# D9.2 Integrated CoRoSect Platform release I

corosect.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016953

| | |
|---|---|
| **Author(s)/Organisation(s)** | ATOS |
| **Contributor(s)** | HSEL, ROB, UM, CERTH, OAMK, AGVR |
| **Work Package** | WP9. Secure platform integration |
| **Delivery Date (DoA)** | 2022-12-31 |
| **Actual Delivery Date** | 2022-12-30 |
| **Abstract:** | The document presents the first version of the Integrated CoRoSect system, with the details of the real deployment of the components needed to build the system. It covers the main MES functionalities and the interfaces to communicate with the shop floor robots. It relies on the final version of the CoRoSect architecture (D2.3/D2.4) and builds on top pf OPC-UA, MQTT and NGSI protocols. Finally, it describes the testbed instance used to evaluate the integration and the performance of their components. |

| **Document Revision History** | | | |
|---|---|---|---|
| **Date** | **Version** | **Author/Contributor/ Reviewer** | **Summary of main changes** |
| 24/10/2022 | 0.1 | ATOS | Table of Contents |
| 15/12/2022 | 1.0 | ATOS, CERTH, ROB, UM, OAMK, AGVR | Contents for corresponding sections |
| 20/12/2022 | 1.1 | ATOS | Internal review version |
| 21/12/2022 | 2.0 | ATOS, UM, HSEL | Version ready for delivery |

| **Dissemination Level** | | |
|---|---|---|
| **PU** | Public | **X** |
| **PP** | Restricted to other programme participants (including the EC Services) | |
| **RE** | Restricted to a group specified by the consortium (including the EC Services) | |
| **CO** | Confidential, only for members of the consortium (including the EC) | |

Funding Scheme: Innovation Action (IA) ● Topic: H2020-ICT-46-2020

Start date of project: 01 January, 2021 ● Duration: 36 months



| CoRoSect Consortium | | | |
|---|---|---|---|
| **Participant Number** | **Participant organisation name** | **Short name** | **Country** |
| 1 | UNIVERSITEIT MAASTRICHT<br>https://www.maastrichtuniversity.nl/ | UM | NL |
| 2 | ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS<br>https://www.certh.gr/ | CERTH | GR |
| 3 | HOCHSCHULE EMDEN/LEER<br>https://www.hs-emden-leer.de/en/ | HSEL | GER |
| 4 | LUONNONVARAKESKUS<br>https://www.luke.fi/ | LUKE | FIN |
| 5 | OULUN AMMATTIKORKEAKOULU OY - OULU UNIVERSITY OF APPLIED SCIENCES<br>https://www.oamk.fi/fi/ | OAMK | FIN |
| 6 | FUNDACION PARA LAS TECNOLOGIAS AUXILIARES DE LA AGRICULTURA<br>http://www.fundaciontecnova.com/ | TECNOVA | ES |
| 7 | KATHOLIEKE UNIVERSITEIT LEUVEN<br>https://www.kuleuven.be/kuleuven/ | KU LEUVEN | BEL |
| 8 | ATOS IT SOLUTIONS AND SERVICES IBERIA SL<br>https://atos.net/en/ | ATOS | ES |
| 9 | ROBOTNIK AUTOMATION SLL<br>http://www.robotnik.es/ | ROB | ES |
| 10 | AGVR BV<br>www.agvegroup.com | AGVR | NL |
| 11 | NASEKOMO AD<br>https://nasekomo.life/ | NASEKOMO | BG |
| 12 | ENTOMOTECH SL<br>http://entomotech.es/ | ENTOMOTECH | ES |
| 13 | ENTOCYCLE LTD<br>https://www.entocycle.com/ | ENTOCYCLE | GB |
| 14 | SOCIETA AGRICOLA ITALIAN CRICKET FARM SRL<br>https://www.italiancricketfarm.com/ | ICF | IT |
| 15 | INVERTAPRO AS<br>https://www.invertapro.com/ | INVERTAPRO | NOR |
| 16 | FIELD LAB ROBOTICS BV<br>https://www.fieldlabrobotics.com/ | FLR | NL |
| 17 | FoodScale Hub<br>https://foodscalehub.com/ | FSH | RS |

| | | | |
|---|---|---|---|
| 18 | AgriFood Lithuania DIH<br>https://www.agrifood.lt/ | AFL | LT |
| 19 | CENTRO INTERNAZIONALE DI ALTISTUDI AGRONOMICI MEDITERRANEI<br>http://www.iamb.it/ | CIHEAM | IT |

# Table of Contents

# Executive Summary

This document takes up the **Advanced CoRoSect's architecture** defined by Task 2.3 in its D2.4 (M24) and details the **implementation of the first instance of the CoRoSect's System**, ready to test the integrations and the performance. Here are presented all the components' first functional implementation schemas, including the protocols, hardware interfaces, connections, programming languages and hardware devices used to build all of them and so, support all their addressed functionalities and integration capabilities.

This text also links with D9.1, as it uses the here provided interfaces (**RAMI4.0 Asset Administration Shells**) to integrate and communicate all the components and layers, and with D9.3 by providing the technical layout for the first round of system's tests.

From the existing technologies and I4.0 compliant initiatives for IIoT architectures, **FIWARE framework** has been selected for implementing the CoRoSect's Information Management System (IMS) and the data sharing interfaces to support AAS-based communications and data storage. This implementation enables the shop floor management and links the OT layer with the IT layer using I4.0.

All the implementations rely on RAMI4.0 compliant standards to expose their connection interfaces, and common network protocols to build the system's network.

In summary, this D9.2 provides a **concise technical description** of the approaches (including protocols, programming languages, deployment environments, etc.) selected for the **deployment**, **integration**, and **instantiation** of the CoRoSect System first release, **as pointed in M18 review**, intended to support the pilots' demonstrators (WP10) and CoRoSect evaluation (T9.3). It also presents the **cloud Testbed** created that includes an instance of this first release, intended to help CoRoSect's developers to integrate and test their corresponding devices.

# List of Figures

| List of Abbreviations and Acronyms | |
|---|---|
| AAS | Asset Administration Shell |
| AGV | Automated Guided Vehicle |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| AR | Augmented Reality |
| CIA | Confidentiality, Integrity, and Availability |
| DMS | Data Management System |
| DoW | Document of Work |
| D-Robot | (Stacking) De-Stacking Robot |
| DS | Decision-Making System |
| DSS | Decision Support System |
| DT | Digital Twin |
| DX.Y | Deliverable X.Y |
| ERP | Enterprise Resource Planning |
| ETSI | European Telecommunications Standards Institute |
| HRC | Human-Robot collaboration |
| ICF | Italian Cricket Farm |
| ID | Identification |
| IdM | Identity Manager |
| IIoT | Industrial Internet of Things |
| IMS | Information Management System |
| IoT | Internet of Things |
| IT | Information Technology |
| JSON | JavaScript Object Notation |
| Mx | Month X |
| MES | Manufacturing Execution System |
| ML/DL | Machine Learning / Deep Learning |
| MQTT | Message Queuing Telemetry Transport |
| M-Robot | Manipulation Robot |
| NGSI | Next Generation Service Interfaces |
| NGSI-LD | NGSI-Linked Data |
| OPC-UA | OLE (Object Linking and Embedding) for Process Control-Unified Architecture |
| OSI | Open Systems Interconnection |
| OT | Operations Technology |
| PDP | Policy Decision Point |
| PEP | Policy Enforcement Point |
| Q&R | Query & Retrieve |
| RA | Reference Architecture |
| RAMI4.0 | Reference Architecture Model for Industry 4.0 |
| RBAC | Role-Based Access Control |
| RCS | Robot Control System |

| | |
|---|---|
| REST | Representational State Transfer |
| RM | Route Manager |
| ROS | Robot Operating System |
| SFM | Shop Floor Manager |
| SLAM | Simultaneous Localization And Mapping |
| SQL | Structured Query Language |
| WP | Work Package |

# 1 Introduction

## 1.1 Scope and objectives of the deliverable

This is the document that presents the first implementation of the CoRoSect's System, based on the advanced architecture provided by Task 2.3 in D2.4. Its objective is to detail the set of hardware and software pieces selected to provide RAMI4.0 integration, supporting I4.0 protocols and interfaces defined by D9.1. This system's first implementation will be used to test and evaluate:

- The **integrations** between all system's components according to the defined interfaces (i.e. defined AAS)
- The **performance** of the **RAMI4.0** connectors developed within the scope of the project
- The data gathering process and information distribution, including synchronous and asynchronous query/retrieve interfaces
- The **commands' flow** between the Information Technologies (IT) layer and the Operations Technologies (OT) layer (i.e., Shop Floor Manager and Shop Floor Devices)
- The **interactions** defined **between integrated components**, (i.e., handling cells) their performance and response times
- The **Digital Twins** implementations usability for data sharing and process management
- The **replicability** and **scalability** of the proposed implementation, by testing and comparing cloud and on-premises deployments and providing different solutions to the farms to implement the CoRoSect's solution.
- The level of fulfilment of CoRoSect's **functional and technical requirements**, supervised by the end-users (i.e. insects' farms responsible) and according to the work done in WP2

For this purposes, Task 9.2 is providing a cloud testbed, based on Kubernetes and Docker containers, so first developed versions of different software components (context brokers, data bases, connectors, interfaces, controllers, etc.) can be deployed and managed by each responsible partner to polish the implementation of their interfaces, digital twins, and data management operations.

This testbed is used to evolve the CoRoSect's pieces to stable versions that conform the on-site orchestrator, based on Docker containers, that quickly deploys an isolated instance of the CoRoSect's system in the farm's intranet. This implementation will be used to run the corresponding pilots on the selected farms.

## 1.2 Relationships with other deliverables and tasks

The CoRoSect's System implementation requires first from the architecture defined to support all the requirements and functionalities derived from the WP2 analysis. This is provided by the Task 2.3 and both versions of the architecture (Initial – D2.3 and Advanced – D2.4) are used first for the initial developments and finally to align with the latest version the implementations' descriptions. From WP4 to WP8 progresses, this task derives the proposed implementations for the systems' building blocks and integration interfaces defined by Task 9.1. Finally, the instantiated system will support the WP10 pilots and so, the evaluation of the CoRoSect's system. These relationships are depicted in Figure 1 and include the following incomes from WPs:

- **WP2 (Use-cases, user requirements and system architectures)**: user requirements and specifications from Tasks 2.1 and 2.2 are initially collected by Task 2.3. This are shown in
    - D2.3 Initial System Architecture (M12) [1]
    - D2.4 Advanced System Architecture (M24) [2]

- **WP4 (Farm-level modelling and orchestration)**: provides the architectures for the SFM, the DSS and the IMS.
    - D4.2 Data analytics to obtain the prediction models (M18) [3]
    - D4.3 IMS Implementation (M30) (preliminary proposals for its implementation)
- **WP5 (AI-enabled perception methods)**: provides the architecture for the Objects detector and the VR tools for the Augmented Reality (AR) simulators.
    - D5.1 Object detection methods for environment analysis (M30) (initial proposals)
    - D5.2 Tools for natural interaction with the VR environment (M30) (initial proposals)
- **WP6 (Robotic actions planning and control)** and **WP7 (Cognitive robots and smart mechatronics)**: provide the architectures for Shop Floor components deployment and integration.
    - D6.1 Documentation of control for handling of crates (M12) [4].
    - D6.2 Documentation of control for insect handling (M12 and M24) [5].
    - D6.4 Safety concept and control in robotic systems (M12 and M24) [6].
    - D7.1 Sensoring solution to support insect rearing process automation (M30).
    - D7.2 Report on and documentation of robot cell for handling crates (M12 and M24) [7].
    - D7.3 Report on and documentation of robot cells for object manipulation and feeding (M12 and M24) [8].
    - D7.4 Report on and documentation of AGV for insect farms (M30)
- **WP8 (Human-robot collaboration schemes)**: architecture for the Routes' Manager deployment and integration.
    - D8.2 Autonomous and human-aware robot trajectory plan for safe and efficient HRC (M32)
- **WP9 (Secure platform integration)**: architecture of the CoRoSect system instances.
    - D9.1 Integration plan and definition of the interfaces (M24) [9]

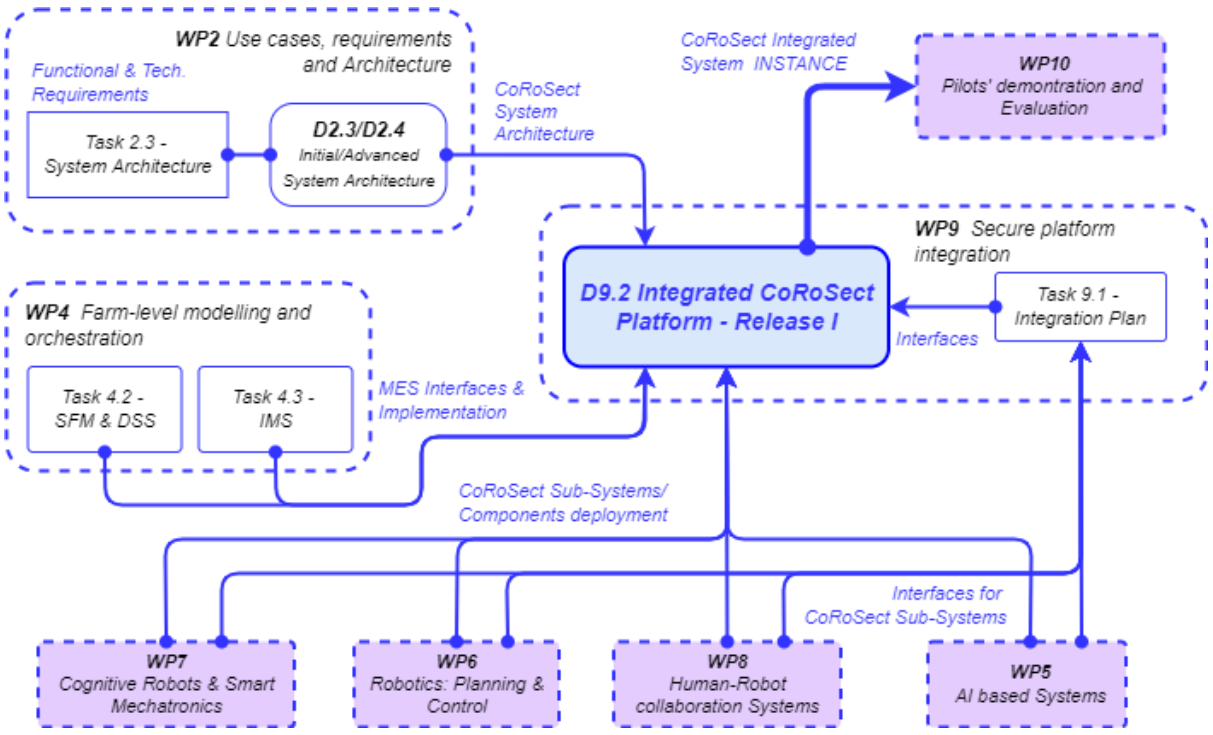


*Figure 1: D9.2 dependences with other CoRoSect's tasks and work packages*

## 1.3 Structure of the deliverable

As introduced, the scope of this document is to detail the **implementation of the first release of the CoRoSect's System**, including the set of selected and developed software and hardware componentes to support CoRoSect's buiding blocks and functionalities. This proposed implementation will follow the Advanced System Architecture and implememt the RAMI4.0 interfaces defined in Task 9.1. According to this, the structure of this deliverable is as follows:

- Section 2 (**Architecure**) takes up the CoRoSect's Advanced Architecture from D2.4 as the guideline to implement and present the CoRoSect's system. Introduces the FIWARE Smart Industry architecure as the core for the IMS.
- Section 3 (**Shop Floor Components**) details the components of the project's Shop Floor and describes their implementation according to their architecture introduced in D2.4.
- Section 4 (**MES**) details the components of the project's Manufacturing Execution System and their implementation. Includes also the definition of the interfaced supported to manage collected information.
- Section 5 (**Robots' planning & control**) details the implementation of the project's handling cells and sub-systems to control the shop floor performance according to project's objectives.
- Section 6 (**Human-Robot collaboration**) covers the elements for the project's safe routing of AGVs and the tools to enhance human and robot learning processes.
- Section 7 (**CoRoSect's deployments**) describes the cloud instance used as testbed and the orchestrator created to instantiate the defined system's implementation.
- Section 8 (**Conclusions**) closes the deliverable and the CoRoSect's firs release description, setting the bases for the pilot's support (WP10) and the final CoRoSect's relase.

