D4.2 Final version of the SHOP4CF components

Grant Agreement No.	873087
Project Name	Smart Human Oriented Platform for Connected Factories (SHOP4CF)
Work Package No.	4
Lead Beneficiary	DTI
Delivery Date	31.12.2021
Author(s)	Jeshith Damsbo (DTI), Aske Bach Lassen (DTI)
Contributor(s)	DTI, FZI, JVERNE, IFF, IMEC, PSNC, TAU. TECNALIA, TUE, TUM, UPM
Editor(s)	Aske Bach Lassen (DTI)

This Project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 873087. Neither the European Commission nor any person acting on behalf of the Commission is responsible for how the following information is used. The views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect the views of the European Commission.

Reviewer(s)	Andreas Dubourg Limkilde (FBR), Genessis Perez Rivera (FZI)
Nature	Other/Software
Dissemination Level	Public

Document Revision History

Version	Date	Modification Reason	Modified by
V0.1	15/12-2021	Initial version of the deliverable	Aske Bach Lassen (DTI)
V0.2	22/12-2021	Internal review by Consortium Partner	Andreas Dubourg Limkilde (FBR)
V0.3	22/12-2021	Internal review by Consortium Partner	Genessis Perez Rivera (FZI)
V0.4	03/01-2022	Final version of deliverable	Aske Bach Lassen (DTI)

Abbreviations

Adra	AI Data and Robotics Association
AGV	Automatic Guided Vehicle
AMD	Amendment
BPMN	Business Process Model Notation
DIH	Digital Innovation Hub
DMP	Digital Manufacuring Platforms
EFFRA	European Factories of the Future Research Association
EURADA	European Association of Regional Development Agencies
F2F	Face-to-Face
FSTP	Financial Support to Third Parties
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HORSE	Smart integrated Robotics system for SMEs controlled by Internet of Things based on dynamic manufacturing processes
IDS	International Data Spaces
IDSA	International Data Spaces Agency
IoT	Internet of Things
ISO	International Organization for Standardisation
KER	Key Exploitable Results
KPI	Key Performance Indicators
L4MS	Smart Logistics for Manufacturing
MES	Manufacturing Execution Systems
MOOC	Massive Open Online Course
PEDR	Plan for the Exploitation and Dissemination of Results
PM	Person Month
PR	Public Relations
RAMP	Robotics and Automation MarketPlace
ROS	Robot Operating System
SME	Small and Medium Enterprises
SQL	Structured Query Language
SUS	System Usability Scale
UC	Use Case
UML	Unified Modelling Language
WoT	Web of Things
WP	Work Package

Consortium Partner Abbreviations

TUM	TECHNISCHE UNIVERSITAET MUENCHEN
Fraunhofer	FRAUNHOFER GESELLSCHAFT ZUR FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V.
DTI	TEKNOLOGISK INSTITUT DENMARK

IMEC	INTERUNIVERSITAIR MICRO-ELECTRONICA CENTRUM
ED	EUROPEAN DYNAMICS SA
FBR	FUNDINGBOX RESEARCH APS
TU/e	TECHNISCHE UNIVERSITEIT
FZI	FZI FORSCHUNGSZENTRUM INFORMATIK
UO	UNIWERSYTET OPOLSKI
JVERNE	INSTITUT DE RECHERCHE TECHNOLOGIQUE JULES VERNE
PSNC	INSTYTUT CHEMII BIOORGANICZNEJ POLSKIEJ AKADEMII NAUK
TECNALIA	FUNDACION TECNALIA RESEARCH & INNOVATION
UPM	UNIVERSIDAD POLITECNICA DE MADRID
TAU	TAMPEREEN KORKEAKOULUSAATIO SA
BOSCH	ROBERT BOSCH ESPANA FABRICA MADRID SA
VWP	VOLKSWAGEN POZNAN SP Z O.O.
ARC	ARCELIK A.S.
SAG	SIEMENS AKTIENGESELLSCHAFT
ISDI	INTERNETSIA, S.L.

Executive Summary

The SHOP4CF components constitute the foundation on which the project is built. They constitute the common denominator of the FSTP projects, are tested in the Pilots and will be a central element in the sustainability of the project. This deliverable details and documents the technical development of 26 components, that will serve as the main toolbox for improving connectivity on the manufacturing shopfloor while also taking human factors into consideration.

Table of Contents

List of Tables

Introduction

This deliverable provides an overview of all the components developed by the SHOP4CF core consortium partners. It details the most relevant technical aspects of each component and is intended to serve as a detailed catalogue to be used by FSTP beneficiaries, system integrators or other organizations that wish to implement SHOP4CF components.

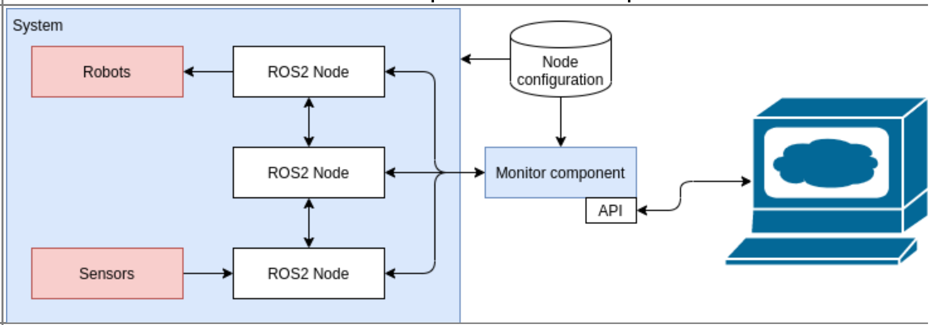
All components are thoroughly tested by the component developer themselves. However, most of the components have been tested by third parties through the Pilot projects. Table 1 shows in which projects the components have been used.

Table 1 – Overview of component use in SHOP4CF Pilots.

Partner	Component	Bosch #1	Bosch #2	Siemens #1	Siemens #2	Siemens #3	Arc #1	Arc #2	VWP #1	VWP #2
DTI	ROS2 Monitoring			x	x					
	Workcell Process Optimization based on Reinforcement Learning				?					
TU/e	MPMS		x	(x)	x				x	
FZI	FBAS-ML			x						
	DTS			x						
	F-TPT	x	?	x	x					
JVERNE (FZI)	Human Aware Mobile Robot Navigation	x	x							
TECNALIA	AR Content: Manual Editor			x						
	AR Content: Teleassistance			x				x		
	VR Creator									
UPM	Interoperability Layer through Web of Things (WoT-IL)		x				?	x		
PSNC	Predictive Maintenance (PMADAI)								x	
	Visual Quality Check (VQC)	x								
	Digital Twin for Intralogistics									x
IMEC	OpenWIFI			x	x	x				
	FLINT (M3RCP)						x		x	
	Wi-POS	?	?	x	x	x				?
	DYAMAND									
TAU	Virtual Reality Set for Robot and machine Monitoring and Training			x						
	Multi-Modal Offline and online programming solutions				x					
	Adaptive Interfaces				x					
	Digital Twin						?			
	C2NET Data collection Framework						?			
IFF	Review of risk analysis			x	x	x			?	?
	Automated Safety Approval			x						
TUM	AR For Collaborative Visual Inspection	x		x		x				

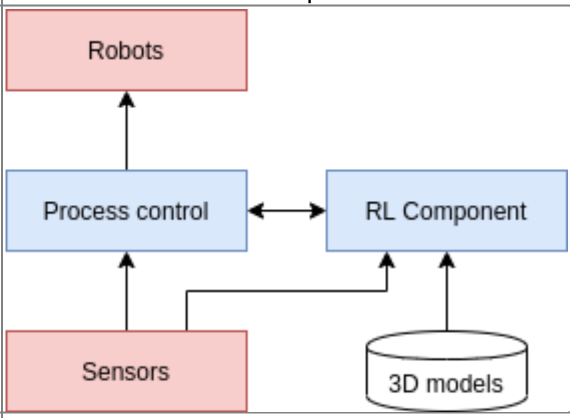
Components

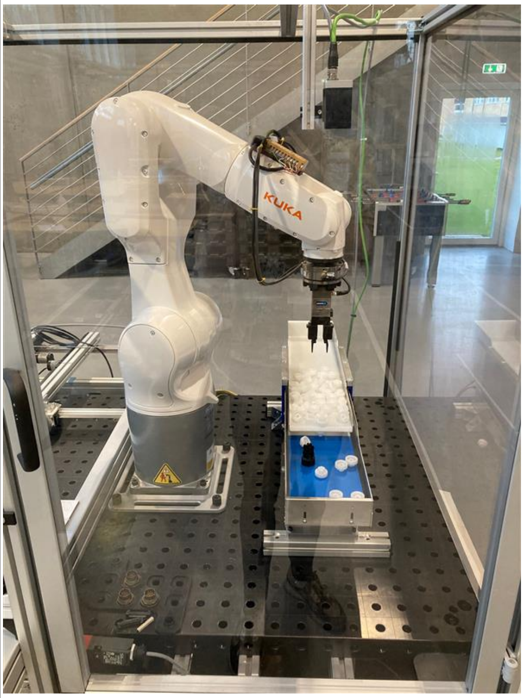
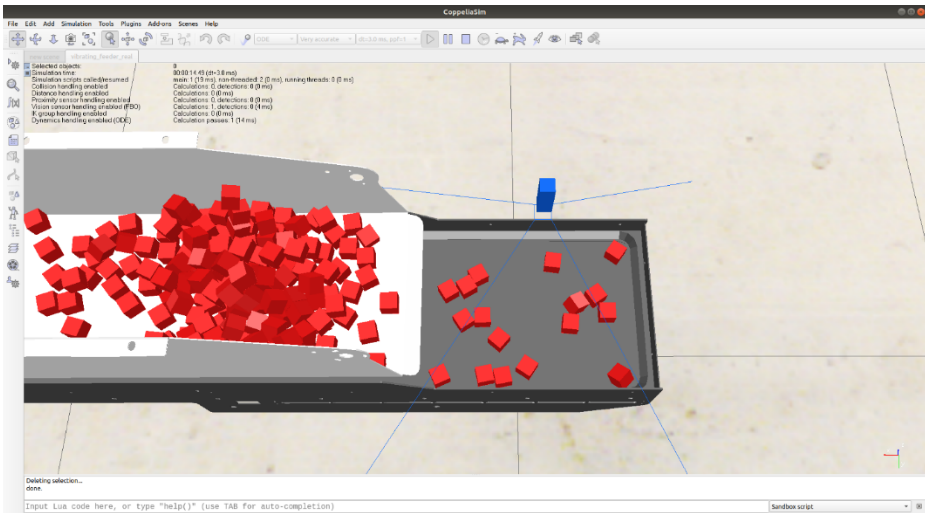
ROS Monitoring

ROS2 Monitoring tool	
Moni2	
DTI	
Application area	Supporting workers on the shop floor by providing system status of the ROS components.
Main functions	Make it easy to monitor a robot system
Interfaces and data - Input	A GUI is used to interact with the component, through the GUI the user can setup the ROS components that the user wants to monitor.
Interfaces and data – Output	Intended use for this component is to provide aggregated data to the user in a dashboard. It's assumed that no output to other components will be needed.
Functional architecture diagram	
Main non-functional requirements	The component has no real-time responsiveness requirements.
Software requirements/dependencies	Platform: Ubuntu, macOS, Windows Requirements: ROS2, Docker
Hardware requirements	64-bit system capable of running: Ubuntu, macOS, Windows and ROS2.
Security threats	The component should operate behind a firewall during production.
Privacy threats	No privacy threats have been identified.
Execution place	Private cloud/PC near production.
Deployment instructions	Deployment instructions for the component can be found on a public repository.
User interface	A dashboard showing the current status of all ROS2 nodes in the system.
Supported devices	Desktop, Laptop, SoC
User defined scenarios (non-technical) and relevant pilot cases	The component can be used to monitor a collection of ROS2 nodes.
Roles/Actors	The component is intended to be used by developers but could also be beneficial for shop floor workers.
Component Type	Web application, PC application.
Development environment	Environment: ROS2, Docker Language: Python, C++, JavaScript
TLR	The current component is TRL 7.
Component usability	The component can be used to monitor ROS2 nodes, to get their status to see if they are active or have been destroyed/deactivated. For instruction video see Instruction video section.
Versions	Free

Instruction video	https://www.youtube.com/watch?v=NbTZB01xRxA
GitHub link	https://github.com/SHOP4CF/moni2
RAMP link	https://docker.ramp.eu/harbor/projects/19/repositories/moni2 (Login required)
Docker registry	N/A

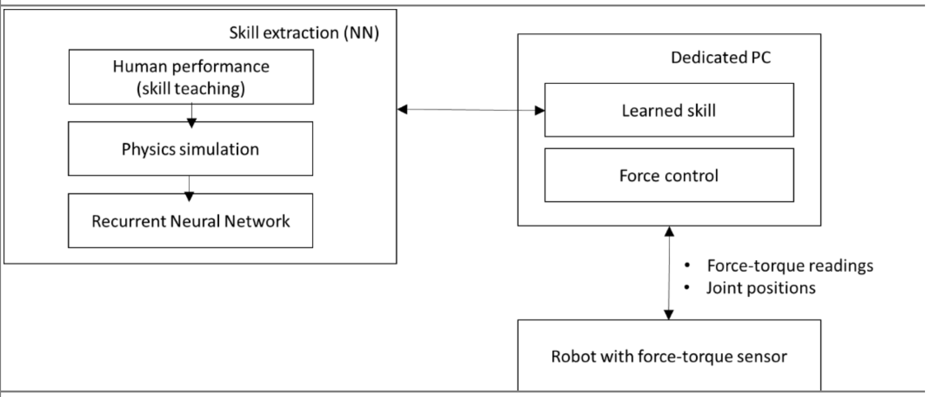
Workcell Process Optimization based on Reinforcement Learning

Workcell process optimization based on Reinforcement Learning	
Shakeit	
DTI	
Application area	The application area is within process control and long-term learning.
Main functions	The component can optimize a process based on reinforcement learning.
Interfaces and data - Input	The expected input is sensor data, e.g., robot state and images as ROS2 messages and 3D models of objects and equipment. The input is expected to be received from components in the design and execution phase.
Interfaces and data – Output	The expected output are the corrections parameters, actions, or another type of decision based on the current state of the system. The interface for these messages could be: ROS2 messages, message queues, and/or socket connections to other components on the local execution and analysis level.
Functional architecture diagram	
Main non-functional requirements	Requirements for real-time responsiveness depends on the application. However, since reinforcement learning optimize next action and not the current, real-time responsiveness requirements are relaxed.
Software requirements/dependencies	Platform: **Ubuntu**, *macOS, Windows.* Requirements: ROS2, Docker
Hardware requirements	64-bit system capable of running: **Ubuntu**, macOS, Windows. High performance PC/Cloud (good CPU and GPU) for training models. Eg: +10 core count, +64 GB ram, RTX 2080ti for training (depending on the application and model).
Security threats	When deployed on-premise a firewall should be enough. If deployed in the cloud work is required to ensure a secure connection between the cloud and the production equipment/PC.
Privacy threats	No privacy threats have been identified.
Execution place	Private cloud/PC near robot.
Deployment instructions	Deployment instructions for the component can be found on a private access repository.
User interface	The component will have multiple user interfaces: A common user interface (dashboard) for developers and end-users containing data visualization, selected actions, and other diagnostics.

	Developers will furthermore have a GUI for yaml-file system configuration and all available ROS2 tools for visualization and diagnostics.
Supported devices	Desktop/Cloud
User defined scenarios (non-technical) and relevant pilot cases	The component can be used to optimize a work cell process with reinforcement learning. Example: optimize the process control of a vibration feeder, such that that an element always is available for a robot to pick up.
Roles/Actors	Integrators, developers
Component Type	PC-application
Development environment	Environment: ROS2, Docker Language: Python, C++
TLR	The component TRL is 7
Component usability	Screenshot of real robot and current simulation   Concept video: https://www.youtube.com/watch?v=EsoY12Xj6Dw

Versions	The use of this component during and after the project is subject to the Consortium Agreement and the detailed background. Premium version.
Instruction video	https://www.youtube.com/watch?v=352dgTaCUac
GitHub link	http://robotgit.localdom.net/ai-box/applications/shakeit.git (Login required)
RAMP link	N/A
Docker registry	N/A

Force-Based Assembly Strategies for Difficult Snap-Fit Parts Using Machine Learning

Force-Based Assembly Strategies for Difficult Snap-Fit Parts Using Machine Learning	
FBAS-ML	
FZI	
Application area	Supporting human workers on the shop floor
Main functions	The component is based on a generic add-on force-control for classical industrial and/or collaborative robots. An innovative force-sensor based strategy is used to fit two or more parts together that require a snap connection. The component is a ROS based control approach
Interfaces and data - Input	Describe the inputs of the component, detailing: • Force-Torque sensor readings at the TCP: sensor mounted on the robot or already integrated (e.g. URe series) • ROS messages • Triggered by FTPT (FZI) via ROS
Interfaces and data – Output	Describe the outputs of the component, detailing: • Joint positions commands to the hardware drivers at a cycle rate of approx. 125 to 1000 Hz (robot dependent) • ROS messages
Functional architecture diagram	
Main non-functional requirements	• Low latency required • The trained assembly skills can be scaled in time and are primarily limited by the force control performance of the robot (F/T sensor)
Software requirements/dependencies	• ROS framework with ROS control (kinetic or melodic) • FZI Custom extension of ROS Cartesian Motion, Impedance and Force Controllers • FZI Custom wrappers for external robotic sensors • Robot ROS driver (e.g. ROS UR) • TensorFlow 2.1 with python 2.7
Hardware requirements	• Robot with wrist force-torque sensor mounted or integrated • Dedicated Pc (e.g. i7 shuttle PC with 8 GB ram)
Security threats	The component should run on a separate network without access to the public internet or to any other network not authorized to use it (ROS1 security)
Privacy threats	No specific privacy requirements, no personal information logging
Execution place	Local
Deployment instructions	Internal development, deployment instructions only upon (approved) request

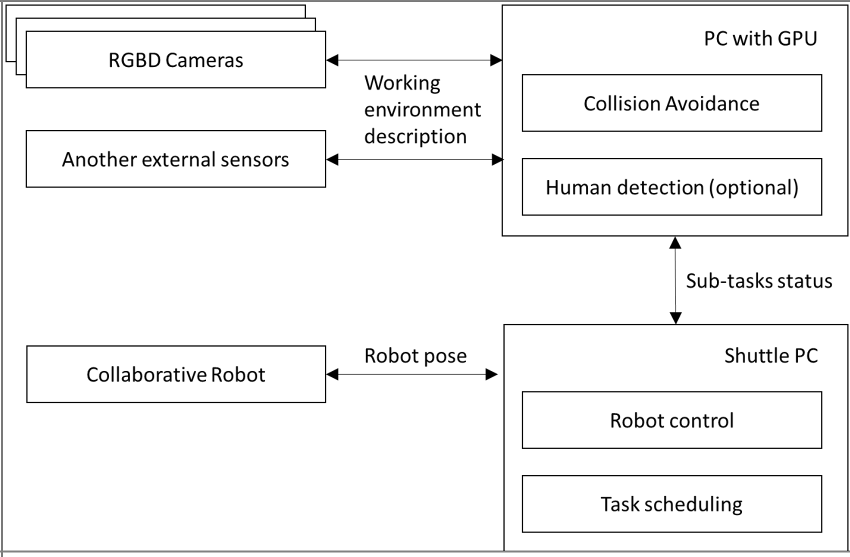
User interface	Text configuration files
Supported devices	Any robot which supports ROS control and can measure end-effector forces and torques (intrinsic or integrated)
User defined scenarios (non-technical) and relevant pilot cases	Force-based assembly tasks which require difficult snap fitting of parts by a robot Pilot cases: Siemens use case 1
Roles/Actors	Collaborative or industrial robots
Component Type	Native app
Development environment	ROS, C++ and Python (2.7)
TLR	Current TRL: 4-5 Expected TRL: 5-6
Component usability	Public videos: https://youtu.be/BX2dWxLMWeQ https://youtu.be/GNZqJz-N6NA The current version of the force controller is open source: https://github.com/fzi-forschungszentrum-informatik/cartesian_controllers 
Versions	Free version: only one deep neural network can be trained for one force-based snap fit assembly task Premium version: Extension to train as many deep NN as needed for all force-based assembly steps
Instruction video	N/A
GitHub link	https://github.com/fzi-forschungszentrum-informatik/cartesian_controllers
RAMP link	N/A
Docker registry	N/A

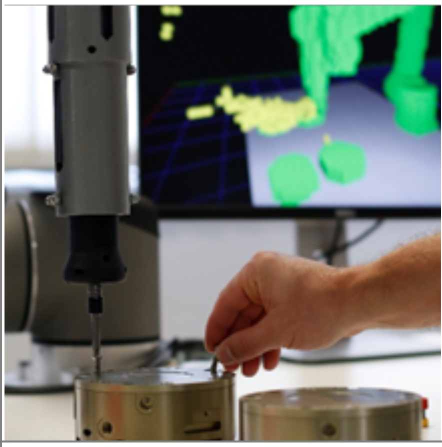
Dynamic Task Scheduling for Efficient Human Robot Collaboration

