



# All-in-one middleware for industrial human-robot-interaction

[arise-middleware.eu](http://arise-middleware.eu)

Coordinator



Consortium partners



Co-funded by  
the European Union

## Document History

Ver.	Date	Description	Author	Partner
0.1	13/11/2024	Initial table of content	Gizem Bozdemir	PAL
1.0	20/11/2024	Main Contribution	Severin Lemaignan	PAL
2.0	11/12/2024	Review	Riccardo.Zanetti	ENG
2.1	13/12/2024	Review	Francisco Melendez	FIWARE
3.0	16/12/2024	Last Version	Severin Lemaignan /Gizem Bozdemir	PAL
3.1	27/12/2024	Final Review	Anibal Reñones, Adrian Lozano	CARTIF

---

### Disclaimer

*The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Union. The European Commission is not responsible for any use that may be made of the information contained therein.*

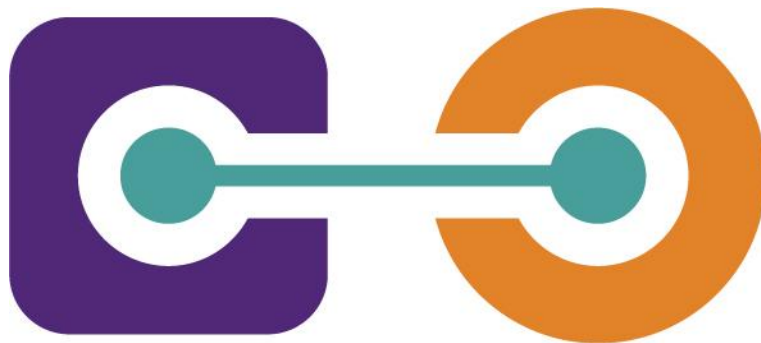
## Publication Details

Grant Agreement Number 101135784

Acronym ARISE

Full Title	Agile, human-centric, and Real-time enabled SourceE technologies advancing industrial HRI in Europe
Topic	HORIZON-CL4-2023-DIGITAL-EMERGING-01-02
Funding Scheme	HORIZON Innovation Actions
Start Date	01/01/2024
Duration	42 months
Project URL	<a href="http://www.arise-middleware.eu">www.arise-middleware.eu</a>
Project Coordinator	CARTIF
Deliverable	D2.5 Library of Open modules for agile Industrial HRI development & Periodic Report on ROS4HRI
Work Package	WP2 Open middleware and technology enablers for industrial HRI
Delivery Month (DoA)	M12
Version	V1.0
Actual Delivery Date	31/12/2024
Type	Other
Dissemination Level	Public
Lead Beneficiary	PAL
Keywords	ROS4HRI, Open modules, HRI, ROS2

# D2.5 - Library of Open modules for agile Industrial HRI development & Periodic Report on ROS4HRI v1



# Table of Contents

<b>1. Introduction</b>	<b>7</b>
1.1 Overview and Progress of Task 2.3	7
<b>2. ROS4HRI: a framework for HRI in ROS</b>	<b>9</b>
2.1 Overview of ROS4HRI	9
2.2 Progress on ROS4HRI migration to ROS 2	10
Table 2: The 8 new ROS4HRI packages created for ROS 2	12
2.3 Description of the ROS4HRI packages	12
<b>3. ARISE software architecture specification</b>	<b>13</b>
3.1 Overview	13
3.2 High-level overview of application creation	14
3.3 Detailed description of the components	14
3.3.1 Application	14
3.3.2 Mission controllers	15
3.3.3 Intents and intent recognition	15
3.3.4 Tasks	16
3.3.5 Skills	16
<b>4. Agile development of ROS modules for HRI: The ARISE template generator</b>	<b>18</b>
4.1. Introduction	18
4.2. Role and Purpose	18
4.3. Main Functions	18
4.3.1 create command	18
4.3.2 list command	19
4.4. Available Templates	19
4.4.5. Availability	21
4.4.6. Benefits of Using rpk	21
<b>5. Conclusion and Outlook</b>	<b>22</b>
<b>Annex A: ROS4HRI Package Details</b>	<b>22</b>

# List of Figures

Figure 1: Open Neural Network Exchange	08
Figure 2: Overview of the ARISE Architecture	12
Figure 3: The output of the llm_chatbot_python app template	19
Figure 4: Example of skeleton detection	22
Figure 5: Example of expression recognition output	23
Figure 6: RViz's Humans plugin	27
Figure 7: RViz's Skeletons3D plugin	28
Figure 8: RViz's TF_HRI plugin	28
Figure 9: Human model, rendered in rviz	29
Figure 10: Screenshot of the included KnowledgeCore explorer	30
Figure 11: <i>rqt_chat</i> screenshot	31

# List of Tables

Table 1: The 17 ROS4HRI packages ported from ROS 1 to ROS 2 09

Table 2: The 8 new ROS4HRI packages created for ROS 2 10

# Acronyms & Abbreviations

ACR.	Description
HRI	Human-Robot Interaction
ROS	Robot Operating System
ROS2	Robot Operating System version 2
LLM	Large Language Model
TEF	Testing and Experimentation Facility
FSTP	Financial Support to Third Parties
REP	ROS Enhancement Proposal
RViz	ROS Visualization
URDF	Unified Robot Description Format
GPT	Generative Pre-trained Transformer
TTS	Text-to-Speech
API	Application Programming Interface
PyPI	Python Package Index

ACR.	Description
HRI	Human-Robot Interaction
ROS	Robot Operating System
ROS2	Robot Operating System version 2
LLM	Large Language Model
TEF	Testing and Experimentation Facility
ONNX	Open Neural Network Exchange

## 1. Introduction

This document, Deliverable D2.5 of the ARISE project, outlines the development of a library of open modules for agile industrial HRI and provides a periodic update on the progress of the ROS4HRI framework. The aim is to present advancements in creating agile, scalable, and modular HRI solutions using open-source tools.

The report is structured as follows:

1. **Overview of Task T2.3:** Objectives, scope, expected outcomes and progress achieved by Month 12.
2. **ARISE Software Architecture Specification:** A detailed framework for HRI application development.
3. **Template Generator for Agile Development:** Insights into the ARISE template generator for ROS2-based HRI modules and LLM integrations.
4. **Conclusion and Next Steps:** Summary of accomplishments and planned future activities.

This deliverable focuses on providing a technical overview of the advancements made within the ARISE project.