# 2 CoRoSect's System Architecture

Current section sets the baseline for the CoRoSect systems implementation. It uses the Advanced System Architecture (Figure 2) provided by Task 2.3 (D2.3 and D2.4) to identify the components that need to be developed and instantiated. This directly links the deployed system with the requirements and objectives of the project, to evaluate their fulfilment and performance. On the other side and extracted from the current architecture implementations presented in D2.3/D2.4, it also introduces a catalogue of open-source enablers to develop and implement the required building blocks.

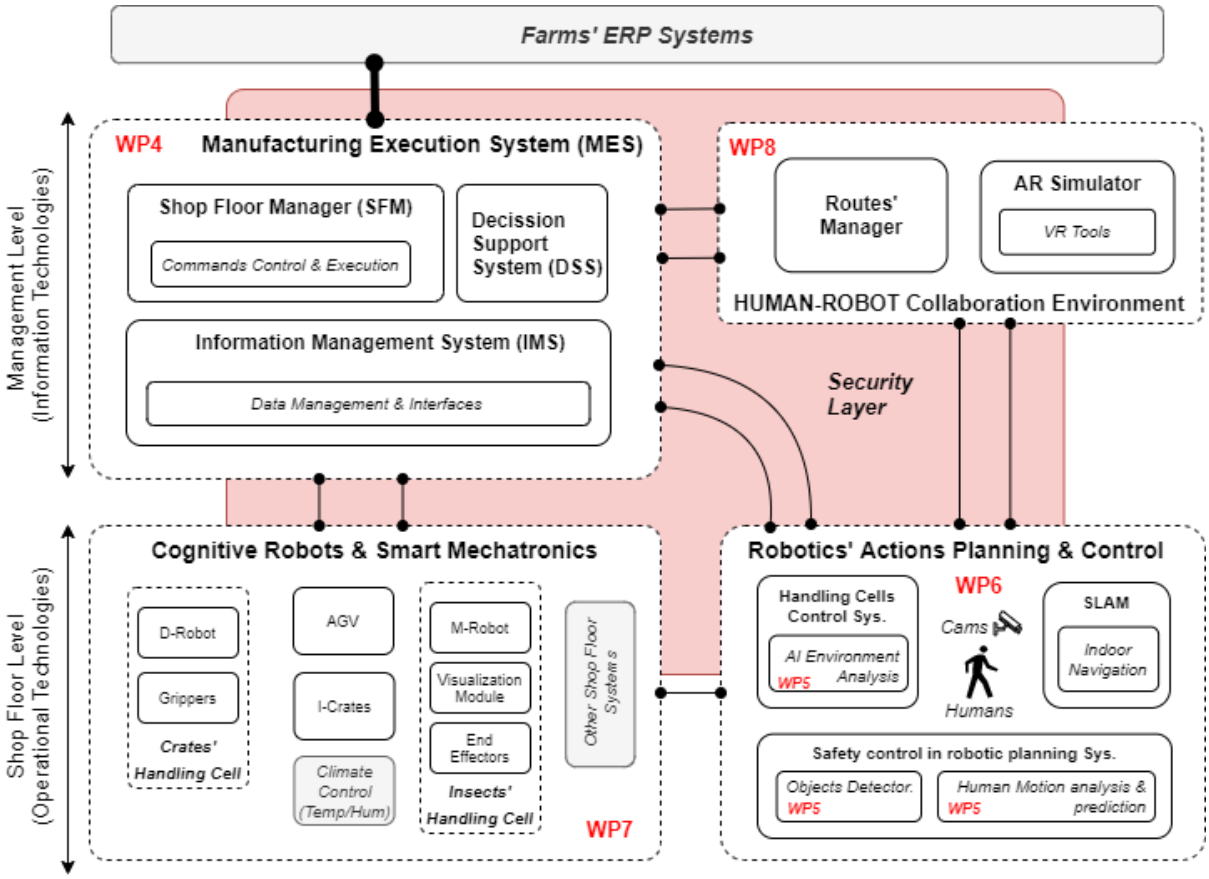


*Figure 2: CoRoSect's Reference Architecture (from D2.4)*

## 2.1 CoRoSect's Advanced System Reference Architecture

Task 2.3 within CoRoSect's WP2 has defined an initial (D2.3) and advanced (D2.4) system architecture to cover project's objectives, scenarios and use cases. These architectures are based on the requirements (both, technical and functional) extracted from the farms automation's needs; insect's rearing processes to be implemented; and the involved shop floor components providers themselves. This Advanced System Architecture (an evolution of the Initial Architecture) is used the planning guide to deploy the software and hardware components that build all the functional pieces and fulfil the requirements specified by the WP2. This architecture (Figure 3) divides the full system into four main functional blocks:

- The **Shop Floor Level** (Operational Technologies); includes the robots, attached devices, IoT infrastructures and mechatronics used to execute the insects' rearing processes identified in CoRoSect. This is represented mostly by the hardware components, but also will deploy the

proprietary software controllers and specific cell controllers that enable the direct interaction with these elements.

- Management Level – **Manufacturing Execution System** (MES); implements the software components to homogenise, stores and manage all the information from the Shop Floor (and any other potential data source) and support the trigger and control actions for the manufacturing process, with assisted decision-making module.

Between these two levels, the system implements the integration layer based on RAMI4.0 interfaces and protocols defined in D9.2

- **Robotics' Actions Planning and Control**; exploits data from the MES and from the direct controllers of the shop floor components; and implements the cell controllers that interacts with all components at the shop floor to ensure an efficient and safe process orchestration at the cell's level.
- **Human-Robot Collaboration Environment**; defines and implements a set of systems that operates at the top of the architecture to manage the AGV at the shop floor. These are intended to optimise their routes while avoiding accidents and blockages. Here are also included the VR tools to enhance human and robot learning processes.

Assisting these two sections, the Simultaneous Localization and Mapping (SLAM) module will be implemented and supported by the AGV controller and the Route Manager sub-system.

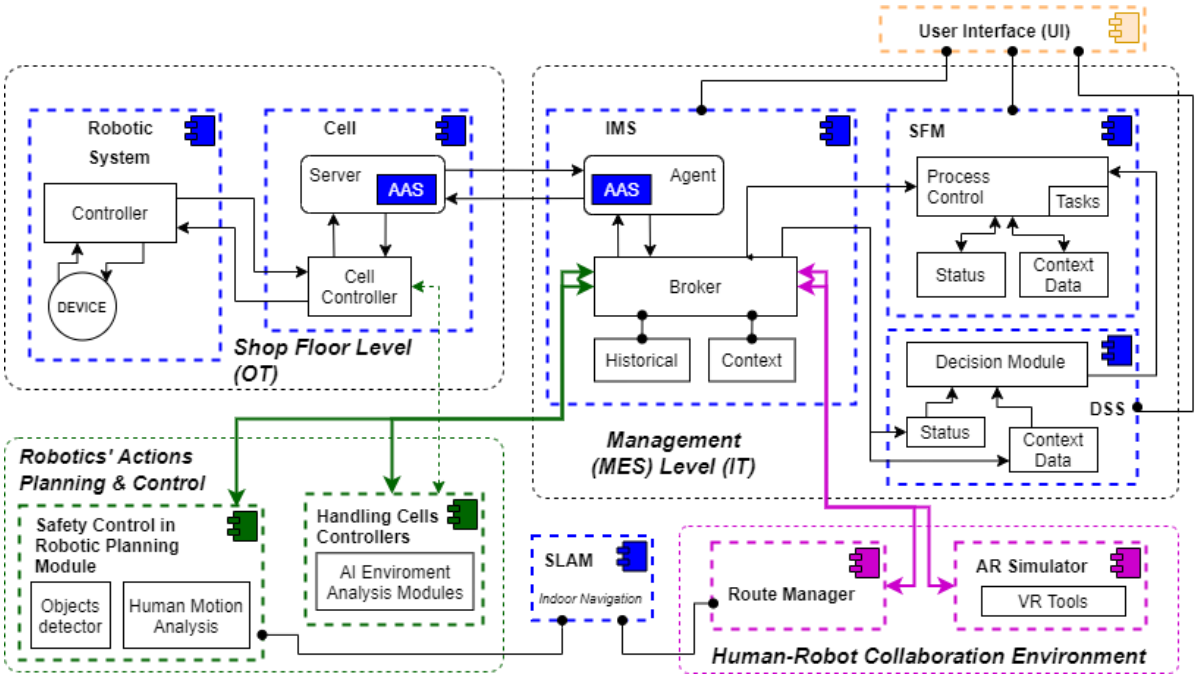


*Figure 3: CoRoSect's Advanced Reference Architecture – Logical View (from D2.4)*

## 2.2 FIWARE Smart Industry Reference Architecture

FIWARE [10] was born in early 2010's as a platform, driven by the European Union, for the development and global deployment of Future Internet (FI) applications. Initially, it was focused on IoT infrastructures exploitation within the smart city's framework to promote open standards and open-source developments to foster smart cities evolution, but, once matured within this environment, it is exporting its expertise and integrating IoT in other cutting-edge scenarios, such as the Smart Industry and I4.0. Its greatest differential values with respect to other initiatives are:

- 100% Open standard (Open Source).
- Open data models and licence-free use, with mass adoption and driven by an specific initiative: Smart Data Models.
- Standard APIs supported by the OMA and the ETSI (ETSI NGSI-LD [11]).
- Wide set of Open-Source library of developed components (General Enablers) to interact and integrate woth different IT infrastructures (coomon data bases, dashboards, ERPs, etc.)
- Wide (and growing) adopters' community (components deployed in more than 250 cities all over the world).

The FIWARE solution for Smart Industry [12] exports its curated architecture to the Industry IoT (IIoT) world, focusing on the decisions' support and the business processes' automation. It proposes a functional architecture (Figure 4) that combines the NGSI standard and the Smart Data Models for easy and agile data management, with the extended and powerful I4.0 compliant protocols. It creates a versatile data space for effective data exchange within the smart manufacturing farm's scenario but also with other related domains, such us smart logistics, enabling the creation of innovative value chains.

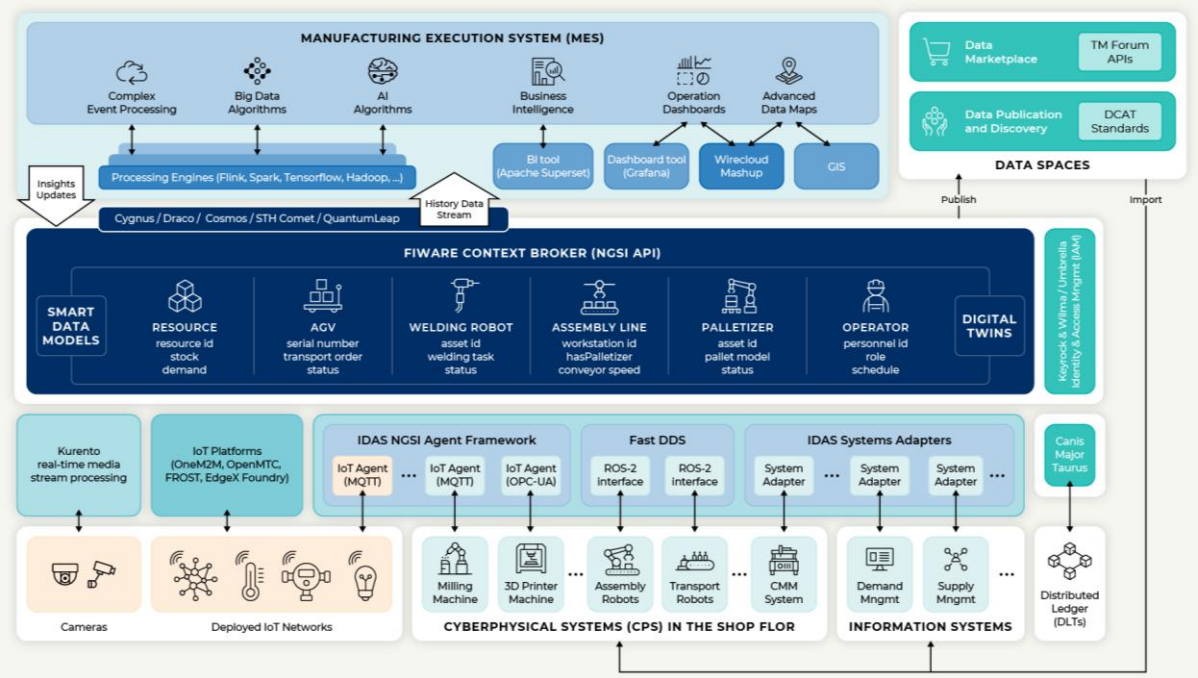


*Figure 4: FIWARE[1] Smart Industry Reference Architecture*

The solution proposed for CoRoSect's System, based on its advanced architecure, builds on top of the management of Context Information: FIWARE framework defines context as a collection of entities that work as Digital Twins of real world assets, both physical (e.g., a gripper, a robot arm or an operator) or conceptual (e.g., a claim, a command, etc.). FIWARE architecture provides enablers to work with the NGSI standard open API [13] and manage information about these entities, as the CoRoSect's Digital Twins. From the achitecture in Figure and from the FIWARE General Enablers' cataloge, CoRoSect extratcs for the implementation of its system:

- **IoT Agents** [14] are specific adaptors for a given communication protocol and its dara structure to the NGSI protocol and JSON models payloads. On one side, they implement a

[1] https://www.fiware.org/about-us/smart-industry/

proprietary protocol (for a dedicated IoT Agent) or a non-ngsi interface (for common communication standards), whilst,on the other side, they provide a REST interface able to manage NGSI updates and requests. They are usually configured to convert a specific data structure, provided by the device manufaturer or the native communication protocol to a selected Smart Data Model JSON payload. FIWARE catalogue provides native MQTT, LoRa and OPC agents, but also the code and structure to code and deploy customised ones.

- The **Context Broker** [15] is the core component of the architecture, implementing and supporting the NGSI API Rest and functionalities. Currently, there are several implementations available in the catalogue, supporting NGSIv2, NGSI-LD or both. All the information of the system passes through the Context Broker.
- **Historical data connectors** and **storage** components connect to the Context Broker to capture selected data and store it. There are different historical data connectors available, able to download data to different data base managers (mongoDB, MySQL, Hadoop, Postgre, etc.) where it can be structured and classified according to time series or to specific customised patterns. This historical datasets can be later used to feed AI engines and/or clients' dashboards.
- **Smart Data Models** initiative [16] provides a wide set of explotable data models for many different devices, sources, scenarios and applications, derived from and enriched by different real use cases. It is a life community that works for the standardisation, homogenisation and disemination of these NGSI-based data models, as well as for the expansion of this set to cover new areas. In this sense, it provides tools and support to define new data models or modify exisiting ones to expand the FIWARE functionalities.