Dynamic Task Scheduling for Efficient Human-Robot-Collaboration	
DTS	
FZI	
Application area	Task manager for safe and efficient human-robot interaction
Main functions	The task scheduler: • Analyses the robot tasks and distributes them into sub-tasks, classifying them as achievable or not, to be done or already completed • Tracks if there are sub-tasks left to be completed, if they can be achieved by the robot without stopping or waiting, and which subtasks have been achieved • Avoids collision with humans and objects in the surrounds of the robots using a volume based prediction, which allows the robot to update the current goal if an unexpected object appears in the trajectory • Could be boosted with acoustic or visual augmented feedback from the robot goal and the status to the user
Interfaces and data - Input	The component expects • A complete description of the robot working environment and relative positions (with respect to the robot) • Real-time sensors data as ROS messages: depth cameras (required), RGB images (optional) and/or laser scanner data (optional) • Robot description URDF file • Real-time joint angles of the robot • ROS messages from FIWARE (using the ROS-FIWARE bridge) from the IFF safety components • triggered by the FTPT component (via ROS messages)

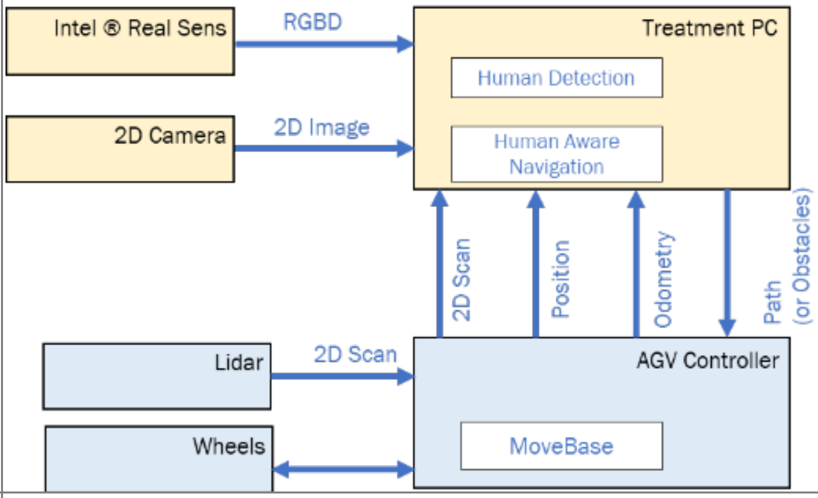
Interfaces and data – Output	The component outputs:
	• State of the robot in the working environment
	• Human pose in the working environment
	• Possible collision with objects/humans in the working environment
	• Schedule of the task as ROS actions:
	    • Reachable sub-tasks
	    • Completed sub-tasks
	    • Problems reaching and completing the sub-tasks
	• Trajectory points as FIWARE context updates

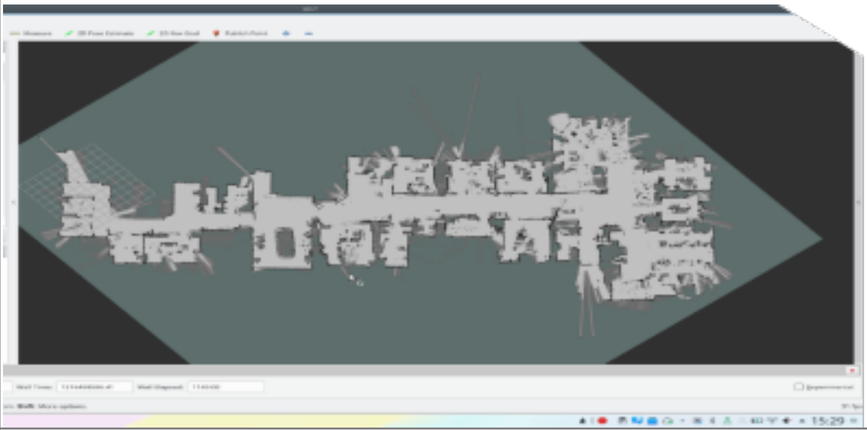
Functional architecture diagram	

	
Main non-functional requirements	• Real time responsiveness is fundamental for the task scheduling to work safely and properly. The supervision of the robot pose requires at least 10 checks per second of the environment, for the robot to accurately react if there is any obstacle on its way • Sensor information (in particular depth information) needs to be as up-to-date as possible
Software requirements/dependencies	• ROS1 Framework • GPU-Voxels • FZI Custom extension of ROS Cartesian Motion, Impedance and Force Controllers • FZI Specific Extension of the FlexBE ROS package or FZI behaviour-Tree Implementation for Task Modelling and Scheduling • FZI Custom ROS wrappers for external robotic sensors • FZI Shared workspace (ROS application of GPU-Voxels) for human-robot-collaboration • FZI Robot Collision Detection ROS package • FZI Human Pose Prediction and Tracking software (optional) • Robot ROS Driver
Hardware requirements	• (Depth) Cameras with fast update rate for the images • Combination of several sensors (one is not enough) • 1 shuttle PC for robot control with real time optimization (low latency) • 1 additional PC with GPU for more computational intense tasks (i.e. collision avoidance, human detection)
Security threats	Run on a separate network (ROS1 security) without access to the public internet or to any network not authorized to use it
Privacy threats	No specific privacy requirements. No personal information, camera or 3D data logging
Execution place	Local
Deployment instructions	Internal development, deployment instructions only upon (approved) request
User interface	Text configuration files
Supported devices	Any robot with ROS driver, URDF description and real time joint angles
User defined scenarios (non-technical) and relevant pilot cases	Efficient Human-Robot collaboration on the shop floor, where the robot needs to fulfil tasks in the proximity of the worker Pilot Use case: Siemens Use Case 1
Roles/Actors	Collaborative Robots
Component Type	Native app

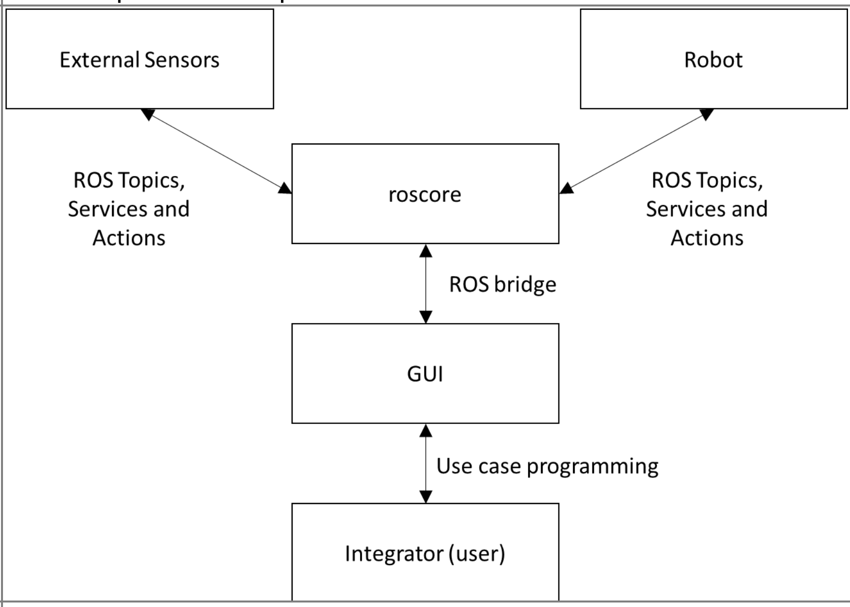
Development environment	ROS1, C++ and Python
TLR	Current TRL: 3-4 Expected TRL: 5-6
Component usability	Public video from a previous version of the component: https://youtu.be/tPZQSKHbyq8 
Versions	Free version: The component detects if a sub-task is blocked, and informs the user that the robot cannot perform its task. The robot will dynamically switch to a new sub-task if available Premium version: The component also could predict human motion and improve the dynamic of the scheduling of the sub-tasks
Instruction video	https://youtu.be/YUBNCw-e3ls
GitHub link	N/A
RAMP link	N/A
Docker registry	N/A

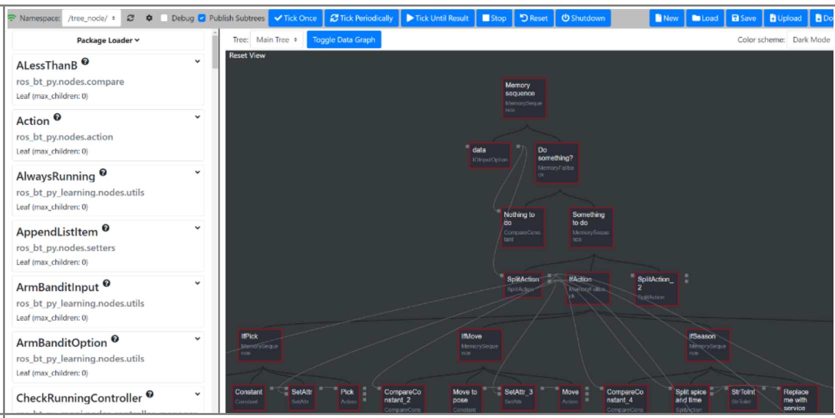
Human Aware Mobile Robot Navigation in Large Scale Dynamic Environments

Human Aware Mobile Robot Navigation in Large-Scale Dynamic Environments	
HA-MRN	
FZI	
Application area	Safety and acceptability of mobile robots
Main functions	Provides a mobile robot the capability to detect humans near its path and to adapt the trajectory according to safety and social rules
Interfaces and data - Input	• Real-time stream from camera (UB3 or GigaEthernet) • Preferably connected to a specific computer vision PC • Alternatively ROS video stream (degrade performance) • Real-time connection to lidar or ROS lidar stream • Optional GUI to define robot destination
Interfaces and data – Output	• Global Plan/Local Plan and local navigation waypoints Alternative: • Mobile Platform commands in the form of a ROS twist • Virtual obstacles for ROS MoveBase stack
Functional architecture diagram	
Main non-functional requirements	• Inputs expected at 10 Hz. Outputs between 2 and 10 Hz • Lower frequencies will influence safety and acceptability severely
Software requirements/dependencies	• ROS1 framework • Google Cartographer ROS • Move_Base and/or Move_Base_Flex ROS packages • AGV ROS driver • External ROS sensor drivers (cameras, lasers) • Open Pose • Wheel Odometry (ROS Topic)
Hardware requirements	• SICK Lidar (for example, SICK) • Intel RealSense and/or 2D camera • Dedicated PC (Intel5, High End GPU)
Security threats	Operates inside of mobile robot or secured WIFI connection (no off premises connection required)
Privacy threats	No personal information logging
Execution place	Local
Deployment instructions	Present: Internal development available on approved request Future: Public access repositories
User interface	• Text configuration files • (optional) GUI

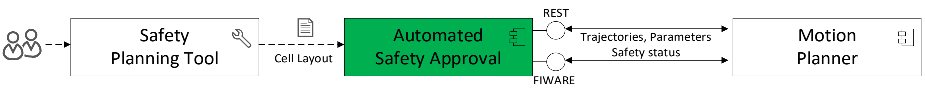
Supported devices	• Specific PC (to be embedded in a compatible AGV) • Any AGV with ROS driver
User defined scenarios (non-technical) and relevant pilot cases	Mobile Robot evolving in an industrial plant or public area with people Pilot use case: Bosch use cases 1 and 2
Roles/Actors	Mobile Robot
Component Type	Native app
Development environment	ROS1, C++ and Python
TLR	Current TRL (in combination): 4-5 Expected TRL (in combination): 5-6
Component usability	Public Video from a previous version of the component: https://youtu.be/Gac5UBkLHAk 
Versions	Free version: all described functionalities will be available on the free version
Instruction video	https://youtu.be/d3c-xlIeFxA
GitHub link	N/A
RAMP link	N/A
Docker registry	N/A

Flexible Task Programming Tool (F-TPT)

Flexible Task Programming Tool	
FTPT	
FZI	
Application area	Programming of robots
Main functions	Graphical front end (GUI) to program new robotic applications by quickly creating new control sequences based on ROS tools. The tool helps to develop or change the collaborative robotic applications, gives monitoring feedback on the status of the process and could be used to model different tasks as well as the interaction between robot and human transparently. It is an alternative to SMACH and FlexBE using Behavior Trees.
Interfaces and data - Input	• ROS topics and services (from robots, sensors) • ROS bridge from topics and services to the GUI • ROS bridge to FIWARE
Interfaces and data – Output	Describe the outputs of the component, detailing: • Feedback on the status of the processes • ROS services, topics and actions • Updates of the processes to FIWARE context
Functional architecture diagram	
Main non-functional requirements	No real time responsiveness required
Software requirements/dependencies	ROS1 Framework
Hardware requirements	A PC
Security threats	Run on a separate network (ROS1 security) without access to the public internet or to any network not authorized to use it
Privacy threats	No specific privacy requirements, no personal information logging
Execution place	Local
Deployment instructions	Internal development, deployment instructions only upon (approved) request
User interface	HTML editor to control the functionalities of the task-programming tool

	
Supported devices	GUI: Any device that allows mouse-like controls
User defined scenarios (non-technical) and relevant pilot cases	Any scenario which involves programming of robots Pilot use cases: Siemens use cases 1 and 2, Bosch use case 1
Roles/Actors	Integrators of the pilot cases
Component Type	Native app
Development environment	ROS1 and Python
TLR	Current TRL: 3-4 Expected TRL: 5-6
Component usability	Public video from a previous version of the component (HORSE): https://youtu.be/hD1vqzykLkU
Versions	Free version: all the functionalities are available in the free version under open source license
Instruction video	N/A
GitHub link	N/A
RAMP link	N/A
Docker registry	N/A

Automated Safety Approval

Automated Safety Approval	
ASA	
Fraunhofer IFF	
Application area	Programming and operating a robot
Main functions	This component is used to determine whether the chosen robot trajectory & speed is safe and the required separation distance has been chosen adequately and can be covered by the sensor configuration. (This uses a calculation of the size of the required separation distances for robots that use the operating mode speed and separation monitoring.)
Interfaces and data - Input	Static data (XML): - robot description (URDF) - geometry of tools and workpieces - sensor properties (e.g. C-value, reaction time, resolution) and poses (e.g. position of laser scanner relative to the robot), and relative to a global coordinate system (e.g. to understand what direction a laser scanner has its field of view) - safety areas resp. working areas of the robot - Dynamic data (JSON): - planned trajectories (including velocities) - current tool and its geometry - current payload and geometry of workpiece Currently, a user should use the commercially available simulation software *Visual Components* together with the *IFF Safety Planning Tool* during Design Phase to set up the initial layout (static data). This will then output the necessary information for the ASA as an XML file. The component features a REST API for retrieving static data, posting trajectories and querying the safety status of the trajectories. In addition, a FIWARE interface subscribes to changes of trajectories and updates the safety status on the context broker accordingly.
Interfaces and data – Output	The output of the component is (JSON): - True/False output info – whether new trajectories respect the safety configuration or not. This is concretely a calculation of the size of the minimum required separation distance for the specific combination of robot trajectories and safety sensors, and a comparison whether this newly calculated area extends beyond specific limits or other constraints. - If safety conditions are *not* met: output of locations where safety/working areas are violated. The safety status of trajectories can be checked via the component's REST API during Execution Phase (prior to performing a robot motion). Additionally the component can post the safety status to a FIWARE context broker.
Functional architecture diagram	
Main non-functional requirements	Trajectories should be checked as early as possible to minimize the delay of execution. Ideally, precomputed trajectories are validated in advance.

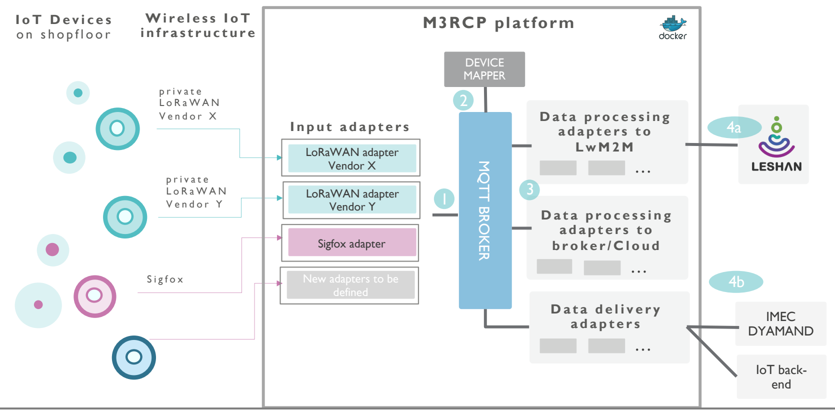
Software requirements/dependencies	Currently *Visual Components* together with the *IFF Safety Planning Tool* are required for setting up the cell layout. In the future, other tools for this process might be available. To activate all features and use optimized calculations, a valid license can be purchased from *Fraunhofer IFF*. The component runs as a Linux Docker Container on Linux and Windows hosts.
Hardware requirements	PC, no special performance features
Security threats	no known issues
Privacy threats	no known issues (no cameras, no collection or processing of personal data)
Execution place	PC next to robot cell or server. Other options possible (e.g. Private cloud).
Deployment instructions	Deployment instructions will be available on the Shop4CF Docker Registry (docker.ramp.eu)
User interface	No user interface available. The REST API is documented using Swagger.
Supported devices	PC, Docker
User defined scenarios (non-technical) and relevant pilot cases	When operating a robot cell that uses speed and separation monitoring for safety purposes, you have to check if a given trajecory is safe. If the trajectories are fixed or worst case trajectories can be defined, the operator can check them during design phase (e.g. using the *IFF Safety Planning Tool*). If trajectories can change (e.g. when using dynamic motion planing), the ASA component allows you to check if a trajectory is safe and the separation distance can be monitored by the sensor configuration. If a trajectory is not safe, the user can calculate a different one or reduce the robot's speed until all safety conditions are met. The ASA component ist utilized in the Siemens Use Case 1, where collision-free trajectories are calculated at run-time.
Roles/Actors	Robot programmer/user.
Component Type	Software service Docker container
Development environment	Docker container Written in C++ and JavaScript (Node.js)
TLR	TRL 5
Component usability	t.b.d.
Versions	*Free Version:* In the free (demo) version, the algorithm always assumes 100% payload and 100% velocity of the robot when calculating the separation distance. This results in a maximum braking distance and comparatively larger safety areas. *Licensed Version:* If a license is purchased, the algorithm uses the values for payload and velocity provided when designing the robot cell with the safety planning tool. Furthermore, it allows the adjustment of a humans approaching speed (normally this value is fixed to 1.6 m/s).
Instruction video	https://www.youtube.com/watch?v=WOvJEgOdQrI&list=PLYSJm5C1pvUVx86dXgHZx7c7enRtCeTMV&index=2
GitHub link	<not available>
RAMP link	https://docker.ramp.eu/harbor/projects/19/repositories/asa
Docker registry	https://docker.ramp.eu/harbor/projects/19/repositories/asa

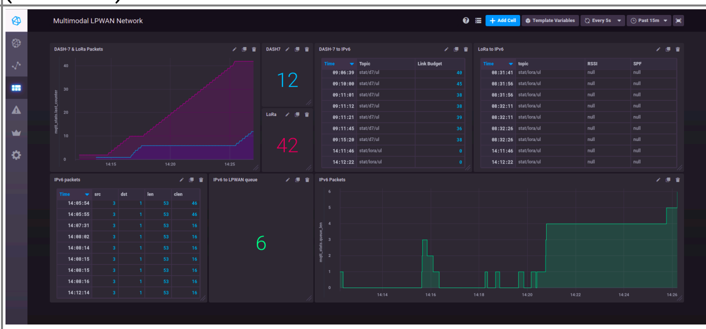
Review of Risk Analysis

Component Title/Name	
RA	
Fraunhofer IFF	
Application area	Hazard identification and risk estimation.
Main functions	The review of the risk analysis supports a safety expert in identifying hazards and estimating risk. The responsible human designer is guided through the formalized process of identifying new hazards based on identified or manually captured system changes (e.g. part changes including geometry and payload; robot changes including speed, reach, tooling; environmental changes including new tables, fencing, etc.). Application highlights where the existing risk estimation requires updates.
Interfaces and data - Input	- user input in web interface - incoming subscription callbacks for status change of Process entity on FIWARE server (if configuration monitoring is enabled, triggered by MES) - reading Process / Resource entities from FIWARE server (if configuration monitoring is enabled)
Interfaces and data – Output	The outputs of the component are: - filled out report with the risk analysis that the user can use as documentation as part of the CE process - history of changes to hazards, areas of risk estimation that need to be revisited based on changes to configuration / process - writing Process entity outputParameters attribute to FIWARE server (when communicating configuration approval to MES), includes approval status and report link - potentially structured data export to FIWARE server (not used)
Functional architecture diagram	n/a (trivial client server SPA)
Main non-functional requirements	Bidirectional network access from RA server to FIWARE server (if not used as standalone tool) Port forwarding, firewall configuration Secret injection via files or environment variables
Software requirements/dependencies	Container runtime, e.g. Docker
Hardware requirements	Server: about 50 MB free RAM / < 1GB disk space. End-user device: min. 1280x720, ideally 1920x1080 screen, modern web-browser, about 500MB free RAM
Security threats	Production application must use HTTPS reverse proxy. Secrets must be generated and injected securely. Network access should ideally be limited. Additional requirements apply depending on threat model. E.g. when using FIWARE integration and local user management with enabled self-registration, the server must be located inside secure network (on the same permissions level as FIWARE server) due to granted lateral access to FIWARE server (alternatively use authentication via identity server for managing trust).
Privacy threats	Process critical data could be part of the data sent between client and server and this could put partners' data at risk.
Execution place	Gateway, private cloud (meaning in pilot premises), cloud, etc. Unrestricted
Deployment instructions	Deployment instructions are a part of the overall documentation and are included in container and available as separate PDF.
User interface	Browser based interface with multiple tabs, available in German and English (with possibility to add additional languages). Documentation (included in container and available as PDF), video see below.

