-art sampling-based motion planning algorithm. Most of the times finds a solution, however, it does not be the best. Moreover, the trajectories generated are usually very long, an unfavourable aspect for industrial robots, since the low-level control is in the robot cabinet, there is no common to be real time communication and each point of the trajectory is considered an independent target, in which it accelerates and slows down. As a result, the robot moves jumping.

As an alternative STOMP [8] has been selected. It is an optimization-based motion planner, that generates smooth well behaved motion plans quickly, without being necessary any post-processing task. The path planning takes in consideration the current pose of the robot and several constraints, such as path length, joints position standard deviation, etc, are configurable. All those advantages make it the best candidate.

However, for the signal cleaning operation mode, the need to execute a set of predefined trajectories arises, some of them being circular. Looking for alternatives among planners in the state of the art, Pilz Industrial Motion Planner [9] was discovered. It provides a trajectory generator for standard robot movements such as PTP, LIN, and CIRC, very close interface to standard robotics. The results obtained have been similar to those of stomp and very satisfactory, so for now it has been decided to use Pilz Industrial Motion Planner as the main path planner, although it is not ruled out to change it, in case it is considered necessary in the future tests.

4.3.4 Collision avoidance

As described in section 3.1, the modular robotic platform prototype is composed by a robotic arm and a container, where all the tools and materials that are necessary for road maintenance operations are located. The arm is in charge of performing pick and place operations inside the container itself, all of them free of collisions.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

To ensure a collision free path planning FCL (Flexible Collision Library) [10] library has been deployed. It supports different object shapes such as geometric primitives, mesh files and octrees and detects any overlap between those type of shapes.

In the implementation a distinction has been made between static and dynamic scenes. Stationary scene refers to the container itself and fixed elements, while dynamic scene concerns any object that may suddenly appear in the environment.

The graphical representation of the container has been exported as a mesh file and is loaded at the beginning of each operation mode. From this moment on, all the trajectories generated by Moveit! are checked point by point against this model and only will be executed if there is no collision.

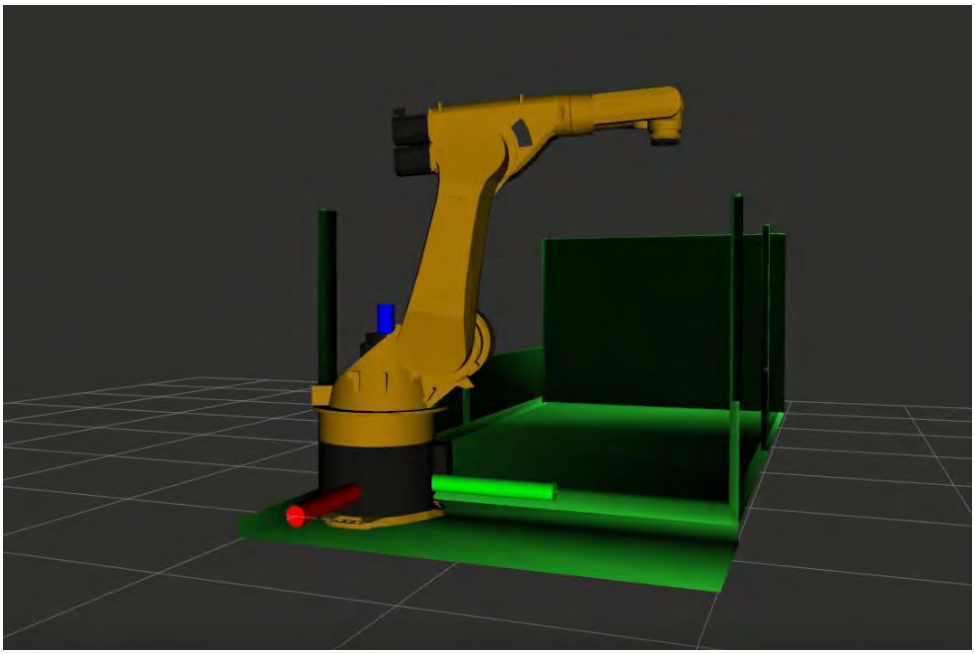


Figure 30: Collision path planning in static scenes

The dynamism of the scene is captured through an RGB-D camera, creating a 3D representation of the environment with the depth image using octomap, whose base unit is the octree. The scene is updated for any changes and in order to detect possible collisions that may arise while the robot is moving, a parallel monitoring system is running, which will stop the motion, re-plan and re-run the new trajectory.

4.4 Perception modules implementation

4.4.1 General considerations

For the implementation of the perception modules, the following have been taken into consideration:

- Use standard ROS messages whenever possible. The main advantage of using standard ROS messages is that they enable interoperability and reusability across different components in the robotic systems.
- Hardware agnostic implementation. The implemented perception modules do not use specific hardware libraries to get sensor data. Instead, they rely on standard ROS messages to get the input data. This means that the same message can be used to communicate with different sensor manufacturers, as long as they provide data in the same format as expected by the message.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

The implementation of the various perception modules follows the general design structure that can be seen in Figure 31Figure 31.

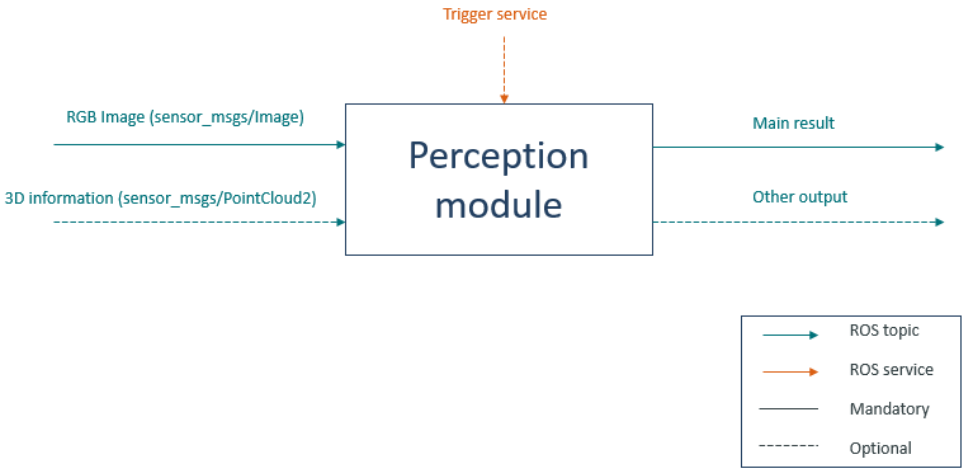


Figure 31. General design structure of the perception modules

All the modules take as input an RGB image of the sensor. The modules subscribe to a ROS topic to get this image, using the standard ROS message type "*sensor_msgs/Image*". This message type allows for the transmission of image data, including height, width, encoding format, and the actual pixel values of the image.

In case the module needs not only 2D image information but also depth information, the module subscribes to another ROS Topic to get this information as a Point Cloud, using the standard ROS message type "*sensor_msgs/PointCloud2*".

When the module takes as input both the RGB Image and the Point Cloud, we need to ensure that the data is synchronized in time so that the correct corresponding data is used. For that, the "*TimeSynchronizer*" class in the ROS "*message_filters*" package is used to synchronize messages from different topics that are timestamped with "*ros::Time*".

All modules provide at least one output with the main result. The module may offer some other output (e.g., for visualisation purposes).

By default, a perception module will perform the detection for each sensor data input it receives from the ROS topics. However, for some operations it is more convenient to be able to trigger the detection at specific times during the operation, instead of being continuously detecting the objects. For those cases, a module can also provide a ROS service to trigger the detection.

To conclude, all perception detectors has been Dockerized in order to be portable and easily executable on any machine.

4.4.2 Cones detector

The cones detector is a perception module for traffic cones detection on RGB images. The purpose of this module is to be used by the truck driver as an assistance system while performing the cone collection operation.

As a solution for the cone's detection, an existing model [11] based on convolutional neural networks has been integrated to meet the needs of the perception module. It uses a Yolov5s model as a base, which has been trained on COCO dataset. Starting from this open-source model, the transfer learning approach has been applied, training the model with images and labels of cones. As a result, the default weights of yolov5s have been adjusted to the cone problem.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

YoloV5s is a cutting-edge object detection model created by Ultralytics. It represents an enhanced version of the YOLO (You Only Look Once) family of object detection models. The "s" in YOLOv5s stands for "small", which means that this model is designed to be smaller and faster, and it can be easily deployed on resource-limited devices like smartphones or embedded systems. Despite its compact size, YOLOv5s is capable of achieving high accuracy and is widely used for various computer vision tasks, such as object detection, instance segmentation, and more.

The dataset used for transfer learning consists of 303 images. For labelling Roboflow [12] website was used, which in addition to the labelling service also provides pre-processing and data augmentation functionalities. In order to increase the number of reduced images available in the initial dataset, augmentation techniques have been applied to obtain a total of 700 images. It is still a small dataset, but the results provided are accurate and with high score.

Figure 32: Cones detector result

The model in reply to an inference returns the bounding box of the cone together with the colour of itself, although the latter is ignored in the context of this project.

The cone detection module has been implemented as a behaviour tree node, in which every time an image of "*sensor_imgs::Image*" type is received via ROS topic, an inference is performed. Only the detection that overcomes a predefined score will be accepted as valid, and the following information will be published for each one of them:

- The bounding box of the cone as well as the original image on which the detection has been performed. A message of type "*vision_msgs::Detection2D*" will be used for this purpose.
- The analysed image with the bounding box overlapped, as well as the theoretical margin of the location of the cones, just for visualization purposes to help the driver to correct in case of deviation.