These components will be deployed and combined to build the CoRoSect's System IMS instance that underpins the system integration as shown in the following sections.

# 3 Shop Floor Component's Integration

Here are described the implementation and integration of the different components at the CoRoSect's Shop Floor (Figure 5). This has been defined within WP2, were the required cyber-physical devices and software systems (the OT and the IT layers respectively) have been put in place to build the project use cases and scenarios. It is described in D2.3 and includes the main modules to handle the crates i.e., the Staking/De-stacking cell and its tools; the insects' manipulation cell, with the corresponding robot arm and the visualization module, the I-Crates and the automated guided vehicles. Here also are mentioned the I4.0 compliant protocols needed to communicate with other components.

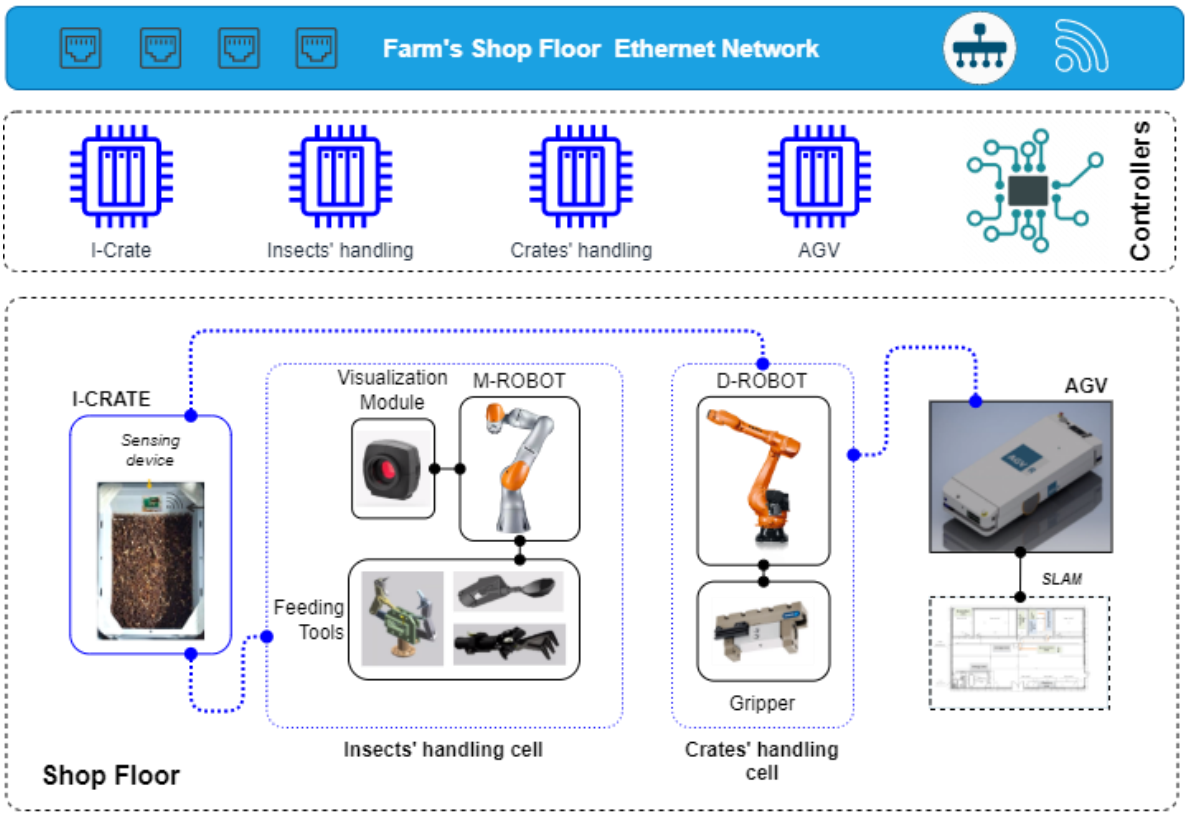


*Figure 5: CoRoSect's Shop Floor (from D2.4 scenario set up)*

## 3.1 Shop Floor Integration Protocols

The standard integration protocols are a key component of the integrated shop floor, as they will support the I4.0 compliant communication. For the physical layers of the communication infrastructure, and ethernet network will be used, supporting TCP/IP protocols (Figure 6). On top of this, and as detailed in D9.1 (Integration Plan and definition of interfaces), CoRoSect solution would support any communication protocol working on top of the TCP/IP stack, release I will work with:

- OPC-UA protocol to communicate Industrial OPC Servers (Robotics): M-Robot, D-Robot, etc.
- MQTT protocol to integrate IoT infrastructures (I-Crates) and those components that already implement any kind of MQTT compliant protocol, such as the AGV or the Routes' manager.

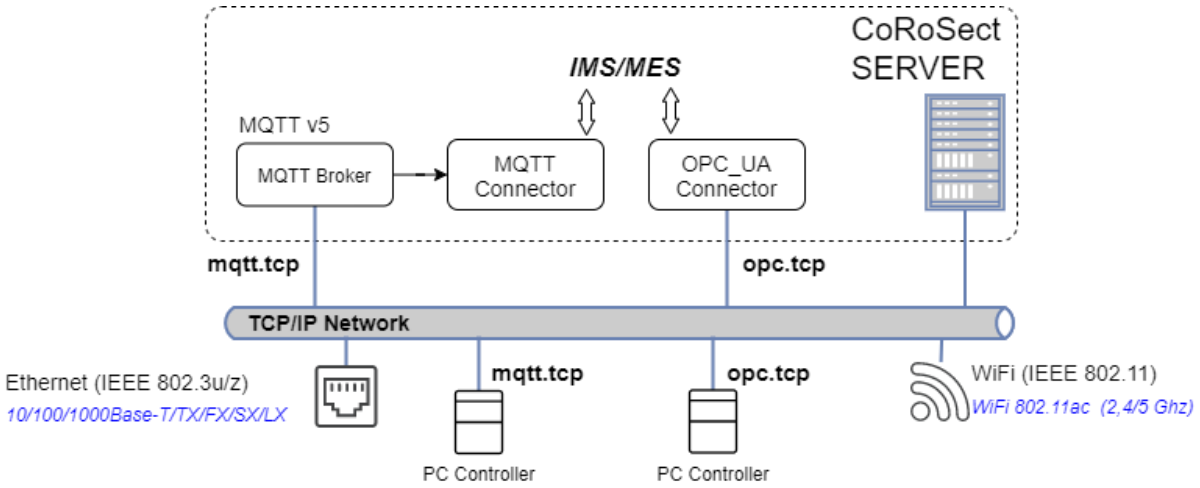


*Figure 6: Network connections and protocols for CoRoSect System (on-site configuration)*

## 3.2 Shop Floor level devices and systems

Following subsections review the CoRoSect's Shop Floor set up (Figure 5) and provides the details of each component's deployment, according to their architectures shown in D2.4.

### 3.2.1 Stacking/De-staking Robot (D-Robot)

The D-Robot is a core component in the CoRoSect Robotic Cell formed by the combination of the D-Robot and attached devices. The D-Robot's main objective is to securely moves crates and boxes from the pallet with stacked crates transported by the AGV to the table of operation, where each crate will be manipulated by the M-Robot, and back again into the empty pallet. The D-Robot is equipped with sensors and robust controllers to manipulate crates filled with insects in a safe way.

As detailed in D2.4, The D-Robot's main **functionality** is divided into two procedures:

- **De-stacking procedure:** D-Robot picks crate from input pallet and places it in operation table.
- **Stacking procedure:** D-Robot picks crate form operation table and places it in output pallet.

The D-robot cell is **composed** by:

- Robotic arm: KUKA KR70 2100[2]. Industrial robot with a reach of 2100mm and a rated payload of 70 Kg.
- Robotic gripper: Schunk pneumatic gripper PSH52[3] equipped with custom jaws for each crate type.
- Lasers and RGB-D sensors for visual servoing.
- Computer with HMI for system monitoring.

The D-Robot **deployment and integration schema** is shown in Figure 7 and follows the general approach proposed for CoRoSect Shop Floor devices. It is divided in:

- **D-Robot Device System deployment,** composed by a KUKA KR70 2100 robotic arm, a KUKA KRC5 controller, and a computer running the D-Robot Control System. The robotic arm is

[2] https://www.kuka.com/-/media/kuka-downloads/imported/6b77eecacfe542d3b736af377562ecaa/0000332117_en.pdf

[3] https://schunk.com/de/en/gripping-systems/parallel-gripper/psh/psh-52-1/p/000000000000302152

controlled through the RobotSensorInterface[4] (RSI) from Kuka which enables the reception of low-level commands from an external computer. RSI is a KUKA proprietary software technology package for implementing applications that require cyclical signal processing as well as high-performance, cyclical influence over the robot motion, it uses XML format for information exchange. The KUKA arm is directly connected to an external PC which runs the ROS D-Robot controller (described in next section) via Ethernet. The KUKA arm is complemented with:

  - Schunk PSH52 pneumatic gripper to grab the crates. The gripper is actuated by a pneumatic system, available in the end user facilities. The actuation of the gripper is controlled by a standard IO system via a PLC which enables the execution of open/close operations. This PLC is connected through ethernet to the computer controlling the D-Robot Control System.
  - Lasers and RGB-D Cameras are used as visual sensors for servoing, which are installed near the end-effector of the robot and connected to via USB to the main computer.

- **D-Robot Control System deployment,** composed by:
  - D-Robot Controller developed using ROS [17], an open-source framework for robotics. A manipulation application based on ROS and MoveIt [18] that enables the robot to robustly and accurately de-stack and stack crates filled with insects has been developed. MoveIt is an open-source tool that allows robot trajectory path planning considering both the robot model and the modelled robot environment, which are used for collision avoidance. The modelled environment can also be modified dynamically, the robot can interact with the modelled environment. This modelled environment is used to move crates around without colliding. An official KUKA ROS driver[5] has also been adapted and integrated in the D-Robot Controller. The driver uses RSI to create an interface to control KUKA robots via ROS. The ROS interface uses joint position commands to control the robot (bridge between ROS and the Kuka low level commands).
  - Gripper Controller is a simple ROS Node that receives commands to operate the gripper, which then activates the corresponding outputs in the in PLC to grab o release the gripper.
  - Visual Sensor Controller are ROS nodes that receive the stream of data from the laser and cameras and output the pose of the crate, which is then used to perform small corrections on the robotic arm movement to compensate the deviation in the positioning of the crates.
  - OPC-UA [19] Server has been developed using ROS. The D-Robot OPC-UA server exposes to the IMS the different command actions available in the D-Robot, it also updates the status of the D-Robot dynamically (whether it is moving, executing commands, or if the commanded actions have succeeded or failed). The D-Robot OPC-UA server is a bridge between the IMS and our software developed using ROS, where ROS actions and services are used to command the robot.

- **D-Robot connection to CoRoSect Network.** The CoRoSect IMS (Information Management System) manages a Digital Twin (DT) for the D-Robot. The command flow through the IMS is implemented using a customized asynchronous PUB/SUB mechanism. In order to trigger the command, the first step is to modify the D-Robot DT. A post request will be sent to the Context Broker (CB) with the command and the value to be executed by D-Robot. This command

[4] https://www.kuka.com/en-gb/products/robotics-systems/software/application-software/kuka_robotsensorinterface

[5] https://github.com/RobotnikAutomation/kuka_experimental/tree/melodic-devel/kuka_rsi_hw_interface/krl/KR_C4

information is automatically addressed to the D-Robot OPC-UA [19] Connector which maps the command request from the SFM to the D-Robot OPC-UA Server to be executed. The D-Robot OPC-UA server receives the command and executes the required ROS action and services needed to operate the D-Robot to carry out the required task. As the D-Robot executes the task the D-Robot OPC-UA server dynamically updates the status of the D-robot. The D-Robot response and/or the commands results are, in turn, written in the D-Robot corresponding DT attribute by its OPC Server/Connector, and then redirected by the IMS to the SFM or to any other interested system.

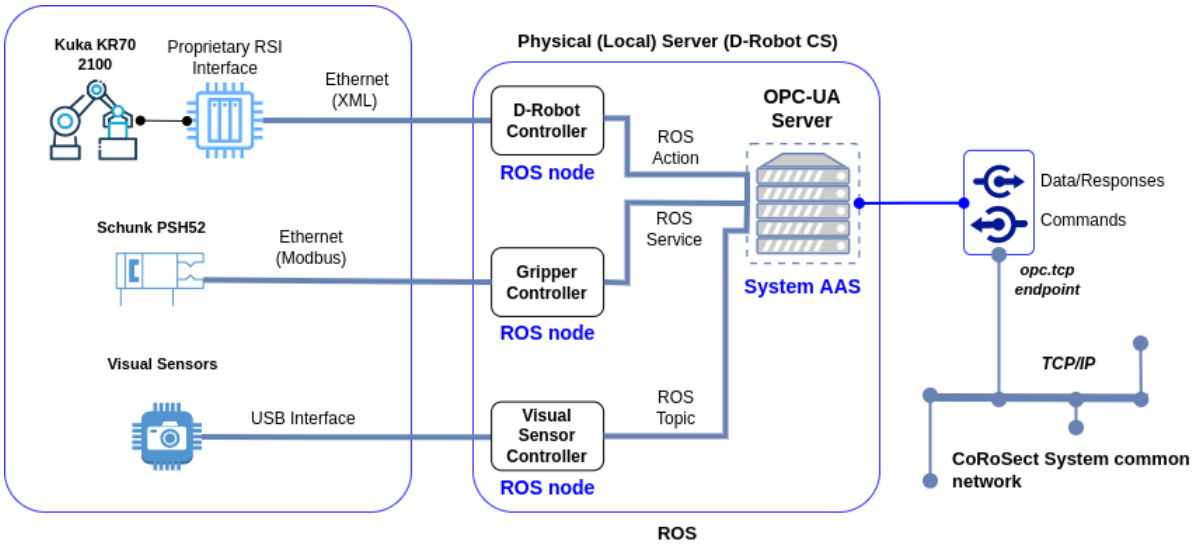


*Figure 7: Stacking/De-stacking Robot (D-Robot) deployment*

### 3.2.2 Manipulation Robot (M-Robot, VI)

The M-Robot is a core component in the CoRoSect Robotic Cell (the entire Cell is formed by the combination of the M-Robot and D-Robot). The M-Robot's main objective is to support the insect rearing processes. For this the M-Robot performs the following tasks (1) manipulation of insects (e.g., picking, placing, sorting), (2) feeding of insects (e.g. adding feed to insect crates), (3) monitoring of insects (e.g. visual monitoring of growth), and (4) material handling for manipulating the insects' environments (e.g. adding/removing support structures into/from crates). The M-Robot is designed to collaborate with human co-workers. To fulfil its tasks the M-Robot is equipped with sensors and robust controllers to perform its tasks in a safe way.

The M-Robot (Figure 8) is **composed** of:

- a KUKA LBR iiwa[6] 14 collaborative robot arm,
- End Effectors such as a pneumatic gripper for safely handling insects and materials,
- a Visual Inspection Sensor providing automatic visual inspections of insects for monitoring of the insect rearing processes and quality control,
- External Visual Sensor(s) for environmental analysis and modelling used for automatic obstacle avoidance, action planning, and human-robot collaboration.