### 1.1 Overview and Progress of Task 2.3

#### Task Objectives

Task T2.3 focuses on the development of an open library of tools and modules for agile industrial HRI by evolving the ROS4HRI framework into a robust, commercially viable solution. The objectives include:

- Migration to ROS2: Transitioning all ROS4HRI components to ROS2, improving modularity, scalability, and compatibility with modern robotic systems.
- Integration of Advanced Perception Tools: Incorporating cutting-edge technologies, including:
  - Face/pose estimation
  - Speech-to-Text (STT)
  - Emotion recognition
  - Plugins for modern Large Language Models (LLMs) like GPT.
- Enhanced Usability and Documentation: Providing comprehensive documentation and updates to the ROS REP-155 data model, ensuring new plugins are well-supported and accessible.
- Iterative Releases and Reporting: Delivering three versions of the library (M12, M24, and M36), with periodic ecosystem updates based on internal advancements and insights from related industrial projects.

### Progress by Month 12 (M12)

As of M12, Task T2.3 has achieved the following milestones:

1. **Completion of ROS2 Migration:** The migration of all ROS4HRI components to ROS2 has been finalized, ensuring better performance, flexibility, and compatibility with the ARISE ecosystem.
2. **Specification of Software Architecture:** A comprehensive software architecture has been developed, serving as a blueprint for HRI application development by ARISE TEFs and FSTP applicants.
3. **Development of a Template Generator:** A tool for creating ROS2 modules tailored for HRI and LLM-enabled functionalities has been designed and implemented. This generator simplifies and accelerates the development process for end-users, aligning with the ARISE vision for agile robotics.

## 2. ROS4HRI: a framework for HRI in ROS

### 2.1 Overview of ROS4HRI

The ROS4HRI framework is a set of ROS packages that provide a set of tools and libraries for developing Human-Robot Interaction applications in ROS. The framework is designed to be modular and extensible, allowing developers to create custom HRI applications by combining existing modules and adding new ones. The framework provides a set of core modules that implement common HRI functionality, such as person detection and recognition, skeleton tracking, speech recognition or gesture recognition. These core modules can be combined with custom modules to create complex HRI applications.

ROS4HRI also includes several visualization tools that allow developers to visualize the output of the core modules in real-time. These tools, fully integrated to standard ROS tools like rviz or rqt, can be used to debug and tune the performance of the HRI system.

The figure below gives an overview of some of the key components of the ROS4HRI framework.

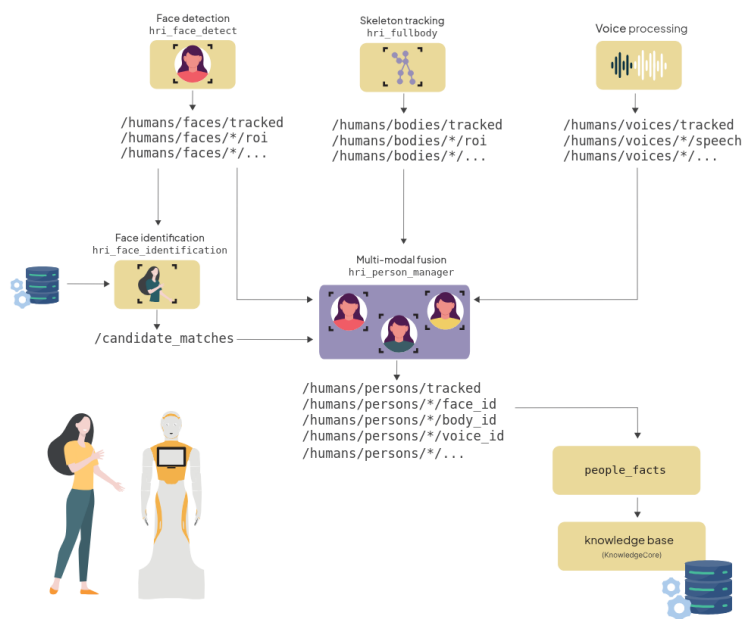


Figure 1: ROS4HRI Overview

### 2.2 Progress on ROS4HRI migration to ROS 2

The tables below show the progress of the migration of the ROS4HRI packages to ROS 2. The tables include the name of the package, the last version released for ROS 1, the current version

for ROS 2, the source repository, and the upstream ROS package page (for packages that have been submitted and accepted to the main ROS repositories).

A description of each package is provided in the next section.

NAME	Last ROS 1 version	Current ROS 2 version	Source repository	Upstream ROS package page
attention_manager_msgs	0.3.3	2.0.0	<a href="https://github.com/pal-robotics/attention_manager_msgs">https://github.com/pal-robotics/attention_manager_msgs</a>	
chatbot_msgs	0.1.0	2.1.0	<a href="https://github.com/pal-robotics/chatbot_msgs">https://github.com/pal-robotics/chatbot_msgs</a>	
hri	0.6.4	2.6.1	<a href="https://github.com/ros4hri/libhri">https://github.com/ros4hri/libhri</a>	<a href="https://index.ros.org/p/hri/github-ros4hri-libhri/">https://index.ros.org/p/hri/github-ros4hri-libhri/</a>
hri_actions_msgs	0.4.3	2.4.2	<a href="https://github.com/ros4hri/hri_actions_msgs">https://github.com/ros4hri/hri_actions_msgs</a>	<a href="https://index.ros.org/p/hri_actions_msgs/github-ros4hri-hri_actions_msgs">https://index.ros.org/p/hri_actions_msgs/github-ros4hri-hri_actions_msgs</a>
hri_engagement	1.0.1	2.0.1	<a href="https://github.com/ros4hri/hri_engagement">https://github.com/ros4hri/hri_engagement</a>	
hri_face_detector	1.5.3	2.0.10	<a href="https://github.com/ros4hri/hri_face_detector">https://github.com/ros4hri/hri_face_detector</a>	
hri_face_identification	0.3.6	2.3.1	<a href="https://github.com/ros4hri/hri_face_identification">https://github.com/ros4hri/hri_face_identification</a>	
hri_msgs	0.9.0	2.2.0	<a href="https://github.com/ros4hri/hri_msgs">https://github.com/ros4hri/hri_msgs</a>	<a href="https://index.ros.org/p/hri_msgs/github-ros4hri-hri_msgs">https://index.ros.org/p/hri_msgs/github-ros4hri-hri_msgs</a>
hri_person_manager	1.1.0	2.0.6	<a href="https://github.com/ros4hri/hri_person_manager">https://github.com/ros4hri/hri_person_manager</a>	
hri_visualization	0.1.7	2.1.0	<a href="https://github.com/ros4hri/hri_visualization">https://github.com/ros4hri/hri_visualization</a>	
human_description	1.0.1	2.0.2	<a href="https://github.com/ros4hri/human_description">https://github.com/ros4hri/human_description</a>	<a href="https://index.ros.org/p/human_description/github-ros4hri-human_description">https://index.ros.org/p/human_description/github-ros4hri-human_description</a>
knowledge_core	2.10.0	3.3.0	<a href="https://github.com/severin-lemaignan/knowledge_core">https://github.com/severin-lemaignan/knowledge_core</a>	
oro	0.1.0	2.2.1	<a href="https://github.com/severin-lemaignan/openrobots-ontology/">https://github.com/severin-lemaignan/openrobots-ontology/</a>	
people_facts	0.2.3	2.0.1	<a href="https://github.com/pal-robotics/people_facts">https://github.com/pal-robotics/people_facts</a>	
pyhri	0.3.0	2.6.1	<a href="https://github.com/ros4hri/libhri">https://github.com/ros4hri/libhri</a>	<a href="https://index.ros.org/p/pyhri/github-ros4hri-libhri">https://index.ros.org/p/pyhri/github-ros4hri-libhri</a>