4.4.3 Signals detector

The signal detector is the perception module in charge of signal detection on RGB-D images for cleaning purposes. Each signal has its own predefined cleaning path, depending on its shape, so the perception module, in addition to locate the signal in the scene, must identify the signal type, in order to determine its shape.

Therefore, for this module a real time detection and classification system is required, and as in cones detector a solution based on convolutional neural networks has been chosen.

A thorough web search was performed in order to find the best model to fit the needs of the detection module. The main pre-selection criteria used were the number of stars and forks on Git, repository

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

activity as well as the latest updates. However, it should be noted that it was not easy to find models that were close to the present day.

Four models were pre-selected and tested: a R-FCN object detection system combined with Resnet V1 101 feature extractor [13], a retrained Mask R-CNN model for traffic signs using Detectron Framework [14] [15], a real time detection and classification model base on YoloV5 [16] and a model based on Yolov3 and Yolov4, with the characteristic of detecting a main class, traffic sings, and classifying on a second level using a customized convolutional neural network, with subclasses such as, prohibition, mandatory, etc. or speed limit, left turn, stop, etc [17].

The best results were obtained with a Yolov5s based model. There is no information on how the model was created, nor on the dataset used for transfer learning or other approaches that may have been used, such as data augmentation, only the final model.

The ROS node has been developed also as Behaviour Three node, and following the design described in section 4.4.1, the signal detection functionality is be available through a ROS service, which message is "*omicron_robotframework_msgs::DetectSignals*". As input data, the node receives RBD data as "*sensor_msgs::Image*" message, and 3D info as "*sensor_msgs::Pointcloud*", both synchronized.

When an analysis request is received, the node retrieves the last available RGB-D information, and an inference is performed just with 2D image as input. If a signal is detected, the model returns the bounding box associated to the signal and the class id, i.e., the type. Then, a post-processing is required, where the point cloud that corresponds to the area of the bounding box is analysed. With this set of points a plane is calculated and with their normals and direction vectors it is determined where the signal regarding to the camera frame. Finally, as the last post-processing task, the shape of the signal is determined according to its type. Both results are encapsulated in the service response message and sent as an output. In addition, the processed image is also published overlapped with the result, for visual purposes only.

[13]

4.4.4 Road cracks detector

The road cracks detector is a perception module for road cracks detection on RGB images. The purpose of this module is to locate the crack as well as its shape, since this data will be the input of the module in charge of generating a trajectory according to that shape for pavement repairing.

As in the preview's detectors, a convolutional neural networks solution has been deployed. Several architectures have been analysed, four exactly. The first one was an approach based on UNet network using Resnet 101 and VGG16 architectures with transfer learning [18]. The idea behind this combination is that VGG16 and Resnet 101 neural network, can extract useful and abstract features from the image that can help improve the accuracy of the UNet segmentation for better characteristic extraction.

The second option was a trainable deep convolutional neural network for crack detections by learning high-level features for representing cracks. It has been built on encoder-decoder architecture of SegNet and implemented pairwise fusion of the convolutional features generated at the same scale in both the encoder and decoder networks. In total four dataset has been used, one for training and the rest for evaluation test. According to the experimental results, DeepCrack achieves an F-measure of over 0.87 on average across the three challenging datasets, demonstrating superior performance compared to the current state-of-the-art methods.

The third approach is a solution based on an enhanced UNet network, specifically GCUNet [19]. It uses graph convolution layers to capture contextual and topological features of images, which allows to improve the UNet's own accurate and contextual segmentation of the images.

The last approximation is a model for low resolution images using joint learning with super resolution (CSSR) [20]. The solution is based on two approaches, on one hand it uses a machine learning approach

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

in which a complete model is trained to optimize super resolution crack detections task. On the other hand, Boundary Combo loss function is used to optimize the segmentation of global and local structures of cracks simultaneously.

The best result has been achieved with a DeepCrack called solution and this solution has been integrated in a ROS node behaviour tree. Crack detection functionality is offered through a ROS service. When a request is received the last captured image is recovered (the node is subscribed to an image topic that is publishing camera info periodically), an inference is executed and detected crack mask is returned.

[14]

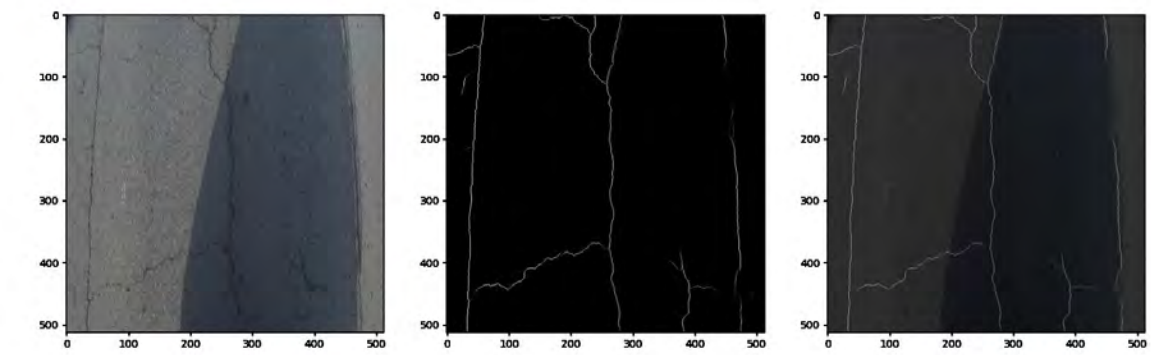


Figure 33: Crack detector result

4.5 Implementation of applications for specific operations

An application has been implemented for each of the operations that the modular robotic platform can perform. Each of these applications implements the set of steps of the procedure for the corresponding road maintenance operation.

4.5.1 General considerations

4.5.1.1 Plugin architecture

These applications, also named operation modes, have been implemented as modular pieces of code inside the general software architecture of the system, according to the design proposed in deliverable D3.1. To achieve this modularity, the ROS pluginlib library has been used. This library provides a way to dynamically load and unload plugins at runtime. This means that new future applications can be added to the modular robotic system without requiring the main software system to be recompiled.

A module named Robot Manager is in charge of managing the initialization and termination of operating modes specified in a configuration file. Its API contains methods for initialising and stopping the loaded modes of operation. The following figure shows the architecture of the implementation:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

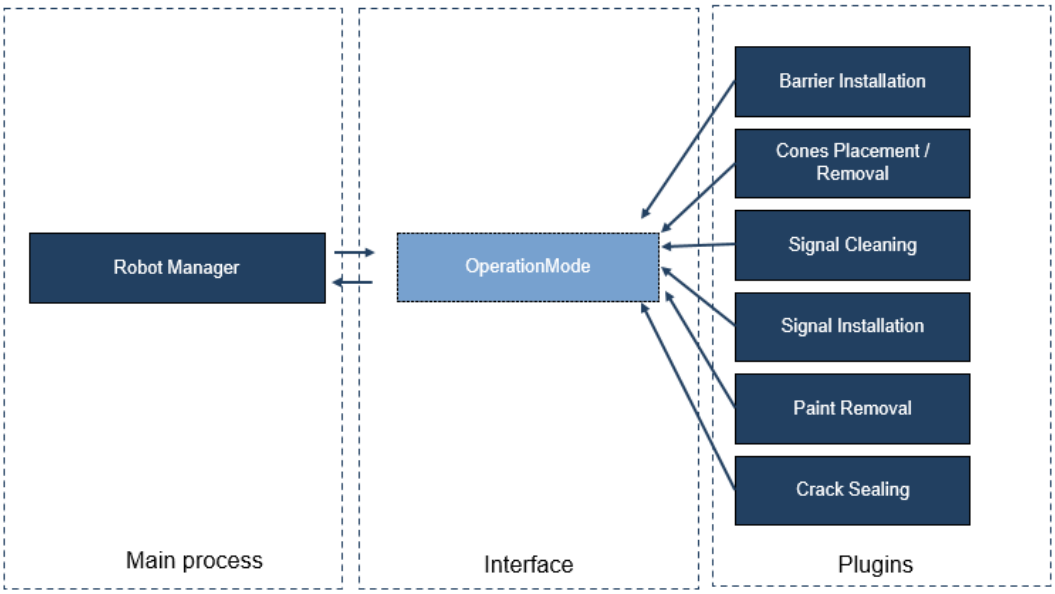


Figure 34. Plugin-based architecture for the implementation of applications for specific operations

4.5.1.2 Behavior trees

The applications have been programmed using the C++ programming language and Behavior Trees. Behavior trees are a type of control architecture used in robotics and artificial intelligence to create complex and intelligent behaviors in robots, game characters, and other autonomous agents. This kind of tree is a hierarchical structure consisting of nodes that represent specific actions. These nodes are connected by edges, which define the order in which the actions should be executed.