The M-Robot deployment and integration schema is shown in Figure 8 and follows the general approach proposed for CoRoSect Shop Floor devices. It is composed of the following components:

[6] https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa

- **M-Robot Device System deployment,** composed of a KUKA LBR iiwa 14 collaborative robot arm, a KUKA SUNRISE controller, and a computer running the M-Robot Control System. The robotic arm is controlled through the KUKA Line Interface (KLI) from KUKA which enables the reception of low-level commands from an external computer. KLI is a KUKA proprietary technology package for connecting to higher level control infrastructure. The KUKA arm is directly connected via Ethernet to an external PC, which runs the ROS M-Robot controller (described in next section). The KUKA arm is complemented with:
    - customised **End Effectors** such as a pneumatic gripper for safely handling insects and materials. The gripper is actuated by a pneumatic system from Festo that allows opening and closing of the grippers. This pneumatic system is controlled from the M-Robot Control System via an EtherCAT module.
    - a **Visual Inspection Sensor**. This high-resolution camera is mounted on the robot arm and provides images to the D-Robot Control System through a USB interface.
    - **External Visual Sensor(s)** for environmental analysis and modelling are installed to provide an overview of the M-Robot's workspace. The visual sensors provide images to the D-Robot Control System through a USB interface
- **M-Robot Control System deployment,** composed of:
    - **M-Robot Controller** developed using ROS [17], an open-source framework for robotics. A manipulation application based on ROS and MoveIt! [18] has been developed that enables the robot to robustly and accurately plan and execute movements for insect and material handling. MoveIt! is an open-source tool that allows robot trajectory path planning taking into account both the robot model and the modelled robot environment, which are used for collision avoidance. The modelled environment can also be modified dynamically, the robot can interact with the modelled environment.
    - **End-Effector Controller** is a ROS Node that receives commands to operate End-Effectors such as a pneumatic gripper. The ROS Node then controls the End-Effectors via an EtherCAT interface e.g. to activate the gripper.
    - **Visual Inspection (VI) Sensor Controller** are ROS nodes that receive the stream of data from the high-resolution Visual Inspection Sensor and output information relevant for quality management of the insect rearing processes such as detected anomalies. The Visual Inspection Sensor is mounted on the KUKA robot arm so that the Visual Inspection Sensor can be positioned in various poses relative to the crates with insects. This is necessary to provide detailed scans of the crates while avoiding occlusions from obstacles such as necessary support material placed inside the crates for insect rearing. The Visual Inspection Sensor Controller plans and provides the desired positions of the Visual Inspection Sensor relative to the crates as required for the image acquisition and quality management.
    - **External Visual Sensor Controller** are ROS nodes that receive the stream of data from the External Visual Sensors and output a 3D model of the workspace to identify the pose of the crate, the crate's content, and the position and movement of human co-workers. This information is then used by the M-Robot Controller to plan safe robot movements that avoid collisions with obstacles and human co-workers.
    - **OPC-UA Server** has been developed using ROS. The M-Robot OPC-UA server exposes to the IMS the different command actions available in the M-Robot, it also updates the status of the M-Robot dynamically (whether it is moving, executing commands, or if the commanded actions have succeeded or failed). The M-Robot OPC-UA server is a bridge between the IMS and our software developed using ROS, where ROS actions and services are used to command the robot and its components.
- **M-Robot connection to CoRoSect Network.** The CoRoSect IMS (Information Management System) manages a Digital Twin (DT) for the M-Robot and its components. The command flow through the IMS is implemented using a customized asynchronous PUB/SUB mechanism. To

trigger the command, the first step is to modify the M-Robot DT. A post request will be sent to the Context Broker (CB) with the command and the value to be executed by M-Robot. This command information is automatically addressed to the M-Robot OPC-UA [19] Connector which maps the command request from the SFM to the M-Robot OPC-UA Server to be executed. The M-Robot OPC-UA server receives the command and executes the required ROS action and services needed to operate the M-Robot to carry out the required task. As the M-Robot executes the task the M-Robot OPC-UA server dynamically updates the status of the M-robot. The M-Robot response and/or the commands results are, in turn, written in the M-Robot corresponding DT attribute by its OPC Server/Connector, and then redirected by the IMS to the SFM or to any other interested system.

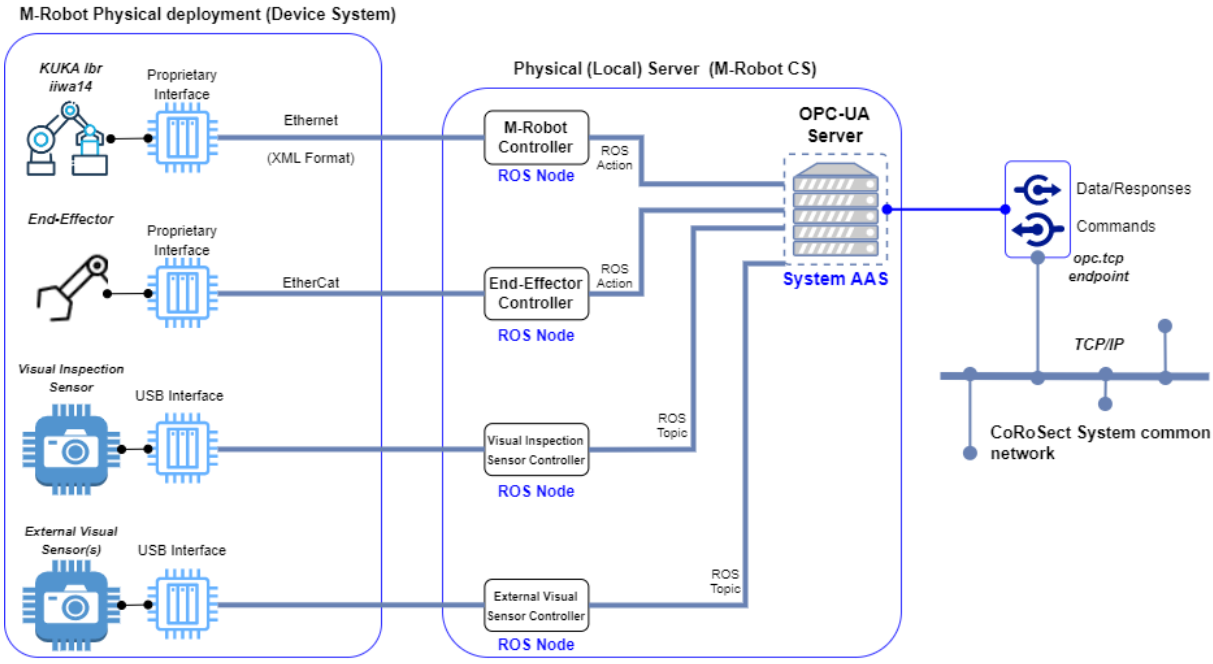


*Figure 8: Manipulation Robot (M-Robot) deployment*

### 3.2.3 Intelligent Crates (I-Crates)

I-Crate architecture has been explained in D2.4. and the interface definition in provided in D9.1. CoRoSect I-Crates contain Intelligent Integrated Sensors. These obtain and transfer sensor data to the I-Crate dedicated gateway device operating as a cell controller system. Sensor data includes the following data: unique ID of the sensor, temperature in Celsius, humidity in %(RH), $CO_2$ in ppm, $NH_3$ in ppm, substrate moisture %, pH value in number and time stamp of each sensor. Figure 9 presents a dashboard view of the sensor values and other data. Sensor data is published with frequency which depends on the need of sensor data to be known. Interval between single measurements of a specific sensor varies between tens of seconds to several hours. This is specified in the requirement specification of sensors.


*Figure 9: Capture from dashboard (CoRoSect MES) with sensor values shown*

In the final solution one or more of the mentioned sensors can be installed, according to the need. This means that there may exist I-Crate with only one temperature sensor (cost-efficient, low power consumption and easy) or I-Crate with all six sensors (expensive, higher power consumption, more complicated to manage). The possible varying sensor configurations, and also the harsh environment is taken into account when developing and implementing the HW and SW-interfaces for the I-Crate.

The I-Crate controller consist of a gateway and included (MQTT [20]) client software. It communicates the I-Crate generated data directly via WiFi or through the I-Crate local server with CoRoSect MES through the IMS as seen in the Figure 10 below.

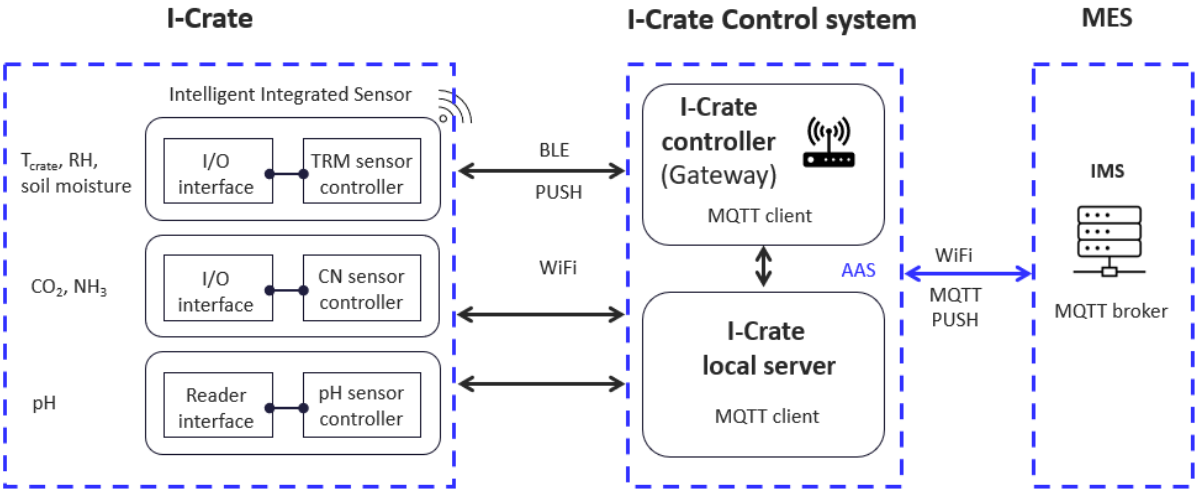


*Figure 10: I-Crate connection to CoRoSect network*

### 3.2.4 Automated Guided Vehicle (AGV)

The AGV is responsible for internal logistics of pallets between Robot Cells. The AGV is designed to operate in the same area as the humans and will anticipate to human detection within predefined distance. The orders received from Route Manager will be routing from point A to Point B. Actions received from Floor Shop Manager for pick, drop, etc. will be performed instantly. The usability of the AGV will be managed by Floor Shop Manager, Floor Shop Manager will request the Route Manager to

send the AGV to a location, after reaching the destination Floor Shop Manager will send an action to the AGV.

**Route manager**

Interface between AGV and Route manager is based on MQTT/VDA5050[7] protocol (Figure 11). VDA5050 enables the integration of different automated guided vehicles in a common system and thus building block for Industry 4.0. Information between the AGV and Master control (Route Manager) is predefined and standardized according to the needs of different AGV suppliers and covers most of commonly used data.

- *Order:* For each new order the AGV will receive current coordinates of the AGV as first coordinates, this is check between Route manager and the AGV that current position is synchronously know by both systems. Additional 2 more coordinates will be added to the first message. After each coordinate is received the AGV will route to next coordinate in the list, at the same time Route Manager will send additional coordinate to enqueue. This construction will prevent stop and drive motion of the AGV and result to smooth operation of the system.
- *Cancel Order:* There is also Cancel Route Action added between Route Manager and AGV for cancelling orders, in this case the AGV will remove already received coordinates that are not released. When a coordinate is released to drive, AGV will move to the last released coordinate and will be in a status for receiving new coordinates, orders or actions.

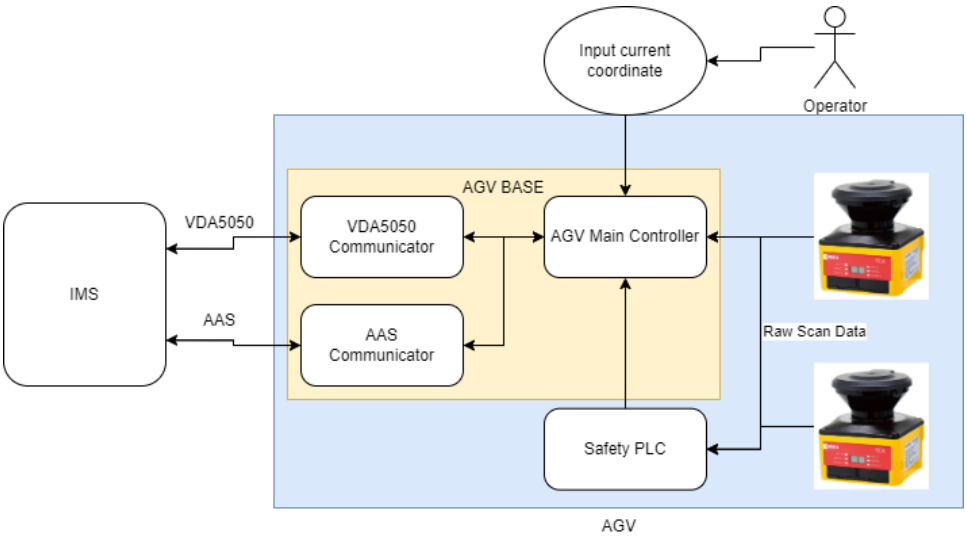


*Figure 11: AGV Data flow*

**Shop Floor Manager**

Interface between AGV and Shop Floor Manager is based on **MQTT/AAS** interface with customized protocol. Floor Shop Manager request route from Route Manager and the AGV will move the destination respectively to the route received from Route Manager. After reaching destination Floor Shop Manager will sends an Operations to AGV to perform (pick, drop, etc., etc.). The AGV will reply that the operation is received and will be processed. Progress of the operations can be received through AGV status topic. When receiving a pick/drop operation, AGV will start 1500 mm in front of the location and use vision or reflector navigation (according to position capability of the area) to drive

[7] https://en.vda.de/en.html, a German interest group of the German automobile industry (manufactures and suppliers)

and position under pallet, raising/lower the lift and drive back 1500 mm on front of the location. Other actions are documented in D6.6 paragraph 6 “Operations”.

# 4 Manufacturing Execution System (MES) components

This section continues with the description of the CoRoSect’s Manufacturing Execution System deployment (Figure 12). It relies on the WP4 outcomes, defining their components and on D2.3/D2.4 that show its architecture. Starting from the network infrastructure and communication protocols introduced by the Shop Floor (Section 3.1), here are detailed the implementation of the MES interfaces and implemented protocols that support their addressed functionalities.