hri_rviz	0.4.2	2.1.0	<a href="https://github.com/ros4hri/hri_rviz">https://github.com/ros4hri/hri_rviz</a>	<a href="https://index.ros.org/p/hri_rviz/github-ros4hri-hri_rviz">https://index.ros.org/p/hri_rviz/github-ros4hri-hri_rviz</a>
rqt_human_radar	0.2.1	2.2.1	<a href="https://github.com/ros4hri/rqt_human_radar">https://github.com/ros4hri/rqt_human_radar</a>	

*Table 1: The 17 ROS4HRI packages ported from ROS 1 to ROS 2*

In addition to the packages listed above, the following packages have been newly created in ROS 2, to further expand the capabilities of the ROS4HRI framework:

NAME	Last ROS 1 version	Current ROS 2 version	SOURCE REPO
hri_body_detect	N/A	3.1.4	<a href="https://github.com/ros4hri/hri_body_detect">https://github.com/ros4hri/hri_body_detect</a>
hri_emotion_models	N/A	1.1.1	<a href="https://github.com/ros4hri/hri_emotion_models">https://github.com/ros4hri/hri_emotion_models</a>
hri_emotion_recognizer	N/A	1.0.1	<a href="https://github.com/ros4hri/hri_emotion_recognizer">https://github.com/ros4hri/hri_emotion_recognizer</a>
hri_face_body_matcher	N/A	2.1.0	<a href="https://github.com/ros4hri/hri_face_body_matcher">https://github.com/ros4hri/hri_face_body_matcher</a>
kb_msgs	N/A	1.1.0	<a href="https://github.com/pal-robotics/kb_msgs/">https://github.com/pal-robotics/kb_msgs/</a>
hri_privacy_msgs	N/A	1.1.0	<a href="https://github.com/ros4hri/hri_privacy_msgs">https://github.com/ros4hri/hri_privacy_msgs</a>
rqt_chat	N/A	1.0.1	<a href="https://github.com/pal-robotics/rqt_chat">https://github.com/pal-robotics/rqt_chat</a>
tts_msgs	N/A	1.1.0	<a href="https://github.com/pal-robotics/pal_tts_msgs">https://github.com/pal-robotics/pal_tts_msgs</a>

*Table 2: The 8 new ROS4HRI packages created for ROS 2*

## 2.3 Description of the ROS4HRI packages

The ROS4HRI framework comprises a set of modular packages that enable robust human-robot interaction capabilities, such as person detection, face and pose estimation, speech recognition, emotion analysis, and more. These packages are designed to integrate seamlessly with ROS2 and follow standardized conventions to ensure interoperability and scalability.

A total of 17 ROS4HRI packages were ported from ROS1 to ROS2, and 8 new packages were developed for ROS2 to extend the framework's functionality. These packages provide tools for key functionalities, including perception, visualization, intent recognition, and privacy-aware interaction.

For a detailed description of each package, including their version history, core features, and repository links, please refer to [Annex A: ROS4HRI Package Details](#).

The Annex includes:

- A list of ROS4HRI packages with their corresponding repositories.
- A summary of functionalities provided by each package.
- Insights into how these packages support the ARISE software architecture and template generator.

This modular package-based approach enables developers to tailor ROS4HRI for a variety of industrial and research applications.

# 3. ARISE software architecture specification

## 3.1 Overview

In order to facilitate the development of complex robotic applications, the ARISE project provides a software architecture specification. This specification defines the structure of the software components that make up a robotic application, including the mission controller, tasks, skills, and intents.

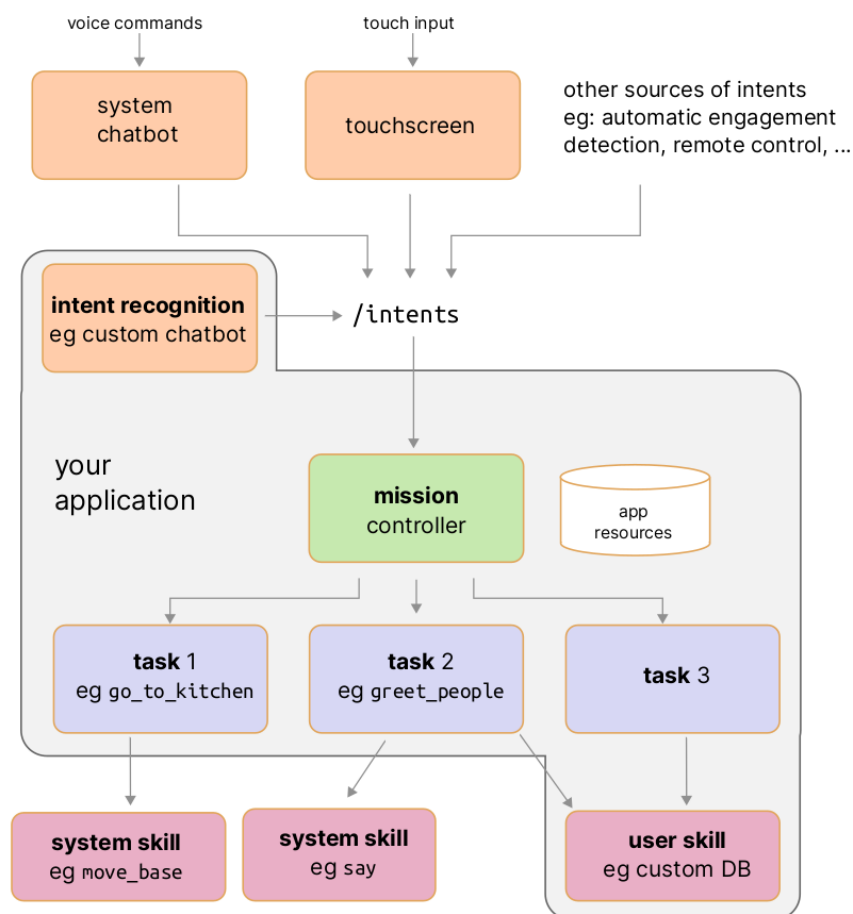


Figure 2: Overview of the ARISE Architecture

We define the main concept as follows:

- The **mission controller** implements the main role or behaviour of the robot. It reacts to incoming **intents** and starts **tasks** to perform actual actions. At a given time, **only one mission can be active**.
- **Intents** represent the user's desires or commands. The mission controller monitors incoming intents and reacts accordingly. Intents are recognised (or *extracted*) by dedicated nodes, the **intent recognition** nodes. Your robot provides a set of standard intents recognition nodes (including, for instance, a chatbot), but you can also create your own.
- A **task** is a (finite) set of actions performed by the robot to achieve a specific goal (i.e. "Doing an inventory", "Getting a glass of water", "Going to the kitchen"). It is time-bound (i.e. it has a finite duration) and can usually be paused and resumed. Tasks rely on the robot's individual skills to perform actions.
- **Skills** are smaller, self-contained operations performed by the robot: saying something using TTS, navigating to a position, performing a gesture are all examples of skills. Some skills are provided out-of-the-box on the robot, we call them **system skills**, some others are custom for a particular application, we call them **user skills**.
- Finally, an **application** is a package made of usually one **mission controller**, several **tasks** implementations, and, based on the needs, custom intent recognition nodes, skills, and generic resources (e.g. images, sounds).

This terminology is based on the RobMoSys conceptual model.<sup>1</sup>

## 3.2 High-level overview of application creation

Creating a new ARISE-compliant application involves the following steps:

1. **Create a new application package.** We provide the ARISE rpk template generator to this effect (see below). This will create a new directory with the basic structure of a robot application.
2. **Customize the mission controller** to implement the desired behaviour. If needed, create as well new tasks, skills, and intent recognition nodes. Templates are provided to help the FSTP applicants to get started with each of those.
3. **Build the application** using the standard ROS 2 build tools
4. **Deploy the application** on the robots and/or TEF platforms.

## 3.3 Detailed description of the components

### 3.3.1 Application

---

<sup>1</sup>[https://robmosys.eu/wiki/general\\_principles:separation\\_of\\_levels\\_and\\_separation\\_of\\_concerns](https://robmosys.eu/wiki/general_principles:separation_of_levels_and_separation_of_concerns)

An **application** is a bundle of one mission controller and all its required resources (for instance data, text, images, models, additional tasks and skills...)

The whole application is bundled as a **robot package** that can be easily distributed and installed.

### 3.3.2 Mission controllers

**Mission controllers** implement the main role or behaviour of the robot. Its main role is to react to incoming **intents** and start **tasks** to perform actual actions. In that sense, the mission controller is the orchestrator of the robot's behaviour.

ARISE does not enforce any specific structure or technique to implement the controller: participants are free to use any control paradigm they wish, from simple imperative-style control scripts, to behaviour trees (BT)<sup>2</sup>, finite-state machines (FSM)<sup>3</sup> or symbolic task planner (eg, HTN<sup>4</sup>).

The rpk tool can be used to create the scaffolding of a mission controller, before customizing for a particular application.

Examples of **mission controller** that could be implemented by FSTP participants include:

- a 'robot receptionist' mission, which welcomes visitors when they enter a building;
- an interactive information robot, which answers questions about a location or an organisation;
- a controller implementing robot teleoperation for a scientific study.

**Alternative terminology:**

**Mission controllers** are sometimes also referred to as **supervisors** or **high-level behaviours** in the literature.

### 3.3.3 Intents and intent recognition

An intent is an abstract description of an operation to be performed by the robot.

Intents are primarily designed to capture user-initiated desires or commands. For instance, a button click on a touchscreen, the result of a chatbot-based verbal interaction, a command started by a remote user interface.

Intents are represented as ROS messages, and they are all published on the same channel (the ROS /intents topic). By monitoring incoming messages on this channel, the active mission can react to user requests.

**Note**

---

<sup>2</sup> [https://en.wikipedia.org/wiki/Behavior\\_tree\\_\(artificial\\_intelligence,\\_robotics\\_and\\_control\)](https://en.wikipedia.org/wiki/Behavior_tree_(artificial_intelligence,_robotics_and_control))

<sup>3</sup> [https://en.wikipedia.org/wiki/Finite-state\\_machine](https://en.wikipedia.org/wiki/Finite-state_machine)

<sup>4</sup> [https://en.wikipedia.org/wiki/Hierarchical\\_task\\_network](https://en.wikipedia.org/wiki/Hierarchical_task_network)

Intents are also an effective way to test/debug your application: you can manually publish a new Intent message on the `/intents` topic to simulate any user interaction, and check the controller's reaction.

In order to facilitate re-usability, certain intentions are standardised. Examples of intents include `ENGAGE_WITH`, `GUIDE`, `PRESENT_CONTENT`, etc. It is however possible to create as many custom intents as needed for specific applications (it is simply a name in the `Intent.msg`). The only requirement is to create the corresponding intent recognition capability, and to add the intent handling to the mission controller.

Indeed, the application must provide a way to extract intents from the user's input. This is done by creating an **intent recognition** node, which listens to the user's input (eg, a chatbot) and publishes the corresponding intent on the `/intents` topic.

ARISE TEFs come with a set of standard intent recognition nodes, including an LLM-based chatbot.

### 3.3.4 Tasks

A **task** represents a finite set of actions performed by the robot to achieve some goal (i.e. "Doing an Inventory", "Getting a glass of water", "Going to the kitchen"). It is time-bound (i.e. it has a finite duration), and can usually be paused and resumed.

The robot's mission controller is responsible to start (and possibly stop/pause/resume) the required tasks to achieve the desired behaviour. It does so by calling the task's control API (usually, a ROS action called `/<task_name>/control`, but this may depend on each particular task).

Tasks themselves then rely on calling **skills** to perform specific actions.

#### Note

When developing simpler applications, the mission controller might decide to directly call skills, without creating a task. This is fine, but not recommended for complex applications, as it makes it harder to monitor and control the robot's behaviour.