Figure 35 shows an example of a behavioural tree. In this example, the rectangle-shaped nodes represent an action node, which performs the action and then returns either success or failure depending on whether the action was successful or not. Ellipse-shaped nodes represent a condition node, that checks a condition and returns success or failure based on the result. Nodes with a question mark are “fallback” nodes, a type of node that executes its child nodes in order until one of them succeeds, at which point it stops executing and returns success. They are used to check if a condition is met, and in case it is not met, to execute a set of actions that as post-condition meets that condition. Nodes with an arrow represent a “sequence” node. They are a type of node that executes its child nodes in order until one of them fails, at which point it stops executing and returns failure.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N°955269

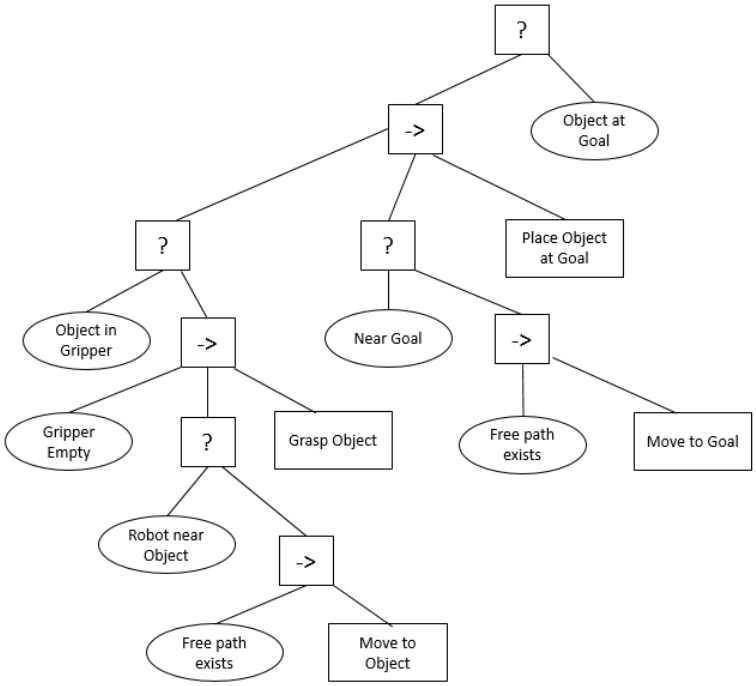


Figure 35. Example of a Behavior Tree to perform a Pick & Place operation.

Behavior trees are a powerful tool for implementing a set of steps of a procedure, providing the following main advantages:

1. Modularity. Behavior trees provide a modular way of organizing and reusing actions. Each node represents a specific action, and the tree structure allows for actions to be composed and combined in different ways to achieve more complex behaviors.
2. Flexibility. Behavior trees provide a flexible way of controlling the behavior of autonomous agents. The structure of the tree can be easily modified, and new actions can be added or removed without affecting the rest of the tree.
3. Transparency. Behavior trees provide a clear and understandable way of visualizing and debugging the behavior of autonomous agents. The structure of the tree makes it easy to see how different behaviors are connected and how they interact with each other.

In the context of OMICRON, we decided to implement the main logic of the applications for the various operations using Behavior Trees, since the operations are made up of a series of tasks that must be carried out in a certain sequence. The main reasoning behind this implementation decision is that (1) some tasks are common across multiple operations, so they can be reused once implemented as tree nodes (e.g., move the robot to a given pose, or activate/deactivate tool) (2) the sequence of tasks in an operation can be modified very easily by editing the tree (which makes it much easier to adjust the application during testing) (3) you can view and track the tasks that are running in the application using a GUI (very useful also in testing phases).

In fact, the application implementations for each of the maintenance operations described in the following sections are a first version, which may be subject to modifications during their development in the WP4 tasks. It is expected that the behaviour tree based implementation will facilitate these modifications.

BehaviorTree.CPP library has been used for implementing behavior trees in C++. This library is designed to be flexible and customizable, allowing developers to create complex behavior trees that can handle a variety of scenarios.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

BehaviorTree.CPP library already provides a graphical user interface (GUI) on top of it for creating behavior trees and monitoring the execution of the behavior tree in real-time. This visualisation is particularly useful during the application testing process.

4.5.2 Application for installing safety barriers

4.5.2.1 Operation steps

The steps that were decided for the barrier installation operation are the following:

1. The truck driver stops the truck in the proper position, guided by the view of a side camera.
2. Two operators exit the truck.
3. The operator presses a button to start the system.
4. The robot picks a barrier.
    a. The robot first moves to a fixed pose next to the barrier.
    b. The robot performs a linear motion to contact the barrier.
    c. The magnet gripper is activated.
    d. The robot performs a linear motion to lift the barrier.
    e. The gripper link clamps are closed to fix the barrier.
5. The robot moves bringing the safety barrier next to the pillars.
6. A notification is sent to the operator that manual guidance is the next step.
7. The operator moves the robot manually to the proper place and screws the barrier.
8. Operator moves away from the robot working area.
9. Operator presses a button and robot moves to home pose.

4.5.2.2 Implementation

The implementation of the application has several parameters to make it configurable. The following table shows the parameters of the application.

Table 9. Barrier installation application parameters.

Parameter	Description
num_of_barriers	The number of barriers stacked.
home_pose	Robot home pose.
external_frame	Pos of an external frame with respect to robot base.
barriers_pose	The pose of the stack of barriers, with respect to external_frame
offset	Distance between two stacked barriers.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

The application allows you to define an external frame and indicate the pose of the barrier stack with respect to said frame. This allows for easy configuration so that the unloading of safety barriers can be done from either the left or right side of the truck.

For this application the following behaviour tree action nodes have been implemented:

- LoadSafetyBarriersOperationParameters
- MoveRobot
- EnableGripperMagnet
- EnableGripperLinkClamps
- NotifyTask
- IsTaskCompleted

Table 10. LoadSafetyBarriersOperationParameters BT action node.

LoadSafetyBarriersOperationParameters	
Type of node	Synchronous action node
What it does	(1) Gets the operation parameters from ROS parameter server (2) computes barrier picking poses

Table 11. MoveRobot BT action node.

MoveRobot	
Type of node	Asynchronous action node
What it does	(1) Plans and executes the movement of the robot to place its end-effector in the given final pose.

Table 12. EnableGripperMagnet BT action node.

EnableGripperMagnet	
Type of node	Synchronous action node
What it does	(1) Enables/disables the gripper magnet.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 13. EnableGripperLinkClamps BT action node.

EnableGripperLinkClamps	
Type of node	Synchronous action node
What it does	(1) Enables/disables the gripper link clamps.

Table 14. NotifyTask BT action node.

NotifyTask	
Type of node	Synchronous action node
What it does	(1) Notifies the given instruction to AR system.

Table 15. IsTaskCompleted BT condition node.

IsTaskCompleted	
Type of node	Condition node
What it does	(1) Checks whether the AR instruction was finished or not.

The behavior tree that has been defined for the operation of installing safety barriers is shown in Annex I.

4.5.3 Application for placing & collecting cones

4.5.3.1 Operation steps

The steps that were decided for the cone placement operation are the following:

1. The driver of the truck presses a button to start the operation.
2. The driver starts driving forward.
3. Repeat N times:
    a. The robot moves to the position to pick the next cone.
    b. The robot picks the cone.
    c. The robot moves to drop position.
    d. The robot waits until a given distance is covered by the truck.
    e. The robot releases the cone.

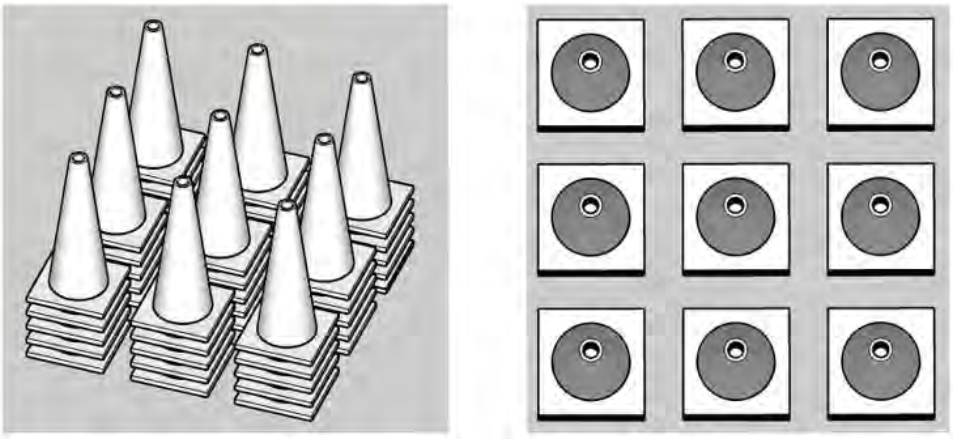
The steps that were decided for the cone collection operation are the following:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

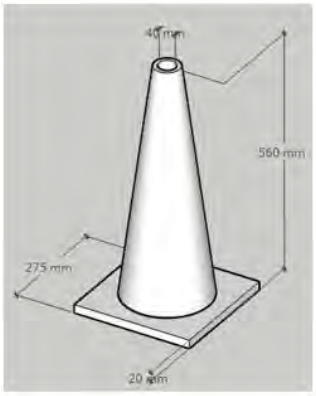
1. The driver of the truck releases the cone retaining tool.
2. The driver of the truck presses a button to start the operation.
3. The driver starts driving backwards.
4. The robot moves to the position to be prepared to pick up the cone.
5. The cone retaining tool has a sensor to detect the cone. When the cone is detected, the robot picks up the cone.
6. The robot moves to the position to release the cone.
7. The robot releases the cone.

4.5.3.2 Implementation

The implementation of the application has been made to be highly configurable, as described below. The cones will be arranged in the container, within reach of the robot, according to the following layout:



The cones taken as reference are according to the Italian regulation, since the final validation will be carried out in Italy. The dimensions of the cones are shown in the figure below:



In order to generalise the layout, a series of parameterizable values have been defined, as shown in the following table:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 16. Cones layout parameters.