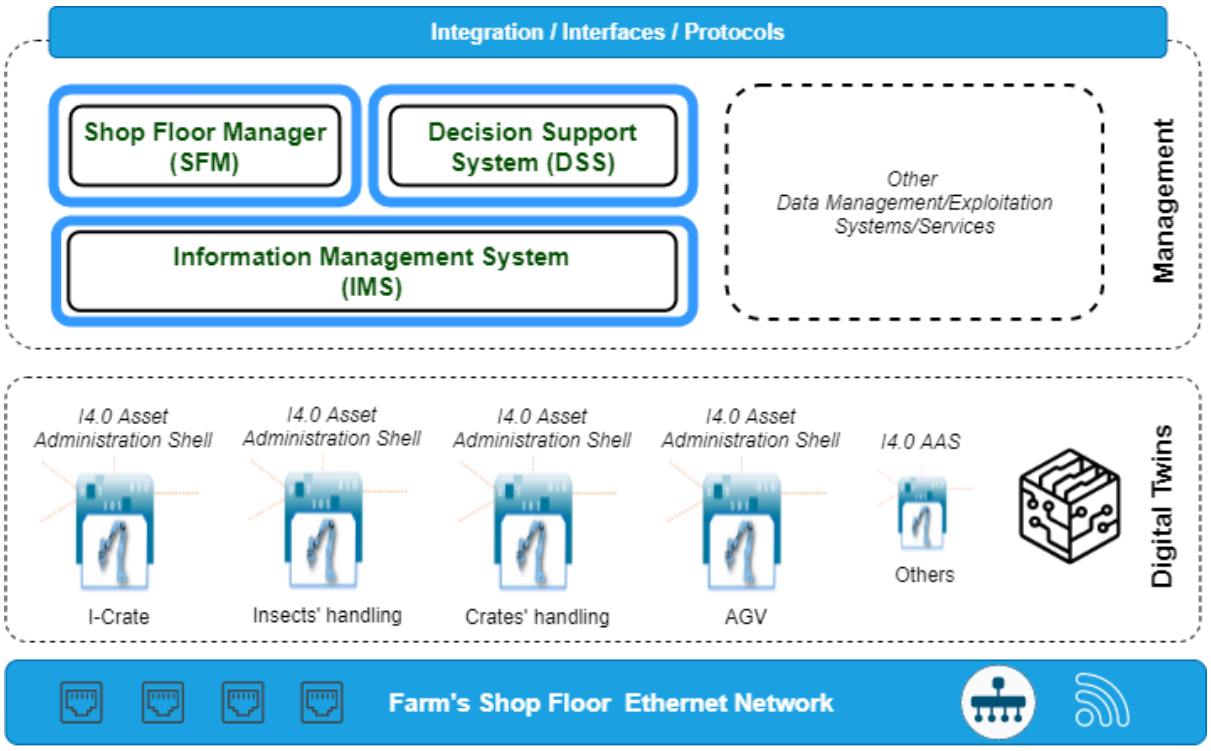


*Figure 12: CoRoSect’s Manufacturing Execution System (from D2.4 scenario set up)*

## 4.1 Information Management System (IMS): MES Interfaces

The CoRoSect IMS, as introduced in D2.3 and D2.4, is the MES component that i) gathers, stores and distributes all the information produced within the CoRoSect System, to support all processes execution and management; and ii) supports the Digital Twins defined in WP4 for the CoRoSect’s components digitalisations. All the details of its covered functionalities are given in D4.1 and D4.2 and fully compiled in D4.3 (M30). This section provides the details for the first functional implementation of the D2.4 IMS architecture mapped on top of FIWARE components (Figure 13). The selected FIWARE enablers are configured to fulfil IMS requirements.

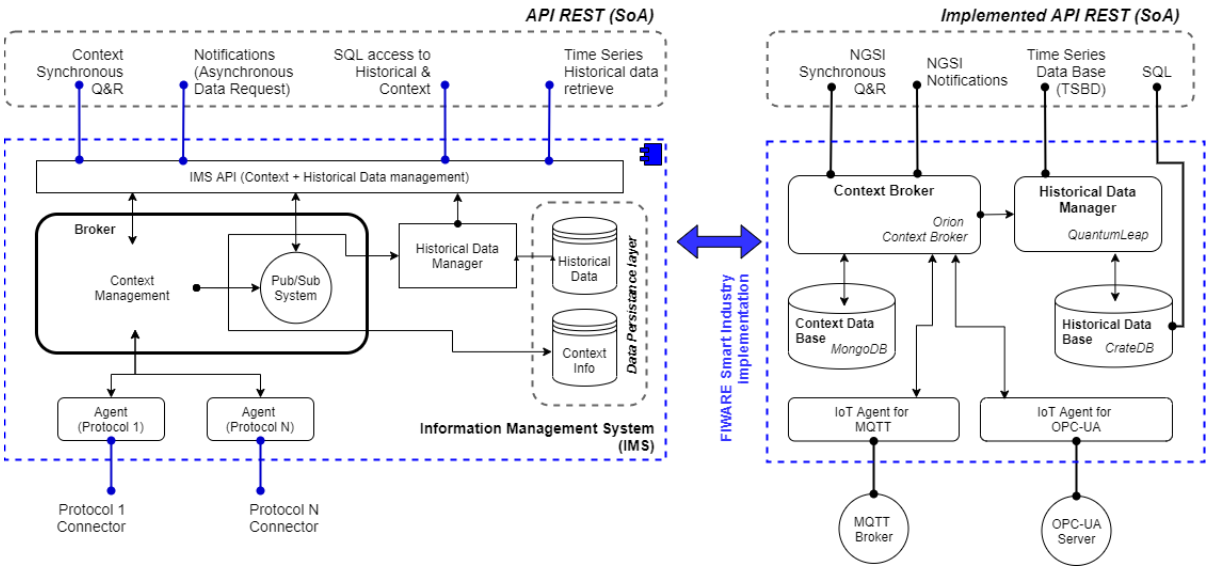


*Figure 13: IMS architecture (from D2.4) mapped in FIWARE Smart Industry enablers*

As mentioned, FIWARE enablers from FIWARE Smart Industry framework has been selected to build the CoRoSect's IMS. To present the implementation we can divide the main enablers into two main blocks, aligned with the CoRoSect's IMS scope:

- **Protocol connectors** (the southbound) provide the link between the industry common protocols and the NGSI REST standard [13]. This way, they present two sides; a) and industry proprietary interface, that directly connects with the specific manufacturer communication protocol; and b) the NGSI side used to connect with the FIWARE context broker. Its main functionality is so to collect data from the manufacturer devices, convert this into a Smart Data Model (a specific JSON structure) and send this data to the NGSI context broker using the NGSI update rest call. Advanced FIWARE IoT agents also implement the way down, receiving an NGSI notification payload with data for the device, converting this into a manufacturer protocol command and sending it to its connected device.
- **Data management interfaces** implement the northbound of the IMS providing the access to query all the gathered data, in a synchronous and asynchronous fashion, aggregated, classified, and homogenised, according to the AAS models defined by Task 9.1 (I4.0) and Task 4.3 (NGSI Smart Data Models [16]). This implementation builds interfaces for context and historical data retrieval.

The approach here described for the IMS implementation is intended to enable the integration of potentially any protocol used to communicate with shop floor devices, by developing specific agents or reuse any of the existing ones. In the CoRoSect use cases, we will develop and integrate a) MQTT systems, oriented to IoT infrastructures; and b) OPC-UA mechatronics from Industrial environments. The selection of these interfaces is detailed in D9.1 [9] and is in line with the project's objective of deploying an IIoT (I4.0) compliant system.

All the information then captured and distributed by the IMS is homogenised according to an NGSI implementation of the CoRoSect's Digital Twins. These will be mapped using new Smart-Manufacturing Smart Data Models, contributing to evolve the FIWARE framework. In turn, these Digital Twins will be managed within the IMS scope using the NGSI interface. Full descriptions of these Digital Twins defined and developed in CoRoSect, and the references to the Smart Data Models' corresponding models' subset are to be provided in D4.3- IMS Implementation (M30).

### 4.1.1 MQTT Connector

The CoRoSect's MQTT [20] connector has been developed to support the integration of native MQTT devices according to the I4.0 compliant interfaces defined in D9.1 and adapt these to the NGSI data models and methods.

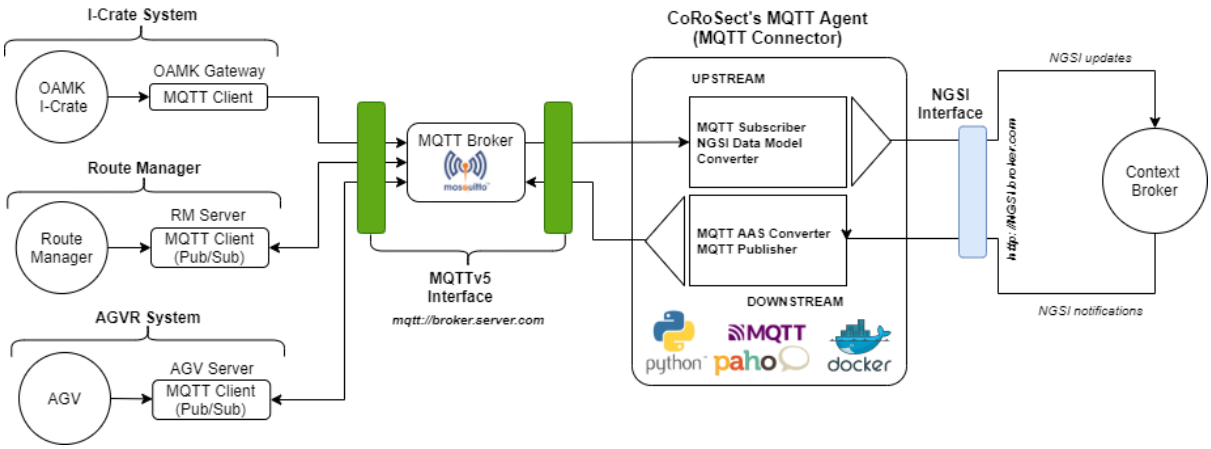


*Figure 14: MQTT's connection implementation for MQTT native CoRoSect's components.*

The CoRoSect's MQTT infrastructure (Figure 14) relies on a broker supporting **MQTTv5** protocol for commands' implementation through MQTT. CoRoSect's first version uses **Eclipse Mosquitto** Open Source MQTT broker[8] to implement this common **MQTT interface** that supports the MQTT Publish/Subscribe mechanisms.

CoRoSect's MQTT Agent (MQTT Connector) is developed using the FIWARE IoT Agent guidelines[9] to build our own Agent. FIWARE already provides an MQTT functional agent, but, due to the particularities of the MQTT Connector for I4.0 AAS, a specific agent (version 1) was designed within CoRoSect's scope.

This CoRoSect's MQTT connector has been developed using python (exploiting the paho[10] librarie) and is distributed and deployed using docker's[11] containers. Its main mission is to read the corresponding topics of each registered MQTT native component and convert the data into the corresponding NGSI Digital Twin, to , later, update accordinly the NGSI data model. This is the way it captures information from MQTT shop floor devices and drives this to the IMS. On the other way around, it receives NGSI notifications from the IMS publish/subscribe interface reated to registeres MQTT devices, converts them and fulfil the corresponding MQTT topics and publish them in the MQTT Broker. This is the basic mechanism to address commands from the Shop Floor Manager to the Shop Floor components.

This way, the MQTT Connector first version is divided into two main blocks:

- The **UpStream** code subscribes to the device topics that maps the device AAS to receive all updates from the MQTT Broker. Then, maps the data on the topic into the corresponding NGSI entity (Figure 15) of the Device Digital Twin, conforming the NGSI update payload (as a JSON file). Once this payload is checked, sends the NGSI update call to update the Digital Twin data in the IMS.

[8] https://mosquitto.org/
[9] https://iotagent-node-lib.readthedocs.io/en/latest/howto.html
[10] https://github.com/eclipse/paho.mqtt.python
[11] https://www.docker.com/

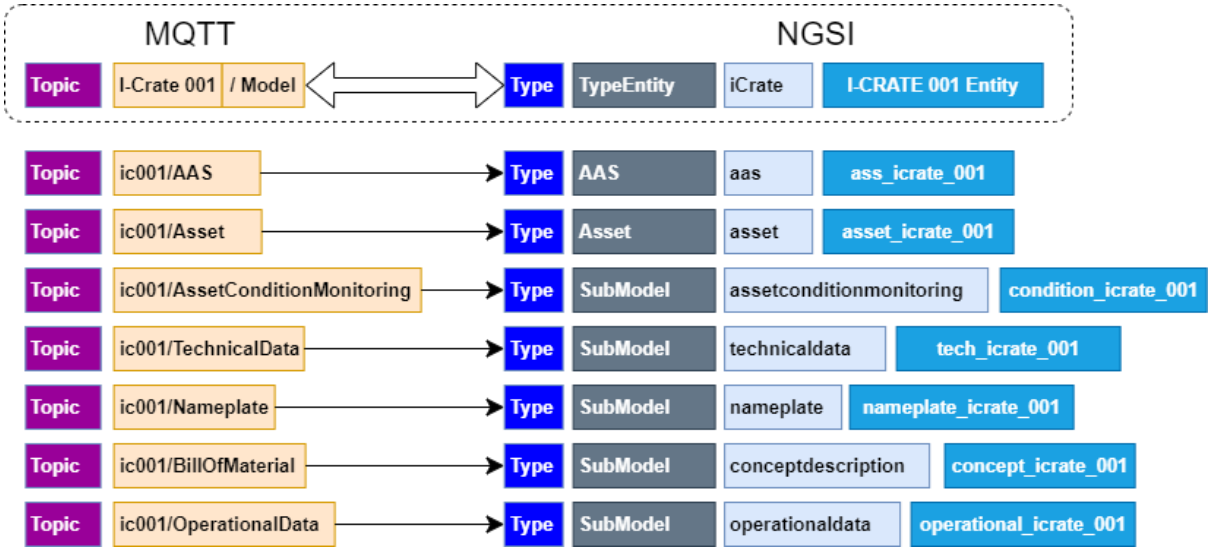


*Figure 15: MQTT's AAS mapping into NGSI entities – Digital Twin. CoRoSect's I-Crate example*

- The **DownStream** code is subscribed to the IMS Context Broker, so it receives any update on the entity that maps a concrete device command. It identifies the device and corresponding topic where to publish the received commands payload and, using MQTTv5 headers, track the command and the response. Then, by MQTT publish/subscribe method, the MQTT Broker redirects this information to the driver's controller.

For CoRoSect's system first version deployment, a dedicated instance for each MQTT integrated device is being deployed. Once its performace is tested, merging these instance in an MQTT conector single one would be analysed.

### 4.1.2 OPC-UA Connector

The CoRoSect's OPC-UA [19] connector is intended to integrate native OPC devices that implement OPC-UA compliant servers. Each of these OPC-Server will manage the Asset Administration Shell of their corresponding devices (Figure 16), according to de defined I4.0 interfaces in D9.1.