Tasks are regular ROS nodes, and users usually create their own tasks, as required by their application.

#### Alternative terminology

**Tasks** are sometimes also referred to as **activities** or **behaviours** in the literature.

### 3.3.5 Skills

**Skills** are 'unit' operations on the robot: navigating to a location, looking at a specific target, grasping an object, etc.

ARISE TEFs each offer a set of off-the-shelf skills, called *system skills*. They are organised in broad categories:

- navigation skills

- gesture-related skills
- social skills
- expressive skills
- communication-related skills
- reasoning and knowledge-related skills
- or direct hardware control skills

The exact list of system skills is still being refined and will be provided to the FSTP participants.

Users can as well create their own custom skills as ROS nodes and bundle them with their application.

#### Alternative terminology

**Skills** are also sometimes referred to as **Actions**, **Atomic actions** or **Atomic task**.

#### Note

**Skills** should not be confused with ROS actions. ROS *actions* are a generic mechanism to perform asynchronous *remote procedure calls* (RPC), while PAL OS' concept of *skills* refers to the implementation of a robot's operation.

That said, in practice, *ROS actions* are often used to implement the robot's skills, especially for asynchronous control.

## 4. Agile development of ROS modules for HRI: The ARISE template generator

### 4.1. Introduction

The `rpk` tool is a command-line utility designed in the ARISE project to streamline the development of robotic applications in the ROS 2 ecosystem. It facilitates the creation and management of standardised application skeletons for robots, enabling developers to quickly bootstrap projects and focus on implementation-specific details. Its flexibility and modularity support a range of robot types, making it a valuable asset for robotics software developers.

This section provides an overview of the tool's functionality, usage, and the templates it offers for different components of robotic applications.

### 4.2. Role and Purpose

The primary role of `rpk` is to provide a structured framework for developing ROS 2-based robotic systems. By offering pre-defined templates, it ensures consistency, adherence to best practices, and reduces the overhead of setting up basic infrastructure for robot tasks, skills, missions, and applications.

This tool is particularly beneficial for teams developing complex robotic systems, as it fosters modularity and reuse of code. The use of `rpk` can significantly accelerate development cycles by standardising the initial setup of robot applications.

### 4.3. Main Functions

In its current form, the `rpk` tool offers two primary commands:

- **create:** This command generates application skeletons for various components of a robot system. Developers can specify the type of component (e.g., skill, task, mission) and select from available templates tailored to specific robot types or use cases.
- **list:** This command displays a catalogue of available templates for intents, skills, tasks, missions, and applications. It provides brief descriptions of each template, including its programming language and example use cases.

#### 4.3.1 create command

The `create` command is central to `rpk`'s functionality, enabling developers to generate skeletons for the following components:

1. **Intent:** Modules for extracting user intentions from inputs, such as chatbot systems or interfaces leveraging Large Language Models (LLMs).
2. **Skill:** Reusable, atomic robot actions. These are basic building blocks for tasks and missions, such as "go to" or "say" actions.
3. **Task:** Time-limited activities that combine multiple skills. Tasks represent intermediate levels of robot behaviour, such as greeting a person or fetching an object.

4. **Mission:** High-level controllers managing the robot's overall behaviour. Missions combine tasks to define the robot's operational goals, such as acting as a receptionist or waiter.
5. **Application:** Comprehensive frameworks combining mission controllers, tasks, and skills, along with necessary resources, to create complete robotic applications.

The create command supports the following options:

- **-r or --robot:** Specifies the target robot type. Available options include generic robots and predefined configurations for PAL's robots like ARI<sup>5</sup> and TIAGo<sup>6</sup>. The 'PAL' variants include examples of calls to PAL specific API, and are not directly relevant to the ARISE project.
- **-p or --path:** Defines the directory path where the generated skeleton will be saved. By default, it uses the current directory.

#### 4.3.2 list command

The list command displays available templates, categorised by component type. For each template, it provides:

- A unique identifier (e.g., base\_python).
- A description of the template's purpose and features.
- The programming language used (e.g., Python).

### 4.4. Available Templates

The rpk comes preloaded with a set of templates for intents, skills, tasks, missions, and complete applications.

As of the current version, the following templates are available:

- **Intent Extractors:**
  - basic\_chatbot: A simple chatbot skeleton.
  - llm\_bridge\_python: An intent extraction module utilizing LLMs via APIs like OpenAI's ChatGPT.
- **Skills:**
  - base\_python: A generic skill template.
  - say\_python: Implements a "say" action as an example.
  - db\_connector\_python: A mock-up for database interaction.
- **Tasks:**
  - base\_python: A generic task template.
  - greet\_task\_python: Demonstrates a "greet" task implementation.
- **Mission Controllers:**
  - base\_python: A generic mission controller template.
  - base\_intents\_python: A supervisor with pre-filled intent handlers.
  - llm\_supervisor\_python: An advanced supervisor leveraging LLMs for user interaction.

---

<sup>5</sup> <https://pal-robotics.com/robot/ari/>

<sup>6</sup> <https://pal-robotics.com/es/robot/tiago/>

- **Applications:**
  - llm\_chatbot\_python: A complete sample application featuring LLM-driven user interaction, with integrated mission controllers, tasks, and skills.

Some of these templates (the base\_\* ones) were designed to be generic and extensible, showcasing best practices in ROS 2 development (including the use of lifecycle nodes, parameter-based configuration, etc.). They are meant to be used as starting points for FSTP users to create their own components.

Other templates are more specific and are meant as examples of how to integrate complete stacks.

For instance, the figure below illustrates the output of the llm\_chatbot\_python app template: *llm\_chatbot\_python*

As seen in the figure, the llm\_chatbot\_python template generates a complete application skeleton, including mission controllers, tasks, skills, intents, as well as the custom ROS 2 message packages required for these nodes to communicate.