Supported devices	The server host is unrestricted. The end-user device should preferably be laptop or PC (due to size and amount of information on the screen).
User defined scenarios (non-technical) and relevant pilot cases	See "Main functions". Additionally, in the FIWARE configuration monitoring mode the RA component will track the resource assignment to the Process on FIWARE server and either automatically communicate previous approval of the configuration or facilitate review by the safety expert.
Roles/Actors	Safety expert, Robot programmer.
Component Type	• Web • Cross-Platform
Development environment	TypeScript, backend: Node.JS, TypeORM (SQlite), frontend: React, Redux, MUI
TLR	TRL 5, Fully functional, tested in SHOP4CF a pilot, not published on RAMP yet
Component usability	Usage and deployment documentation included in container and available as PDF, separate short initial deployment instructions, video demo below Demo version available on request as Docker image archive.
Versions	N/A
Instruction video	https://youtu.be/19jk7Ed0dPk
GitHub link	N/A
RAMP link	N/A
Docker registry	N/A

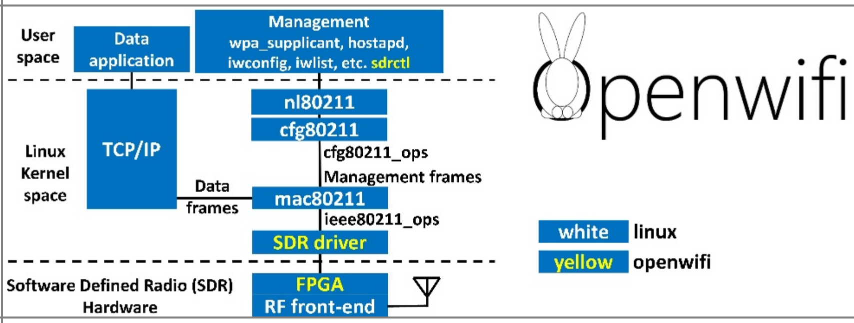
FLINT

FLINT (formerly M3RCP)	
FLINT	
imec	
Application area	The aim of the FLINT platform is to facilitate the incorporation of current/future wireless IoT devices (sensors/actuators) in a factory or shopfloor setting, as well as the required local wireless IoT communication infrastructure to connect such devices (e.g. LoRa gateways, BLE gateways). This component requires horizontal integration. At the left side, it will make use of adapters to interact with wireless IoT devices and long-range wireless communication equipment. At the right, after performing the required data transformations, it will either represent the IoT device as a LwM2M compliant device that can interface in a standardized way with a LwM2M back-end platform (for instance the open source Leshan platform) or deliver the data in a suitable format to a broker (e.g. FIWARE context broker).
Main functions	Facilitate the support of use cases that require the integration of wireless IoT devices. Driven by the SHOP4CF pilot requirements, the component was extended to also be able to take data from a database as input, translate it into the SHOP4CF data model and expose the data on the FIWARE context broker.
Interfaces and data - Input	The platform is modular, using the concept of input adapters for the exchange of data with IoT devices over wireless IoT equipment. A limited number of adapters are available today (e.g. adapter to interface with LoRaWAN network server, with Sigfox Cloud, with BLE gateway, etc.). New input adapters can be designed on an if needed basis. Minimally, an input adapter is assumed to be able to retrieve a unique device ID based on the incoming data and to interact with an MQTT broker for further data processing by the platform.
Interfaces and data – Output	Option 1: interfacing with a LwM2M back-end platform (see figure option 4a) Option 2: interfacing with a broker or IoT platform (see figure option 4b). As the platform is based on the concept of adapters, various output options can be foreseen by designing the appropriate adapters, e.g. REST-based, MQTT. Data format used is JSON.
Functional architecture diagram	
Main non-functional requirements	N/A
Software requirements/dependencies	- Dependency on data formats used by IoT devices / used wireless IoT infrastructure, which requires the 1-time design of suitable input/processing adapters. Similar dependency for output adapters in case no processing to LwM2M.

	- Docker: adapters realized as Docker containers. Implementation of adapters can be done in any language. - MQTT broker: for the information exchange between adapters - LwM2M processing adapters: dependency on Anjay, a C client implementation of LwM2M
Hardware requirements	Server/cloud platform supporting deployment/management of Docker containers.
Security threats	Currently, the internal communication between the adapters and MQTT broker is not secured. However most deployments are done on a secure company network, so the security risk should be limited.
Privacy threats	Privacy threats will depend on the type of data that is collected by IoT devices)
Execution place	Private Cloud
Deployment instructions	Deployment instructions can be found on https://github.com/imec-idlab/flint. Customization will be needed depending on the IoT devices/infrastructure to be used/deployed.
User interface	Dashboard for monitoring data received from / sent to IoT devices (see screenshot below). However, the user interface is not the core of the component, as it can operate without any UI.
Supported devices	The aim of the platform is to be extensible to support a wide range of wireless IoT devices and technologies.
User defined scenarios (non-technical) and relevant pilot cases	Industrial monitoring, asset tracking, environmental monitoring, etc.
Roles/Actors	Factory owner, workers
Component Type	• Web, • Mobile, • Native app , • Cross-Platform, • Operating System, • Other (please specify): Stand-alone platform interacting with other components.
Development environment	Runs on Linux server/Cloud platform, typically Linux-based. Programming language for new adapters can be chosen as long as internal APIs are respected.
TRL	Current TRL = 4 – Technology validated in lab, partial redesign ongoing and further developments taking place Target TRL = 5-6 Adapters have been developed to integrate with the FIWARE context broker (Orion-LD).
Component usability	 Documentation can be found on https://github.com/imec-idlab/flint
Versions	Free version is available on https://github.com/imec-idlab/flint
Instruction video	https://youtu.be/yyDOf8ukvbE

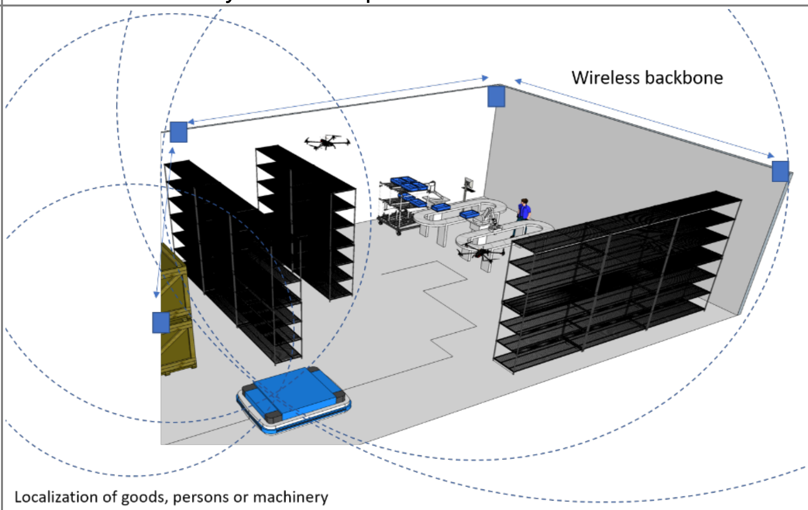
GitHub link	https://github.com/imec-idlab/flint
RAMP link	N/A
Docker registry	N/A

Open-source implementation of 802.11 WIFI on FPGA

Open-source implementation of 802.11 WIFI on FPGA	
OpenWIFI	
imec	
Application area	Supporting human workers on the shop floor by giving them real-time wireless control over aspects such as process management, interactions with robots, collecting sensor data.
Main functions	Providing low-latency network connectivity between WiFi enabled devices. Commercially available WiFi devices (e.g. tablets/smartphones or even sensors) can be used in combination with an openWIFI enabled access point, with much better control over the end-to-end latency.
Interfaces and data - Input	Input can be sensor data in one way, but also control commands in the other direction (e.g. emergency stop from a worker's tablet to a robot)
Interfaces and data – Output	Some monitoring data about the link quality can be exposed to a FIWARE context broker.
Functional architecture diagram	
Main non-functional requirements	N/A
Software requirements/dependencies	Linux OS, GNU toolchain, Xilinix Toolchain
Hardware requirements	SDR board (e.g. Xilinx ZC706 + FMCOMMS2/3/4 or other compliant board see https://github.com/open-sdr/openwifi)
Security threats	WPA2 encryption is available and should be sufficient. Of course, a network firewall is necessary.
Privacy threats	All data transmitted over the same WiFi network can be seen by all connected clients. So SSL encryption might be necessary.
Execution place	Private cloud (meaning in pilot premises)
Deployment instructions	All information and source code is available on https://github.com/open-sdr/openwifi
User interface	Developer: interact with openWIFI through linux WiFi driver (e.g. ath9k), and interface to openwifi specific components with a command line program ("sdrctl") User: openWiFi acts as a regular WiFi access point
Supported devices	All 802.11 WiFi enabled devices are supported (smartphones, tablet, laptops, embedded WiFi hardware, WiFi sensors, …)
User defined scenarios (non-technical) and relevant pilot cases	OpenWIFI could be used to interconnect wireless components in the use-cases. This could be sensors, AGVs, robots, mobile handheld devices, sensors in conveyor belts, …
Roles/Actors	Integrator: configure the openWIFI AP with the correct parameters (e.g. throughput vs range, maximum allowd delay, CSMA/CA parameters, power, bandwidth, …)

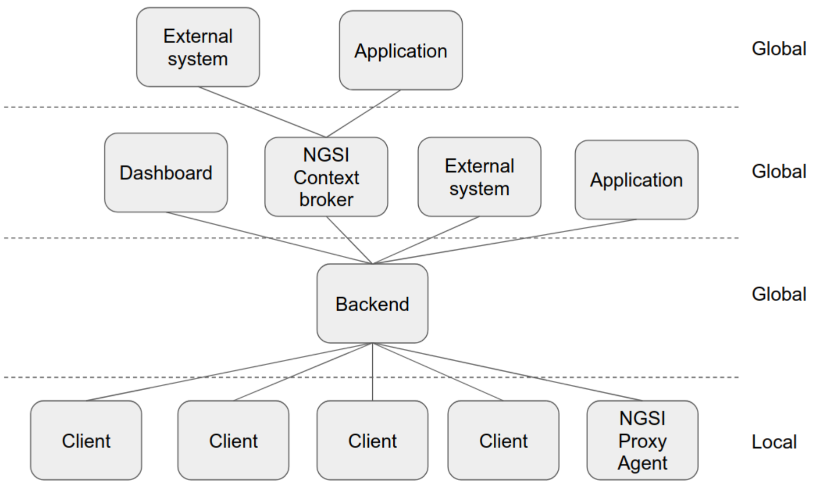
	Shopfloor workers: users of the WiFi AP
Component Type	Other (please specify): Networking equipment
Development environment	Runs on Linux and uses programming language(C & Verilog). For the Linux program & driver, the GNU tool chain is used. For the FPGA programming, the Xilinx toolchain is used.
TLR	Current TRL: TRL 4 – Technology validated in lab Working, initially tested in the test-bed. To achieve the similar level of commercial WiFi chip (regarding standard defined functionality and performance), we still need to add more features, such as MU-MIMO, OFDMA. Regarding our own add-ons, like time sensitive features, are still under development. Target TRL: between TRL 5 and 6. A light integration with FIWARE is foreseen: monitoring information about the wireless link will be exposed on the FIWARE context broker (link quality).
Component usability	https://github.com/open-sdr/openwifi
Versions	AGPLv3 open source license for basic feature set. Advanced features can be requested, but price has to be negotiated per customer.
Instruction video	https://youtu.be/p7zkkdMvPNc
GitHub link	https://github.com/open-sdr/openwifi
RAMP link	N/A
Docker registry	N/A

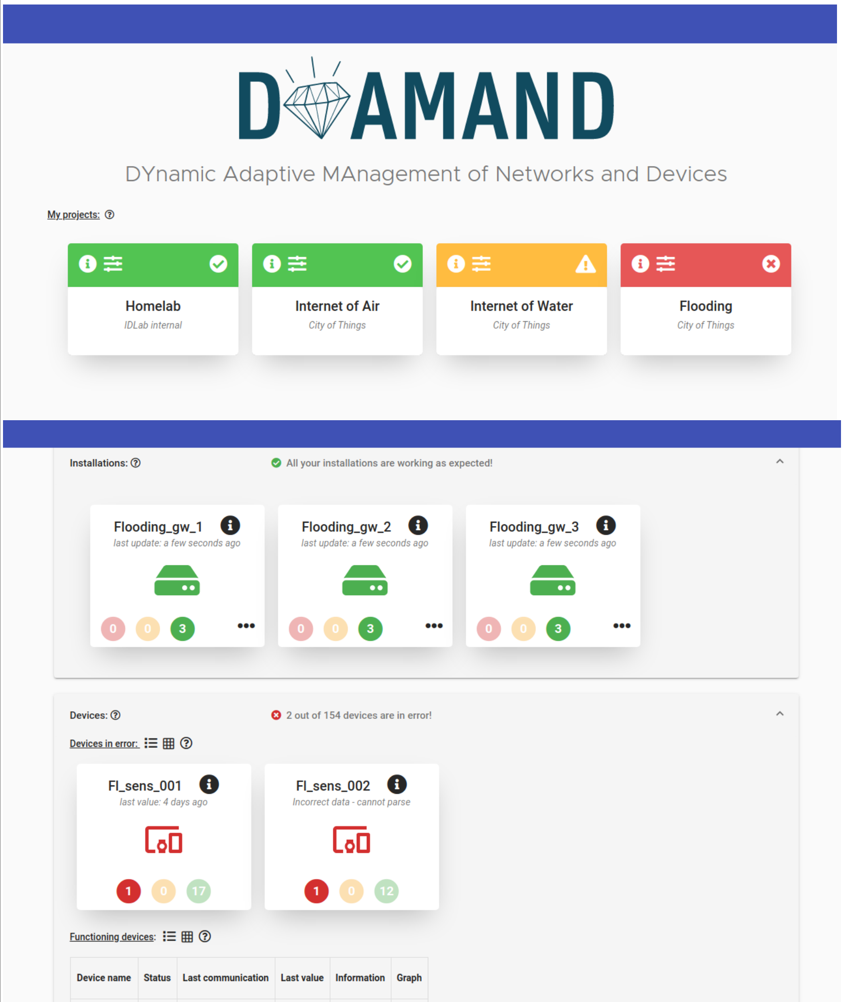
Wi-POS Indoor Localization

Wi-POS Indoor Localization	
Wi-POS	
imec	
Application area	The Wi-POS system is able to accurately determine the position of AGVs, robots or equipment on the SHOP floor. Positioning workers is also possible, but might be difficult for privacy reasons. Its goal is to enhance the safety of humans and support human workers on the shop floor (Relieving repetitive and hard tasks such as moving equipment).
Main functions	Enable automatic driving of AGVs from one conveyor belt to another. Avoiding collisions between AGVs/forklifts and humans. Defining safe zones around robots: the robot automatically stops when a human enters its operation radius. Support workers in finding important equipment on the SHOP floor.
Interfaces and data - Input	N/A
Interfaces and data – Output	Location of equipment/AGVs/robots is pushed to the FIWARE context broker so it can be used by other components.
Functional architecture diagram	
Main non-functional requirements	N/A
Software requirements/dependencies	Standalone software (full-stack) is deployed on anchor nodes and mobile tags.
Hardware requirements	Dedicated embedded hardware is needed.
Security threats	No encryption on wireless sensor network, so positions could be retrieved. The server that collects the hardware should be protected by a network firewall.
Privacy threats	If the position of humans is logged, then privacy concerns might arise.
Execution place	Private wireless network is setup by the Wi-POS system (on-site)
Deployment instructions	Deployment instructions can be obtained on request. The instructions will vary depending on the location and the use case. System should be plug-and-play.
User interface	Not available. Measured coordinates are only pushed to FIWARE context broker.
Supported devices	Only dedicated hardware (proprietary) is supported for now. Other UWB enabled hardware (e.g. new iPhone) might be supported in the future.

User defined scenarios (non-technical) and relevant pilot cases	Determining the position of AGVs on the SHOP floor to allow navigation through the factory. Locating important equipment on the SHOP floor. Defining safe zones around robots to avoid human injuries. Automated inventory management.
Roles/Actors	Factory owner, shop floor workers
Component Type	▪ Web app for visualization of positions ▪ Other (please specify): embedded sensor hardware with UWB antenna
Development environment	Development environment is Linux, using C programming language. Proprietary framework (TAISC) is used for development.
TRL	Current TRL 6. TRL 7 expected to be achieved within the project. The component has integrated support for FIWARE. It is able to push the measured coordinates to the FIWARE context broker (Orion-LD).
Component usability	Video of the system in previous deployments: https://www.youtube.com/watch?v=s2iJVm8Mhmk&t=3s https://www.youtube.com/watch?v=ScVGihe1Bec&t=2s
Versions	Evaluation license is available on request (full functionality, limited time, no source code). Commercial licenses must be negotiated. The backend code will be exposed in a docker container on RAMP. The firmware on the proprietary hardware will remain closed source.
Instruction video	https://youtu.be/penqPbonkMM
GitHub link	N/A, code will not be publicly available.
RAMP link	N/A
Docker registry	N/A

DYAMAND

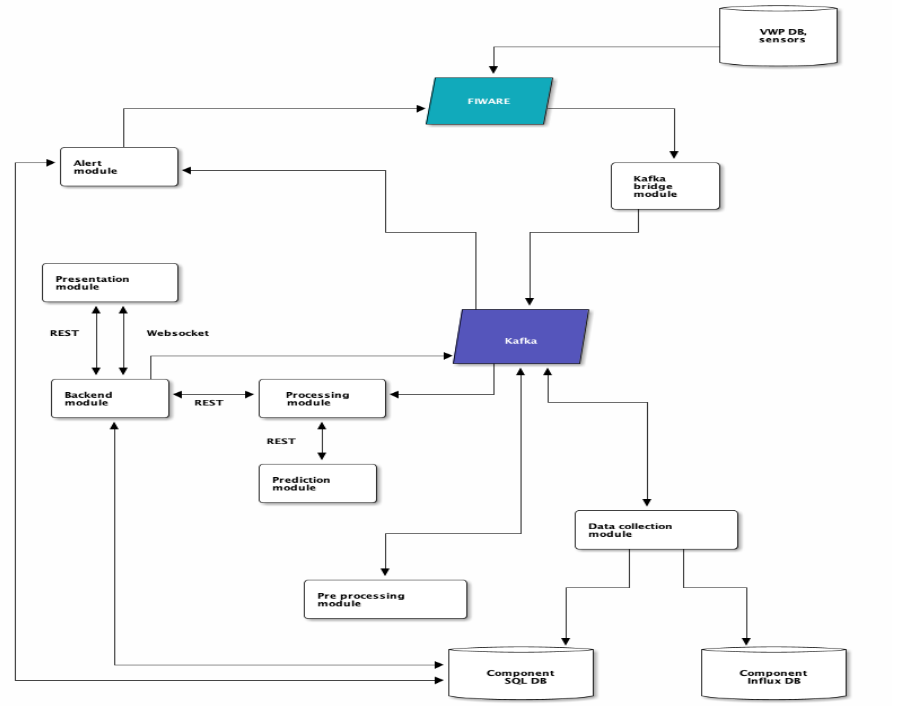
DYAMAND	
DYAMAND	
imec	
Application area	This component is a platform that groups adapters to connect different kinds of input to different kinds of output. For example, it could take input from a positioning engine (location of an AGV, in x,y coordinates), convert the data into the correct format and post it to a PubSub server (e.g. a FIWARE context broker). While the M3RCP component focuses more on adapters for wireless technologies, this component is more mature regarding FIWARE integration and provides already lots of adapters for home automation appliances (e.g. Philips Hue).
Main functions	Enabling applications on the shop floor to take advantage of all available equipment without interconnectivity issues. Monitoring of subsystems during the execution phase to help human workers identify possible issues more quickly.
Interfaces and data - Input	Providing interoperability between different systems of potentially different vendors, flexibility within production lines. Allow remote monitoring and management of systems on the factory floor.
Interfaces and data – Output	Describe the inputs of the component, detailing: - DYAMAND is flexible towards the systems that need to be integrated, creating a future-proof environment for the transition towards Industry 4.0 hence supporting both legacy and future inputs - DYAMAND client is deployed near to the systems that need to be integrated, adjusting to the interface of the system
Functional architecture diagram	
Main non-functional requirements	N/A
Software requirements/dependencies	Client: Linux OS, JVM Backend: environment able to deploy Docker images, e.g. Kubernetes is used internally
Hardware requirements	Server/cloud hardware, not specific.
Security threats	Components are best placed behind the company firewall.
Privacy threats	No specific privacy threats.
Execution place	Gateway, on-premise cloud, private/public cloud

Deployment instructions	Deployment instructions can be obtained on request. The instructions will vary depending on the location and the use case.
User interface	Client/backend do not have user interfaces. Dashboard: 
Supported devices	Client: Linux OS required Dashboard: Laptop, PC, tablet, phone
User defined scenarios (non-technical) and relevant pilot cases	DYAMAND could be used to interconnect sensors/devices/machines/robots in the use-cases to allow seamless communication between them
Roles/Actors	Integrator: use DYAMAND to facilitate integration of different systems Shopfloor worker: use DYAMAND dashboard to keep track of system's health status Application developer: develop specific applications targeted towards the specific use case
Component Type	Standalone platform interacting with different other components. Dashboard is a progressive webapp that can be used on PC, laptop, tablet, phone.
Development environment	Development of new plugins: Java Applications: GraphQL, NGSI
TLR	Current TRL: TRL 7– System prototype demonstration in operational environment Target TRL:

	TRL 8 – System complete and qualified Component is able to push to and read from various context brokers.
Component usability	Client has been validated in several national and European projects. Client has been running in a smart city/building context for several years supporting different types of projects. Publications: https://www.researchgate.net/publication/232659744_DYAMAND_Dynamic_adaptive_management_of_networks_and_devices https://www.researchgate.net/publication/310484323_Remote_Management_of_a_Large_Set_of_Heterogeneous_Devices_Using_Existing_IoT_Interoperability_Platforms https://www.researchgate.net/publication/283551391_Supporting_development_and_management_of_smart_office_applications_A_DYAMAND_case_study https://www.researchgate.net/publication/232659745_Demonstration_of_the_DYAMAND_Framework Webinars can be organised on request.
Versions	Evaluation license is available on request (full functionality, limited time, no source code). Commercial licenses must be negotiated. At the time of writing this deliverable, the DYAMAND team at imec is starting a spin-off company. The continued usage of this component in research projects is unsure at this time, but will be discussed in due time.
Instruction video	https://youtu.be/plQCsQaL8dQ
GitHub link	N/A
RAMP link	N/A
Docker registry	N/A

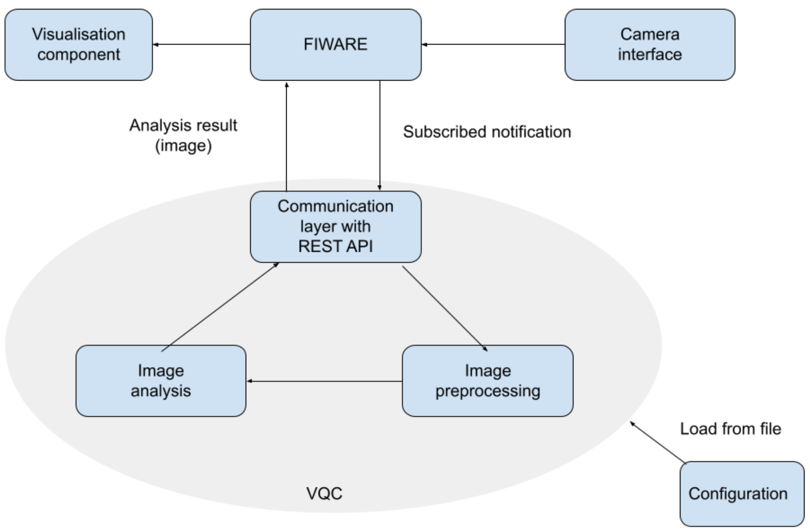
Predictive Maintenance and Anomaly Detection in Automotive Industry

Predictive Maintenance and Anomaly Detection in Automotive Industry	
PMADAI	
PSNC	
Application area	Supporting human workers at production lines.
Main functions	Supporting human workers in predicting or preventing potential failures and incidents; supporting human workers in planning services and repairs.
Interfaces and data - Input	Inputs are preprocessed sensor data (available from DURR EcoEmos MES and other systems) and additional information coming from database. These data are collected in a continuous manner. Data are passed via Orion context broker. Describe the inputs of the component, detailing: • Data are passed via Orion context broker. After reading them, they are in JSON format. The data supposed to be real-time stream data comes from external sensors. • Orion context broker is a bridge between components, with which we can communicate directly. • Data are triggered by actions happened at production line such as the appearance of a new task.
Interfaces and data – Output	The output is information about anomalous or incorrect task that runs on the production line. The alert with the information is passed to the Orion context Broker in JSON format. This information is processed by another component. The resulting process can be initiated in an application that communicates with docker images thanks to REST API and WebSocekts. Information about incorrect data is also delivered to the application. Describe the outputs of the component, detailing: • Data are passed to Orion context broker in a JSON format. • Orion context broker is a bridge between components, with which we can communicate directly. • triggered by anomaly occured during production process.

Functional architecture diagram	
Main non-functional requirements	(non) functional requirements, especially with respect to real-time responsiveness
Software requirements/dependencies	Linux, Windows, MacOS
Hardware requirements	Our app consists of several components (docker images). To use it in a comfortable style, we suggest at least 16RAM, and 60gb of disk space (in order to store OracleDB, InfluxDB, Kafka, Orion). OracleDB spaces is increasing in time in estimate way +/- 0,045MB per one operation (unstable value).
Security threats	Due to Volkswagen security politics, we exchange data between microservices with JWT token. All users, who wants to use our app, should be logged in via LDAP server. In develop mode we can use test user. All address and ports are protected. App is running in internal network without access to external network but it is not required.
Privacy threats	LDAP authentication is required.
Execution place	Everywhere when docker images can be hosted – there is no limitation.
Deployment instructions	Instruction can be found in our bitbucket repository with whole project. Provided upon request.
User interface	The application has a graphical user interface for viewing current waveforms and checking potential anomalous waveforms. The user can view, sort and filter the results. He can also report his own anomaly event if he deems it necessary. The application's capabilities are limited only to viewing waveforms and metadata resulting from current waveforms. Data is obtained from the backend using REST API and websockets.
Supported devices	Laptop, PC
User defined scenarios (non-technical) and relevant pilot cases	Currently two potential use-scenarios for this component have been identified:

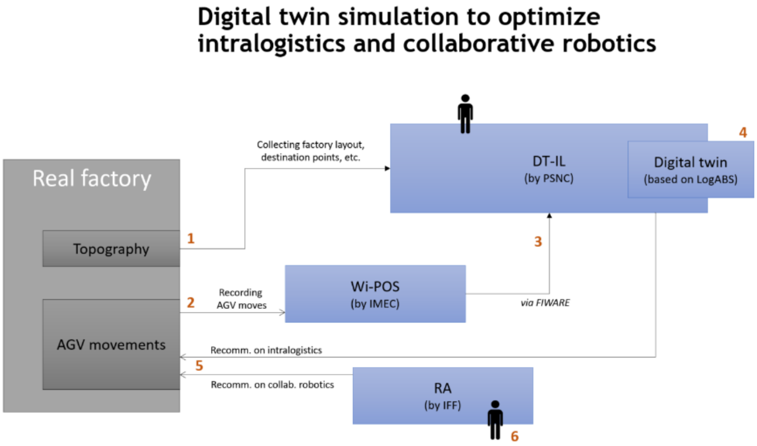
	1.Prediction of failures of a car body lift used in production process. Identification of repair time - which should result in reducing unnecessary interventions by human workers and, at the same time, in preventing future failures. 2. Prediction of repair and maintenance (e.g., cleaning) interventions in parts of the paintshop. Detection of dependencies between observed changes in measurements and quality of paint structure. Again the purpose of this scenario is to reduce unnecessary interventions by human workers and, at the same time, to prevent failures.
Roles/Actors	Expert responsible for production process.
Component Type	• Web • Cross-platform
Development environment	In pilot version component runs on a server with our PREDIO framework, Docker, TensorFlow, scikit-learn, MLflow. Development of pilot version is made mainly in Python. Fiware Orion Context Broker, OracleDB, InfluxDB are alse required.
TLR	TRL 7 achieved. Component is being tested in shopfloor area with real use case scenario. If all tested will be accomplished successfully product will be at TRL 8
Component usability	https://www.youtube.com/watch?v=OozMix2enC8&t
Versions	Product has only one version.
Instruction video	https://www.youtube.com/watch?v=OozMix2enC8&t
GitHub link	Available upon request
RAMP link	N/A
Docker registry	Available upon request

Visual Quality Check

Visual quality check	
VQC	
PSNC	
Application area	Supporting human workers at production lines by monitoring quality.
Main functions	Detecting misassembled components on PCBs.
Interfaces and data - Input	Gray scale images of PCBs – template (correctly assembled PCB) and test sample.
Interfaces and data – Output	Images with indicated places of missing components.
Functional architecture diagram	
Main non-functional requirements	Input requires template image and test sample.
Software requirements/dependencies	OpenCV, Python, Flask. The component can be run with Docker image, then no additional installations are required
Hardware requirements	A standard PC have enough computing power for this component, as no machine learning strategy was used.
Security threats	The components' API should be accessible only in local or private network.
Privacy threats	None
Execution place	PC at the operator stand (in pilot), however this can be also run on the remote server.
Deployment instructions	FIWARE must be already up and running. To deploy the component, you can run the docker image directly or go to project folder and run `docker-compose up -d`.
User interface	There is no user interface – other components should call API function.
Supported devices	In Bosch Pilot AR-CVI was used as GUI.
User defined scenarios (non-technical) and relevant pilot cases	Dedicated application for laptop or desktop PC.
Roles/Actors	Some PCBs are manufactured by human operators. Such PCBs are various types and consists of various number of components. The operator must mount all the components following the instructions. In practice they may forget to mount some of them, so this system should support the operators and indicate the places where the component was not mounted.

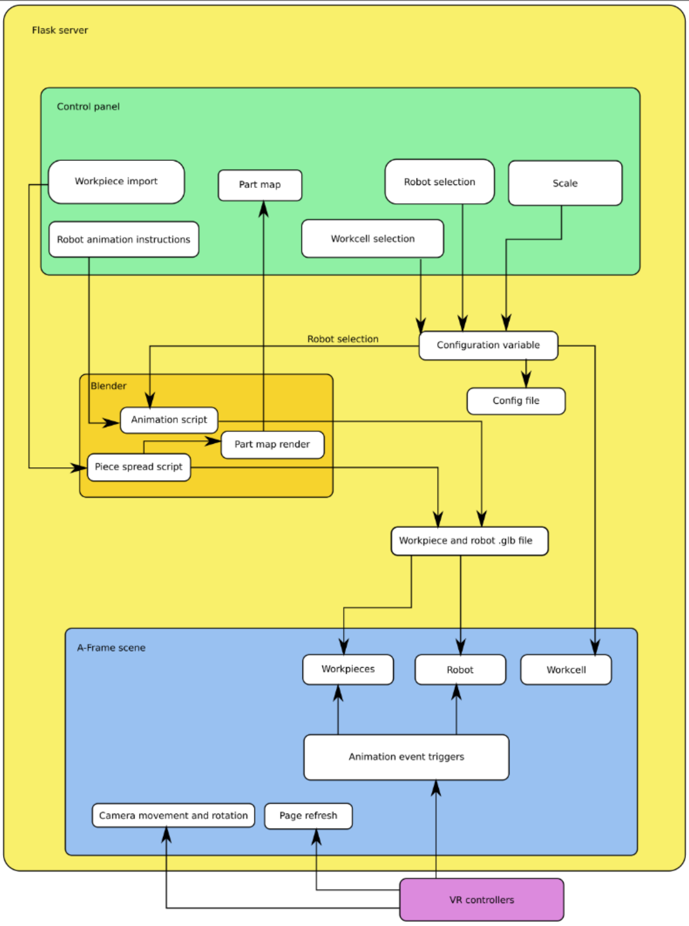
Component Type	Human operator responsible for assembling PCBs.
Development environment	Prototype – WEB. This can be accessed
TLR	Python as programming language, OpenCV for image processing and Flask as server.
Component usability	The component was developed and tested locally. The final integration at Bosch is still in progress.
Versions	Describe the different versions provided (e.g. free, premium) and the functionalities included in those versions.
Instruction video	https://www.youtube.com/watch?v=NMpNN9WCyQ0
GitHub link	N/A
RAMP link	N/A
Docker registry	N/A

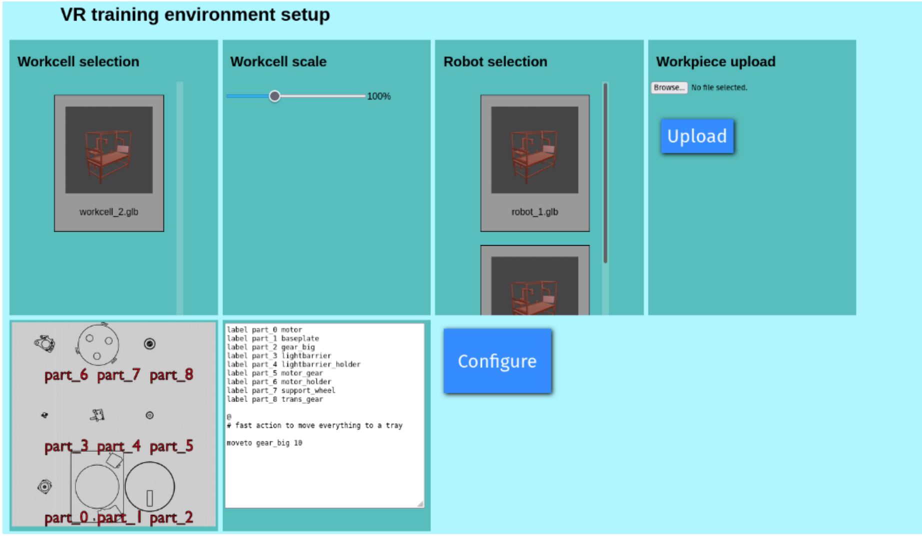
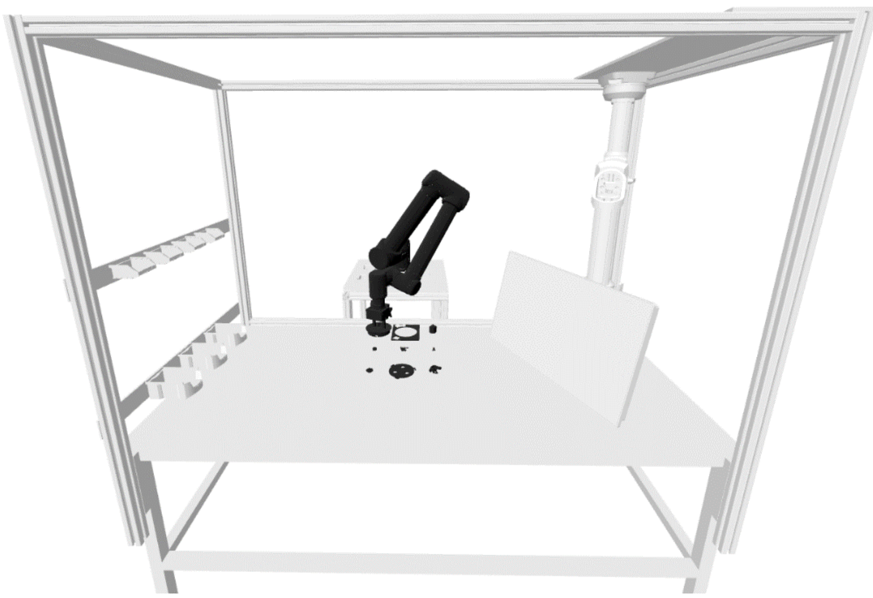
Digital Twin for Intralogistics

Component Title/Name	
Component Acronym	
Developer	
Application area	Supporting human workers on production line.
Main functions	Support intralogistics experts in designing new or modified processes and equipment for the production shop and production lines. Component will help to design altered topography for the production line, plan new AGVs and their movements to be added in the shop and finally simulate and assess the efficiency and risks of newly design or altered logistics processes. Simulation and RA are carried out by external modules or components.
Interfaces and data - Input	Input data will be collected by component's AR application and interface to AGV platform and then transformed and exported as CSV sheets in a form compatible with intralogistics simulation software "LogABS" and RA component. CSV sheets are send over REST interfaces. AR application collects the topography data form redesigned production line, AGV platform collects data describing actual movement of AGVs in the production shop.
Interfaces and data – Output	The simulation output is a visual representation of newly design or redesign logistic process. RA output is provided by a different component.
Functional architecture diagram	
Main non-functional requirements	AR (augmented reality) application precision of measured distances an location of simulated objects – we expect that AR application accuracy will be between 1cm-5cm per 5m
Software requirements/dependencies	Simulation module based on "LogABS" for Windows AR application – native app for Android or iOS
Hardware requirements	Intel i7 CPU with 16GB RAM or better for LogABS Mobile device compatible with AR Core or AR Kit for AR application
Security threats	No specific security requirements.
Privacy threats	No specific privacy requirements.
Execution place	PC and mobile device.
Deployment instructions	Instructions can be found in the code repository with whole project. Otherwise provided upon request.
User interface	Windows GUI for planning and executing logistic simulation, AR app for planning factory locations and equipment.
Supported devices	Laptop, PC, mobile device compatible with AR Core or AR Kit

User defined scenarios (non-technical) and relevant pilot cases	Planning new, safe AGV routes for redesign factory outline e. g. new product is about to enter into production and factory needs to be redesigned to maintain new storage spaces, new machinery etc.
Roles/Actors	Logistics experts planning and redesigning factory spaces and production processes
Component Type	• Mobile, • Android/iOS Native app , • Windows
Development environment	Development environment: Android/iOS, Unity 3D Programming language C#
TLR	TRL 6 – technology demonstrated in relevant environment
Component usability	Videos from pilot demonstration to be released in 2022 . Trainings and documentation provided upon request.
Versions	Component in one demo/test version
Instruction video	To be released in 2022
GitHub link	Internal repo, available upon request
RAMP link	N/A
Docker registry	N/A

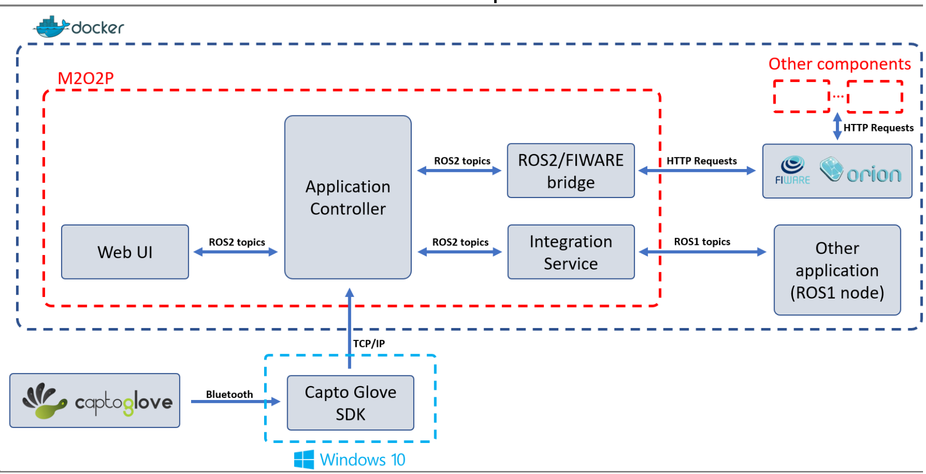
Virtual Reality Set for Robot and machine Monitoring and Training

Component Title/Name	
VR-RM-MT	
TAU	
Application area	Virtual reality-based training for workers
Main functions	Enabling humans' remote visualization and training of collaborative task with robots
Interfaces and data - Input	Dynamic mode: • ROS-messages (e.g., component location) Isolated mode: • An animation instruction file for the robot Applicable to both mode: • Controller input from the VR user • A 3D file of the workpieces in .stl or .glb format This component is positioned in the local design and execution levels. It needs horizontal communication with other components.
Interfaces and data – Output	• Visual output in the VR headset • ROS-messages This component is positioned in the local design and execution levels. It needs horizontal communication with other components.
Functional architecture diagram	

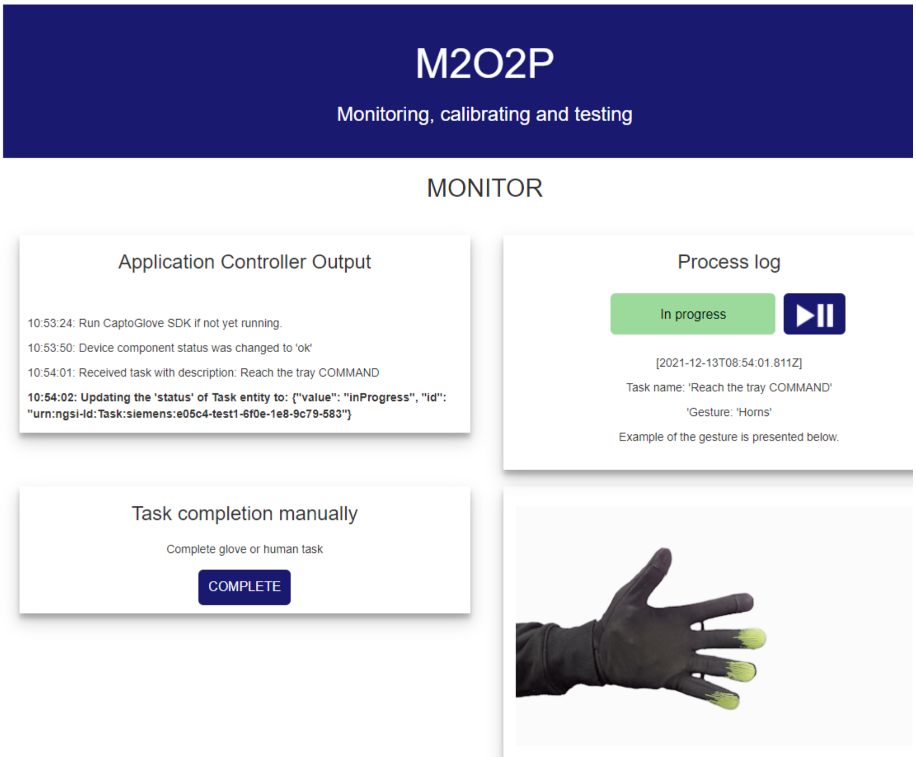
Main non-functional requirements	N/A
Software requirements/dependencies	Windows 10 and compatible browser (Firefox, Chrome, etc)
Hardware requirements	A VR headset supported by A-Frame with controller positional tracking, as listed here: https://aframe.io/docs/1.2.0/introduction/vr-headsets-and-webvr-browsers.html
Security threats	None
Privacy threats	None
Execution place	Private cloud (meaning in pilot premises), cloud
Deployment instructions	Provide information on where deployment instructions for a ready component can be found (e.g. on public or private access repositories or on websites or only upon request, etc.)
User interface	  The above screenshots show the main configuration page and a sample workcell layout in VR mode.
Supported devices	Desktops, Laptops

