Open-Digital-Industrial and Networking pilot lines using modular components for scalable production

Grant Agreement No	:	**101017141**
Project Acronym	:	**ODIN**
Project Start Date	:	**1st January, 2021**
Consortium	:	**UNIVERSITY OF PATRAS – LABORATORY FOR MANUFACTURING SYSTEMS AND AUTOMATION**
		FUNDACION TECNALIA RESEARCH & INNOVATION
		KUNGSLIGA TEKNISKA HOEGSKOLAN
		TAMPEREEN KORKEAKOULUSAATIO SR
		COMAU SPA
		PILZ INDUSTRIEELEKTRONIK S. L.
		ROBOCEPTION GMBH
		VISUAL COMPONENTS OY
		INTRASOFT INTERNATIONAL SA
		GRUPO S21SEC GESTIÓN, S.A.
		FUNDACION AIC AUTOMOTIVE INTELLIGENCE CENTER FUNDAZIOA
		DGH ROBOTICA, AUTOMATIZACION Y MANTENIMIENTO INDUSTRIAL SA
		PSA AUTOMOBILES S.A.
		AEROTECNIC COMPOSITES SL. U.
		WHIRLPOOL EMEA SPA

Title	:	ODIN Core Enabling technologies for smart programming and HR interaction interfaces
Reference	:	D2.2
Availability	:	Public
Date	:	30/06/2022
Author/s	:	TECNALIA, TAU, LMS
Circulation	:	EU, consortium

Summary:

This document focuses on the presentation of the initial prototypes of core enabling technologies for a) easy programming and b) human side interfaces.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

This document provides a detailed description of the initial prototypes of core enabling technologies for easy programming and human robot interaction interfaces.

- Easy programming interfaces – Task 2.4. The following key point technologies will be presented in this document:
  - ODIN use cases specific skill development,
  - Advanced behavior management,
  - On-site interactive skill programming system, and
  - CAD Programming improvements for new skill integration.
- Human robot interaction interfaces – Task 2.5. The initial prototype of the following key technologies will be presented in this deliverable:
  - ODIN Augmented Reality Application,
  - ODIN Smart User interfaces, and
  - Virtual Reality safety training.

Each technology will be integrated inside ODIN pilot lines based on end users' requirements as documented in D1.1.

Running the M18 of ODIN, the first prototypes of the previously documented technologies have been prepared by the partners and will be presented in this deliverable alongside with their first implementation tests.

1. INTRODUCTION

The use of Skill Based Programming (SBP) frameworks allows implementing robotic applications sequencing configurable pre-programmed blocks (skills). It has been demonstrated being very useful for easy and fast deployment of relatively simple operations such as pick and place, inspection, handling a device for performing a process (e.g. drilling, deburring or screwing), navigation, etc. The operations that, can be generalized, may benefit of SBP approaches allowing a high degree of re-usability. The SBP techniques, combined with CATIA software-based CAD Programming concept for skill configuration, allow the plant operators to take advantage of the intrinsic information that CAD models contains in order to configure new processes using pre-programmed skills. The use of an extended and well-known software that the operators are habituated to use, empowers them for easy and quickly adaptation of existing robot programs. In this way, the flexibility of the line is increased. As mentioned above, SBP is ideal for tasks that can be generalized for a future re-use. However, there are operations that could require dexterous movements, or with a limited accessibility which makes difficult the collision free trajectory planning and execution.

Smart User Interface (UI) creates new interaction modalities that enable informative and Real-Time communication in industrial human-robot assembly scenarios. The integration of digital light processing (DLP) projector and depth sensors provides operator's position detection and by utilizing marker-based approaches, a UI can be projected on a surface – even a mobile one. This application is utilized in pick and place tasks where virtual buttons and textual descriptions allow intuitive communication between human and manufacturing resources. The setup can also be used to provide help annotations for the operator, for example highlighting part position in the assembly operation. Visualizing work area of the robot with the projectors can increase operator's awareness of the robot movement and therefore decrease the number of violation events.

A virtual safety training system provides a safe training environment for industrial robotic applications, where the human and robot are working in close proximity. Lack of communication between robots and humans creates challenges for human perceiving and decision making during hazardous events. The virtual training application represents a replica of a robotic work cell and familiarizes the operator with safety measures. Feel of presence in virtual reality applications has been studied and proved to have a positive effect on human perception. In this set-up, the Unity [9] game engine features are utilized for creating 3D visualization content of the pilot line. It is worth mentioning that all visualizations could be exported as a template for further use in other applications. These templates improve the speed of application development at next use cases. The assembly training phase provides assembly tasks while the operator can see effect of violation mistakes with industrial indicators in the manufacturing sector. This two-phase training provides an intuitive immersive environment for acknowledging the operators regarding safety measures before stepping into the real robotics environment.

Easy programming interfaces combined with human side interfaces provides a powerful ecosystem for empowering the operators in different industrial settings belonging to disparate sectors.

The initial prototype of easy programming interfaces will be presented in Section 2, while initial prototypes of human robot interaction interfaces will be documented in Section 3.

2. INITIAL PROTOTYPE OF EASY PROGRAMMING INTERFACES

The main goal of providing easy programming interfaces is to equip the human operator with the required tools to create, modify and manage robotic applications based on pre-programmed robot skills. In order to develop powerful enough tools, but maintaining their usability, special efforts have been made on the following sub-tasks:

- Process specific robot skill development. Previously developed robotic skills can be re-used but depending of the use case peculiarities new skills must be implemented. An analysis, task sub-division and generalization iterations are being performed for new skills development.
- Advanced behavior management: The Skill Based Programming (SBP) could provide with an infinite set of skill combination, but the interaction between blocks determines overall system usefulness. Depending of the required logic complexity, hierarchical states machines or behavior trees will be allowed for handling the required interaction graphically.
- On-site Interactive Skill Programming system (OISP). Some tasks require very dexterous combination of movements (final assembly steps, sealant application, complex trajectory execution, etc.). Enhancing SBP with OISP will enable the SBP for integrating the knowledge that is simple for a human but difficult to be generalized for a robot.
- Easy to use CAD Programming interface: An analysis of required parameters is being done, selecting the key parameters and defaulting the rest of them. In this way the flexibility of the system is increased.. Besides, the CAD Programming interface will allow integrating new skills dynamically.

2.1. ODIN use cases specific skill development

Throughout the ODIN project, diverse set of skills will be developed. These skills will envelop the required sequence of robot movements, sensor captures, IO signals, end effector operations, mobility, etc.

2.1.1. Aeronautic Pilot

For the aeronautic pilot the following skills are being developed:

- **Navigation and docking**

The robotic platform requires being able to navigate and dock in an appropriate location.

Table 1: Navigation skill parameters

Parameter	Description
topic	ROS [5] action topic
goal	Target goal for the mobile platform
frame	Parent frame name

Table 2: Docking skill parameters

Parameter	Description
dock	True for docking and False for undocking
camera	Desired camera
reference	Distance to the marker reference
tolerance	Requested maximum tolerance error

- **Grab/Release tool**

The robotic platform requires being able to equip drilling machines at drilling stations for starting operation.

Table 3: Grab tool skill parameters

Parameter	Description
tool_name	Desired tool name to equip
robot_arm	Desired robotic arm to use

Table 4: Release tool skill parameters

Parameter	Description
tool_name	Desired tool name to equip
robot_arm	Desired robotic arm to use

- **Teach/execute trajectories**

There are some tasks that require dexterous trajectory execution. For that purpose, trajectory teaching, processing and executing skills are being developed.

Table 5: Teach trajectory skill parameters

Parameter	Description
arm	Robotic arm to be used
tool_id	Equipped tool id
cam_id	Camera to be used (optional parameter)
estimate_pose	True if vision is used for recording relative trajectories (optional parameter)
estimate_pose_service	Service for pose estimation (optional parameter)
joystick	True if a joystick is used for guiding the robot (optional parameter)

Table 6: Execute stored trajectory skill parameters

Parameter	Description
traj_id	Desired trajectory id for executing
tag	Refer trajectory by name (optional)
arm	Robotic arm to be used
vel	Desired vel for executing the trajectory
sim_io	True if simulated IO are used
marker_detection	True if marker detection is used for relative trajectory execution

- **Detect drilling template**

The drilling templates must be detected for a precise localization. After template detection the position of the drilling holes can be inferred based on the CAD information.

Table 7: Detect drilling template skill parameters

Parameter	Description
object_id	Object to be detected

- **Drill**

After drilling frames have been detected the drilling process can start. A skill which performs collision free trajectories and drilling machine management is required.

Table 8: Drill skill parameters

Parameter	Description
robot_group	Desired robotic arm
part_name	Drilling template to be used
drill_hole_frame_id_list	List of drill frames to be drilled
tool_frame_id	Desired tool

- **Grasp FC trolley**

The robotic platform requires being able to grasp FC trolley in order to transport along the workshop to other cells.

Table 9: Grasp FC trolley skill parameters

Parameter	Description
grasp_trolley	True for grasping, False for releasing

- **Transport**

When the FC is grasped by the robotic platform skill for transporting the FC to any location is required. Transport skill is a special case for navigation skill.

Table 10: Transport skill parameters

Parameter	Description
topic	ROS action topic
goal	Target goal for the mobile platform
frame	Parent frame name

- **Scan**

The scan skill will allow generating a model which will be used as reference for future inspections.

Table 11: Scan skill parameters

Parameter	Description
fc_id	FanCowl id
arm	Desired robotic arm
trajectory_ids	Trajectories that the robotic arm will follow
camera_id	Camera to be used
topic	Topic with image source

- **Inspect**

The inspection use case requires a skill which allows performing an inspection of installed devices in a FC. Inspect skill is using the data generated by the scan skill.

Table 12: Inspect skill parameters

Parameter	Description
fc_id	FanCowl id
arm	Desired robotic arm
trajectory_ids	Trajectories that the robotic arm will follow

Parameter	Description
camera_id	Camera to be used
topic	Topic with image source
reference_model	Reference model to be compared against

2.1.2. Automotive Pilot

For the automotive pilot the following skills are being developed:

- **Screwing while moving**

The screwing while moving skill allows screwing the provided screws on the motor for required parts installation on it. This skill uses the visual servoing techniques developed on T2.2.

Table 13: Screwing while moving skill parameters

Parameter	Description
arm	Robotic arm to be used
camera	Camera to be used
v_tolerance	Linear tolerance (m)
w_tolerance	Angular tolerance (rad)
time_goal	Desired timespan to complete approximation (sec)
camera_H_marker	Desired marker pose to be controlled
timeout	Max allowed time to complete approximation (sec)
ns_marker	Namespace for the current pose topic

- **Navigation and docking**

Before starting the screwing while moving action, it is required for the robotic platform to able to navigate and dock in an appropriate location (Table 1 & Table 2).