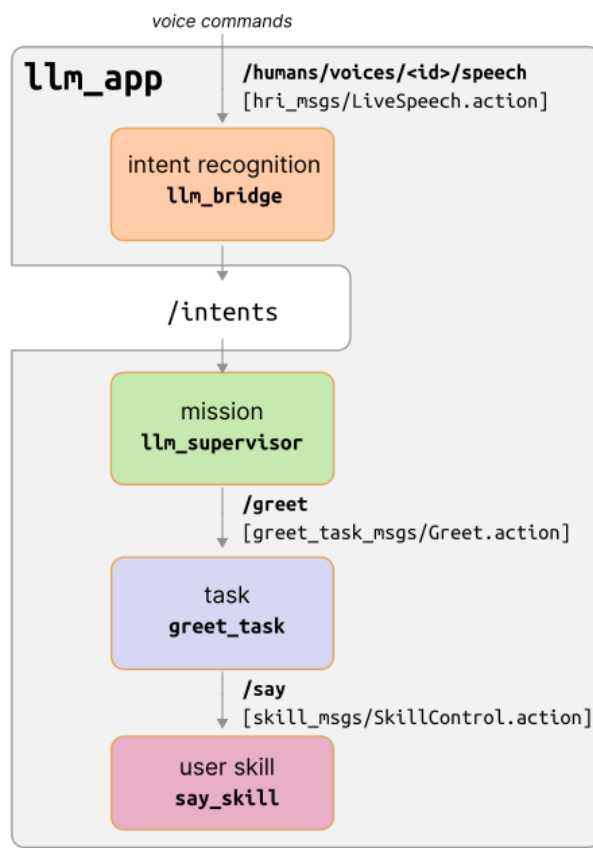


Figure 3: The output of the llm\_chatbot\_python app template

#### 4.4.5. Availability

The `rpk` tool and its templates are available as open-source software under the Apache 2.0 license (<https://github.com/pal-robotics/rpk>).

To facilitate installation and usage, the tool is distributed as a Python package via the Python Package Index (PyPI). Developers can install `rpk` using the following command:

```
pip install rpk
```

#### 4.4.6. Benefits of Using *rpk*

- **Standardisation:** Ensures consistency across robot projects by using pre-defined templates adhering to ROS 2 conventions.
- **Modularity:** Promotes the reuse of code and functionality across different projects and components.
- **Ease of Use:** Simplifies the setup process for robot applications, reducing the learning curve for new developers.
- **Flexibility:** Supports multiple robot types and use cases, making it adaptable to diverse projects.
- **Rapid Prototyping:** Enables developers to quickly bootstrap complex systems, focusing on custom functionality rather than boilerplate code.

The *rpk* tool is a robust solution for generating and managing skeletons for ROS 2-based robotic applications. Its intuitive command structure, extensive library of templates, and support for modular development make it an essential tool for robotics developers seeking efficiency and best practices in their workflows. By simplifying the initial stages of application development, *rpk* empowers teams to dedicate more time to innovation and problem-solving.

## 5. Conclusion and Outlook

This deliverable highlights the progress made in Task T2.3, which focuses on the development of a library of open modules for agile HRI development. By building a library of open modules based on the ROS4HRI framework and integrating advanced technologies like ROS2, we have established a strong foundation for scalable, customizable, and efficient HRI systems.

Key achievements include:

- Migrating ROS4HRI components to ROS2, enhancing modularity, compatibility, and performance.
- Designing a comprehensive software architecture template that streamlines the development process for HRI applications.
- Creating a template generator to accelerate the creation of LLM-enabled and ROS2-compatible HRI modules.

These achievements provide a robust foundation for continued development, addressing the needs of both the ARISE ecosystem and the broader robotics community. They align with the overarching vision of fostering innovation and collaboration by providing open-source tools for industrial and research applications.

Building on these accomplishments, **the next phase of Task T2.3** will focus on:

1. **Defining TEFs System Skills:** Identifying and documenting the system skills provided by ARISE Testing and Experimentation Facilities (TEFs), ensuring they are readily accessible to FSTP participants. This will involve refining the skill catalog and linking them to specific industrial use cases to demonstrate real-world applicability.
2. **Integration with Vulcanexus:** Finalizing the integration of ROS4HRI modules with eProsima's Vulcanexus ROS distribution.
3. **Iterative Refinements and Releases:** Continuing to improve the library through iterative development cycles, incorporating feedback from TEFs, FSTP participants, and industrial stakeholders. Future versions will include additional tools, modules, and documentation to address evolving needs.
4. **Dissemination and Collaboration:** Promoting the use of ROS4HRI modules and ARISE tools through case studies and public dissemination efforts. This will encourage adoption and foster collaboration across the robotics community.

These next steps are designed to solidify the library's impact, ensuring it remains a valuable resource for developing agile and human-centric industrial HRI applications. By iteratively refining our approach and collaborating with stakeholders, ARISE aims to push the boundaries of what is achievable in HRI.

## Annex A: ROS4HRI Package Details

This annex provides detailed descriptions and repository links for the ROS4HRI packages mentioned in Chapter 2. Each package is summarized with its key functionalities, latest version, repository links and some details.

### attention\_manager\_msgs

- *Version: 2.0.0*
- *Code repository: [https://github.com/pal-robotics/attention\\_manager\\_msgs](https://github.com/pal-robotics/attention_manager_msgs)*
- Description: ROS 2 messages used by the attention\_manager package

### chatbot\_msgs

- *Version: 2.1.0*
- *Code repository: [https://github.com/pal-robotics/chatbot\\_msgs](https://github.com/pal-robotics/chatbot_msgs)*
- Description: Messages, services and action definitions useful for chatbots

### hri\_actions\_msgs

- *Version: 2.4.2*
- *Code repository: [https://github.com/ros4hri/hri\\_actions\\_msgs](https://github.com/ros4hri/hri_actions_msgs)*
- Description: Action definitions useful for Human-Robot Interaction

### hri\_body\_detect

- *Version: 3.1.5*
- *Code repository: [https://github.com/ros4hri/hri\\_body\\_detect](https://github.com/ros4hri/hri_body_detect)*
- Description: ROS node implementing multibody 2D/3D full-body pose estimation, using Google Mediapipe. Part of ROS4HRI.
- Details:

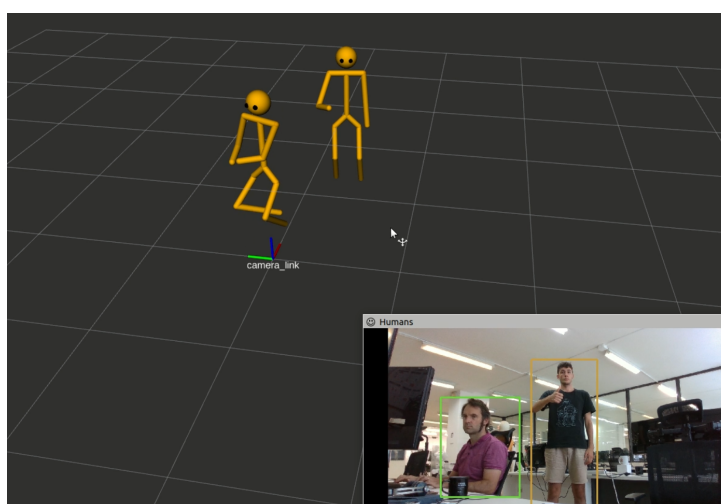


Figure 4: Example of skeleton detection

`hri_body_detect` is a ROS4HRI<sup>7</sup>-compatible 2D and 3D body pose estimation node.