User defined scenarios (non-technical) and relevant pilot cases	This component could be used to train workers in a collaborative assembly process by virtualizing the whole procedure in VR and allowing the worker to interact with the robot and components prior to working in the actual setup. It is important to keep in mind that since this application is meant for training, having a concrete step by step process is required to design and fully benefit the collaborative training.
Roles/Actors	Shopfloor workers
Component Type	Web, Native app
Development environment	Runs on a backend based on Flask, a common Python library for http servers. Animations are generated via the Python API in Blender. The VR environment uses A-Frame, a web VR technology. The Dynamic mode will receive instructions via ROS.
TLR	Current TRL: 4 Expected TRL: 5/6 The component is functional however certain changes may have to be done for communication with ROS based systems and potentially FIWARE based systems, depending on the use case.
Component usability	Documentation will be provided on the Git page.
Versions	Free version: Covers all the core functionalities (setup, workpiece import, simulation) with standard set of predefined work cell and robot Premium: Additional work cell layouts and robots to choose from are under consideration
Instruction video	Link to new video will be made available on Git and project web pages
GitHub link	The git page link will be made available soon.
RAMP link	Waiting on RAMP docker registration. Will be updated once pushed to RAMP
Docker registry	N/A

Multi-Modal Offline and Online programming solutions

Component Title/Name	
M2O2P	
TAU	
Application area	Enables multi-modal human-robot interaction method for controlling robot or any other applicable device.
Main functions	Gives operator natural and intuitive input method for completing operator related tasks. CaptoGlove sensor glove is used as the input device. M2O2P will change the sensor data to states and so on to gestures. These gestures act as commands or as way to proceed in a process. In addition, M2O2P provides Web User Interface for monitoring tasks, calibrating the glove, and testing the glove and the gestures.
Interfaces and data - Input	M2O2P gets input data from the CaptoGlove using TCP/IP connection. Data is formatted as a string and holds raw sensor values from the glove. This data stream is continuous. M2O2P also gets data from FIWARE using subscriptions. This data is used to filter the available gestures. This component is positioned in the local design and execution levels. It needs horizontal communication with other components.
Interfaces and data – Output	M2O2P provides multiple possible output methods. M2O2P is ROS2 based component which means that one output method is ROS2. This output can be accessed from ROS2 topic "/command_id". Using Integration-Service, it is possible to bridge the information from ROS2 to ROS1. ROS1 topic has the same name as the ROS2. These are both "direct connection" methods. For FIWARE connection, ROS2-FIWARE-bridge provides functionalities to publish and subscribe to ROS2 topics and use HTTP requests to communicate with Orion Context Broker. Output is sent when there is gesture to be made, and operator is doing the right gesture with the glove. This component is positioned in the local design and execution levels. It needs horizontal communication with other components.
Functional architecture diagram	
Main non-functional requirements	N/A

Software requirements/dependencies	For the CaptoGlove, there should be Capto Suite installed. Docker installed on the host machine
Hardware requirements	20GB Hard drive space, recommended 2GB RAM
Security threats	None
Privacy threats	None
Execution place	CaptoGlove SDK is ran on host Windows PC and all the other parts of M2O2P are docker images
Deployment instructions	Instructions of application are provided in PDF format, later on video too.
User interface	User interface for the component is mainly in Web UI of the component. This can be reached from host machine navigating to *localhost:54400* on browser. Pictures of the UI:



CALIBRATION

Change thresholds for fingers if the application don't recognize your bent/straight fingers or it recognizes them too easily.

You can also return the original limits or update the original limits by overwriting the volume (original_limits.txt)

	Straight threshold	Bent threshold	Current values
Thumb	+100 -100	+100 -100	[1000, 800]
Index	+100 -100	+100 -100	[1600, 850]
Middle	+100 -100	+100 -100	[1600, 500]
Ring	+100 -100	+100 -100	[1800, 800]
Pinky	+100 -100	+100 -100	[1900, 900]

Return the original limits Update the original limits

Restore the original limits from backup

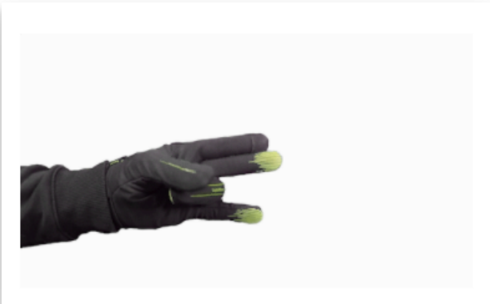
TEST

Activate and deactivate testing mode with the buttons below.

When testing mode is activated, you can change the desired gesture, which will show you the desired states of fi try to replicate the states.

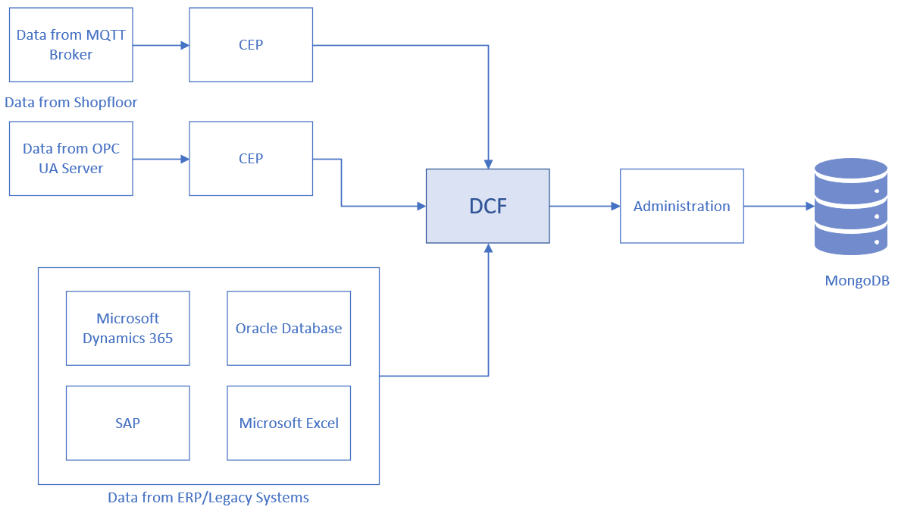
Activate Deactivate ON

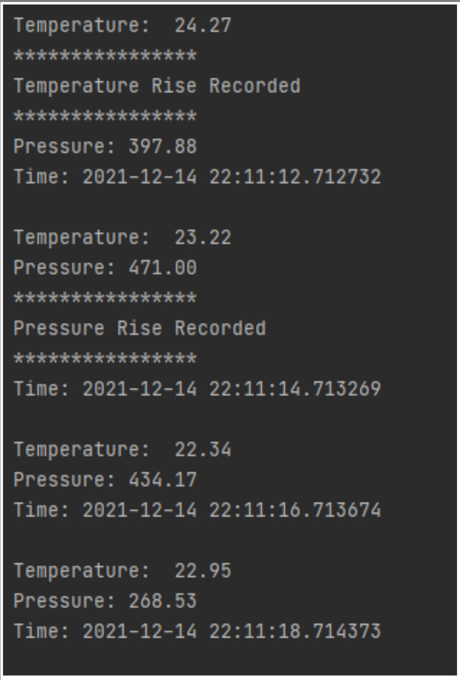
	Raw sensor data	States of fingers	Ges
User	[3286, 3680, 2735, 2432, 3553]	[0, 0, 0, 0, 0]	no
Desired		[2, 0, 0, 2, 0]	Index, middle and pin



Supported devices	Any Windows 10 machine, CaptoGlove.
User defined scenarios (non-technical) and relevant pilot cases	Component can be used in any system where there is a need to send commands or finish tasks by human operator using the glove. In the Siemens pilot case, the component was used to complete tasks in a bin picking collaborative robotic application.
Roles/Actors	Workers
Component Type	Component is Docker based application, and the needed CaptoGlove SDK is Windows executable. This means that the application needs to run on Windows machine. With further development and the use of Linux SDK, component is also usable on Linux machine, but the configuration application Capto Suite is not available on such machine.
Development environment	M2O2P is a component, that uses ROS2 as middleware between its sub-components. The backend of the component is Python based, and in frontend functions are made with Javascript, and visuals are made with HTML+CSS.
TLR	Current TRL: 4 Expected TRL: 5/6 The component is functional however certain changes may have to be done for communication with ROS and FIWARE based systems, depending on the use case.
Component usability	Documentation will be provided on the Git page.
Versions	Free version: covers basic functionality (UI with process log, application controller, etc) Premium: Testing and calibration functioning are under consideration.
Instruction video	Link to new video will be made available on Git and project web pages
GitHub link	The git page link will be made available soon.
RAMP link	This section will be updated once image pushed to RAMP
Docker registry	N/A

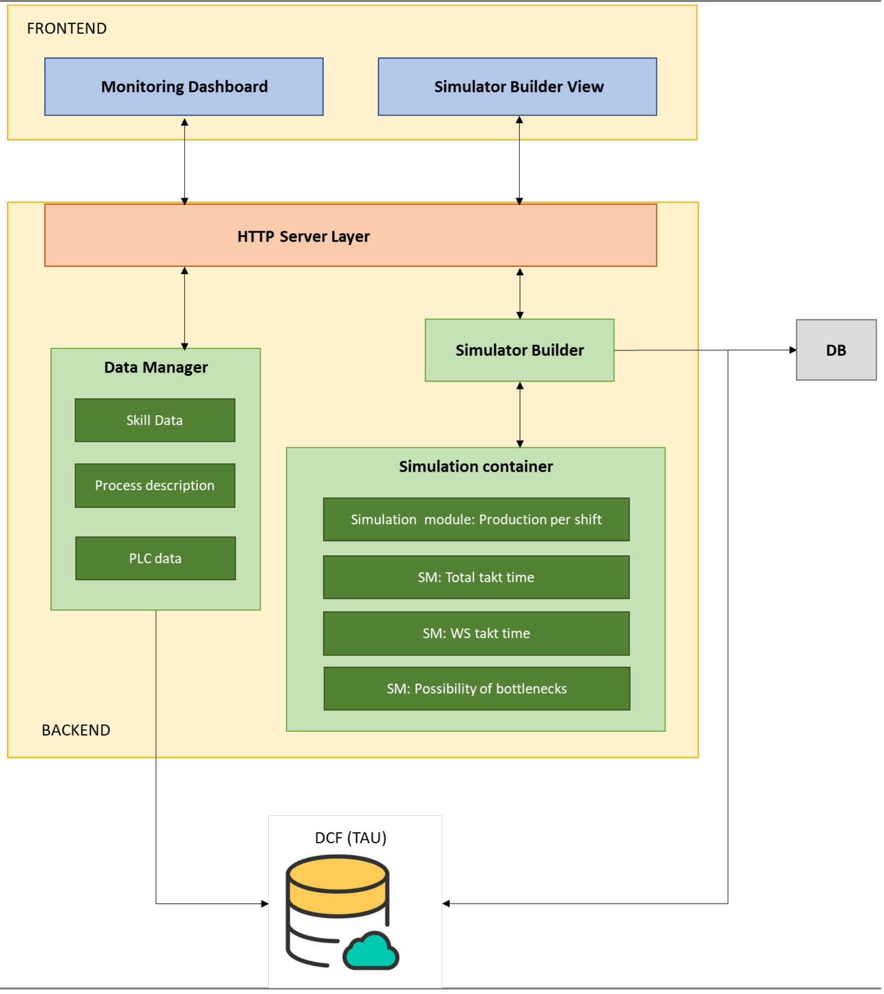
C2NET Data Collection Framework

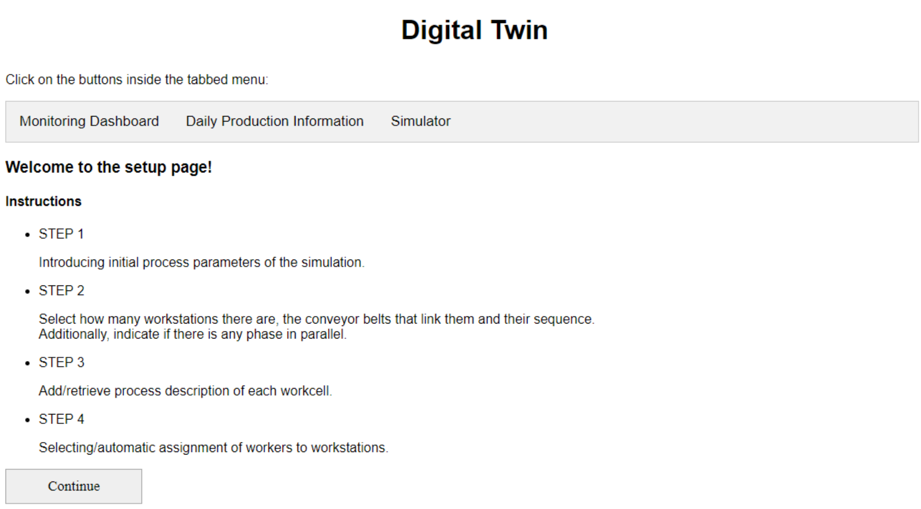
Component Title/Name	
DCF	
TAU	
Application area	Collecting data from shop floor (field devices e.g. sensors and controllers e.g. PLC) and enterprise resource planning (ERP)
Main functions	Monitoring of sensors, saving events in a database
Interfaces and data - Input	• Host address of OPC UA Server or MQTT Broker with user configurations (username, password) • MongoDB Database URI Connection String • API endpoint for ERP data or csv file path This component is positioned in the local execution and analysis levels. It needs horizontal communication with other components.
Interfaces and data – Output	JSON type data will be stored in database which can contain various type of information (time, string, integer). Event/alarm will be triggered by user defined configuration / workstation requirements. This component is positioned in the local execution and analysis levels. It needs horizontal communication with other components.
Functional architecture diagram	
Main non-functional requirements	N/A
Software requirements/dependencies	Data Transfer and Communication within DCF and Database requires Python installed and some Libraries (opcua, paho.mqtt, pymongo, pandas, json, flask, requests, cx_Oracle, hbdcli)
Hardware requirements	• Windows 7 or 10 • x86 64-bit CPU (Intel / AMD architecture) • 4 GB RAM 5 GB free disk space
Security threats	The component requires authentication from server/database before connecting and collecting ERP or shopfloor data and storing the data in database.
Privacy threats	None
Execution place	Both devices connected with local network and on different host address can be connected via MQTT and OPC-UA
Deployment instructions	DCF component will be deployed in docker and relevant instructions will be provided

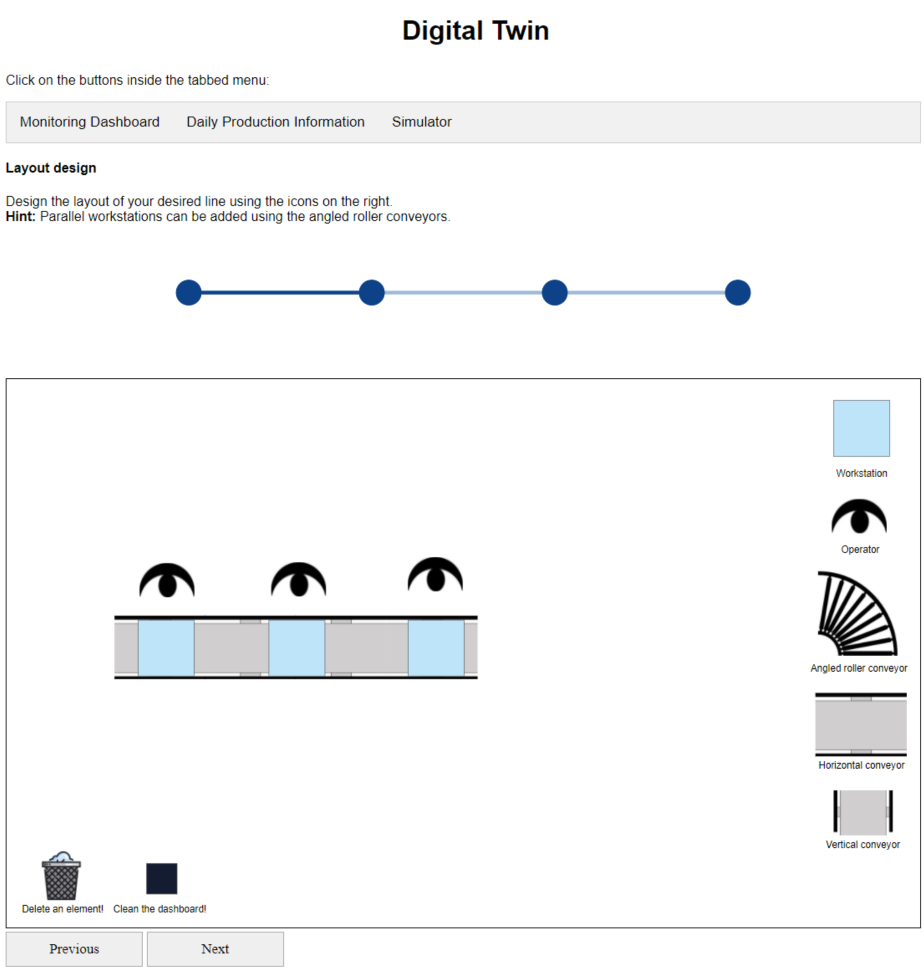
User interface	  After specifying necessary connection configuration, the DCF module is monitoring the temperature and pressure reading through opc-ua server (left image). If the temperature or pressure is more than the allowed, it is logging the information (e.g. time, value, description) in the database (right image). These parameters can be changed according to the use case.
Supported devices	Desktop, Laptop
User defined scenarios (non-technical) and relevant pilot cases	DCF component can be used at production/assembly lines to collect data from workstation/sensors and apply event processing. For instance, if more time is being consumed to complete the task at specific workstation, this activity can be monitored, and relevant data can be logged in database for engineers/supervisor to view and take appropriate action
Roles/Actors	Manufacturing/Assembling Industry Managers, Engineers, Supervisors/Shift Leader
Component Type	Windows/Desktop application
Development environment	Python programming is mainly used for the development of the component. The database used is the document-based database, MongoDB
TLR	Current TRL: 4 Expected TRL: 5/6 The component is functional however certain changes may have to be done for communication with FIWARE based system, depending on the use case.
Component usability	Basic instructions are written in the source code but text file for usability and additional documentation will be provided on the Git page
Versions	Free version: covers core functionalities (with predefined set of data adapters) Premium: Additional data adapter integration under consideration
Instruction video	Link to new video will be made available on Git and project web pages
GitHub link	The git page link will be made available soon.
RAMP link	Waiting on RAMP docker registration. Will be updated once pushed to RAMP
Docker registry	N/A

Digital Twin

Digital Twin (Control and Planning)	
DT-CP	
TAU	
Application area	Production monitoring dashboard and simulator for validation of process modifications.
Main functions	DT-CP provides support on the production line by monitoring production (based on available and defined input sources) and allows simulation to test different production scenarios. Additionally, it provides information to balance the workload of the operator between the work cells based on skill set data.
Interfaces and data - Input	Data Collection Framework (DCF) component developed by TAU is an input of this component that provides: • Real time data stream from PLC. • Description of process sequence as JSON. On the other hand, the user enters some data manually through the user interface to be able to simulate different scenarios (expected production, simulation time, ...). This component is positioned in the global design, execution, and analysis levels. Depending on the use case, It may need both horizontal and vertical communication with other components.
Interfaces and data – Output	Daily reports from the monitoring production. Simulation reports. This component is positioned in the global design, execution, and analysis levels. Depending on the use case, It may need both horizontal and vertical communication with other components.

Functional architecture diagram	
Main non-functional requirements	N/A
Software requirements/dependencies	N/A
Hardware requirements	Device capable of handling web-based applications
Security threats	None
Privacy threats	None
Execution place	private cloud (meaning in pilot premises), cloud
Deployment instructions	Instructions will be provided on the Git page
User interface	The component will be divided into two parts: an online dashboard for monitoring the line in real time and a simulator, to experiment with alternative models to be implemented in the real line. The monitoring dashboard is intended to follow the flow of product from one workstation to another. The simulator allows to modify and test different production strategies that would be more complicated and time consuming to test in the real line. The simulator setup consists of four steps: the introduction of the initial process execution parameters, design of the layout, process description assignment and allocation of resources.