Parameter	Description
N	Number of rows of the cone's layout
M	Number of columns of the cone's layout
X	Max number of cones in each group of the cone's layout
H_c	Height of a cone
d	Distance from the ground to calculate the robot approaching position in Z
d_{offset}	Distance between two stacked cones
P_{i_0}	Ground level position of i^{th} cones group
P_{i_d}	Approaching position of i^{th} cones group

For ease of understanding, Figure 36Figure 36 shows the parameters of Table 16Table 16.

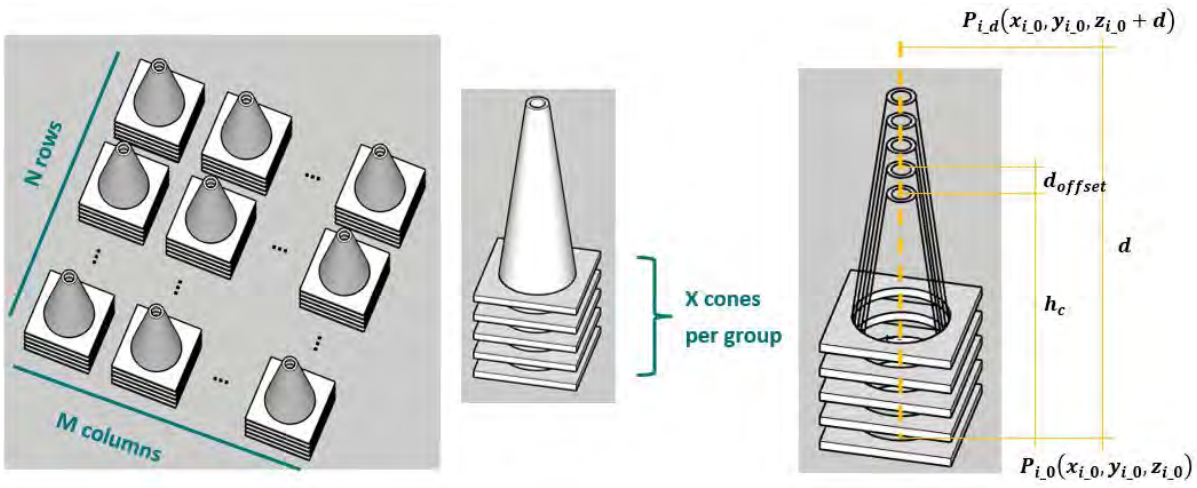


Figure 36. Cones layout parameters.

In addition to the layout-related parameters, the implementation also has parameters for defining the robot's pick-up and drop poses. In this way, it is easy to configure the system to be able to pick up the cones from both the right and left side of the truck.

For this application the following behaviour tree action nodes have been implemented:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

- LoadOperationParameters
- GetPoses
- MoveRobot
- EnableGripper
- IsDistanceCovered
- ResetDistance
- UpdateParameters

Table 17. LoadOperationParameters BT action node.

LoadOperationParameters	
Type of node	Synchronous action node
What it does	(1) Gets the operation parameters from ROS parameter server (2) computes approach poses for each cones group (3) computes poses for each index (5) computes available cones indexes (6) computes first cone index (7) computes first available index

Table 18. GetPoses BT action node.

GetPoses	
Type of node	Synchronous action node
What it does	(1) Sets the current index pose and group approach pose (to be used in further steps for the robot motion)

Table 19. MoveRobot BT action node.

MoveRobot	
Type of node	Asynchronous action node
What it does	(1) Plans and executes the movement of the robot to place its end-effector in the given final pose)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 20. EnableGripper BT action node.

EnableGripper	
Type of node	Synchronous action node
What it does	(1) Enables/disables the gripper by calling a ROS service (to hold or release the cone)

Table 21. IsDistanceCovered BT condition node.

IsDistanceCovered	
Type of node	Condition node
What it does	(1) Checks whether the truck has traveled the given distance, by subscribing to a ROS topic in which the distance measurement sensor provides data.

Table 22. ResetDistance BT action node.

ResetDistance	
Type of node	Synchronous action node
What it does	(1) Resets the traveled distance by calling a ROS service.

Table 23. UpdateParameters BT action node.

UpdateParameters	
Type of node	Synchronous action node
What it does	(1) Identifies the next available index (a cone or free space, depending on whether the mode is place or collect) (2) Updates the available_cones vector (3) increases by one the number of executions

The behavior trees that have been defined for the operations of placing and collecting cones are shown in Annex I.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

4.5.4 Application for cleaning road signals

4.5.4.1 Operation steps

The steps that were decided for the signal cleaning operation are the following:

1. The driver of the truck stops the truck in the proper position.
2. The operator moves the robot, positioning it close to the signal, in such a way that the eye-in-hand camera of the robot sees the signal.
3. The operator presses a button to begin the signal identification.
4. The system identifies the signal.
5. The robot moves to the centre pose of the signal.
6. The robot activates the spraying and executes a trajectory based on its shape.
7. The robot moves to its original position.

4.5.4.2 Implementation

The parameters available to the implemented application are shown in Table 24.

Table 24. Signal cleaning application parameters.

Parameter	Description
home_pose_joints	Robot home pose with respect to robot base (in joints space)
teleop_init_pose_joints	Robot initial pose to start teleoperation with respect to robot base (in joints space)
tool_dist_to_signal	Distance at which to position the robot tool relative to the centre of the signal
camera_frame	Camera frame transformation with respect to robot flange
tool_frame	Tool frame transformation with respect to robot flange
robot_paths	Predefined robot paths to be executed depending on the type of signal

The implemented application has parameters to define a series of predefined trajectories depending on the type of shape of the signal. The following shapes are considered: round, triangle, inverted triangle, square and octagon. Another parameter also defines the distance at which the robotic tool should be positioned with respect to the centre of the signal.

For this application the following behaviour tree action nodes have been implemented:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

- LoadCleaningOperationParameters
- MoveRobot
- EnableTeleoperation
- IsTeleoperationFinished
- DetectSignals
- EnableSpraying
- ExecuteTrajectory

Table 25. LoadCleaningOperationParameters BT action node.

LoadCleaningOperationParameters	
Type of node	Synchronous action node
What it does	(1) Gets the operation parameters from ROS parameter server

Table 26. MoveRobot BT action node.

MoveRobot	
Type of node	Asynchronous action node
What it does	(1) Plans and executes the movement of the robot to place its end-effector in the given final pose.

Table 27. EnableTeleoperation BT action node.

EnableTeleoperation	
Type of node	Synchronous action node
What it does	(1) Enables/disables the option to teleoperate the robot.

Table 28. IsTeleoperationFinished BT condition node.

IsTeleoperationFinished	
Type of node	Condition node
What it does	(1) Checks whether the teleoperation has finished or not.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 29. DetectSignals BT action node.

DetectSignals	
Type of node	Synchronous action node
What it does	(1) Calls the signal perception module (2) computes signal center pose with respect to the robot tool.

Table 30. EnableSpraying BT action node.

EnableSpraying	
Type of node	Synchronous action node
What it does	(1) Enables/disables the spraying system.

Table 31. ExecuteTrajectory BT action node.

ExecuteTrajectory	
Type of node	Asynchronous action node
What it does	(1) Executes a given trajectory

The behavior tree that has been defined for the operation of cleaning road signals is shown in Annex I.

4.5.5 Application for installing road signals

4.5.5.1 Operation steps

The steps involved in the operation for removing horizontal marks are as follows:

1. The operator on the container presses a button to unload a signal.
2. The robot moves to a prefixed position on the container.
3. The robot enters to “sensitive mode”.
4. The operator on the container moves manually the robot and places the tool on the signal. After that, activates the tool. He then moves the robot manually moves the robot to a predefined position to start the automatic motion.
5. The operator presses a button to start the robot automatic movement.
6. Ther robot moves to a prefixed position on the road.
7. The robot enters to “sensitive mode”.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N°955269

8. The operator on the road moves the signal to the road and releases the signal from the tool. The operator moves away from the signal.
9. The operator on the road presses a button to start the robot automatic movement.
10. The robot moves to a home position.
11. The process is repeated.

4.5.5.2 Implementation

The parameters available to the implemented application are shown in.

Table 32. Signalling application parameters.

Parameter	Description
home_pose_joints	Robot home pose with respect to robot base (in joints space)
pose_container_joints	Robot container pose with respect to robot base (in joints space)
pose_road_joints	Robot road pose with respect to robot base (in joints space)

For this application the following behaviour tree action nodes have been implemented:

- LoadSignallingOperationParameters
- EnterSensitiveMode
- WaitForSignal
- MoveRobot

Table 33. LoadSignallingOperationParameters BT action node.

LoadSignallingOperationParameters	
Type of node	Synchronous action node
What it does	(1) Gets the operation parameters from ROS parameter server

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 34. EnterSensitiveMode BT action node.

EnterSensitiveMode	
Type of node	Synchronous action node
What it does	(1) Activates de sensitive mode of the robot

Table 35. WaitForSignal BT action node.

WaitForSignal	
Type of node	Asynchronous action node
What it does	(1) Waits until a given ROS service is called with a specific ID. This is used to wait until the operator sends a signal from the UI (e.g. by pressing a button).

Table 36. MoveRobot BT action node.

MoveRobot	
Type of node	Asynchronous action node
What it does	(1) Plans and executes the movement of the robot to place its end-effector in the given final pose.