It is built on top of Google Mediapipe 3D body pose estimation<sup>8</sup>.

The node provides the 2D and 3D pose estimation for the detected humans in the scene, implementing a robust solution to self-occlusions.

This node performs the body-pose detection pipeline, publishing information under the ROS4HRI naming convention regarding the body ids (on the `/humans/bodies/tracked` topic), the bodies bounding box, and the joint state of the bodys' skeleton.

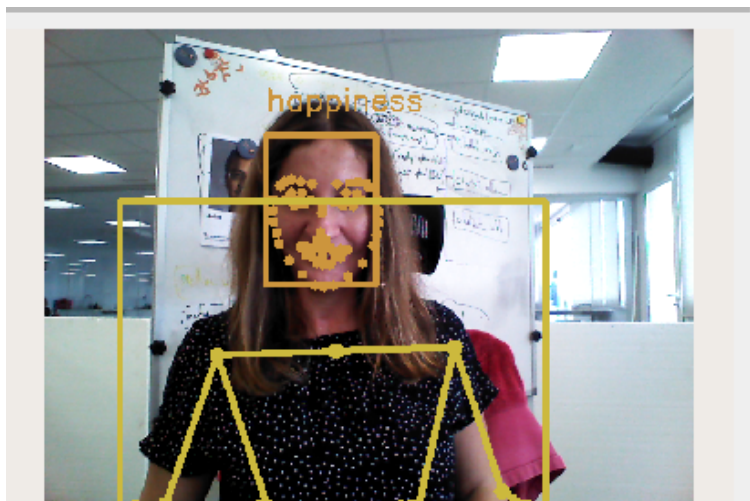
To estimate the body position, the node does not need a RGB-D camera, only RGB is required. However, using an RGB-D camera provides a more accurate depth estimation.

**Important:** to estimate the body depth without using a depth sensor, a calibrated RGB camera is required. You can follow this tutorial<sup>9</sup> to properly calibrate your camera.

Full documentation available in the project's README.<sup>10</sup>

### `hri_emotion_recognizer`

- *Version 1.0.1*
- *Code repository:* [https://github.com/ros4hri/hri\\_emotion\\_recognizer](https://github.com/ros4hri/hri_emotion_recognizer)
- *Description:* ROS4HRI-compliant expression recognizer node
- *Details:*



*Figure 5: Example of expression recognition output*

<sup>7</sup> <https://wiki.ros.org/hri>

<sup>8</sup> [https://ai.google.dev/edge/mediapipe/solutions/vision/pose\\_landmarker](https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker)

<sup>9</sup> [http://wiki.ros.org/camera\\_calibration/Tutorials/MonocularCalibration](http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration)

<sup>10</sup> [https://github.com/ros4hri/hri\\_body\\_detect#readme](https://github.com/ros4hri/hri_body_detect#readme)

The `hri_emotion_recognizer` node is for expression recognition using ONNX (Open Neural Network Exchange) models. It integrates within the ROS4HRI framework by analyzing the output from the `hri_face_detector` node to identify emotional states.

For more information about ONNX Fer Plus models, visit ONNX GitHub repository<sup>11</sup>.

Full documentation available in the project's README<sup>12</sup>.

### `hri_engagement`

- *Version 2.0.1*
- *Code repository:* [https://github.com/ros4hri/hri\\_engagement](https://github.com/ros4hri/hri_engagement)
- Description: Engagement detection for ROS4HRI
- Details:

`hri_engagement` is a ROS4HRI-compatible engagement detector node.

The node estimates the engagement status of persons who are in the field of view of the camera, using the 'visual social engagement' metric defined in "Measuring Visual Social Engagement from Proxemics and Gaze"<sup>13</sup>(by Webb and Lemaignan). This metric combines the distance between the robot and the person, with a measure of mutual gazing (whether the person and the robot are looking at each other).

For each of whose faces are detected by the robot (ie, in `/humans/faces/tracked`), it performs temporal filtering of this metric (over a 2 sec window) and then publishes the resulting level of engagement on the topic: `humans/persons/<person_id>/engagement_status`.

The engagement status is defined according to the `EngagementLevel.msg` defined in `hri_msgs`<sup>14</sup>. Here a quick overview of the different states:

- UNKNOWN: no information is provided about the engagement level
- DISENGAGED: the human has not looked in the direction of the robot
- ENGAGING: the human has started to look in the direction of the robot
- ENGAGED: the human is fully engaged with the robot
- DISENGAGING: the human has started to look away from the robot

Full documentation available in the project's README.<sup>15</sup>

### `hri_face_body_matcher`

- *Version: 2.0.3*
- *Code repository:* [https://github.com/ros4hri/hri\\_face\\_body\\_matcher](https://github.com/ros4hri/hri_face_body_matcher)
- Description: The `hri_face_body_matcher` package
- Detailed description

---

<sup>11</sup> [https://github.com/onnx/models/tree/main/validated/vision/body\\_analysis/emotion\\_ferplus](https://github.com/onnx/models/tree/main/validated/vision/body_analysis/emotion_ferplus)

<sup>12</sup> [https://github.com/ros4hri/hri\\_emotion\\_recognizer#readme](https://github.com/ros4hri/hri_emotion_recognizer#readme)

<sup>13</sup> <https://ieeexplore.ieee.org/document/9900801>

<sup>14</sup> [https://github.com/ros4hri/hri\\_msgs/blob/master/msg/EngagementLevel.msg](https://github.com/ros4hri/hri_msgs/blob/master/msg/EngagementLevel.msg)

<sup>15</sup> [https://github.com/ros4hri/hri\\_engagement#readme](https://github.com/ros4hri/hri_engagement#readme)

`hri_face_body_matcher` is a ROS4HRI-compatible face to body matcher node.

It finds the most likely matches between the recognized faces and bodies based on their relative position in the source image.

#### Algorithm

For each of the possible associations of recognized face and body, a matching cost is computed, linearly decreasing with the distance between the body nose and the face center in the image.

The rate the confidence drops is proportional to the `~confidence_scaling_factor` parameter and the face size, intended as its diagonal length.

$$\text{confidence} = \max(0, 1 - \frac{\text{distance} * \text{c.s.f.}}{2 * \text{face\_size}})$$

Full documentation available in the project's README<sup>16</sup>.