2.2. Advanced behavior management

The robot programming should be a simplified and functional process for generating robotic applications in an easy way. On the one hand, an intuitive GUI is required, and on the other hand, an advanced enough backend which must be able to feed the GUI with the required information is also imperative. The skills must be connected with this human-machine interface to in order to be used.

The sequence proposed in ODIN for easy robot programming process is illustrated in Figure 1. The process is composed by the following steps:

1. Drag and drop skills for sequencing the desired operations.
2. Parametrize skills through the Skill Parametrization Wizard.
3. Teach through Skill Teaching Assistant the skills that require an onsite interactive teaching process (described in Section 3.3).
4. Generate executable files with the configured process.

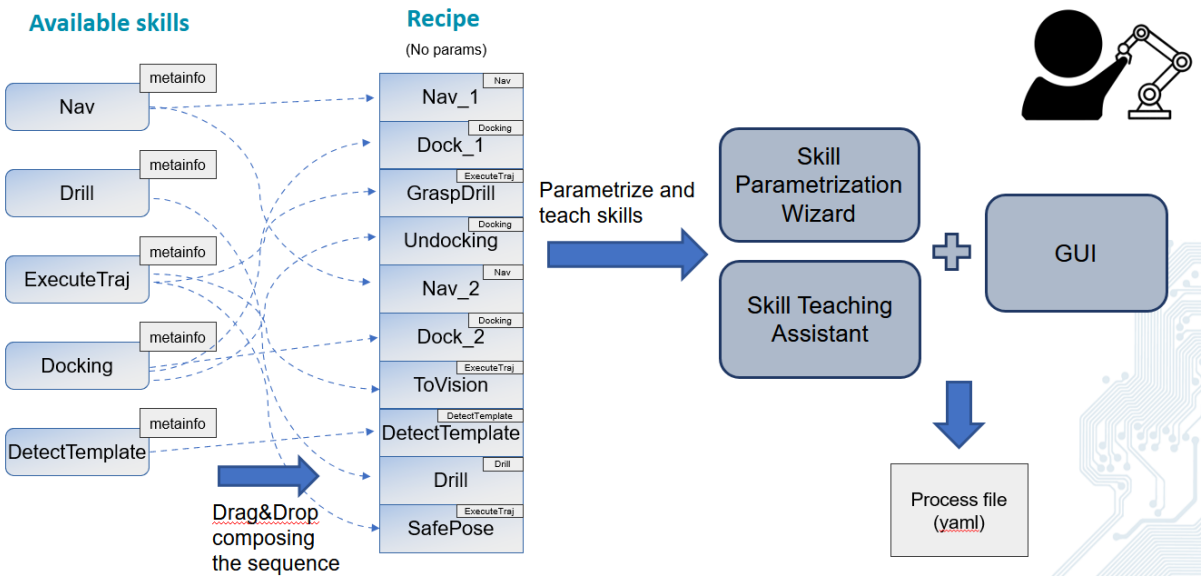

Figure 1: Preliminary design

Regarding the first step of the aforementioned process, the first prototype's implementation is based on Blockly concept [1]. The set of available skills will automatically be recovered by the system thanks to metainformation files which will expose the desired skills for being used in the GUI (Figure 2).

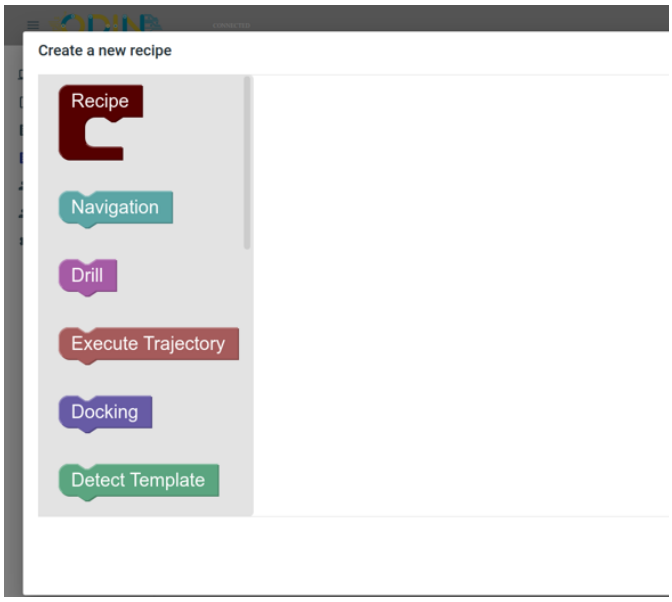

Figure 2: Blockly based first prototype

The first prototype of metainformation developed in the ODIN project contain the following fields:

- Spec: The skill specification, i.e., the location of the skill executable code.
- Params: The parameters of the skills. Contains the type, source, default values, and information.
- Result: The result of the skill.
- Pre-conditions: Conditions that must be met before starting the execution.
- Hold-conditions: Conditions that must be met during the execution.
- Post-conditions: Effects that the skills could produce.
- Teaching assistant: The information related to skills that require a teaching step.

The metainformation is stored in a YAML file and each skill needs to have a corresponding metainfo file to be used by the system (Figure 3).

```
spec: !Skill
      skill: flexbotics_manipulation_utils:ManipulationUtils.move_tcp
params:
  - group:
      type: str
      source: ['right_arm', 'left_arm']
      example: 'right_arm'
      info: Desired robot manipulator
  - pose:
      type: str
      source: !TFROS
              namespace:
              filter: pose
      example: 'right_pose_1'
      info: Desired target pose
pre-conditions:
  ros_com:
    services:
```

Figure 3: Metainformation file for feeding the GUI

The Blockly based interface prompt a window for parametrizing the skills with the information contained in the metainfo YAML files (Figure 4).

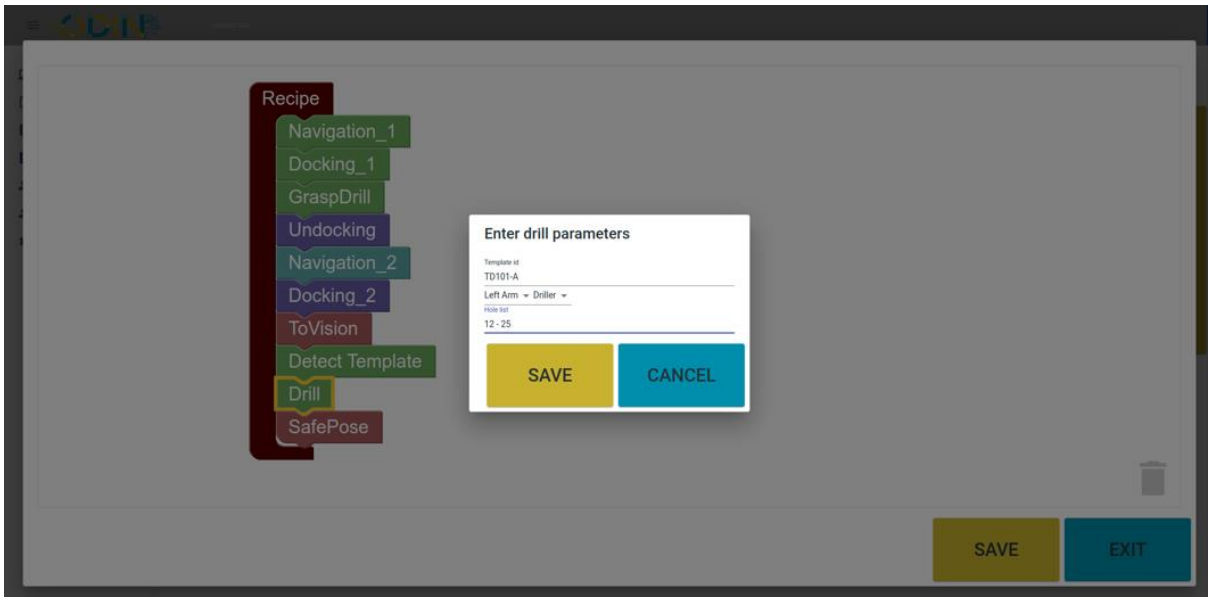

Figure 4: Skill parametrization based on metainfomartion

One of the claimed innovations is the concept of "source" of the parameters. This block allows feeding the interface from different sources of information with a simple syntax. The planned system will allow the following sources of information to be used:

- Fixed lists of elements (implemented in the current prototype)
- ROS TF queries + filters (implemented in the current prototype)
- ROS Topic queries + filters (implemented in the current prototype)
- ROS Service queries.
- YAML files.
- Result values of previously executed skills

2.3. On-site Interactive Skill Programming system

Some complex operations cannot be easily implemented using SBP approaches. For filling this gap an On-site Interactive Skill Programming system is being developed in ODIN project. Some examples of complex operations are the precise movements in narrow spaces, agile pose recording, big part manipulation, etc. The OISP allows taking advantage of human dexterity for guiding the robot or be able to use a joystick in case of dangerous environment (Figure 5).

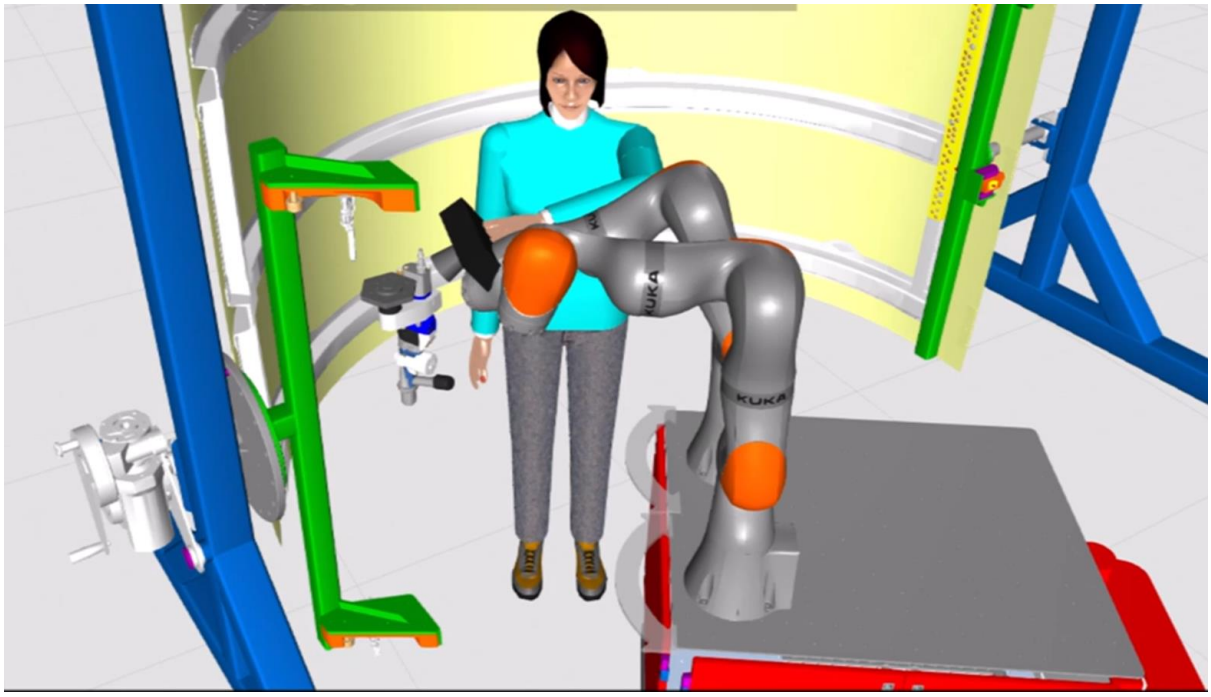

Figure 5: Dexterous trajectory teaching through manual guidance or joystick

Through the GUI, the operator is able to select the arm in order to start the teaching operation using a simple and easy to use interface (Figure 6).