The behavior tree that has been defined for the operation of installing road signals is shown in Annex I.

4.5.6 Application for removing horizontal marks

4.5.6.1 Operation steps

The steps involved in the operation for removing horizontal marks are as follows:

1. The driver of the truck stops the truck in the proper position.
2. The operator. places the robot in the proper horizontal position using the tablet with the feed from an eye-in-hand camera.
3. The operator presses a button to start the identification of a lane mark.
4. The system shows the detected lane mark to the operator.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

5. The operator may modify the detection, and finally confirms.
6. Detected lane mark is converted from pixel points to world coordinates.
7. The robot executes the trajectories activating the laser. The laser is deactivated once finished.
8. The robot moves to a home position.
9. The operator drives to the next position.

4.5.6.2 Implementation

The parameters available to the implemented application are shown in.

Table 37. Removing horizontal marks operation parameters.

Parameter	Description
home_pose_joints	Robot home pose with respect to robot base (in joints space)
camera_frame	Camera frame transformation with respect to robot base reference frame
tool_frame	Tool frame transformation with respect to robot flange
laser_init_pose_joints	Robot pose before starting the removing operation (in joints space)
height_offset	Distance offset in height of the laser tool with respect to the road

For this application the following behaviour tree action nodes have been implemented:

- LoadLaserOperationParameters
- EnableTeleoperation
- WaitTeleoperationFinished
- WaitForSignal
- DetectlaneMark
- GetLaneMarkResult
- ExecuteLaserWithRobot
- MoveRobot

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 38. LoadLaserOperationParameters BT action node.

LoadLaserOperationParameters	
Type of node	Synchronous action node
What it does	(1) Gets the operation parameters from ROS parameter server

Table 39. EnableTeleoperation BT action node.

EnableTeleoperation	
Type of node	Synchronous action node
What it does	(1) Calls a ROS service to enable the teleoperation.

Table 40. WaitTeleoperationFinished BT action node.

WaitTeleoperationFinished	
Type of node	Asynchronous action node
What it does	(1) Waits until a given ROS service is called which indicates that teleoperation has finished. This is used to wait until operator indicates that teleoperation finished.

Table 41. WaitForSignal BT action node.

WaitForSignal	
Type of node	Asynchronous action node
What it does	(1) Waits until a given ROS service is called with a specific ID. This is used to wait until the operator sends a signal from the UI (e.g. by pressing a button).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 42. DetectLaneMark BT action node.

DetectLaneMark	
Type of node	Synchronous action node
What it does	(1) Calls the lane mark detection perception module (2) calls the ROS service to set the detection.

Table 43. GetLanemarkResult BT action node.

GetLaneMarkResult	
Type of node	Synchronous action node
What it does	(1) Calls the ROS service to get the latest detected lanemark result

Table 44. ExecuteLaserWithRobot BT action node.

ExecuteLaserWithRobot	
Type of node	Asynchronous action node
What it does	(1) Transforms detected lanemark points to real world coordinates (2) Executes trajectories of the robot by activating the laser when starting, and deactivating it when finished.

Table 45. MoveRobot BT action node.

MoveRobot	
Type of node	Asynchronous action node
What it does	(1) Plans and executes the movement of the robot to place its end-effector in the given final pose.

The behavior tree that has been defined for the operation of removing of horizontal marks is shown in Annex I.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

4.5.7 Application for sealing pavement cracks

4.5.7.1 Operation steps

The steps involved in the crack sealing operation are as follows:

1. The driver of the truck stops the truck in the proper position.
2. The operator presses a button to start the cracks identification using the rear top-down camera.
3. The system shows the detected cracks to the operator.
4. The operator may modify the detection, and finally confirms.
5. Detected cracks are converted from pixel points to world coordinates.
6. The robot executes the trajectories activating / deactivating the tools during the execution. (The robot may be moved horizontally to the possible three positions during this step, to reach the cracks).
7. The robot moves to a home position.
8. The truck moves 1,5 meters forward and the process is repeated.

4.5.7.2 Implementation

The parameters available to the implemented application are shown in Table 24.

Table 46. Crack sealing application parameters.

Parameter	Description
home_pose_joints	Robot home pose with respect to robot base (in joints space)
rear_camera_frame	Camera frame transformation with respect to world reference frame
tool_frame	Tool frame transformation with respect to robot flange
p1_frame	P1 position transformation with respect to world reference frame
P2_frame	P1 position transformation with respect to world reference frame
P3_frame	P3 position transformation with respect to world reference frame
sealing_init_pose	Robot pose before starting the sealing (in joints space)
blowing_height_offset	Distance offset in height of the tool with respect to the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Parameter	Description
	road for the blowing operation.
zone_limits	Array containing the limit values defining the robot working zones (P1, P2 and P3).

For this application the following behaviour tree action nodes have been implemented:

- LoadSealingOperationParameters.
- WaitForSignal.
- DetectCracks.
- GetResult.
- ExecuteSealingWithRobot.
- MoveRobot.

Table 47. LoadSealingOperationParameters BT action node.

LoadSealingOperationParameters	
Type of node	Synchronous action node
What it does	(1) Gets the operation parameters from ROS parameter server

Table 48. WaitForSignal BT action node.

WaitForSignal	
Type of node	Asynchronous action node
What it does	(1) Waits until a given ROS service is called with a specific ID. This is used to wait until the operator sends a signal from the UI (e.g. by pressing a button).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Table 49. DetectCracks BT action node.

DetectCracks	
Type of node	Synchronous action node
What it does	(1) Calls the cracks perception module (2) calls the ROS service to set the detection.

Table 50. GetResult BT action node.

GetResult	
Type of node	Synchronous action node
What it does	(1) Calls the ROS service to get the latest detected cracks results

Table 51. ExecuteSealingWithRobot BT action node.

ExecuteSealingWithRobot	
Type of node	Asynchronous action node
What it does	(1) Transforms detected cracks points to real world coordinates (2) Determines the working areas needed for the robot to seal the cracks (2) Executes trajectories of the robot by activating the tools, and moving the robot horizontally if necessary.

Table 52. MoveRobot BT action node.

MoveRobot	
Type of node	Asynchronous action node
What it does	(1) Plans and executes the movement of the robot to place its end-effector in the given final pose.

The behaviour tree that has been defined for the operation of sealing cracks is shown in Annex I.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

5 Integration of elements in the system

This section describes the steps that have been taken to integrate the elements into the system.

5.1 Robot integration into the container platform

A clamping adapter plate was produced for the KUKA KR60 robot to secure it in place, in accordance with the manufacturer's specifications. The clamping plate that was produced is shown in the figure below.

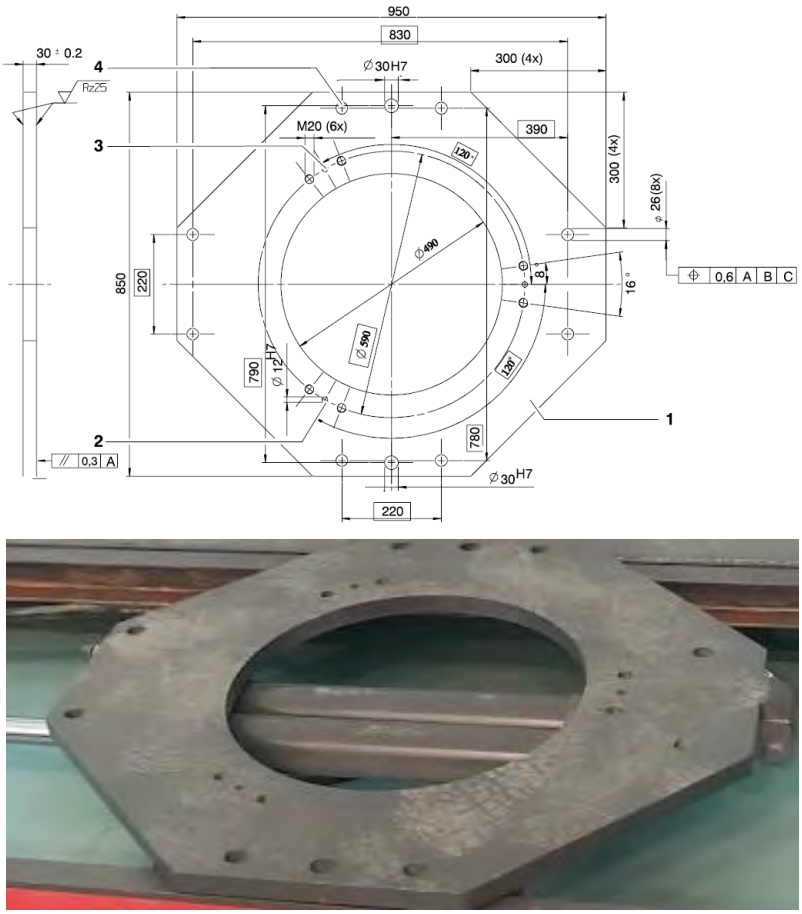


Figure 37. Clamping adapter plate for KUKA KR60 robot.

The robot has screw holes in its base, which are used to secure it to the clamping adapter plate.

After the container was manufactured, stress tests were conducted at the manufacturer's own facilities before being transported to TEKNIKER's facilities. These tests involved placing a weight of 1000 kg on the adapter plate where the robot is fastened, thereby simulating a weight that is somewhat higher than the robot's own weight.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Figure 38. Stress weight tests at container manufacturer's facilities.