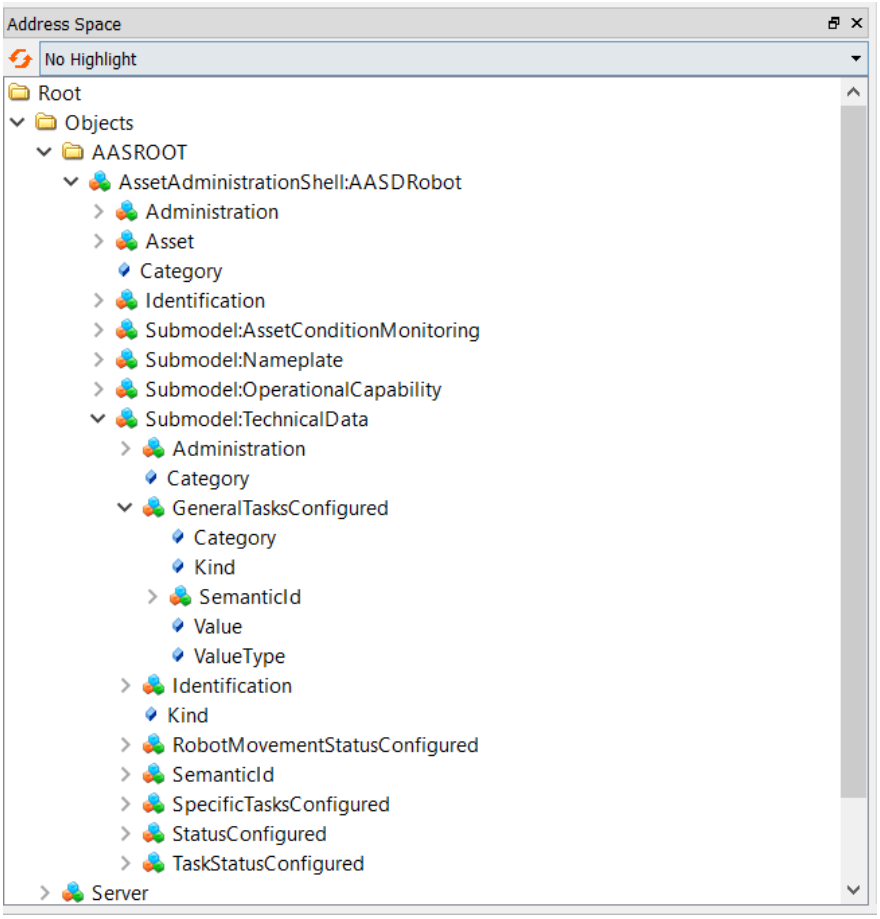


*Figure 16: OPC-UA Asset Administration Shell exposed by the OPC Server. CoRoSect's D-Robot example.*

The CoRoSect's OPC-UA integration infrastructure (Figure 17) relies on these OPC servers, exposed, each of them, through a dedicated OPC.TCP endpoint. The CoRoSect's OPC-UA agent (OPC-UA Connector) follows a development process similar to that of the MQTT connector, based on the FIWARE IoT Agent guidelines12. FIWARE framework already provides a basic OPC-UA IoT Agent[13] which works at a different level from that the one faced by the project, but in a complementary way. CoRoSect is developing its own OPC-UA IoT Agent in close collaboration with the FIWARE Foundation and the Smart Manufacturing area, relying on the RAMI4.0 Asset Administration Shell.

[12] https://iotagent-node-lib.readthedocs.io/en/latest/howto.html
[13] https://iotagent-opcua.readthedocs.io/en/latest/

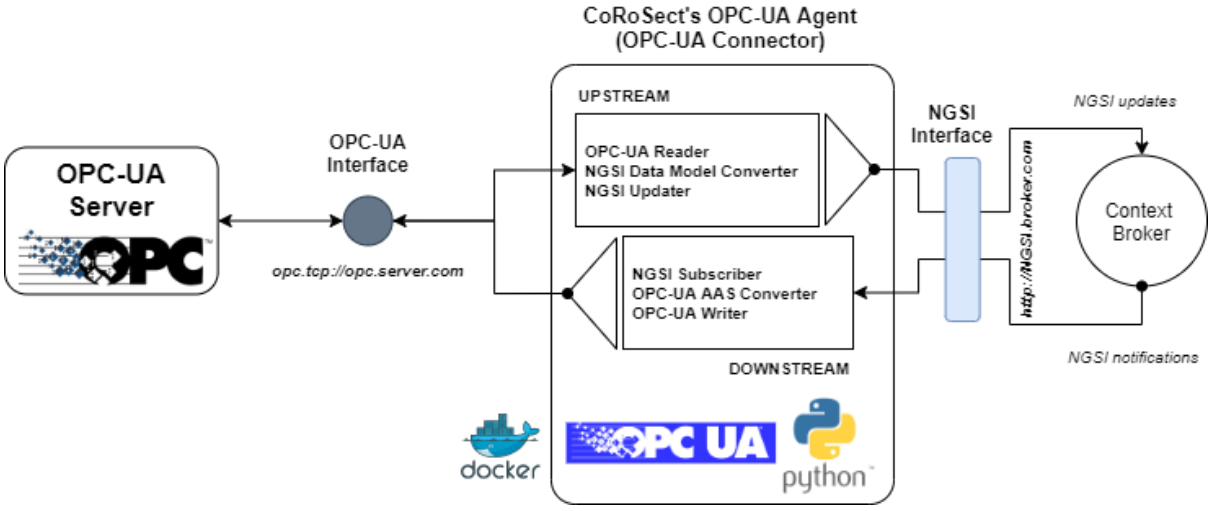


*Figure 17: OPC-UA's connection implementation for OPC native CoRoSect's components.*

This new OPC-UA connector is written in python (exploiting the Pure Python OPC-UA[14] library) and it is distributed and deployed using docker's[15] containers. Unlike the MQTT connector, this one directly connects to a specific OPC Server using OPC protocol and works through alarms and events to read the server data. Through the same connection, it directly writes on specific server's objects to send commands to the controlled device. For its design, an initial approach similar to that of the MQTT connector is followed. This is divided into two main paths:

- The **UpStream** code is aware of the server events that update the OPC objects (device AAS) to read them, adapt the new information to the NGSI Digital Twin derived from its AAS and trigger the NGSI update in the CoRoSect's IMS.
- The **DownStream** code is subscribed to the IMS Context Broker to receive any update on the NGSI's Digital Twin entity linked to a concrete command, through a NGSI notification. It reads the command's inputs and writes in the corresponding server's OPC object, transferring the command to the device controller. The response to the command is sent back to the IMS through the upstream path.

The OPC-UA Digital Twins are defined in WP4 (D4.3). For this first system's implementation, an specific instance of the CoRoSect's OPC connector is deployed for each OPC cell and OPC native integrated device.

### 4.1.3 NGSI Interface

The NGSI interface is directly provided (and supported) by the FIWARE Context broker [15]. For this first version we're implementing the Orion Context Broker[16] (Version 3.6.0) which deploys a full NGSIv2[17] interface. The NGSI protocol works based on entities and subscriptions (notifications) (Figure 18). The entities aggregates and stores attributes and values related to a physical asset (e.g. a device, a room, a robot, a sensor or an operator) and the NGSI API provides methods to create, update, modify and delete these entities and their attributes. The subscriptions implement a Publish/Subscribe mechanism to get asynchronous notifications every time an entity or set of attributes of the interest of the subscriber are updated. This implementation provides:

[14] https://python-opcua.readthedocs.io/en/latest/
[15] https://www.docker.com/
[16] https://fiware-orion.readthedocs.io/en/master/
[17] https://fiware.github.io/specifications/ngsiv2/stable/

- NGSI **Synchronous data Query/Retrieve interface**:
  - [GET /v2/entities/{}] root retrieves list of entities, specific entities, list of attributes and values that match different criteria by id, type, or pattern matching.
- NGSI **Context Data update/modify/delete interface**:
  - [POST /v2/entities/{}] root creates, append and updates entities and attributes, with new values.
  - [PUT /v2/entities/{}] root replaces attributes and updates attributes' data on entities that match specific criteria
  - [PATCH /v2/entities/{}] root updates existing entity attributes
  - [DELETE /v2/entities/{}] root removes specific entities and or attributes
- NGSI **Asynchronous data query interface** (Subscriptions)
  - GET /v2/subscriptions to list existing subscriptions
  - POST /v2/subscriptions to create a new subscription
  - GET /v2/subscriptions/{subscriptionId} to retrieve a specific subscription status
  - DELETE /v2/subscriptions/{subscriptionId} to delete a specific subscription
  - PATCH /v2/subscriptions/{subscriptionId} to update an existing subscription

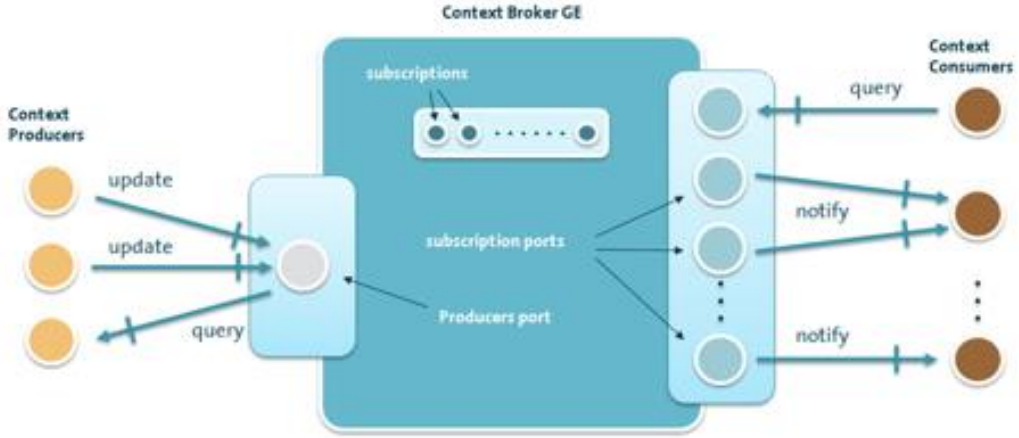


*Figure 18: Context Broker in a nutshell[18].*

The persistence layer, where all the context information, subscriptions and digital twins' structure is stored, is supported by a MongoDB[19] (documents-based data base) instance.

### 4.1.4 Eclipse BaSyx Interface

BaSyx is an open-source platform oriented to the implementation of Industry 4.0 in small and large companies interested in automation. It was born in 2016 within the "Basic System Industrie 4.0" (BaSys 4.0), funded by the German Federal Ministry of Education and Research (BMBF) [21] and currently defines an architecture of seven closely coupled component types that realize an Industry 4.0 production chain, relying on I4.0 Asset Administration Shell (AAS).

Eclipse BaSyx [22] offers an open-source middleware to implement this BaSys 4.0 architecture that covers most of the I4.0 scenarios, also within the scope of CoRoSect, such as:

- End-to-end digitisation of the production (the shopfloor)
- End-to-end connectivity between shopfloor and IT
- Peer-to-peer communication between devices and the IT

[18] https://ec.europa.eu/cefdigital/wiki/pages/viewpage.action?pageId=82773700
[19] https://www.mongodb.com/home

- Digital process models, value streams, and product tracking
- Digital twins for processes, products, and devices
- Big data analysis of production processes
- Automated tracing and documentation of production processes

This framework proposes an HTTP compliant with Industry 4.0 API, intended to interact with AAS: BaSyx Asset Administration Shell HTTP REST-API[20]. This API defines a set of REST operations for reading and modifying AAS properties (submodels and submodel elements) and to send commands (submodel element operations) to an administrated device.

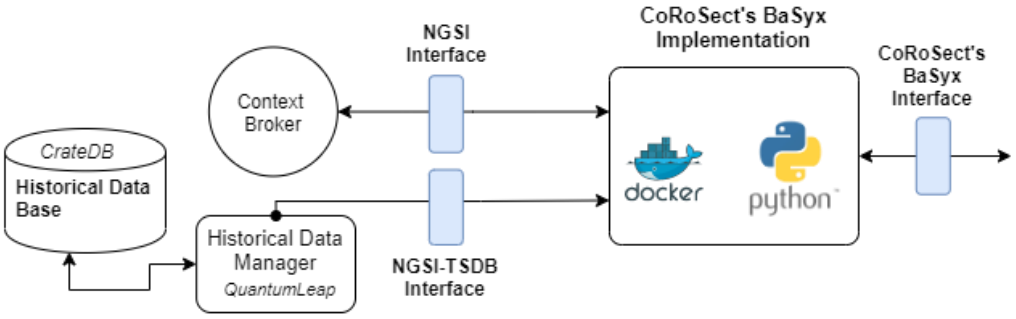


*Figure 19: CoRoSect's BaSyx interface deployment.*

CoRoSect develops and deploys its own I4.0 compliant REST API implementation, based on the methods defined by BaSyx and extending it by adding subscriptions. This implementation is built on top of the NGSI and NGSI-TSBD API RESTs, supported by the FIWARE components to work with the CoRoSect's IMS (Figure 19). The current version supports:

**AAS reading and commands** (from BaSyx API)

- [GET /aas] retrieves all AAS in IMS.
- [GET /aas/{aas_id}] retrieves AAS info identified by its <aas_idshort>.
- [GET /aas/{aas_id}/submodels/{submodelIdShort}] retrieves a specific submodel from a given AAS.
- [GET /aas/{aas_id}/submodels/{submodelIdShort}/submodel/submodelElements] retrieves a specific submodel Element from a given submodel of a given AAS
- [GET /aas/{aas_id}/submodels/{submodelidShort}/submodel/submodelElements/{idShort}/value] retrieves the value of a specified submodel element (property)
- [GET /aas/{aas_id}/submodels/{submodelidShort}/submodel/submodelElements/{idShort}/output] retrieves the oputput value of a specified submodel element (operation)
- [POST /aas/{aas_id}/submodels/{submodelIdShort}/submodel/submodelElements/{idShortPathToOperation}/invoke] sends a command (operation) to a given AAS by writing on the corresponding Submodel Element Operation

**Subscriptions management** (CoRoSect's BaSyx extension)

- [POST /aas/{aas_id}/submodels/{submodelIdShort}/submodel/submodelElements/operation/{idSh

[20] https://app.swaggerhub.com/apis/BaSyx/basyx_asset_administration_shell_http_rest_api/v1#/

ortPathToOperation}/subscription] subscribes to a Submodel Element Operation, to track commands progresses

- [POST /aas/{aas_id}/submodels/{submodelIdShort}/submodel/submodelElements/property/{idShortPathToProperty}/subscription] subscribes to a Submodel Element Property, to track status of an AAS object
- [POST /subscription] creates a custom subscription
- [GET /subscription] retrieves all the current subscriptions and their status
- [DELETE /subscription/{subscriptionID}] removes a specified subscription.

### 4.1.5 NGSI-TSDB Interface

The NGSI Time Series Data Base (TSDB)[21] interface directly interacts with the historical datasets captured and stored by the IMS, structured as temporal data records. This API is used to retrieve the temporal evolution of specific AAS element values (properties), or commands (AAS operation outputs) progresses along time.

This REST API is directly supported by the IMS' Historical Data Manager and is specified by the NGSI-LD [13] standard and implemented by the FIWARE QuantumLeap[22] enabler. CoRoSect's first version directly exposes its HTTP REST interface[23] on its V0.8.3.

### 4.1.6 SQL Interface

In addition to the NGSI and the NGSI-TSBD interfaces to retrieve information form the IMS, CoRoSect's system also offers an SQL-based REST API to access historical datasets. This REST endpoint is supported by the system's historical SGBD, here implemented by the CrateDB[24] an open-source distributed SQL database. This new API entry[25] allows the user to perform a direct SQL sentence to read the historical data and structure the outcome according to its needs.

## 4.2 Shop Floor Manager (SFM) and Decision Support System (DSS) Integration

SFM and DSS work as a unit hand in hand together, but with separation of concerns. As explained in D2.4, the SFM is responsible for the overall process management of the shop floor. Human operators interact in first place directly with the SFM. The DSS supports the SFM in decision-making in case of a necessary intervention or regarding the shop floor process flow (inclusively if pre- and post-conditions of executable tasks are met). A more detailed description of the specific functionalities/services of the SFM/DSS can be found in D2.4 and D4.2.

For the necessary integration with the IMS the **BaSyx API** [22] is mainly used (see Figure 20). The SFM invokes commands (operations) for a given asset by using the BaSyx API to execute them. The DSS receives, on the other hand, current operational data and responses by subscription from the IMS, which will then be evaluated. The DSS provides a REST API callback function for subscriptions when needed. All necessary AASs from the belonging shop floor components are retrieved through the IMS for analysis purposes. SFM & DSS will be deployed together with the IMS on the provided CoRoSect server (see Figure 6).