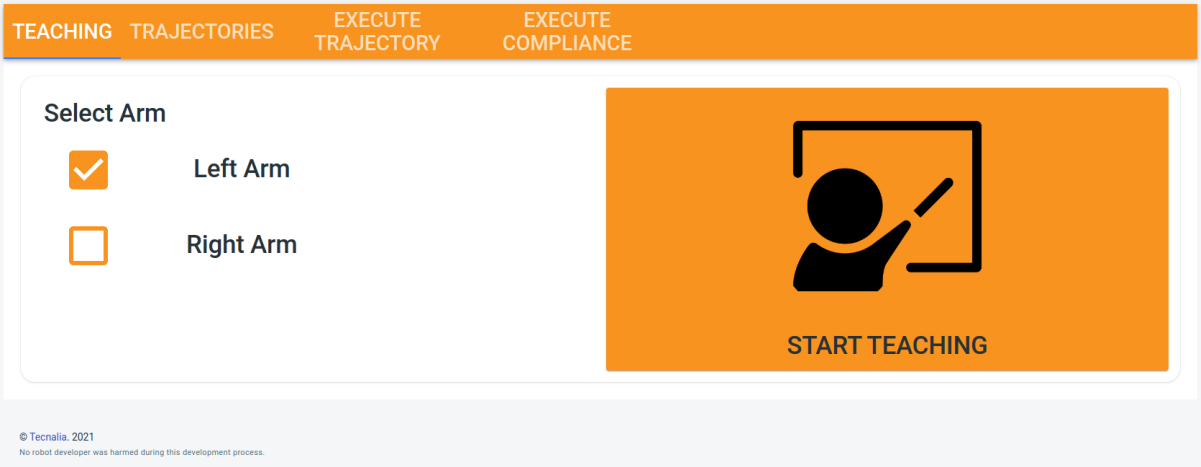

Figure 6: Start Teaching Assistant window

Relevant poses may be added during the teaching of new robot trajectories. These relevant poses add functionalities to the trajectory. Through the “Add relevant pose” button, the system will ask the typology of the relevant pose and it will be stored with the trajectory in a database (see Figure 7).

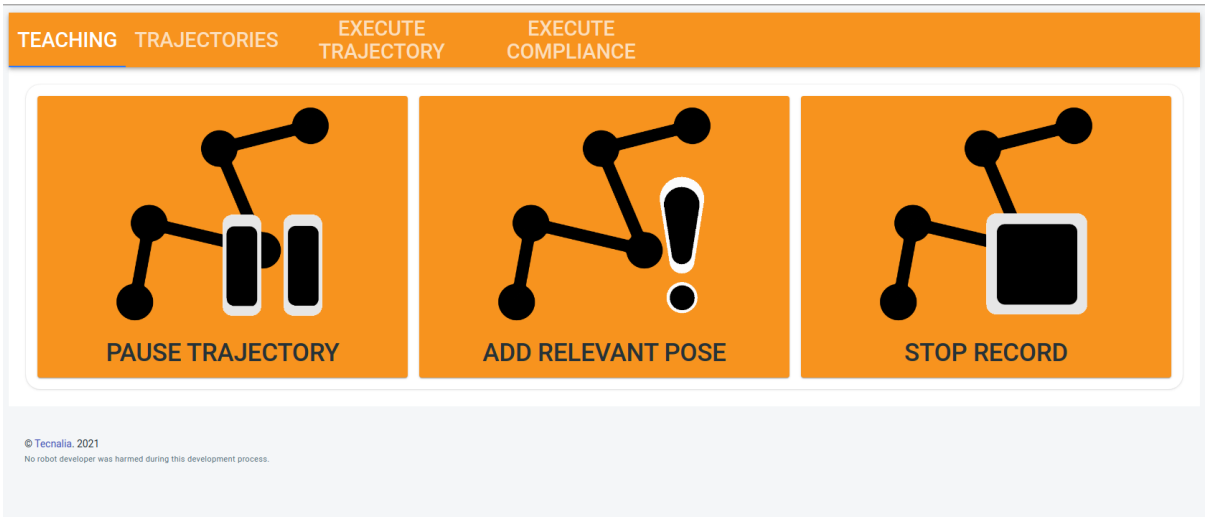

Figure 7: GUI for adding relevant poses in the taught trajectory

This approach allows the creation of relative trajectories based on a vision marker detection solution, i.e., the trajectory can be stored as relative to a marker. In this way, regardless any changes on robotic platform's location in the layout, the robot trajectory might be reproduced without any changes. Additionally, Figure 8 visualizes the set of relevant poses that can be added for other scenarios.

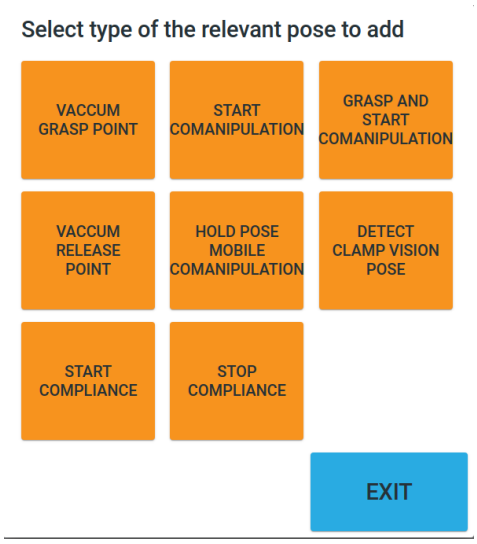

Figure 8: Available relevant poses

2.4. CAD Programming improvements for new skill integration

The dynamic definition of skills is investigated in ODIN to increase the flexibility of the CAD programming. Figure 9 and Figure 10 presents how the list of the skills from an XML file can be visualized in the CAD Programming interface.

The skills have been associated with a type of field which represent the parameters that will be retrieved from the CATIA cell. Thanks to the experience of previous projects (VERSATILE, RECAM, THOMAS) the most common types of interaction with CATIA information have been compiled in different types of skills. This will determine if a skill require selecting a frame, a part, a pick and place operation, or different combinations of them.

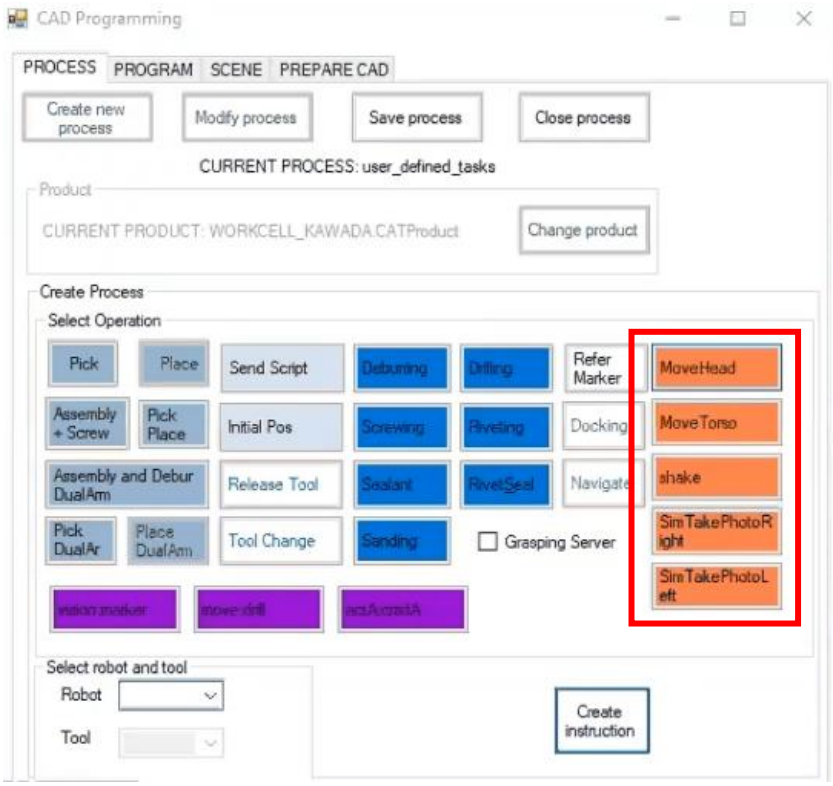

Figure 9: CAD Programming module with dynamically loaded skills

```
<user_skills>
  <skill name="MoveHead" type="0" method="move_head" />
  <skill name="MoveTorso" type="0" method="move_torso" />
  <skill name="shake" type="0" method="shake" />
  <skill name="SimTakePhotoRight" type="0" method="sim_take_photo_r" />
  <skill name="SimTakePhotoLeft" type="0" method="sim_take_photo_l" />
</user_skills>
```

Figure 10: Skills defined in the XML file will be dynamically added to the GUI

3. INITIAL PROTOTYPE OF HUMAN ROBOT INTERACTION INTERFACES

3.1. ODIN Augmented Reality (AR) Application

An AR application is being developed in ODIN by LMS aiming to provide support to human operators during the execution of the required assembly tasks. This application will be deployed in AR devices such as AR glasses or/and Tablets (Figure 11) based on end users' requirements. Using these devices, the operators have a better overview of the production status and the running tasks assigned to the different resources but also interact with the system using virtual objects visualized in the virtual environment of the AR application.

Figure 11: ODIN AR devices: a) AR glasses and b) Tablet

ODIN AR application will be used in the White Goods and Automotive pilots providing different type of support to the operators.

- **Automotive Pilot**

The Automotive pilot of ODIN focuses on the assembly process of motor and gearbox. In more details, the automotive pilot is divided in three operations namely a) the motor & gearbox assembly, b) the additional parts assembly and c) the quality check.