To secure the robot control cabinet, threaded holes have been drilled in the floor of the container, allowing the cabinet to be securely attached to the container.

Figure 39. The robot control cabinet fixed to the container

5.2 Sealing machine integration into the container platform

The base of the PAVASAL sealing machine was modified by adding adapters, enabling it to be affixed to the floor of the container (already explained in Section 3.1.8). As with the robot control cabinet, threaded holes were drilled into the floor of the container to facilitate the securing of the sealing machine.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

Figure 40. Sealing machine secured to the container.

This attachment was validated at PAVASAL's facilities prior to the container being transported to TEKNIKER's facilities.

5.3 General control cabinet

The modular robotic platform has a general control cabinet in charge of managing the following elements:

- Hydraulic unit. Hydraulic unit that feeds two movements of the platform on which the robot is mounted: horizontal movement (left/right) and vertical movement (up/down). The unit consists of 2 solenoid valves, each with two solenoid coils. One of the solenoid valves will control the raising and lowering of the platform, from 2 relays that will be activated by the portable pushbutton panel.
  The other solenoid valve, which will manage the horizontal movement of the platform, will be managed from 2 digital outputs of the robot cabinet.
- Jay Electronique radio remote control. Wireless control unit that has a receiver to be mounted on the control cabinet. The control unit has an emergency button, 4 function buttons, and a reset button. Each function button activates a relay on the receiver, pressing the emergency button opens 2 guided relays on the receiver, and the reset button activates another relay.
- KUKA KR60 robot. The robot will use various tools. At present, the tools to be used on the robot flange are managed from the robot's own wired outputs. Two 24V outputs from the robot's output board manage the left/right movement of the platform on which the robot is mounted. These outputs act on the coils of a solenoid valve of the hydraulic unit.
- Pressure washer. The pressure washer is a commercial system, which needs to be powered at 220VAC.
- Voltage generator. The system also incorporates a generator, which generates a three-phase voltage of 400V to power the robot and the rest of the elements. A 220VAC transformer and 24V power supply feeds the control panel.
- Sealing machine. The sealing machine has its own control cabinet, with which it communicates from the general control cabinet.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

5.4 Safety system

The safety system is comprised of a wireless emergency alarm provided by Jay Electronique, as well as a laser scanner. The emergency stop button has the capability to shut down the following systems:

- The Kuka KR60 robot via its emergency circuit.
- The power supply to the hydraulic group's valves, consisting of 4 coils: left/right and up/down.
- The power supply to the pressure washer.
- The power supply to the sealing machine push buttons.

The scanner is responsible for stopping the following systems:

- The Kuka robot via its door circuit. This allows for manual movement of the robot, even if the scanner is being invaded by the person operating the robot.
- The power supply to the valves, preventing the platform from being raised or lowered, as well as the robot's ability to move from left to right.

The following figure shows the block diagram of the safety interconnections.

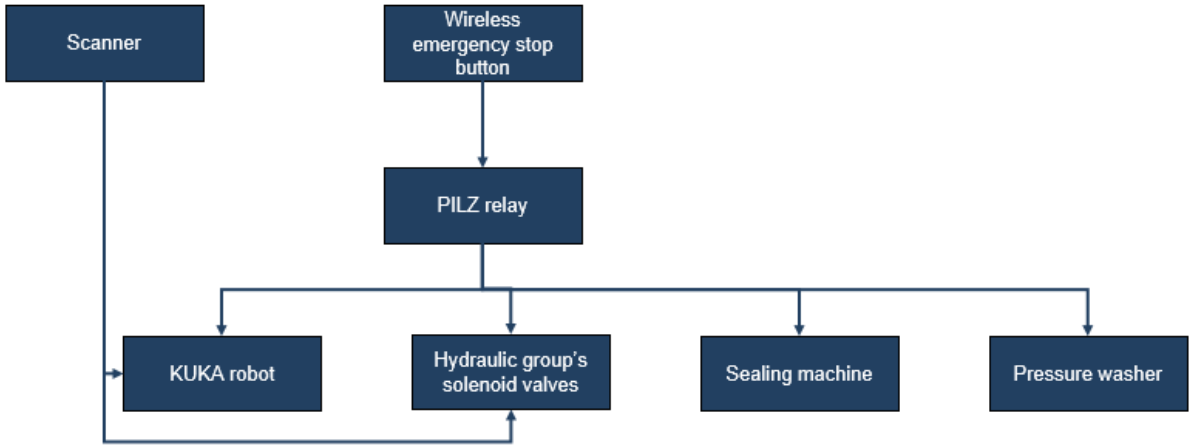


Figure 41. Safety interconnections diagram.

The next immediate step will be to manufacture the general cabinet that integrates all the elements that make up the complete system of the modular robotic platform. This step has been postponed due to the need to have all the components available in order to be able to make a proper definition of how to power and manage the system, especially from a safety point of view. We have waited until all the necessary elements are available, such as the container and the hydraulic unit, in order to design and manufacture a cabinet that integrates all these elements in a safe and efficient way. The detailed description of the manufactured cabinet will be included in the intermediate report of OMICRON's robotic developments, D4.2.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

6 Conclusion

This document describes the implementation of the modular robotic platform prototype done in the OMICRON project. The document includes both the hardware and software implementation, along with the integration of the elements in the system. The features that the developed modular robotic platform offer are the following:

- **Perform various maintenance operations**. The modular robotic platform prototype can be used for various maintenance operations, by means of small changes in the configuration (such as changing the robot's tool and repositioning the elements in the box).
- **Easy extensibility to other functions.** The architecture of the modular robotic platform has been implemented with distributed computing in mind. This approach offers several advantages over a centralized system, such as the ability to perform multiple tasks on individual computers while presenting a unified system to users, and the ease of expanding the system.
- **Safe operation.** The overall system is equipped with components that ensure the safe operation of the robot.
- **Ease of use.** The system provides interfaces so that the operator can control the process and at the same time receive information to facilitate its operation.

The modular robotic platform described in this document will be used as the basis for the development of robotic solutions for a number of interventions being developed in WP4 of the OMICRON project. These operations are barrier installation, cone placement and collection, traffic sign cleaning, paint removal and crack sealing.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

7 References

[1] V. Tadic, A. Toth, Z. Vizvari, M. Klincsik, Z. Sari, P. Sarcevic, J. Sarosi y I. Biro, «Perspectives of RealSense and ZED Depth Sensors for Robotic Vision Applications,» *Machines,* vol. 10, nº 183, 2022.

[2] [Online]. Available: https://github.com/IntelRealSense/realsense-ros

[3] [Online]. Available: https://github.com/luxonis/depthai-ros

[4] [Online]. Available: https://github.com/stereolabs/zed-ros-wrapper

[5] [Online]. Available: https://orocos.org/wiki/orocos/kdl-wiki.html

[6] [Online]. Available: https://en.wikipedia.org/wiki/OpenRAVE

[7] [Online]. Available: https://traclabs.com/projects/trac-ik/

[8] [Online]. Available: https://ompl.kavrakilab.org/

[9] [Online]. Available: https://github.com/ros-industrial/stomp

[10] [Online]. Available: https://ros-planning.github.io/moveit_tutorials/doc/pilz_industrial_motion_planner/pilz_industrial_motion_planner.html

[11] [Online]. Available: https://github.com/flexible-collision-library/fcl

[12] [Online]. Available: https://github.com/jhan15/traffic_cones_detection?utm_source=pocket_mylist

[13] [Online]. Available: https://roboflow.com/

[14] [Online]. Available: https://github.com/aarcosg/traffic-sign-detection

[15] [Online]. Available: https://www.vicos.si/research/traffic-sign-detection/

[16] [Online]. Available: https://github.com/skokec/detectron-traffic-signs

[17] [Online]. Available: https://github.com/aryan0141/RealTime_Detection-And-Classification-of-TrafficSigns

[18] [Online]. Available: https://github.com/Alzaib/Traffic-Signs-Detection-Tensorflow-YOLOv3-YOLOv4

[19] [Online]. Available: https://github.com/aryan0141/RealTime_Detection-And-Classification-of-TrafficSigns/tree/main/Codes

[20] [Online]. Available: https://github.com/khanhha/crack_segmentation

[21] [Online]. Available: https://github.com/TachibanaYoshino/Road-Crack-Segmentation--Keras

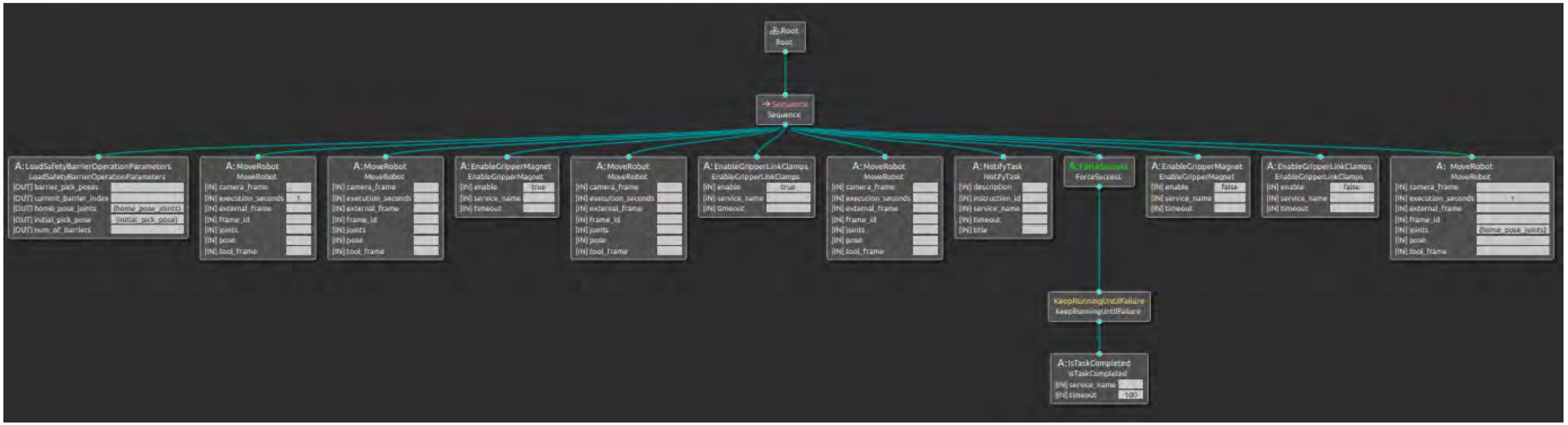
[22] [Online]. Available: https://github.com/Yuki-11/CSSR