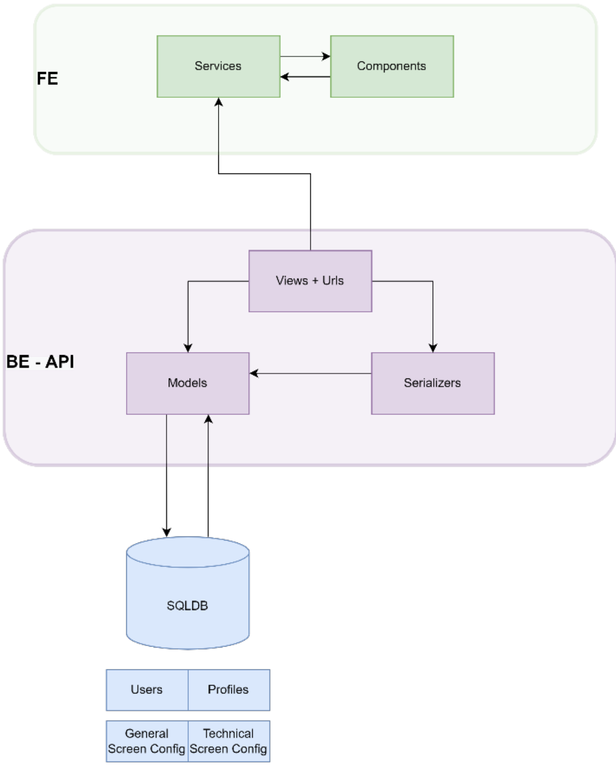


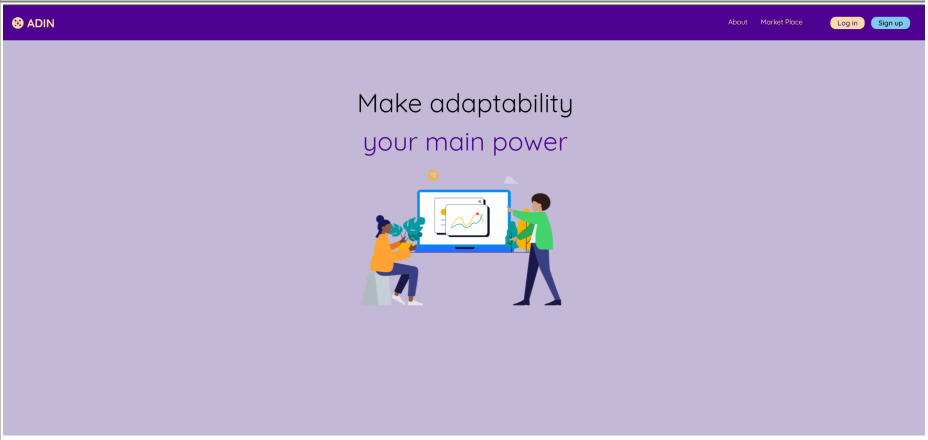
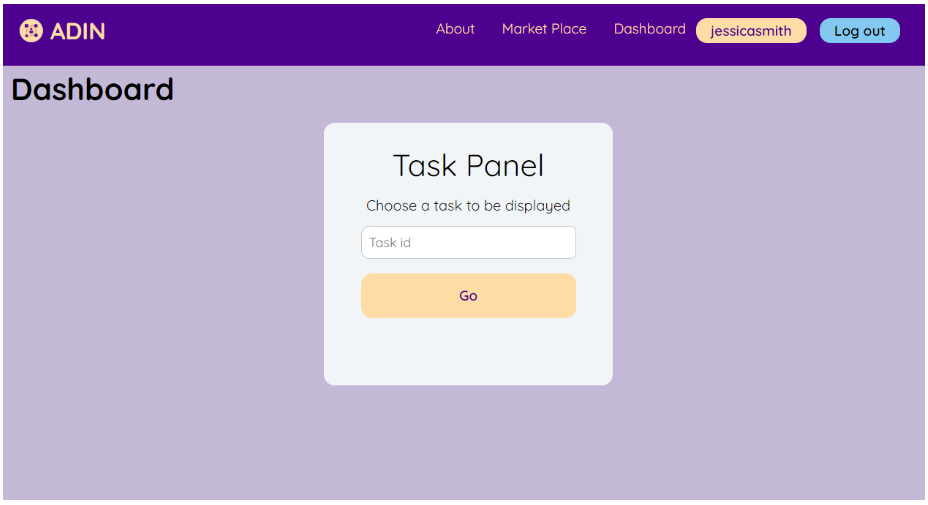
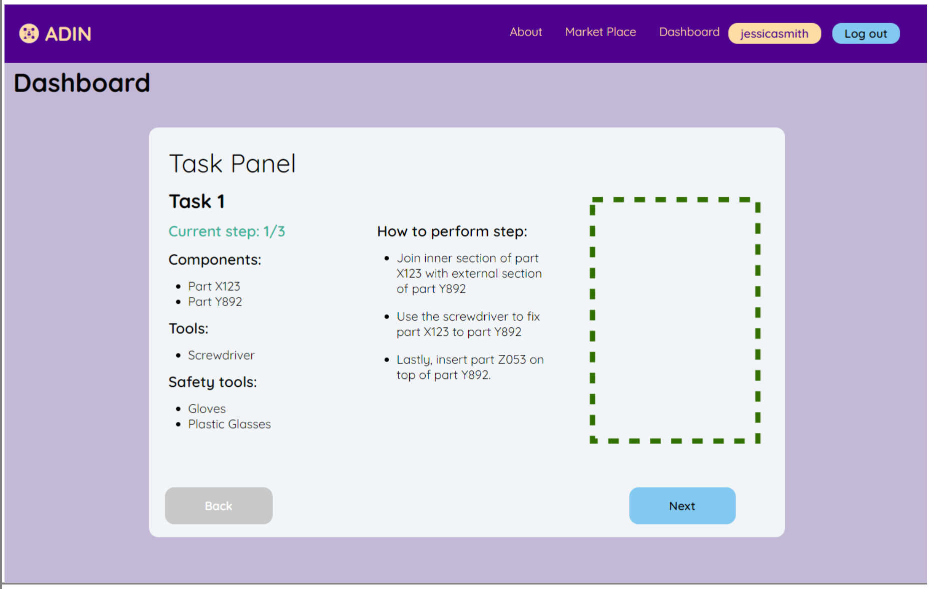


Supported devices	Desktop, Laptop
User defined scenarios (non-technical) and relevant pilot cases	Coupled with data collection applications, the monitoring part of the component could be used to have an overview on the process and provide notification and control mechanisms. Data inputs, notification handling, and control mechanisms may be to be further adapted as per use case.

	The simulator allows the user to define a time-based simulation of a process assembly by setting concrete configuration parameters (e.g., takt time, shift time) and assigning resources (e.g., workers) with additional resource parameters (e.g., skill set). Th result would be a report on the pre-provided production targets. Specific configuration parameters and settings may have to be further adapted as per use case.
Roles/Actors	Production engineering and production management engineers
Component Type	Web
Development environment	Development environment: Visual Studio Code and Webstorm. Programming language: the frontend is based in HTML, CSS and JS to create a web application for the user and it is implemented on NodeJS backend, using ExpressJS.
TLR	Current TRL: 4 Expected TRL: 5/6 The component is functional however certain changes may have to be done for communication with FIWARE-based systems, depending on the use case.
Component usability	Documentation will be provided on the Git page.
Versions	Free version: covers core functionality (simulator setup + monitoring dashboard with limited set of data sources) Premium: More advanced notification handling mechanisms for monitoring dashboard are under consideration.
Instruction video	Link to new video will be made available on Git and project web pages
GitHub link	The git page link will be made available soon.
RAMP link	Waiting on RAMP docker registration. Will be updated once pushed to RAMP
Docker registry	N/A

Adaptive Interfaces

Adaptive Interfaces	
ADIN	
TAU	
Application area	Supporting human workers on the shopfloor
Main functions	ADIN is a web-based application capable of assisting employees in the manufacturing sector during their working time. It combines user profile data such as skill level, preferences, experience, and role, among other characteristics at runtime. In addition, ADIN helps avoid human error and reduce stress by providing the most relevant and specific information needed by the user, for example by guiding workers to perform tasks.
Interfaces and data - Input	Structured data from the data base. Inputs by the user for system setup. Sensory inputs via FIWARE This component is positioned in the global analysis level. Depending on the use case, It may need vertical communication with other components.
Interfaces and data – Output	Information of the selected task (Steps for completing it, components required, tools and safety tools needed, and information on how to perform each step). Specific displayed information for the current user profile. Factory notifications and warnings relevant for the user. This component is positioned in the global analysis level. Depending on the use case, It may need vertical communication with other components.
Functional architecture diagram	
Main non-functional requirements	N/A
Software requirements/dependencies	N/A
Hardware requirements	Device capable of use web-based application (e.g.: PC or laptop)
Security threats	None

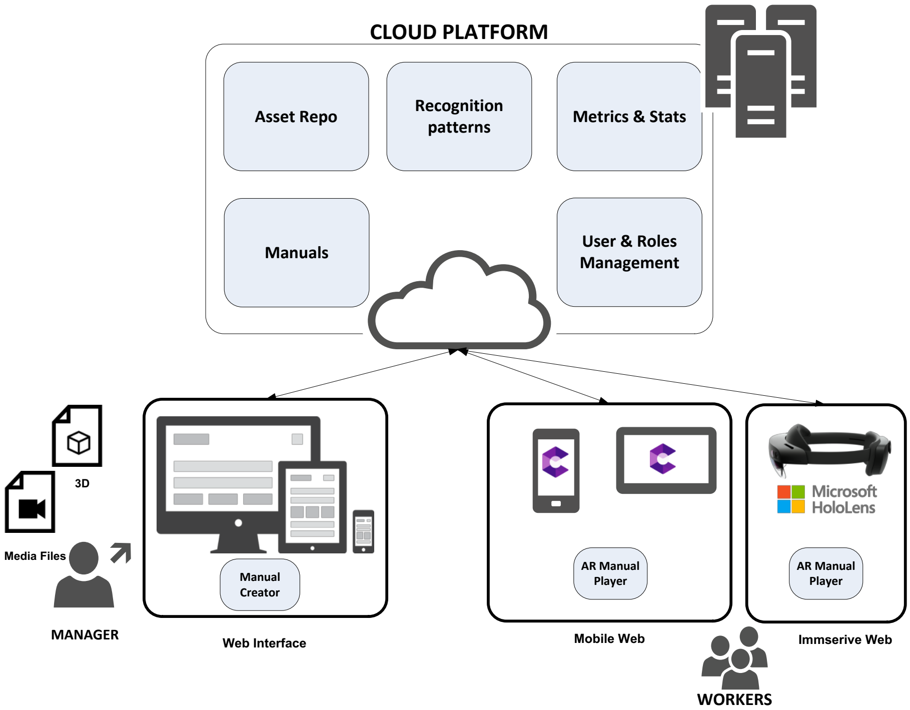
Privacy threats	None
Execution place	private cloud (meaning in pilot premises)
Deployment instructions	Instructions will be provided on the Git page
User interface	  
Supported devices	Desktop, Laptop
User defined scenarios (non-technical) and relevant pilot cases	ADIN can be used by workers in an assembly line for assisting them on the task, giving them the specific and relevant information for fulfilling the duty.

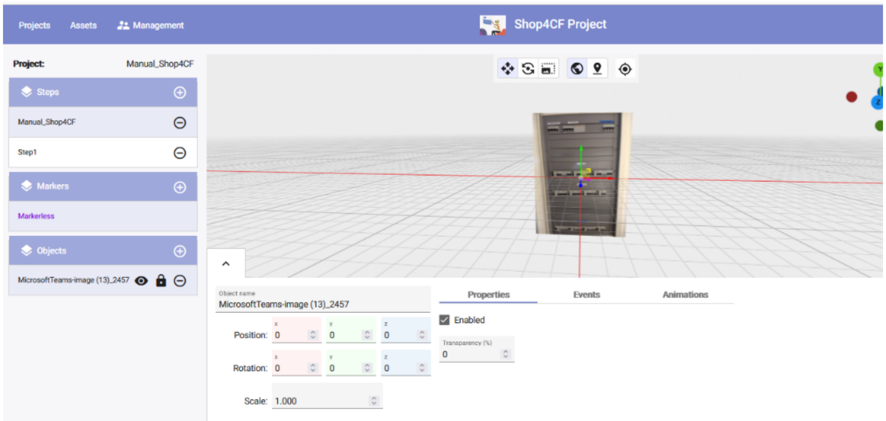
	It also can be used in collaborative task with cobots where the worker receive instructions on the steps of the collaboration task.
Roles/Actors	Shopfloor workers
Component Type	Web
Development environment	Development environment: Visual Studio Code and PyCharm. Programming language: JavaScript (FE) and Python (BE).
TLR	Current TRL: 4 Expected TRL: 5/6 The component is functional however certain changes may have to be done for communication with FIWARE based system, depending on the use case.
Component usability	Documentation will be provided on the Git page.
Versions	Free version: visualization dashboard for process assemblies Premium version: additional sensory inputs for dynamic visualization are under consideration
Instruction video	Link to new video will be made available on Git and project web pages
GitHub link	The git page link will be made available soon.
RAMP link	Waiting on RAMP docker registration. Will be updated once pushed to RAMP
Docker registry	N/A

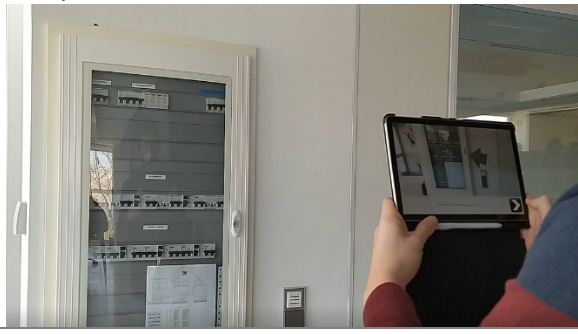
AR Manual Editor

Augmented reality-based content editor: Manual Editor	
AR Manual Editor	
TECNALIA	
Application area	Supporting human workers on the shop floor
Main functions	Assistance and training for operators during customised product assembly process, and maintenance operations including recognition of objects, sequence of operations and AR guidance to operators. This tool is based on code and developments done previously by TECNALIA before SHOP4CF as detailed in the Consortium Agreement. The goal of SHOP4CF for this component is to enable the migration to web. We would like to remark that additional features are included in the tool in another project called 872570-KYKLOS4.0 so SHOP4CF will take advantage of these new features and improvements and use it to facilitate the deployment and usage of the tool.
Interfaces and data - Input	• Main Inputs: : Assets (2D/3D objects, images, pdf, audio, video) and sequence of operations for the product assembly. • Input Data from Partner: The designer needs the manuals (video or pdf) to create the AR guidance to operators. • Nature of Expected Input: JSON, XML, plain text, glb, jpg, png, mp4, pdf, etc. • Interfaces with other platform/systems: The user/role login services to be integrated into the plant/company information system. • Triggered by: AR visualization will be triggered by 3D/context/image/BIDI recognition, based on a camera input. About the position of the component within the architecture, this component is provided at local level giving a functionality to support individual work cells during execution phase. The component does not require vertical and/or horizontal communication with other components.

Interfaces and data – Output	• Main Outputs:
	  ◦ For shopfloor designer: an AR manual design tool for a specific product assembly process (ready to be reused in similar/other processes, to add/update sequences easily, etc.)
	  ◦ For operators: an AR guidance in the customised product assembly process for his assistance.
	• Output Data from Partner: All the data related to the manuals
	• Nature of Expected Output: JSON, XML, plain text, glb, jpg, png, mp4, pdf, etc.
	About the position of the component within the architecture, this component is provided at local level providing a functionality to support individual work cells during execution phase. The component does not require vertical and/or horizontal communication with other components.

Functional architecture diagram	 Main functional Requirements: • The component will require a user validation. • Different content will be showed based on user role. • User should upload all the multimedia files (assets) through the editor/creator. • A manual/guidance could be marked as WIP/Published to be shared.
Main non-functional requirements	• Mobile devices should be compatible with ARCore/ARKit frameworks. • Wi-Fi connection is required at the shop floor. • Minimum brightness levels are required for the AR vision algorithms to work correctly. • If workers are required to wear gloves, specific mobile device models will be required.
Software requirements/dependencies	WebXR, Django, ARCore, Arkit, Microsoft Mixed Reality toolkit
Hardware requirements	• Mobile devices/HoloLens • A server
Security threats	The component need an account management. There is one already implemented but in case to be integrated on RAMP or another platform, further adjustments should be needed.
Privacy threats	None.
Execution place	Private cloud provided by TECNALIA with access to pilots. In case there is a need for pilot sites to deploy the component there, it can be done using dockers.
Deployment instructions	In RAMP marketplace and upon request.
User interface	Interface for the developer: an editor to create AR content in an easy way to guide operators in the shop floor.

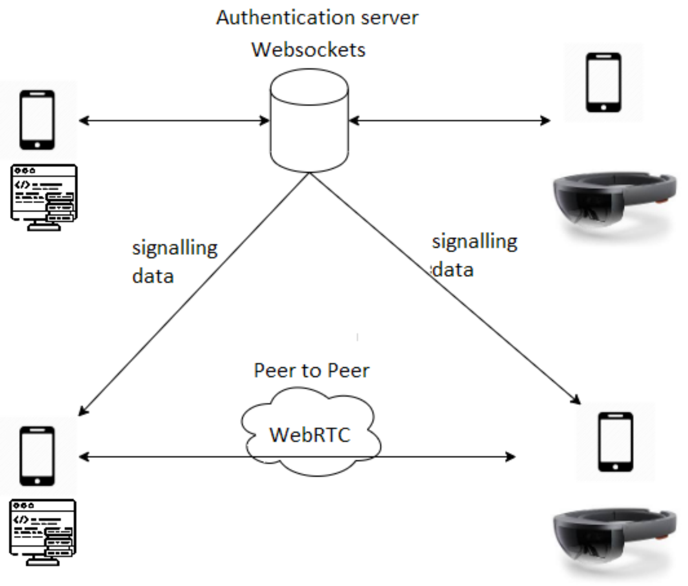
<table>
<tr><td></td><td>

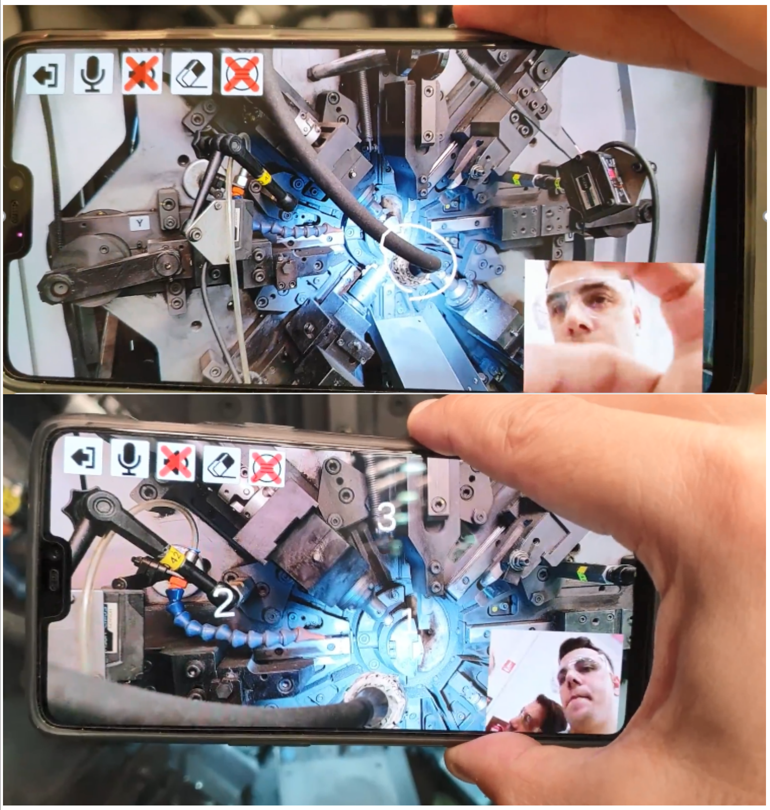
Interface for the Operator (customised by the editor): the AR guidance to be visualised with a mobile devices/HoloLens through different steps and different objects (2D/3D objects, images, video, documents, animations, etc.).
</td></tr>
<tr><td>Supported devices</td><td>Laptop + mobile devices/HoloLens</td></tr>
<tr><td>User defined scenarios (non-technical) and relevant pilot cases</td><td>AR guidance to operator during the assembly of the base plate in collaboration with a robot in the Siemens Use case.
The shop floor manager/developers add a new manual to the system using the editor, adding all the multimedia assets (3D files, pdf, videos, photos, etc.) and defining, step by step, the entire manual.
The shop floor manager, once the manual has been added, defines through the editor what type of trigger will activate the augmented reality display. In this case, through the Context Broker we will see in a task for HoloLens is raised to show the different steps of the manual.
Then he publishes from the editor that manual, so that it can be consumed by HoloLens, defining the type of users and roles that will have permission to do so.
From that moment any worker of the plant with permissions working on the assembly of a base plate will be able to trigger with his device the visualization in augmented reality of that manual based on the status of the collaboration with the robot supporting him/her.</td></tr>
<tr><td>Roles/Actors</td><td>Editor: shop floor manager/developers.
Visualizator: operators.</td></tr>
<tr><td>Component Type</td><td>Editor: Web
Visualizator: Android Mobile/HoloLens</td></tr>
<tr><td>Development environment</td><td>Angular WebXR, 3JS, AFRAME.</td></tr>
<tr><td>TLR</td><td>Current TRL: 5-6
Expected TRL: 6-7

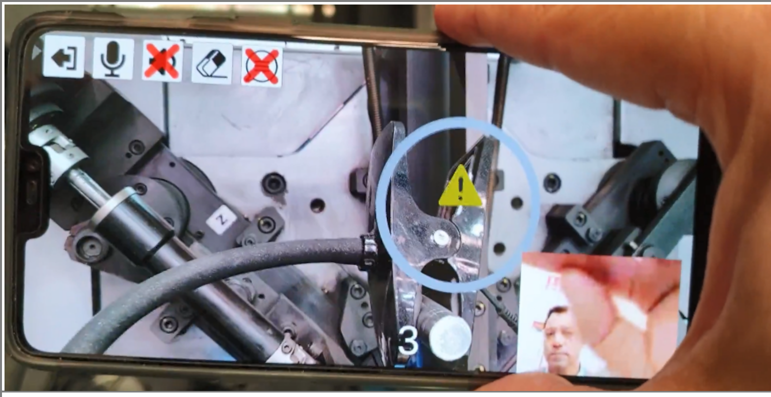
Although the component is fully functional at this moment, further adaptations are required to enable integration with FIWARE Context Broker in order to receive subscribed events to trigger manual instructions as required by pilots.</td></tr>
</table>