During the motor and gearbox assembly operation, the operator (Operator_1 in Figure 12) can receive an order through the AR application and he/she is responsible for positioning and connecting the motor with the gearbox. An amount of physical effort is needed in order for the operator to perform this kind of connection and thus it would be beneficial to support this task with the use of an industrial manipulator. Furthermore, the human operator is performing some screwing tasks in order to finalize this connection.

Regarding the additional parts assembly operation, the target is to connect some further parts on top of the motor and gearbox. Depending on the model of the motor and the customer, the kind of parts may vary, and the operator should be precise. The operator (Operator_2 in Figure 12) can be informed about the exact parts which are required to be installed on a motor and gearbox through the AR application.

Finally, the quality check operation is focused on validating that all the required parts and connections are correctly assembled. This operation is extremely important for STELLANTIS, because in case of any fault is bypassed during this task, then there will be a loss of cost and the faulted motor will proceed to the body & white task. As a result of the quality inspection, in case some parts need to be checked by a human, an operator enters this working area and using the AR application he/she may digitally mark a part as correctly installed though virtual buttons of the AR application.

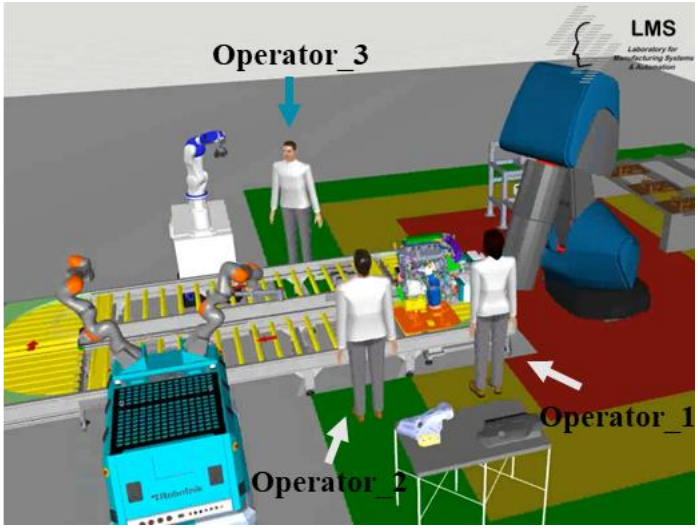

Figure 12: Operators of Automotive pilot

- **White Goods Pilot**

This pilot focuses on the efficient collaboration of a Universal Robot UR10 [3] and an operator for the installation of required components in an Oven and a Gas cooktop. In more details, a human operator and the selected cobot work together in a shared workstation to install a transformer and a set of cooktops and knobs in an Oven and a Gas cooktop accordingly. The operator equipped with AR glasses is aware of the assigned tasks to him/her through textual information visualization inside the virtual environment of the AR glasses (see Figure 13).

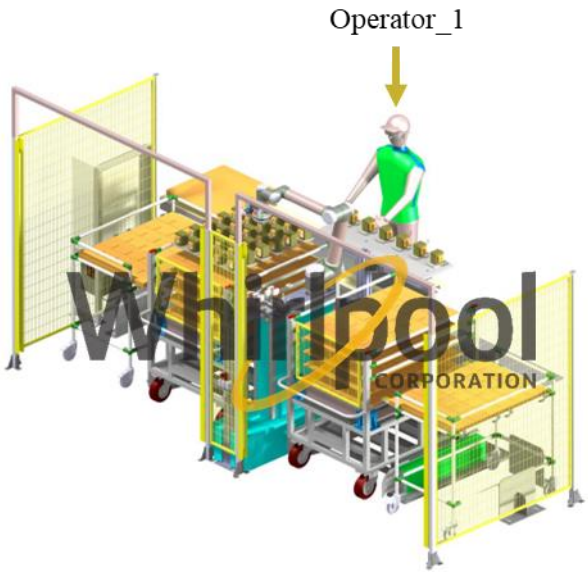

Figure 13: Operator of White Goods pilot

This deliverable focuses on the development of an initial prototype of ODIN AR application.

3.1.1. Features of ODIN AR application

Based on end users' requirement, a set of AR features have been defined for the ODIN AR application. The features of ODIN AR application are presented in the following table alongside with the ODIN pilots each feature will be implemented in.

Table 14: ODIN AR features and implemented pilots

Feature	Title	Automotive Pilot	White Goods Pilot
AR_F1	Visualization of Human assigned tasks through AR	X	X
AR_F2	Task Completed virtual button	X	X
AR_F3	Robot trajectory visualization inside the virtual world	X	X
AR_F4	Easy robot programming using interactive virtual tool	X	X
AR_F5	Security alarm visualization	X	X
AR_F6	Error handling in case of detection process failure	X	X
AR_F7	Resilience – execution error recovery	X	X
AR_F8	Mobile robot's safety zones visualization through AR Glasses	X	
AR_F9	Quality inspection results visualization, digitally mark for validation and instructions provision for corrections	X	
AR_F10	Alarm indicating that the robot is working in collaborative area		X

3.1.1.1. AR_F1: Visualization of Human assigned tasks through AR

The main usage of ODIN AR application is to inform human operators about the assembly tasks assigned to them. In ODIN, human operators are aware of their assigned tasks through textual information visualized in the virtual world of the AR devices. The sequence diagram of message exchange between the OpenFlow and the AR application is presented in Figure 14.

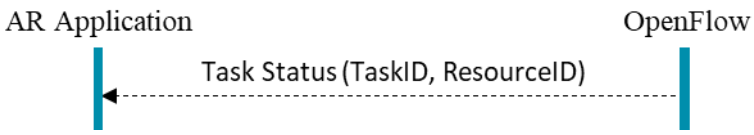

Figure 14: AR_F1 – Message exchange sequence diagram

3.1.1.2. AR_F2: Task Completed virtual button

One of the most important features of the AR application is the triggering of OpenFlow about human tasks completion. This feature is based on a virtual "Task Completed" button visualized inside the virtual world of the AR devices. After the execution of each human assigned assembly task, operators are able to inform the overall system about this task's completion and OpenFlow continues with the assignment of the pending tasks. This feature is based on a ROS service providing information about the status of each human task from the AR application to the OpenFlow. The OpenFlow returns information about the successful or failed execution of the AR application's request to the AR application server. Figure 15 presents the message exchange between the OpenFlow and the AR application during this feature's usage by the operators.

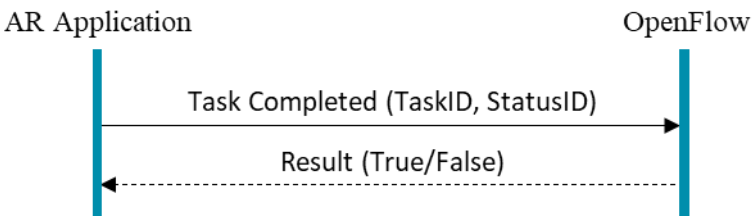

Figure 15: AR_F2 – Message exchange sequence diagram

3.1.1.3. AR_F3: Robot trajectory visualization inside the virtual world

Human operators are able to check the trajectory of upcoming robot motions inside the virtual world of the AR devices thanks to the AR_F3 of ODIN AR application. Robot's trajectory is visualized with a hologram of the selected robot executing the planned motion trajectory inside the virtual environment

of the application. The message exchange between the OpenFlow and the AR application during this feature's usage by the operator is presented in the following figure (Figure 16).

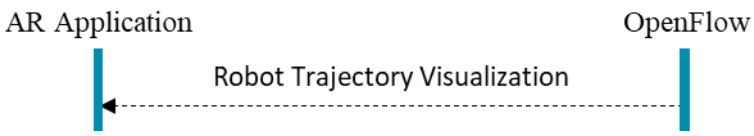

Figure 16: AR_F3 – Message exchange sequence diagram

3.1.1.4. AR_F4: Easy robot programming using interactive virtual tool

Using the AR application of ODIN, the operators are able to easily program a new robot motion using an interactive hologram of equipped robot's tool. This feature can be accessed through the main menu of the application by the operator. The operator may set the target pose of the robot using the interactive virtual tool and trigger OpenFlow to send a motion command request to robot controller by using a virtual button. Figure 17 presents the message exchange between ODIN OpenFlow and AR application during robot's easy programming using the interactive marker.

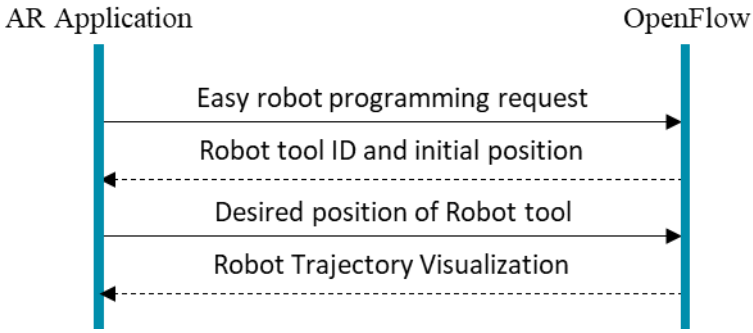

Figure 17: AR_F4 – Message exchange sequence diagram

3.1.1.5. AR_F5: Security alarm visualization

One of the two networked component's modules is the Cyber Security. This module focuses on the deployment of cyber security policies in ODIN pilot cases ensuring the networking safety of the pilots. This module is connected with the AR application through the OpenFlow in the form of security alarms' visualization inside the virtual world of the AR application. The operator is informed about any network's attack and accordingly either resume or pause the assembly process. Messages' exchange between AR application and OpenFlow modules' is presented in the following figure (Figure 18).

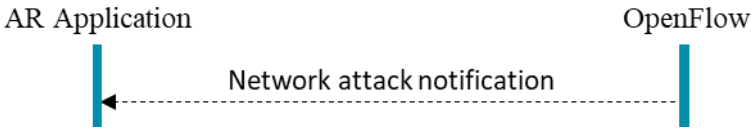

Figure 18: AR_F5 – Message exchange sequence diagram

3.1.1.6. AR_F6: Error handling in case of detection process failure

Through the AR application of ODIN, operators are notified in case of objects' detection process failure. In case of such an event, textual information is visualized inside operator's field of view indicating the robot resource which was not able to execute the specific detection action and the object which was not detected. The operator is able to easily program a new robot motion using the AR_F4 and request again this detection action's execution from the OpenFlow using a virtual button. The message exchange between the OpenFlow and the AR application during errors' handling is visualized in Figure 19.