[23] [Online]. Available: https://github.com/qinnzou/DeepCrack

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

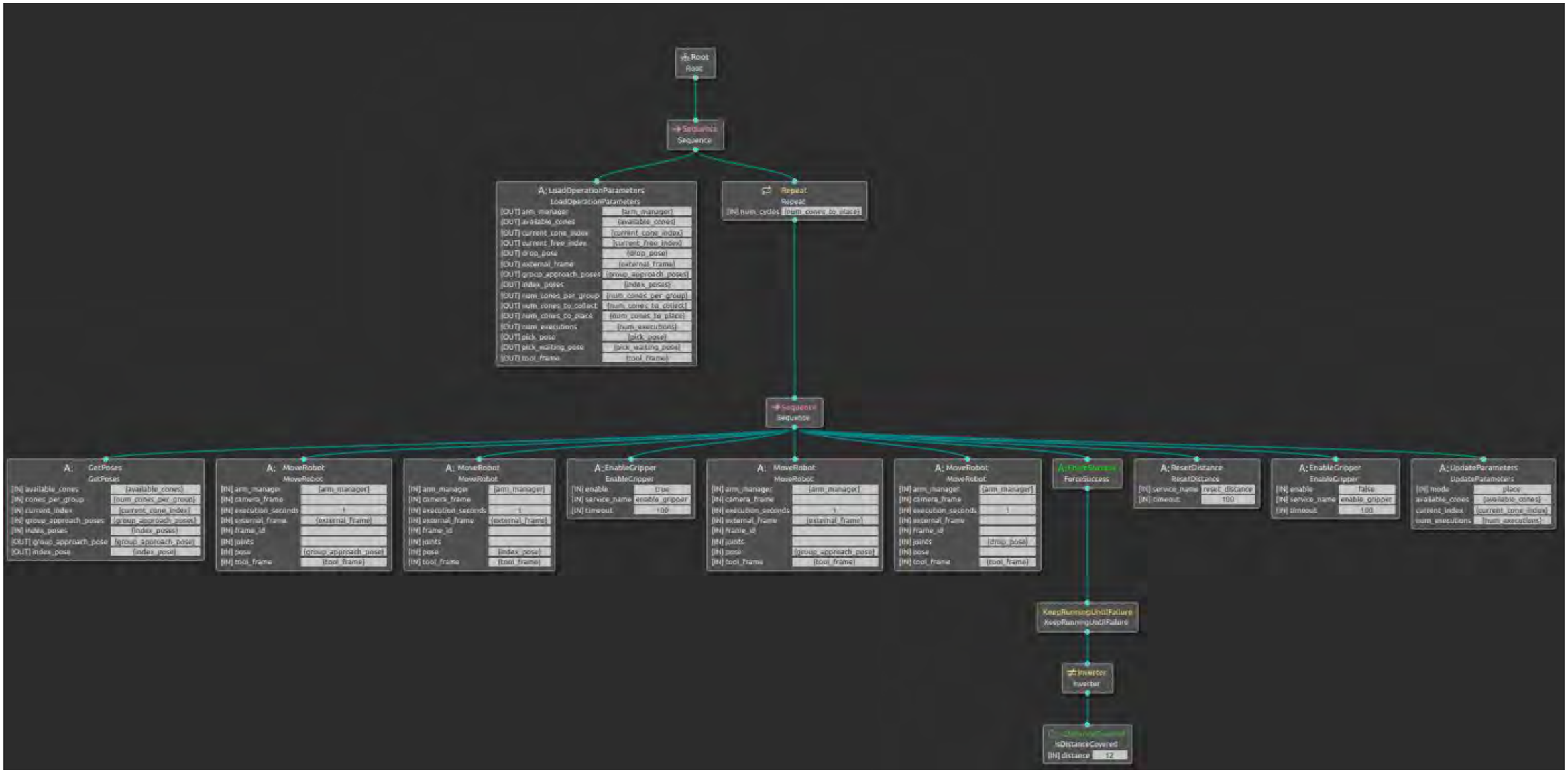
Annex I: Behaviour Trees of the applications

a) Application for installing safety barriers



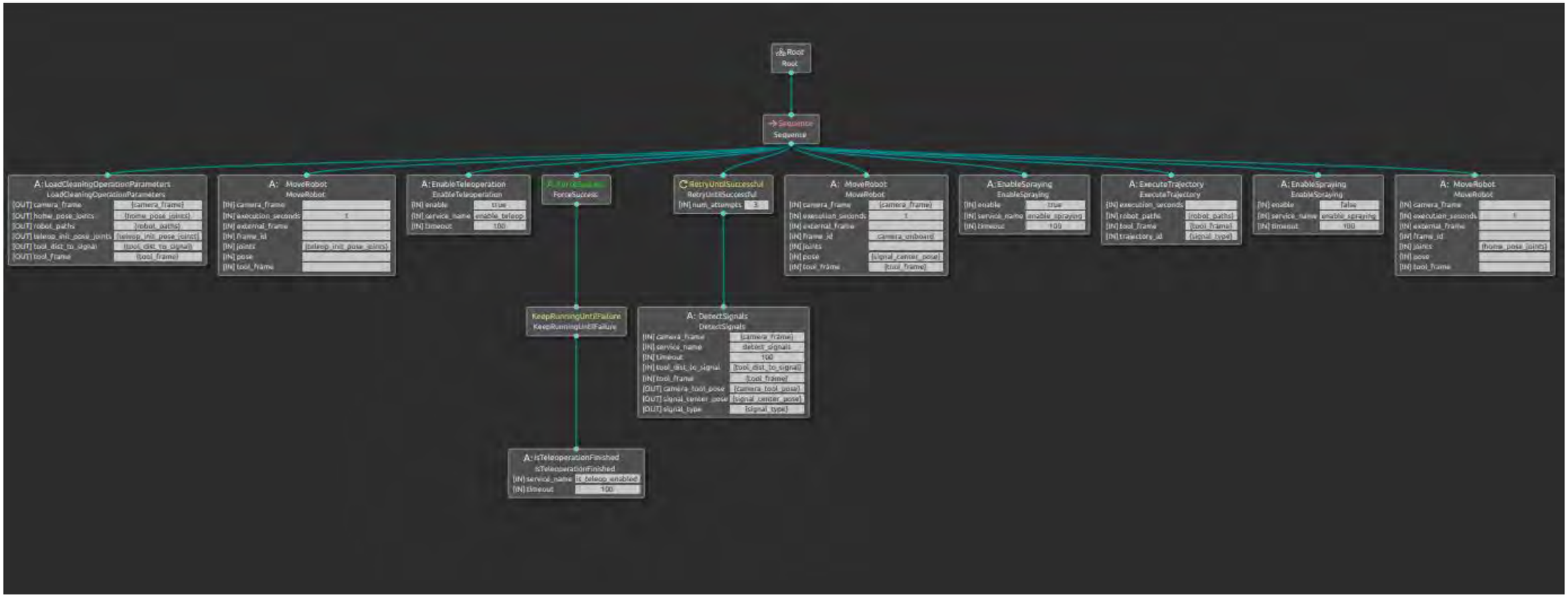
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