[21] https://ngsi-ld-tutorials.readthedocs.io/en/latest/time-series-data.html
[22] https://quantumleap.readthedocs.io/en/latest/
[23] https://app.swaggerhub.com/apis/smartsdk/ngsi-tsdb/0.8.3
[24] https://crate.io/
[25] https://crate.io/docs/crate/reference/en/5.1/interfaces/http.html

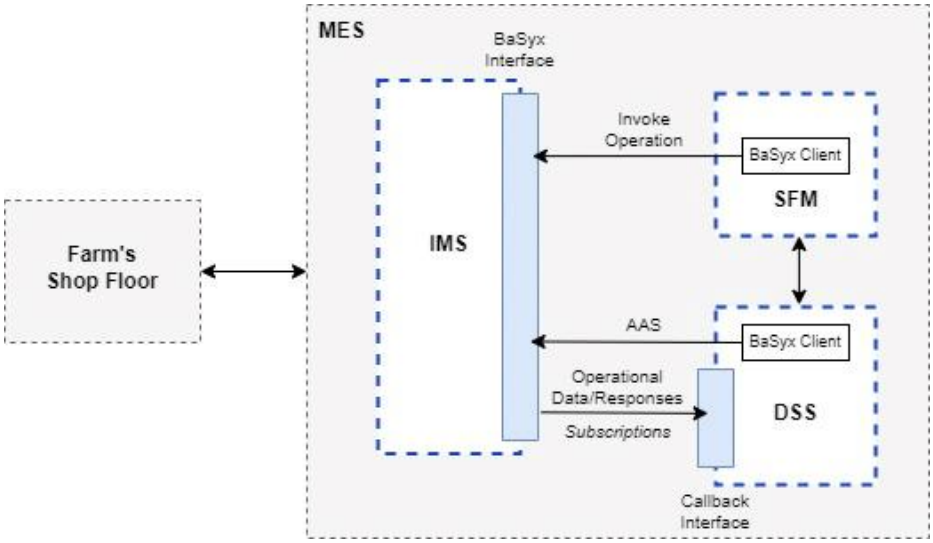


*Figure 20: SFM and DSS integration*

All use cases (see D2.1) of the insect farm are defined as BPMN configuration files (describing the farm processes), which include all necessary tasks related to a shop floor asset (see also D4.2). These BPMN configuration files are directly loaded into the DSS to start the ongoing dispatching process, but the SFM is always in full control of executing the next process steps.

# 5 Robotics' Actions planning and control

As depicted in D2.4 and addressed in WP6, the Robotics'Actions Planning and Control develops services to assist the shop floor components on specific operations, reading and exploiting data from several devices through the CoRoSect MES and actuating directly on the cell controllers. At this level, CoRoSect focuses on the handling cell's system controllers (for handling insects and handling crates). Unlike the cell controllers, which expose the interface of the managed devices, these handling cell's systems can read data from other cells (or devices) at the shop floor and execute automated actions based on the context. Also here is covered the Object's detector, to asist the route manager and the common SLAM module.

## 5.1 Handling cell's system controllers

CoRoSect project identifies two handling cells:

- The Crates' Handling cell: composed by the Stacking/De-Staking Robot (D-Robot) plus the attached devices (gripper), intended to orchestrate the specific actions to identify and move the crates.
- The Insects' Handling cell: including the Manipulation Robot (M-Robot), the Visualization Inspection module and the tools attached to the robot arm, built to implement actions for the breeding and rearing of insects within a crate.

These handling cells are being developed toguether with the corresponding cell controllers (those intended for exposing their interfaces). Due to this, the CoRoSect first release doesn't include these elements and will be implemented within the final release.

## 5.2 SLAM module

As introduced in D2.4, SLAM is used for autonomic localization and mapping of AGV's for integration or changes in areas. For detailed description of SLAM integration in current concept is described tin D6.6 Paragraph 4.

**Deployment and Integration**

SLAM is use in AGV system to decrease integration time. Initial start-up can be followed by SLAM the area and use for localization. Additional markers (Reflectors, Transponders, Magnets can be added to increase the accuracy when needed. Localization can be started after SLAM is performed. Syncing coordinate system with every power cycle will be needed to work with external systems. Benefit of the system is quick adjusting of layout with re-SLAM, starting from known area and driving to changed areas will adjust the layout.

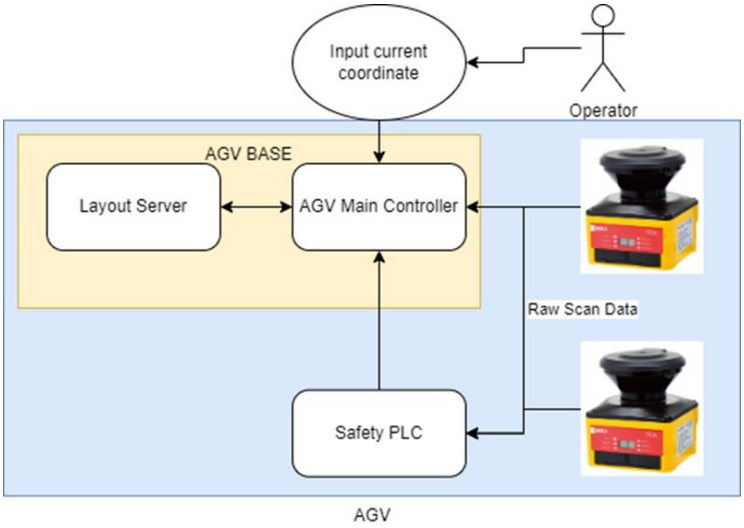


*Figure 21: SLAM module data flow and building blocks*

## 5.3 Obstacles' detector

Obstacles detector deals with the detection of static or dynamic obstacles in the shoop floor. A complete description of Obstacles Detector can be found in D6.7 Safety concept for robotic systems (planning)(M12) and spetially in D6.8 Safety concept for robotic systems (creation) due to M24.

**Deployment**

Obstacles Detector is a pure software component asociated with one or many fixed IP cameras. It will be deployed in a dedicated laptop with GPU capabilities connected to the same network as the rest of components of CoroSect. The camera should be connected to the same network in order to have its streaming available. Minimal requirement for GPU will be Nvidia graphics card 1070 model.

Camera can be physically deployed in a fixed place (in a high place with a clean view without obstacles). It needs electricity and can be connected to the network via Ethernet or Wifi.

This component interfaces via MQTT with the rest of modules. Description of the interfaces can be found in *D9.1 Integration Plan and definition of Interfaces*.

**Interfaces with the rest of the CoRoSect's components**

Obstacles detector interfaces with Route Manager. Obstacles detector sends the detection of obstacles and its projection to the Route Manager. The camera sends its streaming via RTSP protocol (Figure 22).

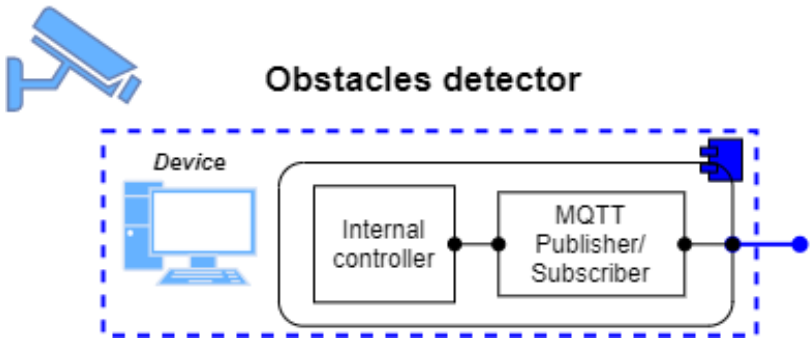


*Figure 22: Obstacles detector physical view.*

# 6 Human-Robot Collaboration (HRC) Environment

In D2.4 this layer is introduced as the components that combines data from the shop floor (connected to the IMS) to enhance its interoperability from the point of view of specific services for humans to interact with robots and vice versa. These components exploit ML/DL technologies to analyse human behaviour and develop AI models to improve learning processes (for both, human and robots); or detect possible obstacles and prevent accidents. Everything for a safer and more efficient human-robot interactions.

Within CoRoSect project, this contains part of the innovative solutions: the augmented reality simulation environment and the Routes' manager.

## 6.1 Augmented Reality simulation – HoloLens System

Microsoft's HoloLens 2 is a see-through-based augmented reality mobile device and is a core component on the CoRoSect system, regarding the situation awareness of users, during human-robot collaboration schemes.

**Covered functionalities**

A complete description of the augmented reality situation awereness module and its supported interface can be found in D9.1-Integration Plan and definition of interfaces.

**Deployment and Integration**

Microsoft's HoloLens 2 (Figure 23) receives data from the necessary components through the IMS, utilizing an OPC-UA server corresponding to an Information Model (XML format). The HoloLens 2 AAS contains information regarding its capabilities and functionalities which are exchanged between the modules and the IMS. The required data are transferred through ROS Nodes to HoloLens 2 via Wi-Fi as depicted on the figure below. The aforementioned communication pipelines also work in reverse providing information regarding the HoloLens 2 to the IMS module.

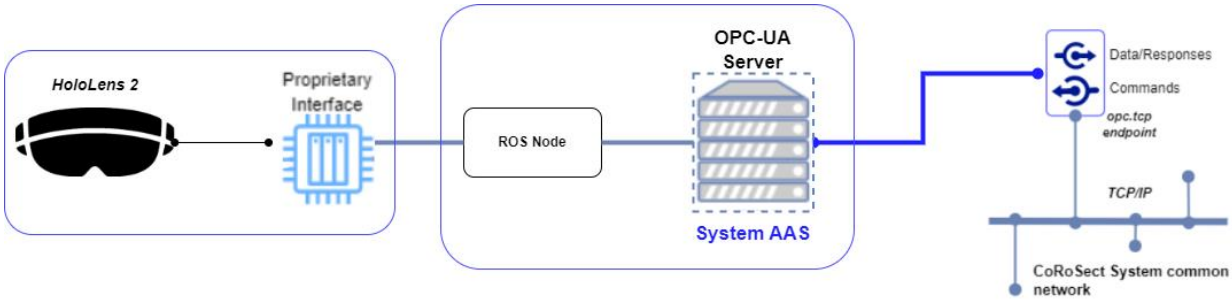


*Figure 23: Hololens (Augmented reality simulationb module) deployment*

## 6.2 Routes' Manager

This module (introduced in D2.4) deals with the creation and tracking of the routes of the robots. A complete description of Route Manager and its functionalities can be found in *D6.7 Safety concept for robotic systems (planning)*(M12) and spetially in *D6.8 Safety concept for robotic systems (creation)* due to M24.

**Deployment**

Route Manager is a pure software module. It will be deployed in a dedicated laptop with GPU capabilities connected to the same network as the rest of components of CoroSect. For obtaining a minimal latency it needs GPU capabilities from a dedicated graphical card (minimal Nvidia 1070).

This component interfaces via MQTT with the rest of modules. Detailed descrition of the interfaces can be found in *D9.1- Integration Plan and definition of Interfaces*.

**Integration with other components**

Route Manager interfaces with three other components:

- Shoop Floor Manager
- Route Manager
- AGV

*1) Shoop Floor Manager to/from Route manager*

- Shop Floor Manager will order a new Route to the route manager for a named AGV (e.g. "AGV1")
    - As an answer, we can accept (that is, the new route has been created) or reject the order (we cannot create the route requested)
- We will send a message of "route complete" when the named AGV reaches the last point of the route
- The SFM can send a message of "Cancel" for the route of a named AVG in any moment (e.g. if the status of the AGV is not "OK")

*2) Obstacles detector to Route Manager*

- Obstacles detector sends the detection of obstacles and its trajectory to the Route Manager.

*3) Routes manager to/from AGV*

- Routes manager sends the next points of the route to the AGV. The points include the orientation for the "exit" point of the next section.
- AGV sends updates of its position periodically.
- Optionally, the Routes manager can send a command of "stop" to the AGV.

# 7 CoRoSect's System Deployment

Previous sections within this deliverable present each system layer with the corresponding components deployment. These represent the approaches each developer has planned (and is following) to build and integrate its corresponding sub-system within the full CoRoSect framework. This first release (Figure 24) is used to test these development paths and prototypes, to ensure RAMI4.0 compliant interconnectivity and the proper performance of data and commands flows.

In this line, this CoRoSect's first release is focused on the interfaces' implementation and deployment, supported by the core IMS block which will enable the data processing and distribution, including the commands' flow, based on Context Publish/Subscribe mechanisms.

For the Shop floor components, and to check their corresponding AAS (interfaces) instances, each developer provides their own OPC/MQTT server/gateway implementation connected to a device simulator (D-Robot, M-Robot, I-Crate, AGV, Route Manager, etc.) that emulates the data generated by the actual mechatronic and its commands responses. These simulators' prototypes are intended to expose (and evaluate) their corresponding full interfaces, to be integrated with the IMS. However, and during the pre-pilots' sessions (M18), basic data gathering and commands paths (simplified interfaces), using TCP, OPC, MQTT and ROS protocols were tested against the real devices (D10.2 [23]).

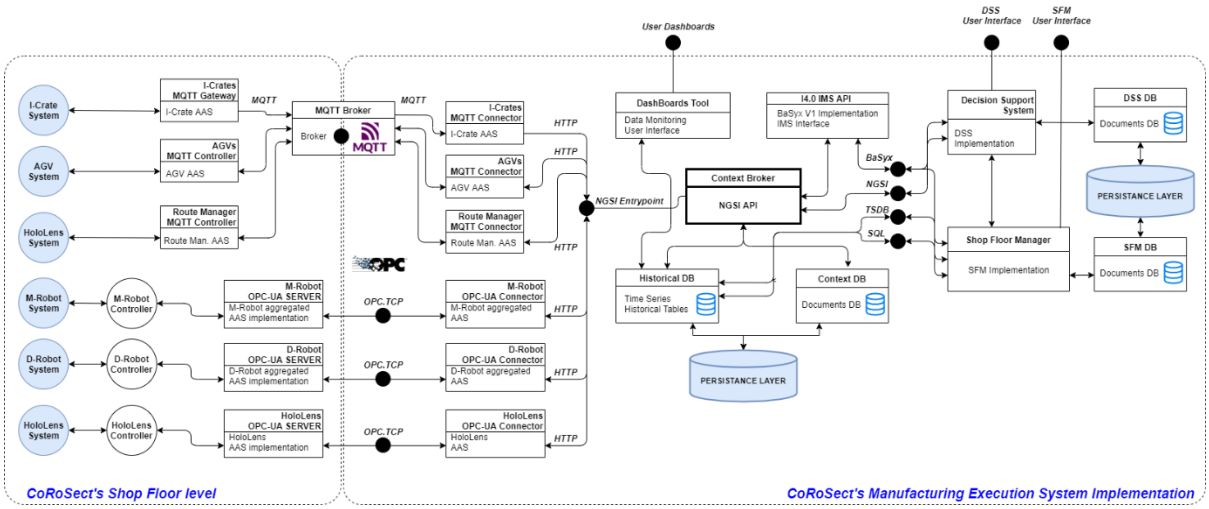


*Figure 24: CoRoSect's System deployment schema – Version 1.*

For the combined Decision Support System and Shop Floor Manager, this first release is devoted to the data collection and commands execution tests, in order to evaluate the RAMI4.0 interfaces (BaSyx and Time series queries) to collect the data from the shop floor and to send specific commands (and retrieve the responses) to seleted devices. These are the basis to implement the manufacturing processes control, which will be deployed and evaluated during incoming pilots.