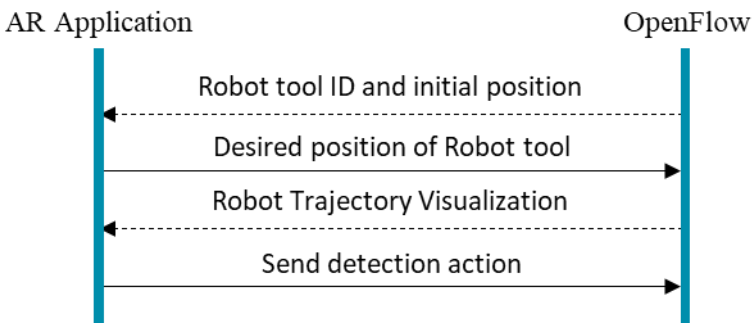

Figure 19: AR_F6 – Message exchange sequence diagram

3.1.1.7. AR_F7: Resilience – execution error recovery

ODIN AR application provides an easy recovery mechanism for the recovery of the system in case of unexpected events such as:

- Safety zones violation.
- Robot's failure to execute a specific motion.
- Failed execution of a human or robot task.

Recovery instructions are provided to the operators in form of textual information. Following the provided instructions, operators are able make the corresponding recovery actions and trigger the OpenFlow to continue tasks' execution using a virtual button. Finally, Figure 20 visualizes the message exchange between ODIN AR application and OpenFlow modules during system's recovery by the operator.

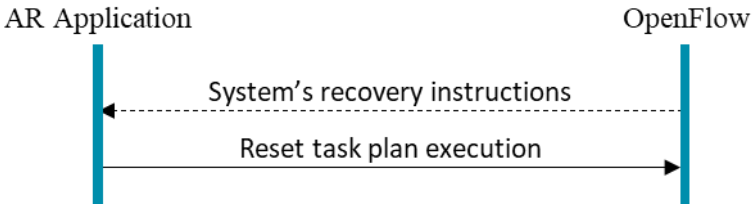

Figure 20: AR_F7 – Message exchange sequence diagram

3.1.1.8. AR_F8: Mobile robot's safety zones visualization through AR Glasses

This feature of ODIN AR application aims to increase the safety awareness of human operators when acting in the same working areas with robots. It is focused on safety zones' visualization inside the virtual world of the selected AR devices. Information about safety zones' location, dimensions and status are provided by the OpenFlow to the AR application. This feature may be enabled through the main menu of the AR application. The message exchange between the ODIN AR application and the OpenFlow is presented in Figure 21.

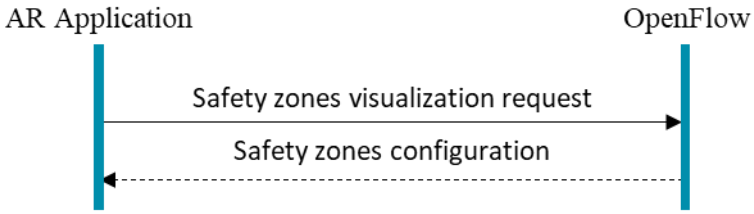

Figure 21: AR_F8 – Message exchange sequence diagram

3.1.1.9. AR_F9: Quality inspection results visualization, digitally mark for validation and instructions provision for corrections

The main task of the human operator be involved in the 3rd operation of the Automotive pilot focuses on the quality inspection of motor and gearbox connection as well as their components (connectors,

screws etc.). This feature of the AR application focuses on the visualization of quality inspection results inside the virtual world of the AR application and instructions' provision for corrections. The operator is able to digitally mark the components he/she is checking using this feature. The exchange of messages between the AR application and the OpenFlow modules for this feature is presented in the following figure (Figure 22).

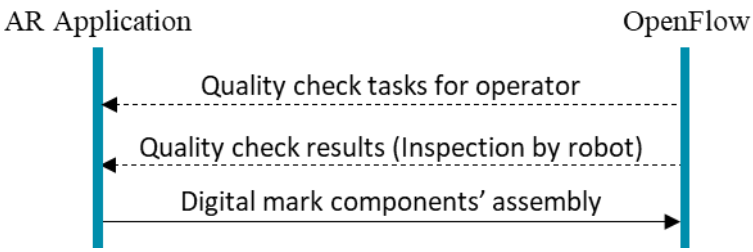

Figure 22: AR_F9 – Message exchange sequence diagram

3.1.1.10. AR_F10: Alarm indicating that the robot is working in collaborative area
Different collaborative areas are defined in the layout of ODIN pilots. In these areas, human operators and robots are working together for the execution of specific assembly tasks. When robots are entering these kind of working areas, corresponding alarms are sent to the operators through the AR application. In this case, the exchange of messages between the AR application and the OpenFlow is presented in Figure 23.

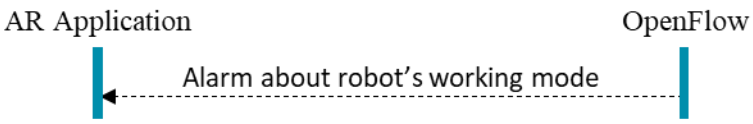

Figure 23: AR_F10 – Message exchange sequence diagram

3.1.2. Integration in the ODIN OpenFlow

An important aspect of the AR application is its integration to the OpenFlow. The application is depended on the OpenFlow to which is connected, receives messages and uses its services. ODIN AR application communicates with the OpenFlow to exchange information depending on the case and features used. An initial version of the AR Operator Support module interfaces has already been presented in deliverable D1.4 "Open architecture for ODIN networked large pilot lines" (Figure 24).

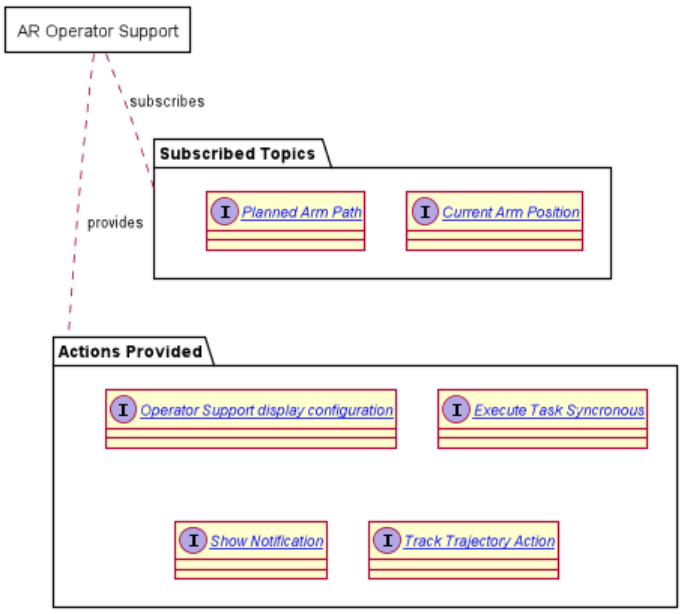

Figure 24: AR Operator Support HMI Module Interfaces

The connection and messages' exchange between the AR devices with the computer running the OpenFlow, robots' controllers and ODIN database is presented in Figure 25.

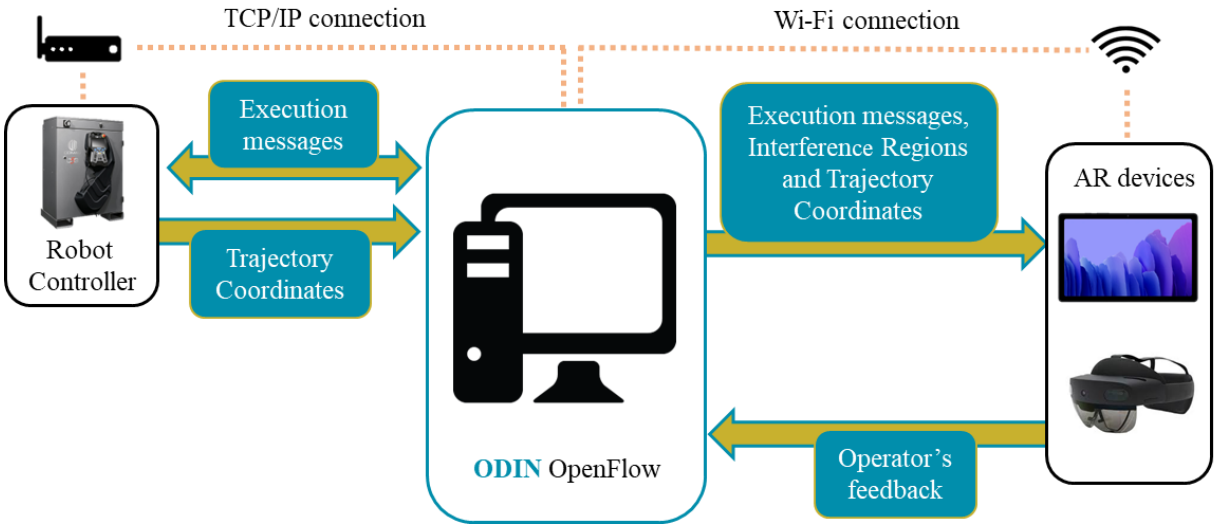

Figure 25: Communication of AR based solution

3.2. ODIN Smart User Interfaces

The goal of TAU's work is to provide accessible, easily configurable, and responsive projector-based User Interface for interacting with collaborative robotic cell. The application consists of a set of laser projectors, installed around the robotic cell area, a set of depth sensors for monitoring the environment and a workstation that runs the application. The RGBD sensor monitors the environment for change in depth information, which can be then used to detect interactions with the UI elements or check for safety violations inside the cell.

ROS messaging system is used to connect different components of the demonstrator between each other (Figure 26). The following nodes define the components of the modules:

- RGBD Sensor nodes: These external nodes provide image/3D data required for the functionality of the module. That data includes RGB and point cloud data from the RGBD sensor used in the setup, as well as camera parameters of the sensor. For TAU demonstrator, Azure Kinect [4] and its ROS driver [6] is used as a sensor node, but other sensors can also be used if they provide the driver.
- Robot Node: Robot node provides data on current robot position, as well as interfaces for controlling the robot program.
- Projection zone nodes: This set of nodes define different areas where content can be displayed. Separate zones can be used for displaying instructions (table level), displaying safety border (ground level). ROS actions for displaying instruction content and visualizing border in the given zone are defined.
- Projector node: Projector nodes account for displaying the content with the selected projector in the robotic cell. First, the system loops over projection zones and generates projection images for each of the projection spaces for the given projector. These images are then merged together by simply adding them together.
- Depth Monitoring Node: This node monitors the cell space using Kinect sensor. The interaction zones are defined within specific projection zone and then reprojected into global depth space of the robotic cell. When interaction with the zone is detected, a message is published in a pre-specified topic, containing zone id and interaction zone id.
- Process control Node: The node monitors messages from Depth Monitoring node, which are then mapped to the control operations with the robot. The simple case of that functionality is

starting/stopping the robot, although more advanced operations are possible. This node will be replaced with OpenFlow in the future.