b) Application for placing & collecting cones



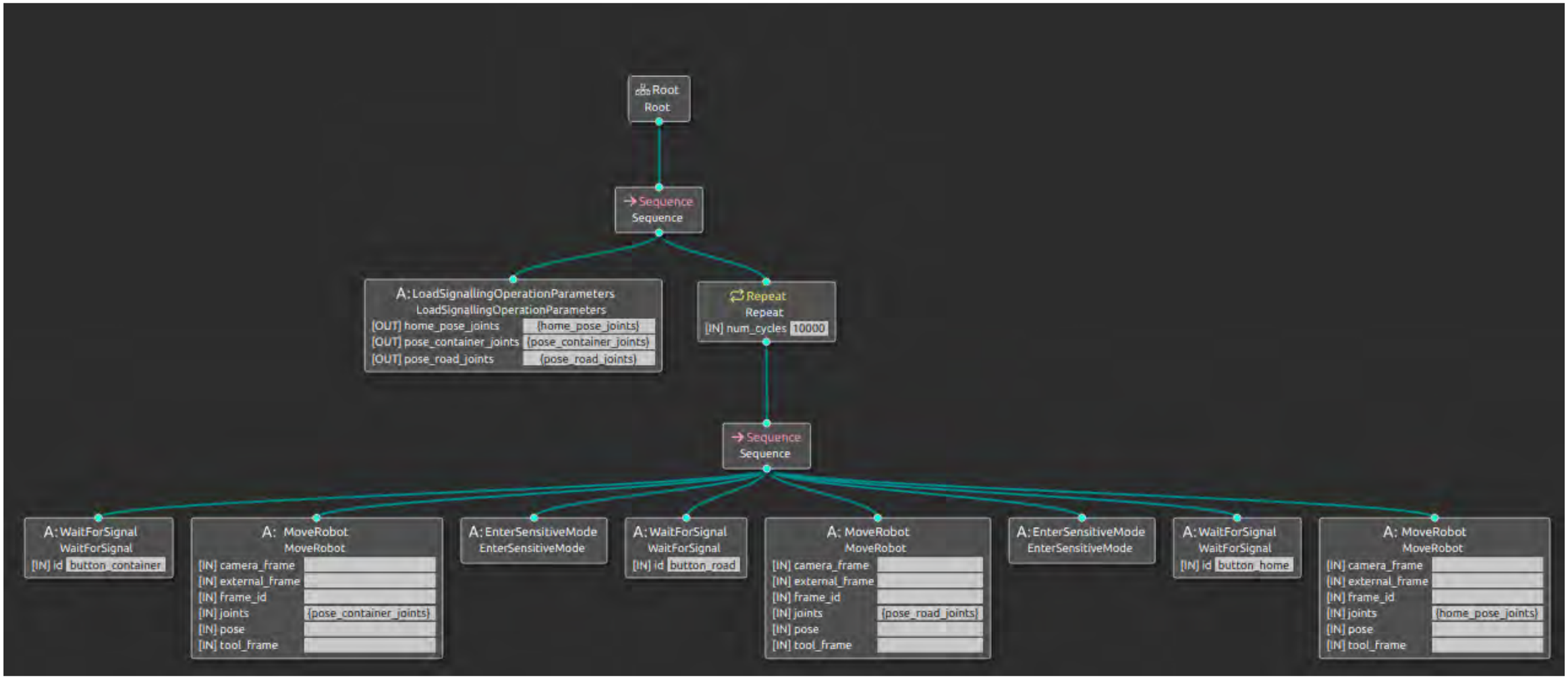
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

c) Application for cleaning road signals



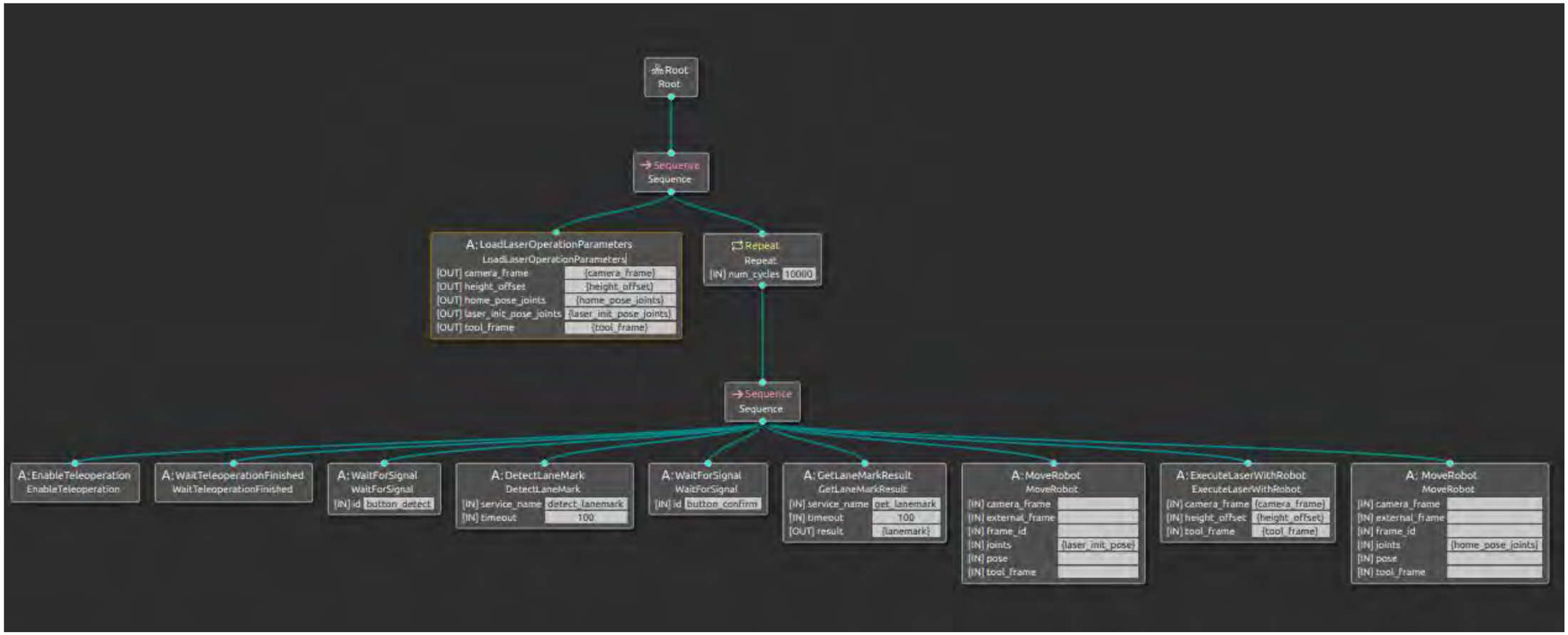
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

d) Application for installing road signals



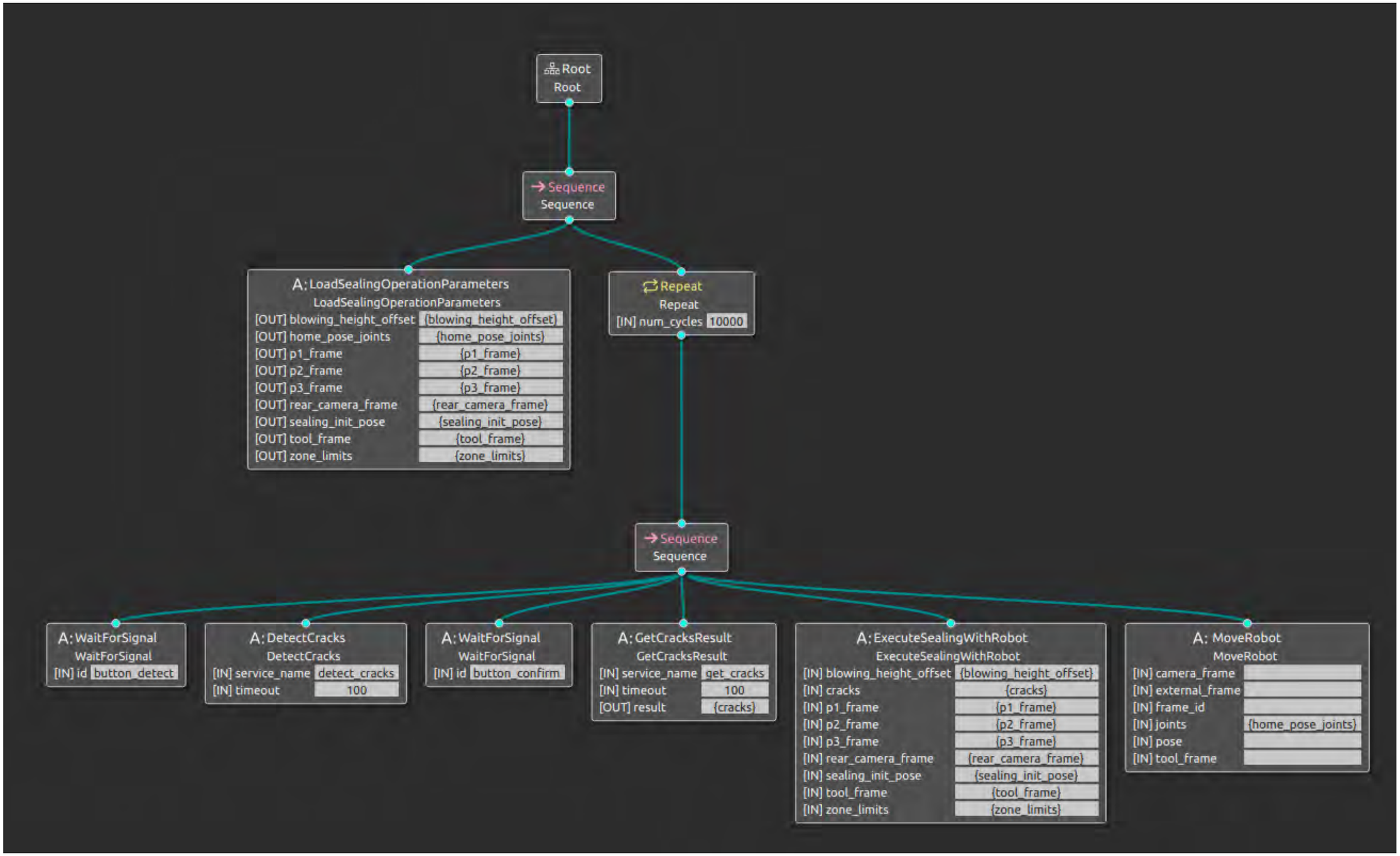
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

e) Application for removing horizontal marks



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269

f) Application for sealing pavement cracks



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°955269