Component usability	https://www.youtube.com/watch?v=J4PrJScjRMY
Versions	The use of this component during and after the project is subject to the Consortium Agreement and the detailed background. • Free version: for the visualizator part of this subcomponent. • Premium version: All functionalities will be available in the premium version allowing the creation, generation and management of AR manuals.
Instruction video	https://www.youtube.com/watch?v=J4PrJScjRMY
GitHub link	Not available. As stated in the CA, the tool is not open source so no code will be provided.
RAMP link	N/A
Docker registry	N/A

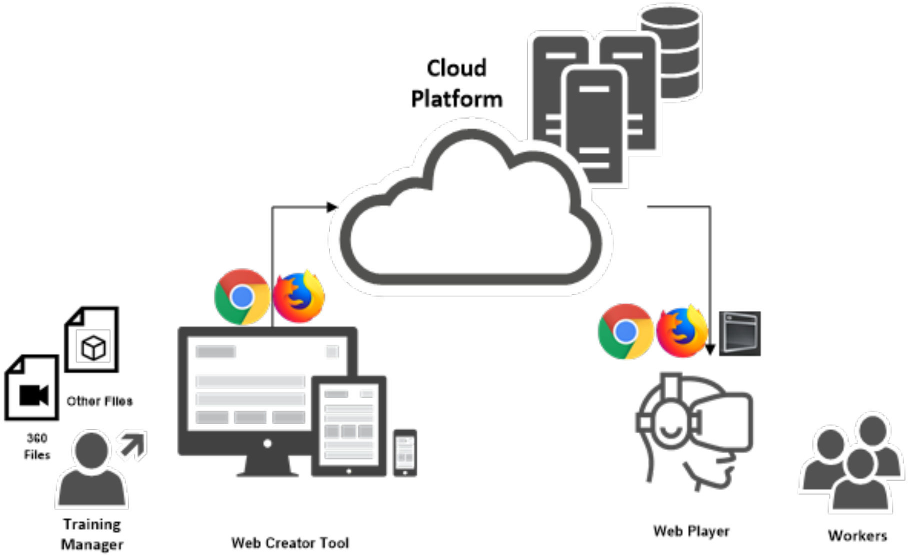
AR-based Teleassistance

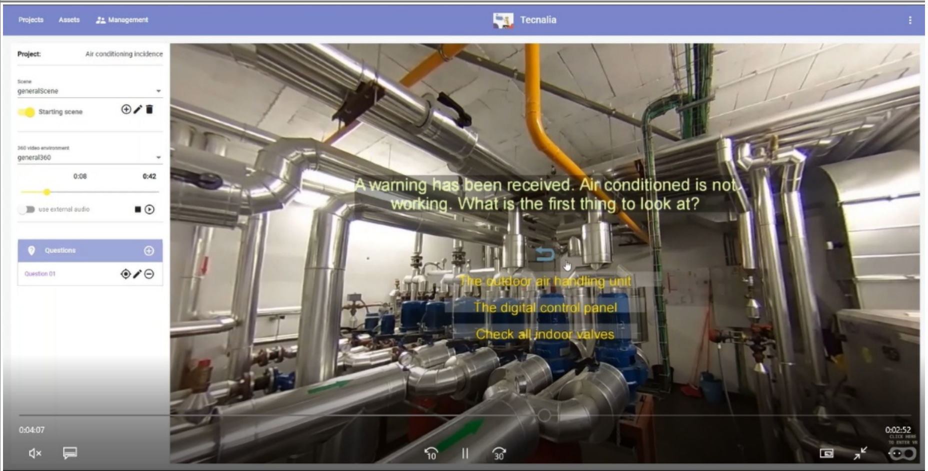
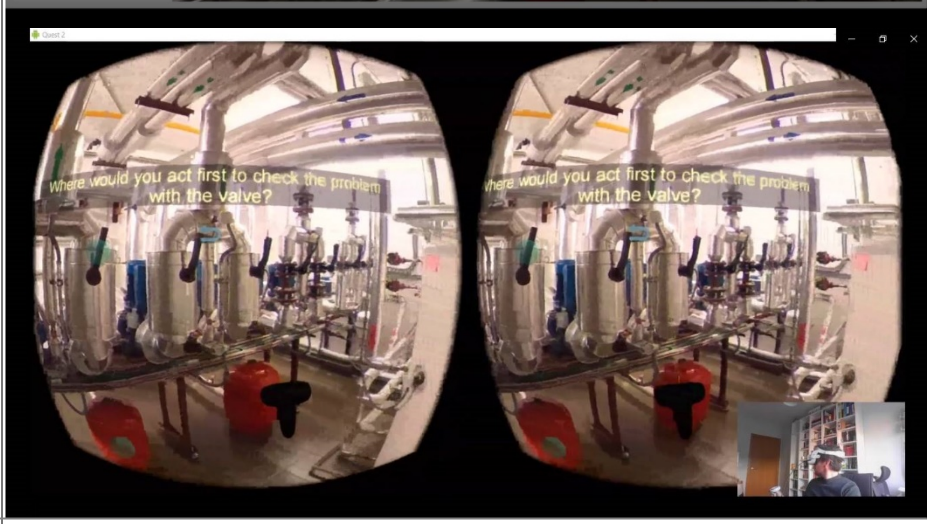
Augmented reality-based content editor: Teleassistance	
AR-based Teleassistance	
TECNALIA	
Application area	Supporting human workers on the shop floor: Tele-assistance in maintenance for long distance workers
Main functions	Communication between workers and experts through video streaming and augmented reality indications supporting operators with the maintenance and collaboration of working processes.
Interfaces and data - Input	• Nature of expected input: real-time stream from the device camera (smartphone, Microsoft HoloLens) and the touch coordinates in the screen. • Interfaces with other platform/systems: all client applications communicate each other using a server that uses REST and web socket services, and the real-time stream uses the WebRTC technology, sending a JSON file with the data to the server. • Triggered by the execution is triggered by the interaction between a worker and the expert using the client application. About the position of the component within the architecture, this component is provided at local level providing a functionality to support individual work cells during execution phase. The component does not require vertical and/or horizontal communication with other components.
Interfaces and data – Output	• Nature of expected output: video and audio in each client application with the generation of 3d objects, such as drawing lines, signalling objects, numbers, etc. • Interfaces to other platform/systems: as it is said in the input part (before), all client applications communicate each other using a server that uses REST and web socket services, and the real-time stream uses the WebRTC technology. • Triggered by the execution is triggered by the interaction between a worker and the expert using the client application. About the position of the component within the architecture, this component is provided at local level providing a functionality to support individual work cells during execution phase. The component does not require vertical and/or horizontal communication with other components.
Functional architecture diagram	

	First, users get identified in a server, and send a notification to the user they want to connect. The moment each peer accepts the request, they get connected to the real time stream and can interact with the drawing screen system.
Main non-functional requirements	• An Internet connection (Wi-Fi connection recommended). • Android app: ARCore framework is needed to use the AR functionalities. • Browser client: right now, the best/recommended browser to use it is Firefox. • Depending on the resolution of the real time streaming, the bandwidth has to be in accordance with it, with higher resolution is recommended to use Wi-Fi-connection.
Software requirements/dependencies	• Server side: NodeJS, Websockets, Express • Client side: ARCore
Hardware requirements	• Server side: UNIX/Linux environment, with enough bandwidth for the number of users to use it • Client side: smartphone device compatible with ARCore; browser can´t be Chrome (Firefox recommended)
Security threats	The component need an account management. There is one already implemented but in case to be integrated on RAMP or another platform, further adjustments should be needed. Server side must use an SSL certificate to send the communication data with https protocol.
Privacy threats	Right now, the authentication part only requires having the client application to log in, so it is necessary not to share them with unknown users.
Execution place	Private cloud provided by TECNALIA with access to pilots. In case there is a need for pilot sites to deploy the component there, it can be done using dockers.
Deployment instructions	In RAMP marketplace and upon request.
User interface	Android app client 

	
Supported devices	Laptop + mobile devices/HoloLens
User defined scenarios (non-technical) and relevant pilot cases	UC2 in Arcelik for equipment maintenance. A worker that needs support with any type of physical component or machine, call to an expert colleague in the field. In order to do that, the worker opens the application installed in his smartphone and calls the previously connected expert. The application connects with the expert sharing the back camera of the worker and the frontal camera of the expert. The worker scans the component area or the machine area, and the expert draws through the mobile screen creating indications, as a drawing mode to the worker, giving an augmented reality support. When the support is finished, both exit the application.
Roles/Actors	• Operator worker who wants support in the field, and an office/expert worker that gives that support.
Component Type	• Editor: Web • Visualizator: Android Mobile/HoloLens
Development environment	• Server side: NodeJS, Express, Websockets (Languages: Typescript and Javascript) • Client side: Unity3D for all platforms (WebGL, UWP, Android) (Languages: C#)
TLR	Current TRL: 5-6 Expected TRL: 6-7 Although the component is fully functional at this moment, further adaptations are required to enable integration with WoT component to visualise info from sensors as required by pilot.
Component usability	https://www.youtube.com/watch?v=br7-aPbR7y4&t=16s
Versions	The use of this component during and after the project is subject to the Consortium Agreement and the detailed background. Premium version.
Instruction video	https://www.youtube.com/watch?v=br7-aPbR7y4&t=16s
GitHub link	Not available. As stated in the CA, the tool is not open source so no code will be provided.
RAMP link	N/A
Docker registry	N/A

VR Creator

Virtual Reality Creator	
VR Creator	
TECNALIA	
Application area	Formation/training of workers
Main functions	It will allow workers to be trained in the operation of a machine, or manufacturing line for example, through the use of virtual reality. With this web tool it will be possible to create and consume immersive VR experiences (with glasses) oriented to training.
Interfaces and data - Input	• Main Inputs: 3D objects, sequence of training steps. • Input Data from Partner: The designer needs the manuals (video or pdf) to create the VR trainings. 360 photos/videos of the warehouse, machine, line. • Nature of Expected Input: plain text, glb, jpg, png, mp4, pdf, etc. • Interfaces with other platform/systems: Synchronization with the company's login/user system. • Triggered by: Nothing technological. Just company's training plan. About the position of the component within the architecture, this component is provided at local level providing a functionality to support individual work cells during execution phase. The component does not require vertical and/or horizontal communication with other components.
Interfaces and data – Output	• Main Outputs: Creator tool: Web oriented solution to easy generate VR interactive trainings based on 360 photos/videos. • Operator: Training player tool: Web oriented VR player to consume the prepared trainings. • Output Data from Partner: All the data related to the VR trainings. • Nature of Expected Output: VR trainings. About the position of the component within the architecture, this component is provided at local level providing a functionality to support individual work cells during execution phase. The component does not require vertical and/or horizontal communication with other components.
Functional architecture diagram	
Main non-functional requirements	• If video 360 are big files web creator tool will require a PC/Laptop with dedicated (and "good") graphic card. We will specify more. • Wifi connection is required during training experience creation and consumption • PC/Laptop and VR HMD should contain a WEBXR compatible browser.

Software requirements/dependencies	WebXR, Django, WebGL
Hardware requirements	• PC/laptop and VR HMDs • A Server
Security threats	The component need an account management. There is one already implemented but in case to be integrated on RAMP or another platform, further adjustments should be needed.
Privacy threats	None.
Execution place	Private cloud provided by TECNALIA with access to pilots. In case there is a need for pilot sites to deploy the component there, it can be done using dockers.
Deployment instructions	In RAMP marketplace and upon request.
User interface	 
Supported devices	PC/Laptop + VR HMDs
User defined scenarios (non-technical) and relevant pilot cases	Training the operators when using new machines. At this moment we do not have any pilot interested in the component.
Roles/Actors	• Training manager for experience creation • Workers to receive the VR training experiences
Component Type	• Editor: Web • Visualizator: Android Mobile/VR headsets
Development environment	Python, C#, JavaScript
TLR	Current TRL: 5-6

	Expected TRL: 6-7
Component usability	https://www.youtube.com/watch?v=BiWTpLE3Qj0
Versions	The use of this component during and after the project is subject to the Consortium Agreement and the detailed background. • Premium version: Fully functionalities will be available following a SaaS approach.
Instruction video	https://www.youtube.com/watch?v=BiWTpLE3Qj0
GitHub link	Not available. As stated in the CA, the tool is not open source so no code will be provided.
RAMP link	N/A
Docker registry	N/A

Manufacturing Process Management System

Manufacturing Process Management System	
MPMS	
TUE	
Application area	End-to-end (i.e., from order reception until product delivery) manufacturing process management, i.e., design and enactment of manufacturing processes.
Main functions	MPMS includes the functionality to design processes and describe agents, and execute in automated way the processes by assigning activities to agents. It provides orchestration of activities in a global level, i.e., covering all work cells/production lines of a factory. MPMS supports dynamic agent allocation by selecting the best agents to perform a task for most optimal utilization, exception handling on agent, task and process level, process monitoring for a complete status overview of the manufacturing processes.
Interfaces and data - Input	MPMS: • Reads agents data and their capabilities from data structures in a relational database, to use during execution for task allocation • Requires step data from the local level to design and execute tasks. These steps/tasks are defined in a structured way in datatables which are shared by both MPMS and local components. The above data input require an interface to a DB, e.g. to PostgreSQL DB. • Receives task status (e.g., completion confirmation), through the local orchestrator components, through FIWARE (or other middleware, e.g. web-socket message bus with JSON-formatted messages). (can also receive the same input with REST API calls (HTTP POST requests from local components)) • Receives changes on resource status through subscription on FIWARE • Receives events and alerts from FIWARE (or with REST API calls (HTTP POST requests))
Interfaces and data – Output	MPMS: • Describes agents and their capabilities in data structures in a relational database, during design phase • Describes tasks (based on step data) in a structured way in datatables which are shared by both MPMS and local components. The above data output require an interface to a DB, e.g. to PostgreSQL DB. • Sends task assignments requests to agent (through the local orchestrator components or control systems), through FIWARE (or other middleware (e.g. web-socket message bus with JSON-formatted messages). (can also send task assignments with HTTP GET/POST requests • Updates resources on FIWARE. • Sends events on FIWARE (or with JSON-formatted messages by HTTP GET/POST requests)

<table>
<tr>
<td>Functional architecture diagram</td>
<td>
MPMS features:

<u>Design modules:</u>

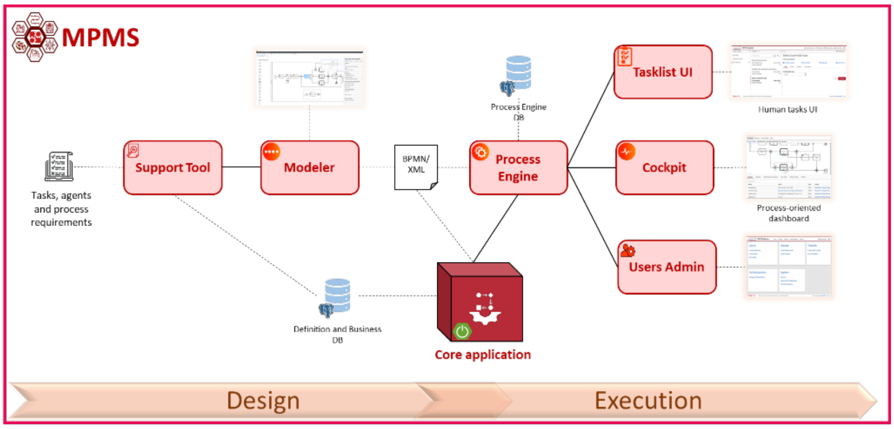
Support Tool – captures requirements analysis (tasks, agents and process requirements) (covered by the Support Tools development activities, added here for sake of completeness)
Modeler – design process models in BPMN 2.0

<u>Execution modules:</u>

Process engine – automates the enactment of the process models
Core application – implements business logic for:

Task delivery – message delivery to both human agents (Tasklistweb app UI) and auto agent tasks (on Context Broker)
Agent allocation – mechanism to select the most suitable (team of) agent(s) to perform a task
Exception handling – processing of events, alerts and exceptions

Cockpit – dashboard for process monitoring (web application)
Users Admin – usersadministration/configuration (web application)



</td>
</tr>
<tr>
<td>Main non-functional requirements</td>
<td>
As a logical functional component, MPMS shall be able to:

Automatically execute a sequence of activities
Monitor agents' availability
Monitor agent's performance including at least task estimated completion time and task actual completion time
Monitor process current state
Provide right information to agents to perform a task
Handle exceptions on agent, task and process level by halting/resuming their activities and initiating out-of-normal action processes
(re-)allocate appropriate agents to perform a task based on abilities, skills, authorizations, cumulative workload, overall manufacturing system status and availability
Re-allocate agents in response to external events such as safety alerts or sensor failures.

As a software technical component, MPMS shall be able to:
</td>
</tr>
</table>

	• provide a modeler application to model processes
	• provide a process engine to automatically enact process models
	• provide tasklist applications to deliver tasks to human operators
	• support integration to custom UIs as tasklist applications
	• provide integration to local components to deliver tasks to robotic agents
	• support various platform environments
	• support various DBMS
	• be deployed both on premise/cloud
	• provide security/authorisation mechanisms
	• integrate to middleware/context broker and other components
	    ○ support web services
	    ○ support REST/JAVA APIs
	    ○ support SOA/Interoperability (NF)
	• be robust (NF)
	• be runtime scalable (NF)
	• be easy to use by both process modelers, developers and end users (e.g., human operators) (NF)

Software requirements/dependencies	MPMS is built on Camunda Platform 7.15.0, Community Edition and runs in every Java-runnable environment. It can support the following environments: <u>Container/Application Server for runtime components</u> • Apache Tomcat 7.0 / 8.0 / 9.0 • JBoss EAP 6.4 / 7.0 / 7.1 / 7.2 • Wildfly Application Server 10.1 / 11.0 / 12.0 / 13.0 / 14.0 / 15.0 / 16.0 / 17.0 / 18.0 <u>Databases</u> • MySQL 5.6 / 5.7 • MariaDB 10.0 / 10.2 / 10.3 • Oracle 11g / 12c / 18c / 19c • PostgreSQL 9.4 / 9.6 / 10.4 / 10.7 / 11.1 / 11.2 • postgres:14-alpine (Docker image) • Microsoft SQL Server 2012/2014/2016/2017 • H2 1.4 • Adminer:4.8.1 (UI for DB management) (Docker image) <u>Web Browser</u> • Google Chrome latest • Mozilla Firefox latest • Internet Explorer 11 • Microsoft Edge <u>Java</u> • Java 8 / 9 / 10 / 11 / 12 / 13 (if supported by your application server/container) Java Runtime • Oracle JDK 8 / 9 / 10 / 11 / 12 / 13 • IBM JDK 8 (with J9 JVM) • OpenJDK 8 / 9 / 10 / 11 / 12 / 13

	○ openjdk:11.0.13-jre-slim (Docker image) Camunda Modeler • Windows 7 / 10 • Mac OS X 10.11 • Ubuntu LTS (latest) The Camunda Community Platform is provided under various open source licenses (mainly Apache License 2.0 and MIT). Third-party libraries or application servers included are distributed under their respective licenses. Detailed info on licences is provided in T7.2.
Hardware requirements	For deploying MPMS in a local desktop PC, not any special requirements are needed. A powerful processor, a lot of RAM memory and a decent graphics card is sufficient. The following specs should do: • Processor: Intel Core i7-7700 @ 3.60GHz / Intel Core i7-6700K @ 4.00GHz / Intel Core i7-7700K @ 4.20GHz / Intel Core i7-8700K @ 3.70GHz • Storage: SATA 2.5 SSD (e.g. 256 GB) • RAM: 32GB DDR4-2133 DIMM (2x16GB) (well, even 16GB will not be a problem) • Graphics Card: Any modern standard graphics card • Monitor, keyboard and mouse are essential. Touchscreen for operators might be handy. A laptop could also work: • Processor: Intel Core i7-7700HQ @ 2.80GHz / Intel Core i7-6770HQ @ 2.60GHz.
Security threats	MPMS connects to DBs with password access. Users of web applications have access with password.
Privacy threats	Agents (and specifically human operators) should be described with due diligence wrt privacy data.
Execution place	MPMS shall be deployed on local PCs on premises. It can also be deployed on a cloud server (e.g., on TUE premises) but extra security is required.
Deployment instructions	Deployment instructions and user manuals can be provided upon request. Can also be uploaded on RAMP if there's a specific repository.
User interface	Three different types of users: Process modelers They use the Modeler application to model manufacturing processes (with BPMN 2.0) Application developers They turn the process models designed by the process modelers into executable process models (i.e., the ones that the Process Engine can interpret and enact). Also, they can build custom applications (e.g. tasklists, cockpits, smartwatch apps) (Human) Managers and operators (end-users)

	Managers use the Cockpit/Dashboard and Admin applications (default or custom) to see the processes state and manage the users of MPMS Operators use the Tasklist applications (default or custom) to receive tasks and provide input (e.g. task completion confirmation) For each type of user, there are manuals and webinars to provide instructions.
Supported devices	• Modeler runs on PC/Laptop (see SW/HW requirements above) • Process Engine runs on PC/Laptop (see SW/HW requirements above) • Web applications run on PC/Laptop/tablet/smartphones (additionally, a prototype tasklist application has been built for smartwatch)
User defined scenarios (non-technical) and relevant pilot cases	MPMS can be used in any pilot/open call for: • process modelling (for bottlenecks identification and enabling automated execution), • dynamic agent allocation, • process orchestration, • automated process execution, • integration to other IS (e.g., ERP) for getting the right information and providing it to agents during execution, • process status monitoring, • task monitoring for job safety and quality of human operators, • re-allocation of agents when job safety and quality criteria are violated, • etc.
Roles/Actors	MPMS is used by: • Process modelers • Application developers • (Human) Managers and operators (end-users) • Robotic agents are implicit users as they are instructed by MPMS to perform a task through local orchestrator components.