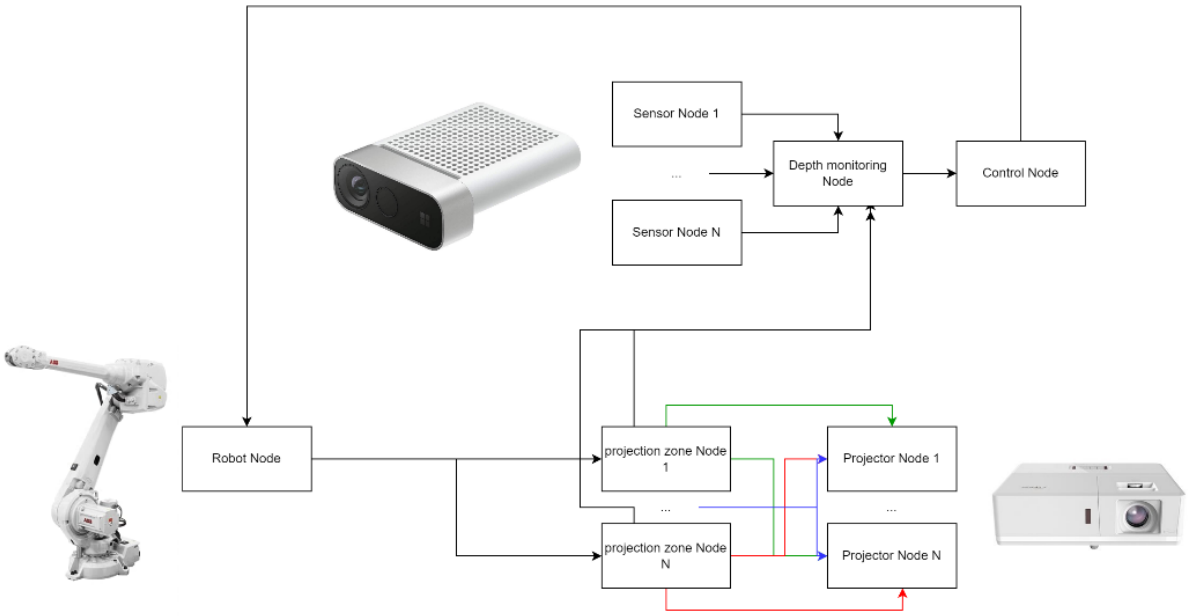

Figure 26 : Communication architecture for projector-based AR solution

3.2.1. Features of ODIN Smart User Interfaces

The developed ODIN interface provides the following functionalities:

3.2.1.1. Displaying content related to the assembly process of the work cell. These instructions can consist of text and graphical information needed for the operator to complete his task.

3.2.1.2. Interactive elements for process control. The user can interact with these elements by hovering his/her hand over them. This can be used to start/stop the robot, switch mode of the operation or confirm completed part of the task.

3.2.1.3. Safety/annotation border. The interface can project a border around the object to highlight its position to the user, or to define safety line between the operator and the device.

3.2.2. Set-up and Calibration

A set of calibration procedures is defined to find transformations between sensors, projector spaces and the robot. Calibration of camera-projector space takes place based on a pattern of Aruco markers [7]. The transformation is computed as a standard openCV Homography transform between coordinates in camera space and coordinates of the markers in the image. Bird-view alignment of the sensor's RGB image is done using checkerboard pattern.

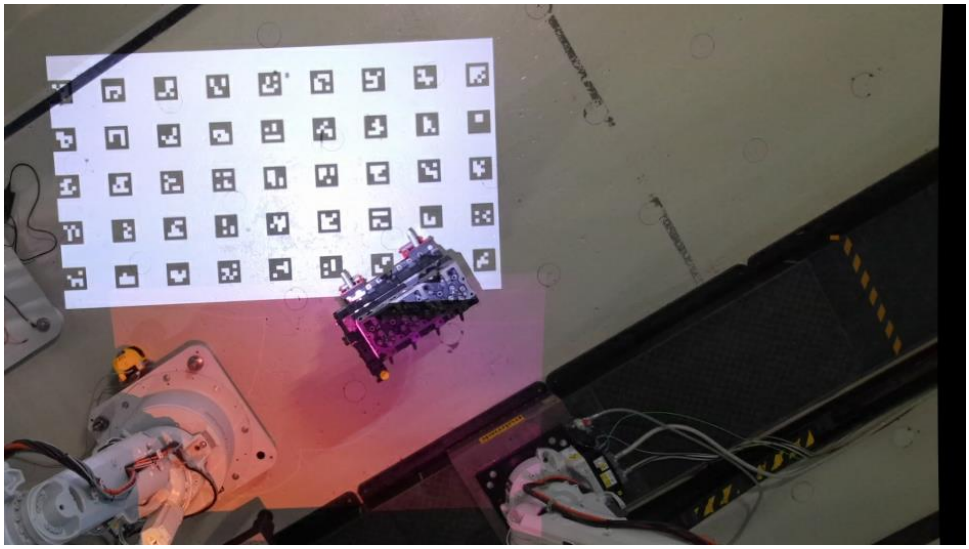

Figure 27: Example of calibration pattern from Kinect sensor

To calibrate robot-camera transformation, a color marker installed on the end effector of the robot was used to get rotation-translation between the base of the camera and the robot.

Calibrated setup can be split into projection zones. (a set of designated rectangular areas which can be used for displaying content). Two types of zones can be used:

- Static Projection zone (the zone which does not change during execution). An example of such zone can be ground area around the robot.
- Mobile Projection zone (the zone that can change its position and orientation during the operations in real time). This can be used for a mobile table that can be moved around the cell.

For static usage, a tool for defining borders of a zone was developed. The tool allows you to check whether the new zone aligns well the setup in real time and it also computes all needed calibrations automatically. The result is stored as a JSON file.

Figure 28: Interface for defining static projection zones

In case of mobile projection zone, the zone is defined through one detectable marker (ARUCO), placed on the corner of the surface. Size parameters of the projection area should also be known beforehand.

3.2.3. Runtime procedure

During runtime of the module, the user can define which projection zones he/she wants to use, and what content should be displayed inside those zones. In case of mobile projection zone, the marker that defines the zone is tracked and the system automatically recalibrates projection area based on the position of the marker.

Interactions with the environment are monitored through changes in Depth/Point Cloud of the robotic cell. For larger environments, data from multiple RGBD sensors may be needed to reach the required

accuracy of the system. To fuse the data from the different sensors, they are transformed to the robot base frame. This solves two problems. First, it provides a unique frame of reference, which is simple to calibrate against. Second, the axis of base frame of the robot in general provide natural orientation relative to the robot cell (XY axis are parallel to the ground, Z axis is perpendicular). This in turn makes the depth map of the fused point cloud simple to work with, since the depth value is checked to see whether user has interacted with button/border or not.

The pipeline for initializing interactive element is shown in Figure 29 and Figure 30.

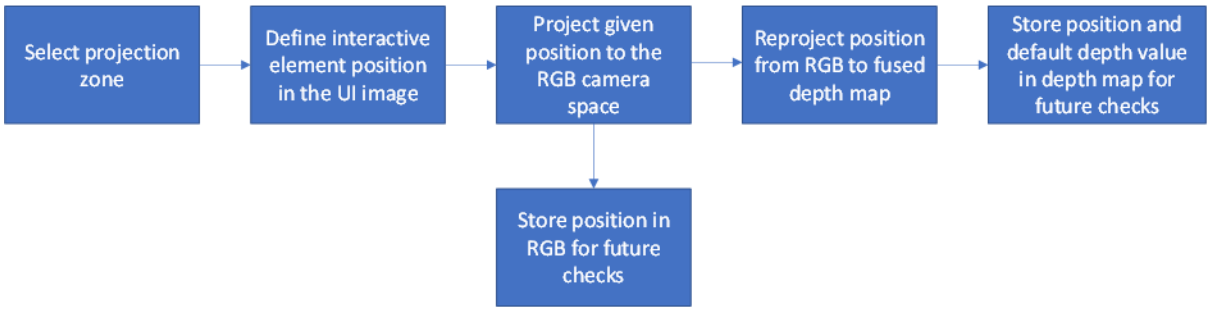

Figure 29: Initialization of interactive element

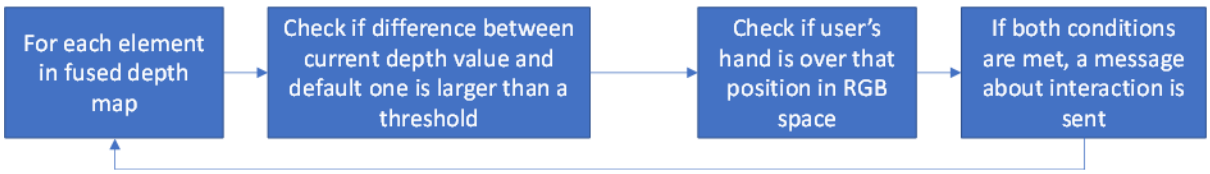

Figure 30: Example of interaction check with button elements

Safety border is defined as a 2D polygon around the robot based on the position of the robot's joints sent through ROS. The border is updated in real time and therefore increases user's awareness of the robot. The size of the area inside the border can be scaled up and down.

This implementation will focus on the White Goods case, providing a safety border around the UR10 robot that would increase awareness of the operator and therefore increase the overall efficiency of the line. Projector interface will provide information about the cell to the user and can be used as a learning tool for new operators.

3.3. Virtual Reality safety training

The goal of virtual reality safety training developed by TAU is to provide a safe environment for operators to familiarize him/herself with working spaces, safety measures and distances. The application consists of a VR headset to run virtual environments on a laboratory scale. Users would follow instructions and UI elements to observe the environment and utilize 3D visualization for perceiving the real work cell. This application follows the risk assessment requirement for risk reduction (training). The second part of the training is dedicated to the assembly of products while users observe safety measures. Optimum results would be reducing operator mistakes in violation of hazard zones.

3.3.1. Features of virtual reality safety training

The VR application provides the following functionalities to the training system:

- 3D elements represent the working spaces of robot where it demonstrates and the volume of robot's reachability by expanding effects.
- Demonstration of safety borders which is a replica of the safety device's area definition. In this application, safety borders are defined based on safety laser scanners in TAU's HRC laboratory (one of the small-scale pilots), which consists of warning zone and danger zone. These are illustrated in Figure 31 with orange and red circular fences correspondingly.