Handling cells control systems, operating at OT and IT levels, are currently being developed, and will be part of the next release to be deployed and checked in pilots' premises.

All the CoRoSect components here presented are still evolving towards their final fully fuctional instances. These may involve some variations on the instantiation schemas to cover and implement all addressed funtionalites, including the hardware (robots and devices) deployment. All these modifications and refinements will be included in the CoRoSect's final release .

## 7.1 Test Environment

As one of its targets, Task 9.2 has built and provide a test environment for whole CoRoSect's developers partners to perform integration tests. This test environment uses a **Kubernetes**[26] framework deployed in a **cloud server** managed by ATOS to deploy the CoRoSect's backbone that enables the data sharing and the commands driving according to the defined models that implement the sub-systems AAS. This cloud environment allows to test and polish the integration interfaces of all the CoRoSect components, by providing a full instance of the CoRoSect Information Management System (IMS) that expose all the interfaces listed in Section 4.1.

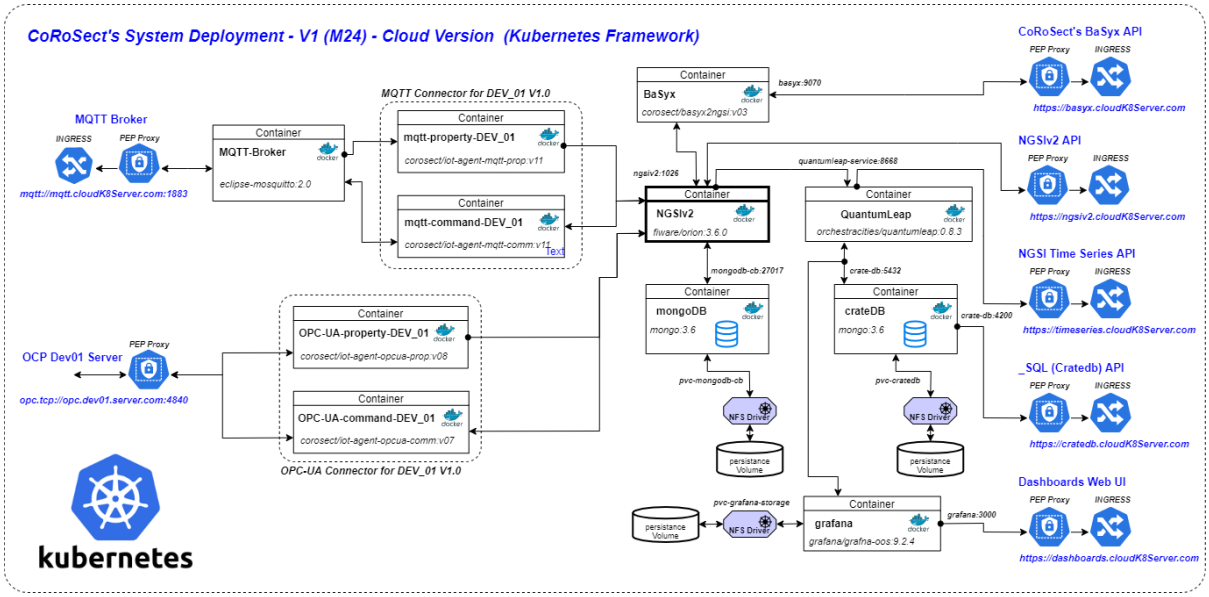


*Figure 25: Cloud deployment for CoRoSect's IMS (Test environment – version 1).*

This instance connects the Shop Floor Manager and the Decision Support System with the Shop Floor components' servers; and the Routes' manager with the AGV, creating a distributed full CoRoSect System that enables the remote development of each of its sub-systems. Its purpose is to evaluate the performance of each of the selected components of both, the could IMS first release and the sub-systems being developed within partner's premises, in terms of covered functionalities, RAMI4.0 compliance and response times. The tested configurations, analysis and modifications done within this testbed will be used to define and deploy the CoRoSect System on farms premises, to execute and evaluate the pilots (WP10).

This cloud testbed relies on docker[27] containers to instantiate the required services and components, as derived from section 4.1 and shown in Figure 25. It also includes room for the deployment of OPC simulators that helps with the OPC-UA data and commands flow refinement. Currently, D-Robot is available and M-Robot simulator is being developed. In the same line, a set of AAS instances with mock-up data that replicates the data structures of a real scenario populates the testbed datasets to enable the queries through the exposed interfaces. These structures are also connected with the simulators (remote and cloud ones) to support the commands' performance.

[26] https://kubernetes.io/
[27] https://www.docker.com/

As an important note, the security layer presented in D2.4 has been deployed, relying on KeyCloak[28] as Identity manager and OAuth server and Kong[29] as pep-proxy, to provide access control mechanisms. However, this security layer has not been activated for the sake of easier integration tests since the testbed only works with simulated data. As soon as all the interfaces are perfectly integrated, the security layer will be activated.

## 7.2 Orchestrator

As mentioned in the previous section, the cloud tested is used to obtain an optimized configuration of the IMS components (including protocols connectors), the controllers of the different Shop Floor components (OT layer) and the developed software components at the IT layer included in this first release: Shop Floor Manager, Decision Support System, Routes' manager, HoloLens component and objects' detector. These components' configurations and latest's software versions developed within CoRoSect are distributed using containers and orchestrated using Docker-Compose[30] to create a self-deployed multi-container application. This CoRoSect's Orchestrator, on its first version, is represented by a YAML file to be deployed on top of a Docker environment within the farm's dedicated CoRoSect's server.

The purpose of this first orchestrator is to easily deploy the first release of the CoRoSect System presented all along this document just to be configured with the farms' intranet parameters. The initial YAML file will include the centralised software components: IMS, SFM, DSS, protocol connectors, Route Manager, Object detector and SLAM core module. Once this is done, all the embedded cell controllers of the shop floor (D-Robot, M-Robot, AGV, HoloLens, etc.) will automatically get connected through the farm's intranet.

A first version of this orchestrator was tested during the pre-pilots' sessions in M18. An evolution of this orchestration environment, based on Kubernetes, is envisioned for the final release of the CoRoSect System.

[28] https://www.keycloak.org/
[29] https://konghq.com/
[30] https://docs.docker.com/compose/

# 8 Conclusions

This document has presented **the first release of the CoRoSect System**, built according to the Advanced System Architecture designed within WP2. This ensures that this prototype is intended to **meet the functional requirements** extracted from the project scenarios defined by the end-users (farms), as well as **the technical ones** derived from the RAMI4.0 compliance and the characteristics of the CoRoSect's Shop Floor devices. Note here that this is an instance of an **IIoT (I4.0) compliance architecture** that merges (as also remarked in D2.4) IoT infrastructures with industrial mechatronics, that demonstrates the convergence of both technologies through a common data model and common interfaces, which enable the full management of a heterogeneous shop floor (this will be expanded in D4.3).

The first release focuses on the **interconnection between the Shop Floor level and the Manufacturing Execution System** (the MES) to implement and check the protocols, interfaces and data models defined during the first half of this project, that support reading data from the CoRoSect sub-systems (real time shop floor monitoring – Figure 26) and control them (commands execution support – Figure 27). Here is implemented an I4.0 compliant system, according to CoRoSect architecture, that integrates all its components and enables the process management and data sharing required to develop the final versions of the Shop Floor Manager, the Decision Support System, the handling cells' controllers and the human and robot collaboration systems.

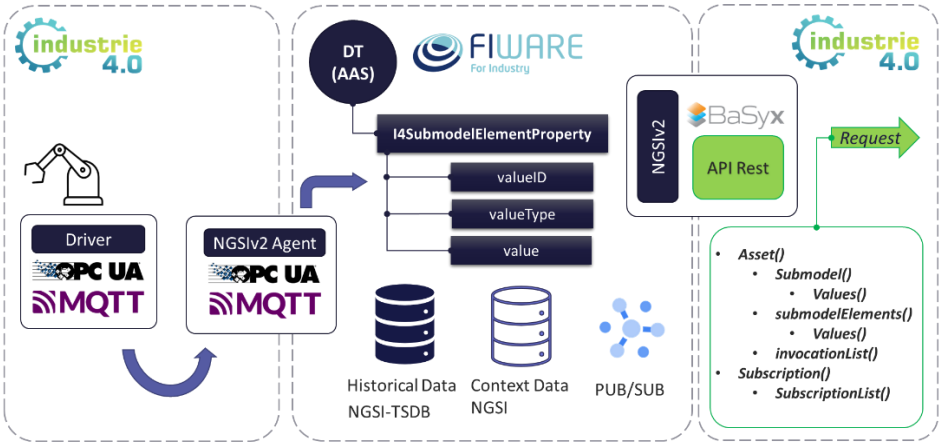


*Figure 26: Data gathering & serving from Shop Floor to MES.*

To achive this, **a testbed instance of this release is deployed in a cloud environment**, accessible by all CoRoSect's partners, to enable the integration checks, the interfaces improvement and the commands tracking. This is to foster the development and improvement of the CoRoSect processes' and cells' controlling during the second project's stage. It also supports technical partners in the development and refinement of their corresponding controllers (and information systems), to get them fully integrated for WP10 pilots.

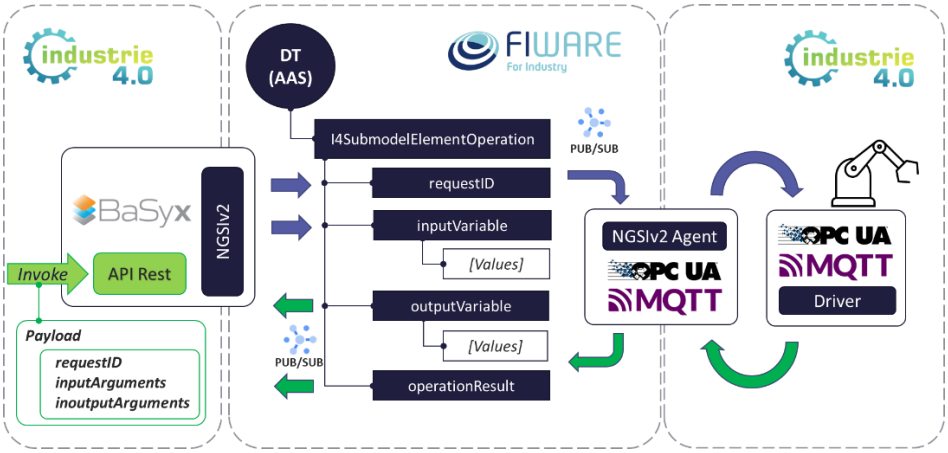


*Figure 27: Commands flow from MES to Shop Floor.*

**Next steps** involve the evolution of this first system release towards it final deployment (M34) getting feedback from the pilot's second stage:

- Supporting Task 9.3 **testings and evaluations** (M27 – M36) of the integrated components and the integration paths (RAMI4.0 compliant)
- Supporting and adapting to the **interfaces** implemented by the project's tech partners, enabling also the checking of the exposed commands
- Enabling the **evaluation of interactions between components** to properly perform manufacturing processes
- Implementing the layout for the **shop floor manager evaluation**
- Supporting **pilots' deployment and performance** in farms' premises

All this feedback, and their impact on the final CoRoSect System Realease, will be detailed in the Integrated CoRoSect Platform (final) release, in M34.

Last, but not least, this prototype has been deployed using **Open-Source components** from an existing Open framework supported by the European Commision, to implement the frameworks' core that links and interconnects all subsystems. This makes the presented deployment **easily portable and scalable**, with potential to support other smart manufacturing scenarios.

# 9 References


[1] CoRoSect, “D2.3- Initial System architecture,” European Union’s Horizon 2020. G.A. No 101016953, 2021.

[2] CoRoSect, “D2.4- Advanced System Architecture,” European Union’s Horizon 2020. G.A. No 101016953, DEC-2022.

[3] CoRoSect, “D4.2- Data analytics to obtain the prediction models,” European Union’s Horizon 2020. G.A. No 101016953, JUN-2022.

[4] CoRoSect, “D6.1- Documentation of control for handling of crates,” European Union’s Horizon 2020. G.A. No 101016953, 2021/2022.

[5] CoRoSect, «D6.2- Documentation of control for insect handling,» European Union’s Horizon 2020. G.A. No 101016953, 2021/2022.

[6] CoRoSect, “D6.4- Safety concept and control in robotic systems,” European Union’s Horizon 2020. G.A. No 101016953, DEC 2021/DEC 2022.

[7] CoRoSect, “D7.2- Report on and documentation of robot cell for handling crates,” European Union’s Horizon 2020. G.A. No 101016953, DEC 2021/DEC 2022.

[8] CoRoSect, “Report on and documentation of robot cells for object manipulation and feeding,” European Union’s Horizon 2020. G.A. No 101016953, DEC 2021/DEC 2022.

[9] CoRoSect, “D9.1- Integration plan and definition of the interfaces,” European Union’s Horizon 2020. G.A. No 101016953.

[10] FIWARE Foundation, “FIWARE | Open Source Platform for the Smart Digital Future,” 2022. [Online]. Available: https://www.fiware.org/.

[11] E. T. S. Institute and [ETSI], “GS CIM 009 - V1.4.1 - Context Information Management (CIM),” 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.04.01_60/gs_cim009v010401p.pdf .

[12] F. Foundation, “FIWARE for Smart Industry,” 2022. [Online]. Available: https://www.fiware.org/about-us/smart-industry.

[13] F. Foundation, “NGSI-LD Step-By-Step,” 2022. [Online]. Available: https://ngsi-ld-tutorials.readthedocs.io/en/latest/.

[14] F. Foundation, “FIWARE IoT Agent Framework,” 2022. [Online]. Available: https://iotagent-node-lib.readthedocs.io/en/latest/.

[15] F. Foundation, “FIWARE Orion Context Broker,” 2022. [Online]. Available: https://fiware-orion.readthedocs.io/en/master/.

[16] “Smart Data Models,” FIWARE Foundations, 2022. [Online]. Available: https://www.fiware.org/smart-data-models/.

[17] Open Robotics, “ROS - Robot Operating System,” Open Robotics, 2021. [Online]. Available: https://www.ros.org/. [Accessed 20 12 2022].

[18] PickNik Robotics, “MoveIt - Moving robots into the future,” 2022. [Online]. Available: https://moveit.ros.org/. [Accessed 20 12 2022].

[19] OPC Foundation, “OPC Unified Architecture (UA),” OPC Foundation, 2022. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/. [Accessed 20 12 2022].

[20] “MQTT: The Standard for IoT Messaging,” OASIS, [Online]. Available: https://mqtt.org/.

[21] D. T. Kuhn, “Fraunhofer- Institute for Experimental Software Engineering,” 2022. [Online]. Available: https://www.iese.fraunhofer.de/content/dam/iese/dokumente/innovationsthemen/basys_40-en-fraunhofer_iese.pdf. [Accessed 20 12 2022].

[22] F. Schnicke, “Eclipse BaSyx,” Eclipse Foundation, 2022. [Online]. Available: https://wiki.eclipse.org/BaSyx. [Accessed 20 12 2022].

[23] CoRoSect, “D10.2- Pilot preparation and planning,” European Union’s Horizon 2020. G.A. No 101016953, 2022.

[24] CoRoSect, “D4.1-Adaptive farm process modelling,” European Union’s Horizon 2020. G.A. No 101016953, 2021.

# CoRoSect

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016953