Component Type	MPMS consists of (see also Functional Architecture):
	
	• Modeler (Camunda Modeler) – Desktop app
	  ○ a web-based app can also be used (Cawemo)
	• Process Engine (Camunda Process Engine) – Java application or remote REST service (running in background)
	• Core application – Java application (running in background)
	• Tasklist / Cockpit (Dashboard) / Admin applications – Web-based on PC/Laptop/Smartphone/Tablet
	  ○ Custom applications can also be built, even for smartwatch

Development environment	The supported environments have been listed above. Below we specify the ones that MPMS runs on its current state (as developed, tested and used in the HORSE project pilot cases and in SHOP4CF pilot cases) MPMS (runtime components) runs in every Java-runnable environment and through Docker. Any IDE can be used to develop the executable process models (e.g. Eclipse, IntelliJ IDEA)

	Container/Application Server for runtime components • Wildfly Application Server 18.0 Databases • PostgreSQL 10.4 ○ postgres:14-alpine (Docker image for Docker deployment) • H2 1.4 • Adminer:4.8.1 (UI for DB management) (Docker image for Docker deployment) Web Browser • Google Chrome latest • Mozilla Firefox latest • Internet Explorer 11 • Microsoft Edge Java • Java 8 Java Runtime • Oracle JDK 8 • openjdk:11.0.13-jre-slim (Docker image for Docker deployment) Camunda Modeler • Windows 7 / 10
TLR	According to the TRL definitions of "Technology readiness levels (TRL); Extract from Part 19 - Commission Decision C(2014)4995" (PDF). ec.europa.eu. 2014, MPMS is on TRL 6 (Technology demonstrated in relevant environment) towards TRL 7 (System prototype demonstration in operational environment), as demonstrated in HORSE and EIT OEDIPUS projects. The core components such as the Modeler, the Process Engine, and the default web applications for Tasklist, Cockpit/Dashboard (and Admin) are stable (dependant on the robustness of the Camunda Platform Community edition). Extra plugins and features have been built and need further tests/improvements. Moreover, extra functionality is under development. The functionality of the dynamic agent allocation and exception handling are dependent on the use cases and require a good design by modelers and developers.

Component usability

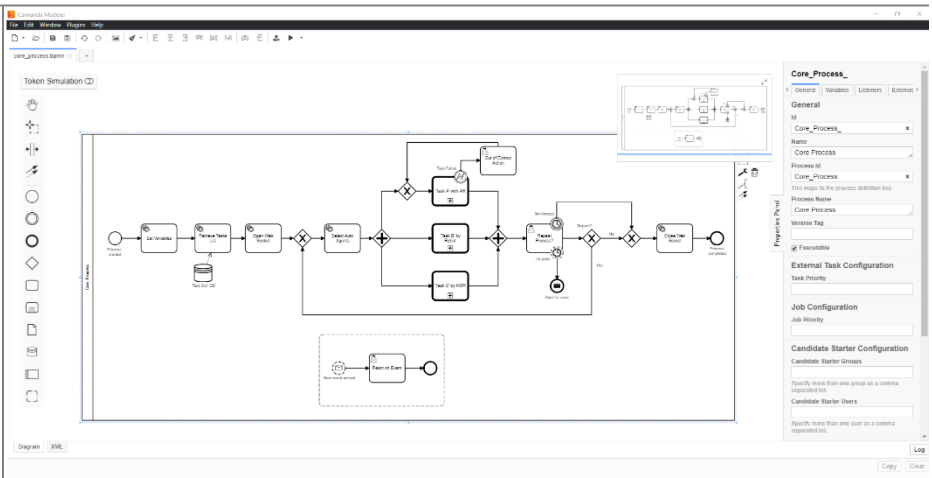


Figure 1 Process Modeler

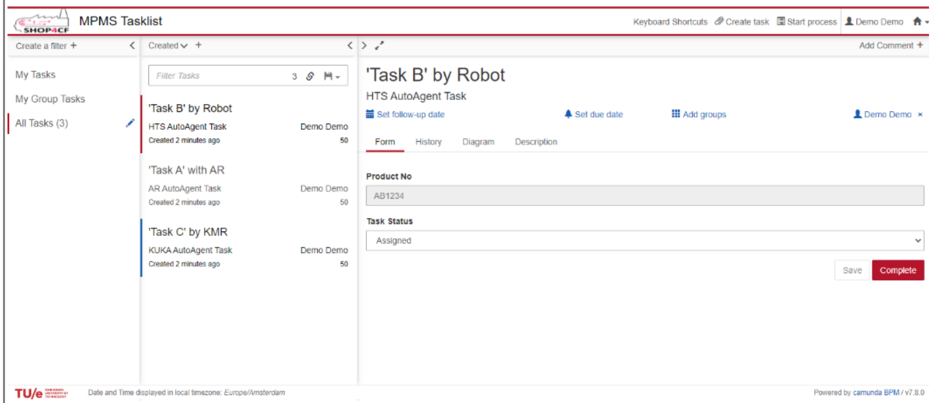


Figure 2 Tasklist application

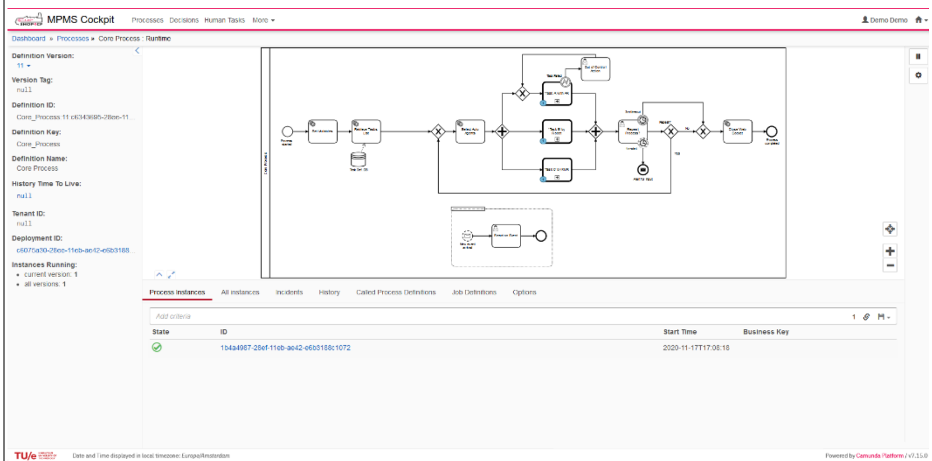


Figure 3 Cockpit application

User manuals, videos and webinars for each type of MPMS subcomponents can be provided upon request

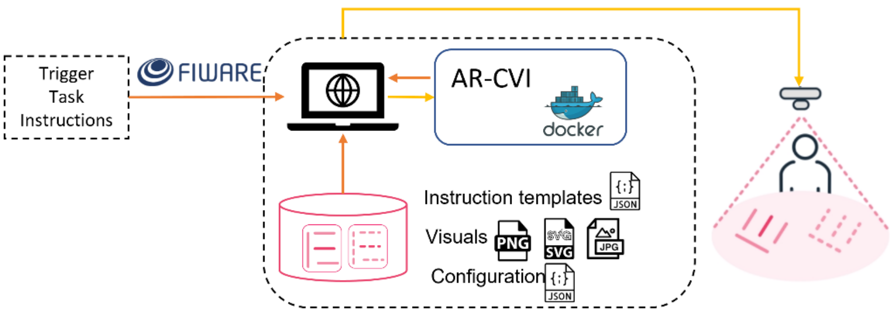
Versions

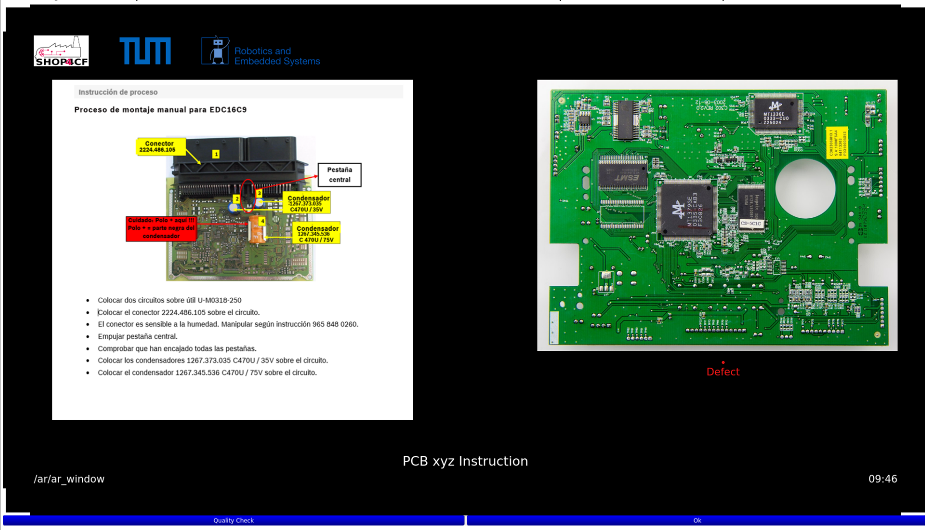
As said before, MPMS runs on Camunda Platform. Currently used version is:

- Modeler (Camunda Modeler) – version 4.11.1
- Process Engine platform (Camunda Community Platform) – version 7.15.0
- Core package Available as freeware
- Premium version (with advanced functionality) Upon agreement with TUE

	Detailed info on versions provided in T7.2 (Licensing Questionnaire)
Instruction video	https://www.youtube.com/watch?v=gdgRm6DkAyc
GitHub link	Initial version of source code provided on https://github.com/SHOP4CF/MPMS There are plans to provide a new repo.
RAMP link	N/A
Docker registry	N/A

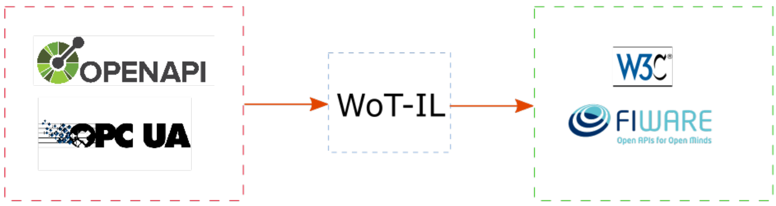
AR for Collaborative Visual Inspection/AR for Task Instructions

AR for Collaborative Visual Inspection	
AR-CVI	
TUM	
Application area	Supporting human workers on the shop floor
Main functions	The component provides visual support in manual assembly and inspection tasks. Human worker is guided by the visualized instructions while performing the associated tasks. The component can project the instructions on a surface or a screen. This provides a clean and structured working environment.
Interfaces and data - Input	Inputs: • Configuration file (JSON) • Visual instructions to be projected (PNG, JPG, SVG) [Optional] • Instruction templates (JSON) [Optional] • Trigger to start visualization of the instructions (Fiware message in REST API using the defined Task Data Model)
Interfaces and data – Output	Outputs: • Visualized instructions on the screen or on the projected area • User input by clicking the displayed buttons (Fiware message in REST API using the defined Task Data Model) [Optional] (This can be taken as inputs to other components)
Functional architecture diagram	
Main non-functional requirements	The Fiware messages are checked at 2 Hz.
Software requirements/dependencies	Ubuntu: • Ubuntu 18.04 or later (for running on Ubuntu) • Docker version 20.10.6 (Previous versions later than v19 are also supported) Windows: • Windows 10 • WSL 2 • Docker version 20.10.6 (Previous versions later than v19 are also supported) • Windows X Server (for running on Windows)
Hardware requirements	• A screen with at least 1920x1080 (HD) resolution support • A projector [Optional] with at least 1920x1080 (HD) resolution. Front surface mirrors might be necessary for projecting down to a table. High brightness according to the illumination of the environment (~4400 Lumens) • A PC or laptop (Preferably a 4-core CPU with 8 GB RAM, 15 GB HDD space)

Security threats	No specific security threats
Privacy threats	No specific privacy threats.
Execution place	The component should be executed in the assembly cell pc (local deployment). Necessary files can be mounted to the Docker container from the host machine or from a remote machine.
Deployment instructions	Currently in the public Github repository: https://github.com/emecercelik/ar-cvi/ Later the instructions will be available in RAMP.
User interface	• The instructions are displayed in a full-screen mode. • User can provide inputs to the component using the buttons on the screen if defined with the display messages (templates). A screen view is provided below: Instruction images, PCB quality check outputs (provided by another component), a written instruction, and buttons (at the bottom).  For a developer demonstration refer to the following video: https://syncandshare.lrz.de/getlink/fiC2jAGB9vBsmaRqtVVmzEg4/ar-cvi_demonstration.mp4
Supported devices	PC, Laptop, Projector, Screen
User defined scenarios (non-technical) and relevant pilot cases	Support on the assembly cell for the human worker by displaying manuals of an assembly or inspection task. The operator can provide inputs to start the inspection (provided by another component).
Roles/Actors	Human worker on the shop floor performing assembly or inspection.
Component Type	PC program that runs on Docker.
Development environment	ROS, QT, C++, Python
TLR	Tested for the Fiware support. The functionality is being tested in the SHOP4CF Pilots. Currently, TRL 4 is reached. The expectation is to reach TRL 5-6 by the end of project.
Component usability	The documentation can be reached from https://github.com/emecercelik/ar-cvi/ A demonstration video can be downloaded from https://syncandshare.lrz.de/getlink/fiC2jAGB9vBsmaRqtVVmzEg4/ar-cvi_demonstration.mp4
Versions	A free version with the full functionality is offered as open source (BSD License).
Instruction video	https://youtu.be/zYbppS1ExBQ
GitHub link	https://github.com/emecercelik/ar-cvi/

RAMP link	Will be available soon.
Docker registry	emecercelik/ar-cvi:ar-cvi_v1

Interoperability Layer through Web of Things

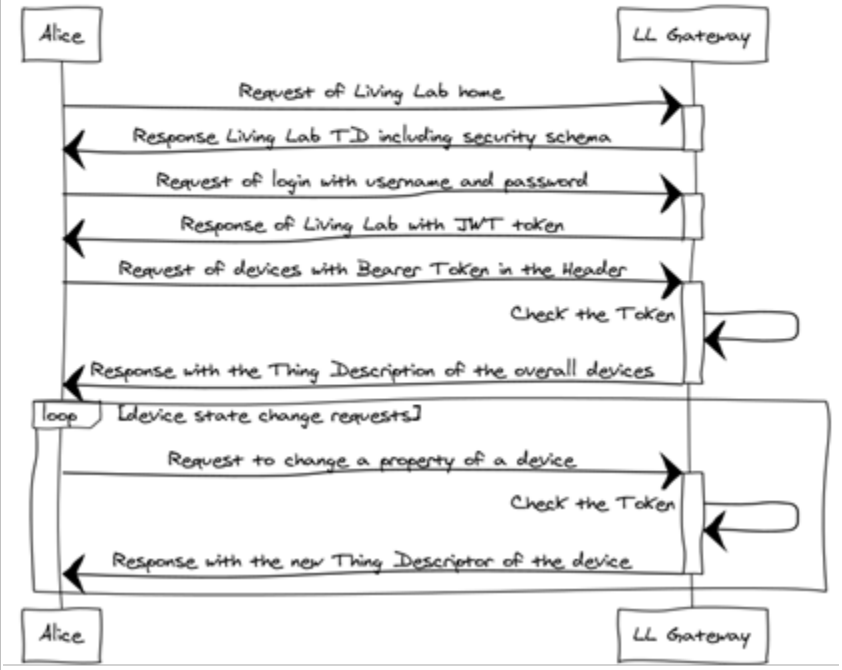
Interoperability Layer through Web of Things	
WoT-IL	
UPM	
Application area	It supports the process management improving the interoperability of the system. It addresses the ability of the system to be standard interoperable.
Main functions	1) Translation of OpenAPI specification into Web of Things- Thing Description. 2) As an evolution within this project, now the component allows communication between OPC UA and Fiware, applying the SHOP4CF data model
Interfaces and data - Input	For function 1: • Input: A JSON object that describe the REST-API (OpenAPI) For function 2: • Input: Data from an OPC UA server
Interfaces and data – Output	For function 1: • Output: A JSON object that describe the Thing Descriptor of the REST-API For function 2: • Output: Data described applying the SHOP4CF data model
Functional architecture diagram	
Main non-functional requirements	In the case of OPC UA communication, the server must be configured and deployed. The Node ID (s) must be provided to be able to read the information in real time.
Software requirements/dependencies	• Node JS and/or Java but it can be developer also with other programming languages. • Docker and Docker compose • Orion LD Context Broker
Hardware requirements	• 16 Gb RAM, 10 Gb HD • Linux - Ubuntu
Security threats	It wraps REST APIs in another descriptor, the component does not manage the security of the APIs described. The server that serves the descriptor will be secured through HTTPS and certificate.
Privacy threats	No specific privacy threats.
Execution place	The component should be executed in a local PC on the shopfloor
Deployment instructions	We provide a docker container with a configuration file.
User interface	In this version a Command Line Interface has been implemented. It is expected to develop a graphical interface for the next version.
Supported devices	PC/Laptop
User defined scenarios (non-technical) and relevant pilot cases	A pilot or a SHOP4CF component owner wants to provide to the external word some functionalities he wants to test with external users (for instance for open

	calls), he builds a WoT (Web of Things) interface with our component that wraps its component to be used by third party developers.
Roles/Actors	Developers
Component Type	Software module
Development environment	Visual Studio Code / Node JS / TypeScript
TLR	From TRL 4 to TRL 6
Component usability	See extended info.
Versions	Free version: component with script Paid version: component with an UI for including semantic contexts
Instruction video	Provide a link to a video describing the component.
GitHub link	https://gitlab.lst.tfo.upm.es/shop4cf/upcua https://gitlab.lst.tfo.upm.es/shop4cf/cli
RAMP link	Will be available soon.
Docker registry	N/A

Extended info

Web of Thing Gateway	
A home gateway in order to map KNX devices and web-based devices within the Web of Things interface	
Scope/Target	This is a part of the Living Lab of the UPM. The system can be scaled on any other KNX installation and also web based devices can be plugged.
Application Type	Home Appliance Service
Application execution place	Gateway
GPU	No
Actors	The service is designed for other application that want to control home environment (e.g. lights, doors, windows, etc..)
User interface	-
Need of training	Yes, it needs knowledge on Web of Things and how to get/set properties it is based on the Mozilla WoT Specification using HTTP GET in order to read a property and PUT in order to set a property: https://iot.mozilla.org/wot
TLR	4
System Validation	Validated within Living Lab at UPM
GDPR Features embedded in the system	no
Support	Ask to UPM

IoT Platform/Cloud component		
Where it is running?	On UPM premises	
	HW requirements	16 Gb RAM, 10 Gb HD
	SW requirements	Linux - Ubuntu
	Statistics on resource usage	No statistics included
	Virtual machine	Now is running in a Docker installed in a VM Linux - Ubuntu

	Containers	Docker
	Bandwidth requirement	Low
	Internet access	Yes
	Data collection	No sensible data collected, stateless application
Interoperability capabilities	REST API and Web of Things	
Data flow		
Communication standards	Web of Things and REST-API, Based on Web of Things Mozilla implementation, each device is implemented as a property: https://iot.mozilla.org/wot/#property-resource	

```
{
  "security": [
    {
      "authorizationUrl": "/auth",
      "scheme": "bearer",
      "format": "JWT"
    }
  ],
  "@type": [
    "Thing"
  ],
  "name": "Smart Home Living Lab Devices",
  "@context": [
    "http://iot.schema.org"
  ],
  "properties": {
    "devices": [
      {
        "name": "lights_OvenWindow",
        "type": "Thing",
        "forms": [
          {
            "http:methodName": "GET",
            "href": "/things/5c581b15f900581c329585d2"
          }
        ]
      },
      {
        "name": "door_BathRoom",
        "type": "Thing",
        "forms": [
          {
            "http:methodName": "GET",
            "href": "/things/5c581b15f900581c329585d3"
          }
        ]
      },
      {
        "name": "lights_DiningRoominterior",
        "type": "Thing",
        "forms": [
          {
            "http:methodName": "GET",
            "href": "/things/5c581b15f900581c329585d4"
          }
        ]
      },
      {
        "name": "lights_TVRoom",
        "type": "Thing",
        "forms": [
          {
            "http:methodName": "GET",
            "href": "/things/5c581b15f900581c329585d5"
          }
        ]
      },
      {
        "name": "blind_BedRoom",
        "type": "Thing",
```


Security mechanisms	Json Web Token
Privacy mechanisms	HTTPS

Conclusion

A comprehensive library of components has been developed, to provide modular solutions for the process automation industry. By using Docker and having instruction videos, implementing these components can be done with minimal effort, allowing more European companies to create connected factories and automation solutions with a human-centric approach. The unified interfaces guarantee easy integration and allow users to mix and match components to create their own unique Industry 4.0 solutions.