- UI panels which provide information and instructions for the user such as:
    - Tooltips which demonstrate annotation of important instructions based on the position of user in the work cell.
    - UI guides which provide general information about triggering events where user could face in real environment.
    - UI guides which provide instructions how to react and avoid hazardous events.

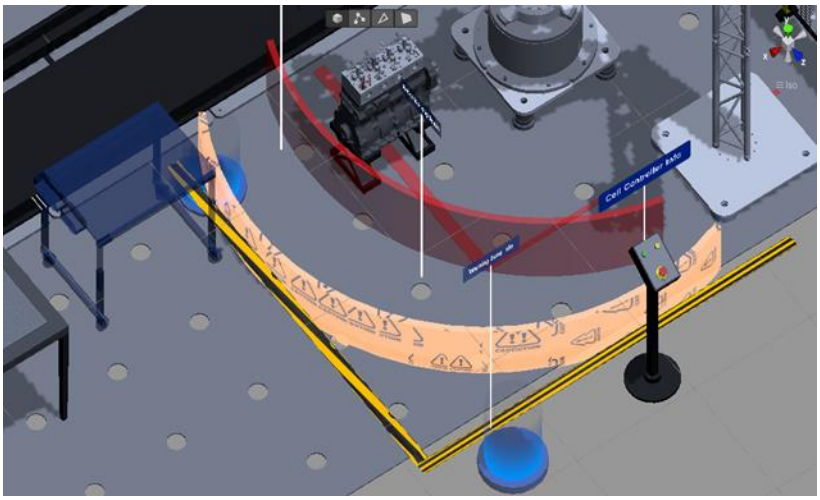

Figure 31: representation of safety areas of laser scanner

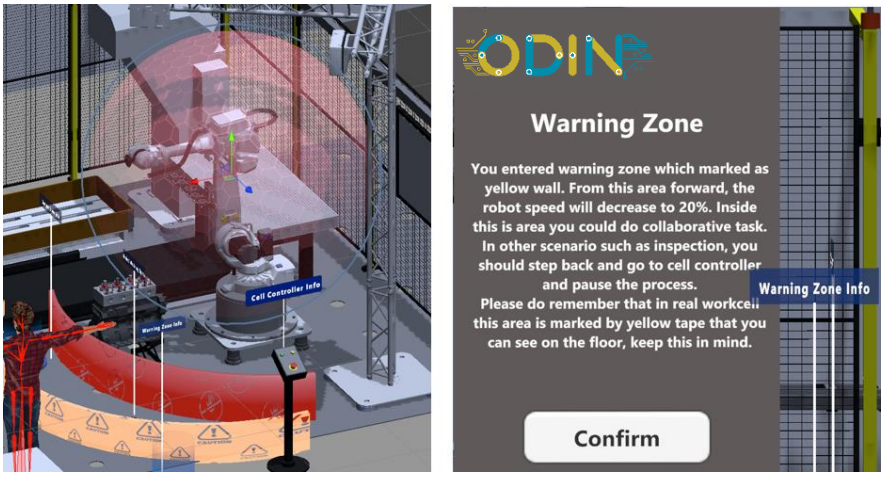

Figure 32: robot workspace representation in sphere shape (left), information of warning zone for operator (right)

For creating a more realistic environment, character rigging is utilized to mimic human body and hands movement. This implementation is done directly in UNITY environment, which control of movement is following headset's and controller's position. For the second phase, the product CAD models are imported to UNITY to produce assembly scenario and with socket feature, the position of assembly products are highlighted for the user as snap-it feature. Assembly scenarios consist of two categories, a) human movements, which follow user tracking by headset and controller to assemble product in sequence, and b) robot movement where it visualizes robot's tasks with proper animation which are triggering based on status of assembly.

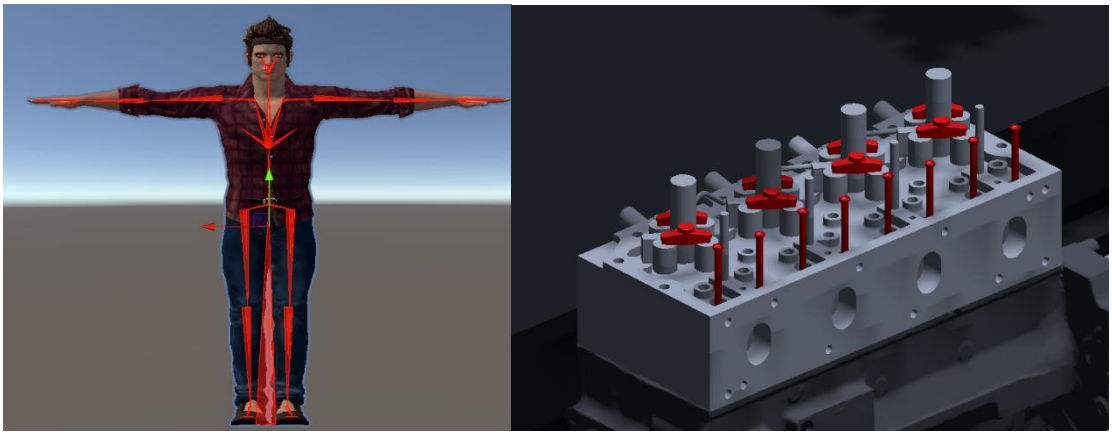

Figure 33 left: component sockets for snap it feature, right: character rigging

3.4. Prototype implementation

At M18 phase of ODIN, the human side interfaces are implemented and prototyped at partners' own facilities. This includes two preliminary setups – one at LMS and another one at TAU. Activities at those setups are presented at next sub-chapters.

3.4.1. White Goods preliminary setup at LMS

The aforementioned AR application has been tested in the preliminary version of white goods setup at Laboratory for Manufacturing Systems (LMS) premises using AR Microsoft HoloLens 2 glasses [2] running the developed application (Figure 34). This setup is a preliminary version of the white goods small-scale pilot at LMS and the first tests and results will be documented in D2.3 scheduled to be submitted on M24 of the project. Additionally, a first version of the OpenFlow was running on a PC, which contained a demo task sequence. Both devices, AR glasses and PC, were connected on the same network for real-time data exchange.

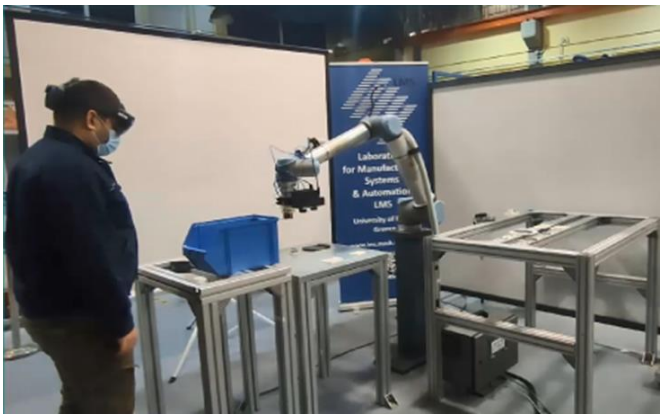

Figure 34: Preliminary White Goods setup at LMS premises

The following features of the AR application tested in this preliminary setup:

- AR_F1: Visualization of Human assigned tasks through AR and AR_F2: Task Completed virtual button

When OpenFlow assigns an assembly task to the human operator, assembly instructions in the form of textual information are visualized inside the virtual world of the AR application. For example, by the time the cobot of white goods pilot places the transformer on the assembly table, human operator must grasp the transformer and place it in an oven. OpenFlow sends assembly instructions to the human operator through the AR glasses as visualized in Figure 35. The operator executes the assigned assembly task to him/her and presses the virtual "Task Completed" button in order to trigger the OpenFlow to continue tasks' allocation and execution (Figure 35).

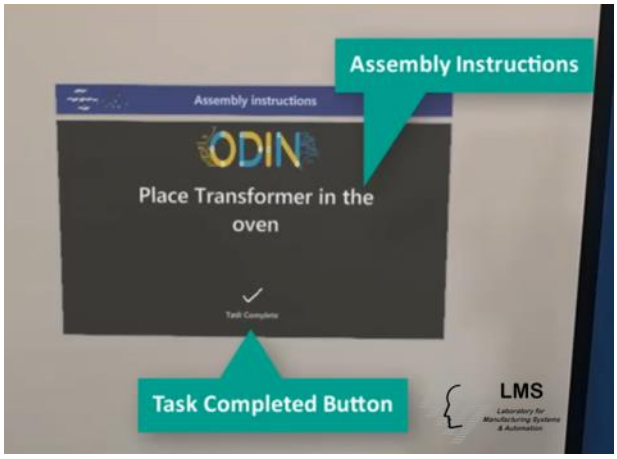

Figure 35: Assembly instructions provision and virtual Task Completed button

- AR_F3: Robot trajectory visualization inside the virtual world

Human operator is able to check the trajectory of an upcoming robot motion using an orange hologram of the selected robot inside the virtual world of the AR devices as presented in Figure 36.

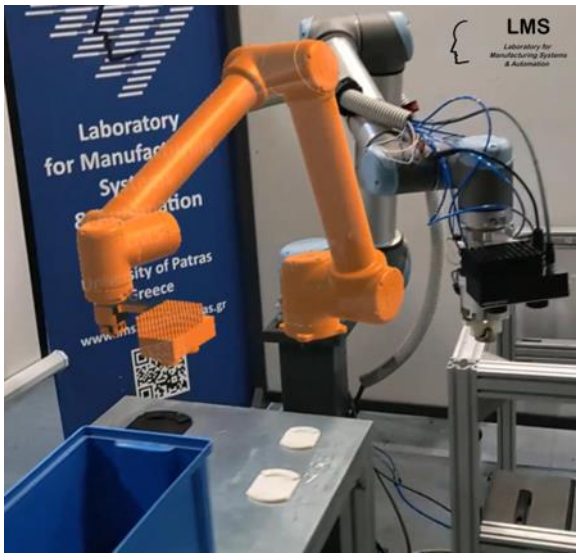

Figure 36: Robot trajectory visualization inside the virtual world

- AR_F4: Easy robot programming using interactive virtual tool

Through the main menu of the AR application, the user is able to access the Easy robot programming feature of the ODIN application. When this feature is enabled, a virtual interactive object with the same geometry of the robot equipped tool is visualized inside the virtual world of the AR application. The user is able to move this tool inside the virtual world using his/her hand. The operator can define a new target pose for the robot using this interactive tool and set the desired location of the tool after the end of the new robot motion (Figure 37). When the operator has placed the tool at the desired location, he/she is able to send the corresponding motion command to the robot controller through the OpenFlow using a virtual button to trigger it.

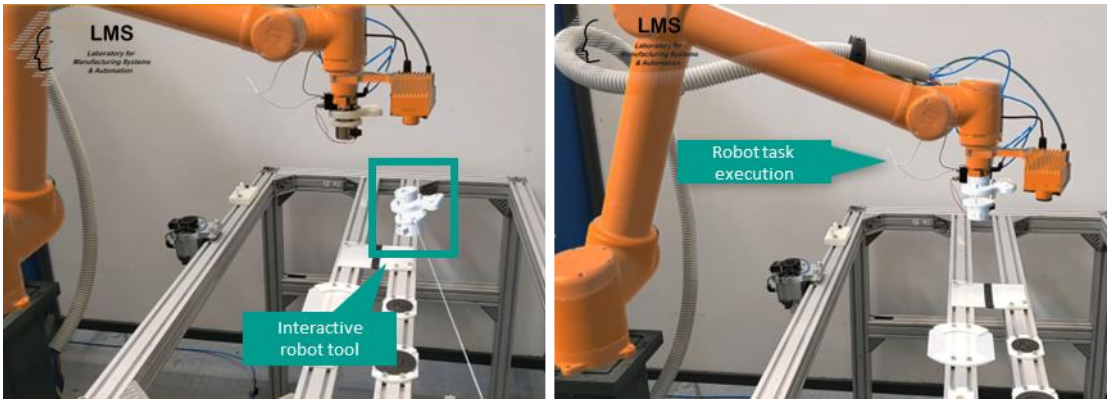

Figure 37: Easy robot programming using interactive virtual tool

- AR_F9: Quality inspection results visualization, digitally mark for validation and instructions provision for corrections

This feature is implemented in the Automotive pilot of ODIN and more specifically it is used in the third operation focusing on the quality inspection of installed components on the motor. The result of a quality inspection action is transferred via the OpenFlow module to the AR application with the form of a photo informing the operator about missing or wrong installed components on the motor (Figure 38). Based on the visualized information, the operator equipped with the AR glasses is able to execute corrective actions on the motor and digitally mark this inspection issue as resolved.

Figure 38: Quality inspection result visualization

3.4.2. Smart User Interfaces preliminary setup at TAU

The preliminary setup at TAU premises is connected with the upcoming TAU's small-scale pilot, which will use as a case study the assembly of a diesel engine. The assembly tasks will provide overall context for the technologies developed until M18 of the project. This setup includes an ABB 4600 industrial robot [8] with a payload of 60kg, performing robotic assembly operations (Figure 39 and Figure 40). The robot is located near/under a stand, onto which sensor and projector devices have been installed. The UI setup consists of two standard DLP projectors with the support of full HD resolution and two Azure Kinect sensors which can be seen in Figure 39 as white boxes attached on truss over the right robot. The devices are installed approximately 3.5 meters above the ground, in close proximity to each other. The devices are connected to a laptop that runs a ROS setup. USB 3.0 is used for Kinect devices connection with the laptop, while Projectors are connected using HDMI/DisplayPort cables. The robot's controller is connected to the same laptop via ethernet connection. Kinect, Robot and projector data

streams are defined through their respective ROS nodes. A single projector covers an area of 2.5 x 1.4 meters at the ground level, which combined creates an augmented area approximately 2.5 x 2.8 meters in size. A movable table with AruCo marker is used for testing the mobile interface of the setup.

The projectors are calibrated to the ground level of the robotic cell, as well as to the table level. The mobile interface and static projections were tested in the technological pilot with double-projector setup and were used to display instructions (3.2.1.1). The mobile projection interface is tracked with decent accuracy, although some deviations can be seen through movement of the table.

The camera was calibrated against the robot base frame, which allows the visualization of the safety border around the robot in real time based on the positions of the robot's joints (3.2.1.3). To connect with the robot, RWS-based interface is used to transfer robot's data through ROS. The update frequency of the messages for the driver is 250ms, which makes it hard to use for faster robot movements and swift violation detection of the border. Alternative methods for robot's connection are investigated at the moment. Interactions with elements in static projection space were tested, with projected buttons used to Pause/Restart the robot program. Up to M18, interactions are checked with the single Kinect. Procedures for fusing multiple Kinect streams were tested but not implemented into until the running month of the project (3.2.1.2).

Figure 39: Virtual model of TAU's HRC environment used for ODIN technological piloting

Figure 40: TAU's HRC environment used for ODIN – engine placed at the front of one ABB robots

3.4.3. Virtual Reality-based safety training at TAU

The VR technical pilot for M18 contains HTC VIVE pro headset [10] with two base stations. Base stations are set up 4 to 5 meters away and facing each other (Figure 42). A gaming desktop is selected for the development, and it is mandatory for this specific headset whereas it requires PCI-e card for providing a wireless connection to the headset. The application also can be run with a laptop if the wired HTC VIVE is selected. The area required for the full experience should be on the laboratory scale. However, there are a couple of teleportation areas that were implemented for important areas such as warning and danger zones.

It is required to Use Unity version 2020.3.3f1 to match the packages that have been utilized up to this date (Figure 41). The application also requires the Steam VR app, where the working areas should be defined and make connections for the controller and headset. After setting up the environment and activating the aforementioned application, the application will be run from UNITY. User is able to start the training from its first phase (safety training) and afterwards, continue with its second phase (assembly training).

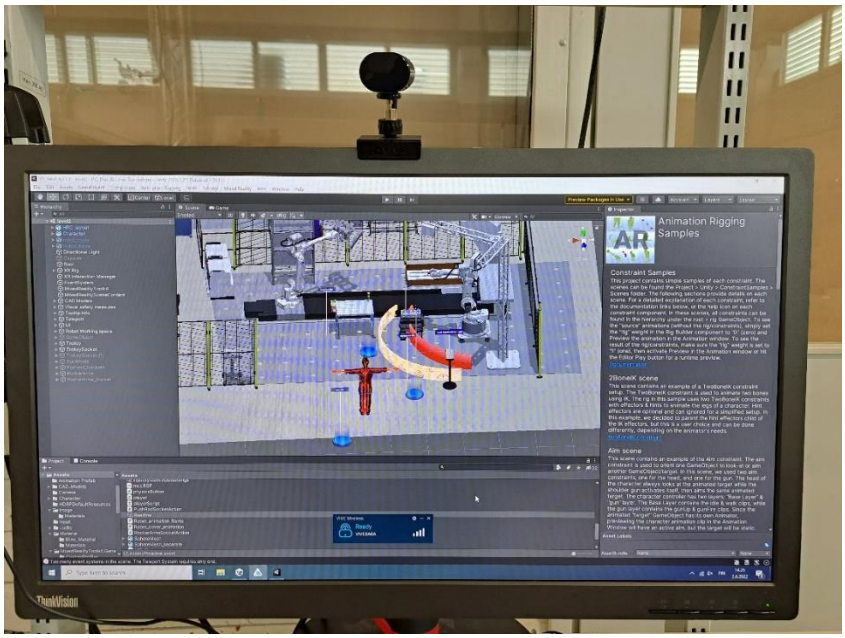

Figure 41: View to the operator safety training environment

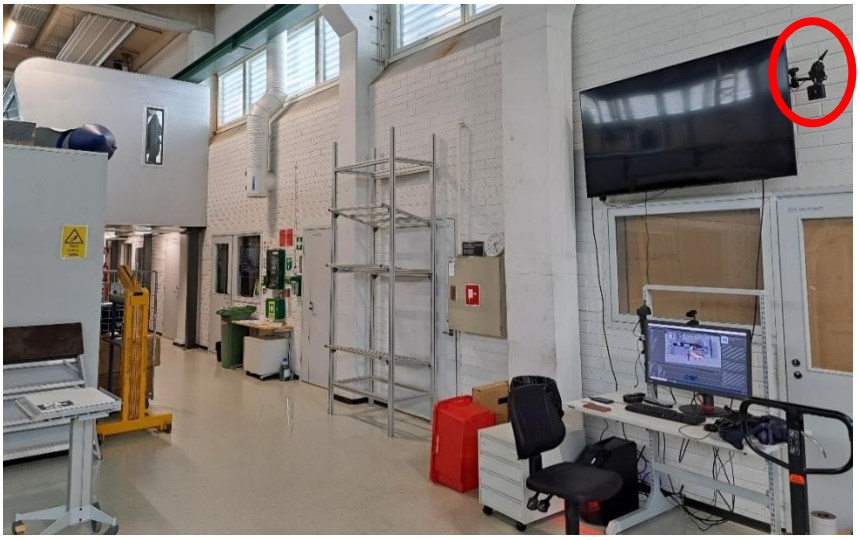

Figure 42: Setup of TAU's VR environment in the lab including one of the base stations

4. CONCLUSIONS

The first prototypes of easy robot programming and human robot interaction modules have been presented in this document. These modules are progressing steady and their implementation in preliminary setups has been documented in this deliverable. Examples of how different types of user interfaces can be used in shared environment were shown. The presented prototypes allow the end users of ODIN to have a clear picture of the available technologies in order to provide their very valuable feedback for successful implementation.

The initial version of the aforementioned technologies is presented in this deliverable, and thus the descriptions presented might be modified, enhanced or improved during the development phase of WP2 according to the relevant tasks in this work package. The next and final version of the presented technologies will be presented in D2.5 which is planned to be submitted on M36.

5. GLOSSARY

CAD	Computer Aided Design
DLP	Digital Light Processing
FC	Fan Cowl
GUI	Graphical user interface
HRC	Human Robot Collaboration
IO	Input/Output
JSON	JavaScript Object Notation
OISP	Onsite Interactive Skill Programming
RGB	Red Green Blue (used in colour sensors or displays)
RGBD	Red Green Blue Depth (used in sensors with colour and depth image)
ROS	Robot Operating System
SBP	Skill Based Programming
UI	User Interface
YAML	YAML Ain’t Markup Language

6. REFERENCES

1. E. Pasternak, R. Fenichel and A. N. Marshall, "Tips for creating a block language with blockly," *2017 IEEE Blocks and Beyond Workshop (B&B)*, 2017, pp. 21-24, doi: 10.1109/BLOCKS.2017.8120404.
2. Microsoft HoloLens 2 AR glasses, https://www.microsoft.com/en-us/hololens/
3. Universal Robot UR10, https://www.universal-robots.com/products/ur10-robot/.
4. Azure Kinect sensor, https://azure.microsoft.com/en-us/services/kinect-dk/
5. Robot Operating System (ROS), https://www.ros.org/
6. Azure Kinect ROS Driver, https://github.com/microsoft/Azure_Kinect_ROS_Driver
7. Aruco Markers, https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
8. ABB IRB 4600 industrial robot, https://new.abb.com/products/robotics/industrial-robots/irb-4600
9. Unity Real-Time development platform, https://unity.com/
10. HTC VIVE pro headset, https://www.vive.com/eu/product/vive-pro/