#### `hri_face_detect`

- *Version: 2.0.10*
- *Code repository: [https://github.com/ros4hri/hri\\_face\\_detect](https://github.com/ros4hri/hri_face_detect)*
- Description: Google Mediapipe-based face detection for ROS4HRI
- Details:

A ROS4HRI-compliant ROS node to perform fast face detection using YuNet face detector<sup>17</sup> and Mediapipe Face Mesh<sup>18</sup>. The former performs well at greater distances (depending on image resolution and image scaling applied) and extracts 5 keypoints. The latter works only at close distances and extracts all the ROS4HRI-defined landmarks.

Full documentation available in the project's README<sup>19</sup>.

#### `hri_face_identification`

- *Version: 2.3.1*
- *Code repository: [https://github.com/ros4hri/hri\\_face\\_identification](https://github.com/ros4hri/hri_face_identification)*
- Description: `hri_face_identification` is a ROS4HRI-compatible package for face identification
- Details:

A ROS4HRI-compatible face identification package. It is built on top of dlib's face recognition pipeline<sup>20</sup>. It performs face recognition at > 100fps on a GTX10xx generation mobile GPU.

---

<sup>16</sup> [https://github.com/ros4hri/hri\\_face\\_body\\_matcher#readme](https://github.com/ros4hri/hri_face_body_matcher#readme)

<sup>17</sup> <https://github.com/ShiqiYu/libfacedetection>

<sup>18</sup> [https://github.com/google/mediapipe/blob/master/docs/solutions/face\\_mesh.md](https://github.com/google/mediapipe/blob/master/docs/solutions/face_mesh.md)

<sup>19</sup> [https://github.com/ros4hri/hri\\_face\\_detect#readme](https://github.com/ros4hri/hri_face_detect#readme)

<sup>20</sup> <http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html>

Because this node is meant to be used in a broader face processing pipeline, it does not perform face detection itself. It expects instead faces to be detected and published under `/humans/faces/...` (see below the list of subscribed topics).

Full documentation available in the project's README.<sup>21</sup>

### libhri

- *Version: 2.6.2*
- *Code repository: <https://github.com/ros4hri/libhri>*
- *Description: A wrapper library around the ROS4HRI ROS topics*
- *Details: Full documentation available in the project's README.<sup>22</sup>*

### hri\_msgs

- *Version: 2.2.0*
- *Code repository: [https://github.com/ros4hri/hri\\_msgs](https://github.com/ros4hri/hri_msgs)*
- *Description: Messages, services and action definitions useful for Human-Robot Interaction. They are directly related to the ROS REP-155.*

### hri\_person\_manager

- *Version: 2.0.6*
- *Code repository: [https://github.com/ros4hri/hri\\_person\\_manager](https://github.com/ros4hri/hri_person_manager)*
- *Description: Combines information about people's face, body, voice into on consistent representation, following the ROS4HRI conventions*
- *Details:*

A ROS4HRI-compatible person manager node. It aggregates information about detected faces, bodies and voices into consistent *persons*, and exposes these detected persons with their links to their matching face, body, voice.

The aggregation relies on additional ROS nodes to provide candidate 'matches' between persons and/or body parts. For instance a face recogniser, that would publish candidate matches between faces and unique persons.

Algorithm: This node collects the candidate matches between features (face, body, voice) or between feature and person. Then it creates a probabilistic graph from the ones with likelihood higher than a threshold.

Finally it looks for a partition of that graph such that: - each subgraph is connected - each subgraph has at most one feature of each type - the overall likelihood of associations is maximised

The resulting most likely graph is published.

Full documentation available in the project's README.<sup>23</sup>

---

<sup>21</sup> [https://github.com/ros4hri/hri\\_face\\_identification#readme](https://github.com/ros4hri/hri_face_identification#readme)

<sup>22</sup> <https://github.com/ros4hri/libhri#readme>

<sup>23</sup> [https://github.com/ros4hri/hri\\_person\\_manager#readme](https://github.com/ros4hri/hri_person_manager#readme)

### hri\_privacy\_msgs

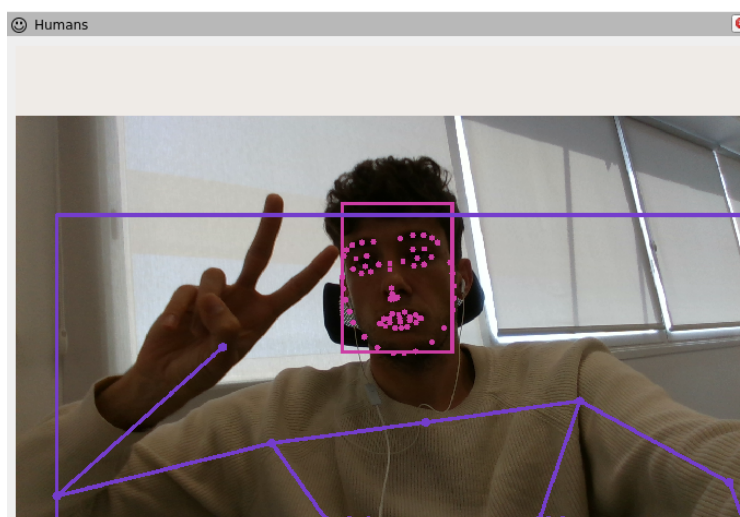
- *Version: 1.2.0*
- *Code repository: [https://github.com/ros4hri/hri\\_privacy\\_msgs](https://github.com/ros4hri/hri_privacy_msgs)*
- *Description: ROS message definitions for declaring privacy-sensitive data flows*

### hri\_rviz

- *Version: 2.1.0*
- *Code repository: [https://github.com/ros4hri/hri\\_rviz](https://github.com/ros4hri/hri_rviz)*
- *Description: Set of rviz plugins for ROS4HRI data visualization*
- *Details: This package provides a list of rviz2 plugins for human-related data visualisation. It is part of the ROS4HRI ecosystem. Full documentation available in the project's README<sup>24</sup>.*

### Plugins

A plugin for visualising 2D information overlaid on a camera stream (ideally, the stream used to detect it). Currently, the plugin can visualise: - Face bounding boxes - Face landmarks - Body bounding boxes - 2D skeleton keypoints



*Figure 6: RViz's Humans plugin*

### Skeletons3D

A plugin for visualising the estimated 3D poses of the detected humans.

<sup>24</sup> [https://github.com/ros4hri/hri\\_rviz#readme](https://github.com/ros4hri/hri_rviz#readme)

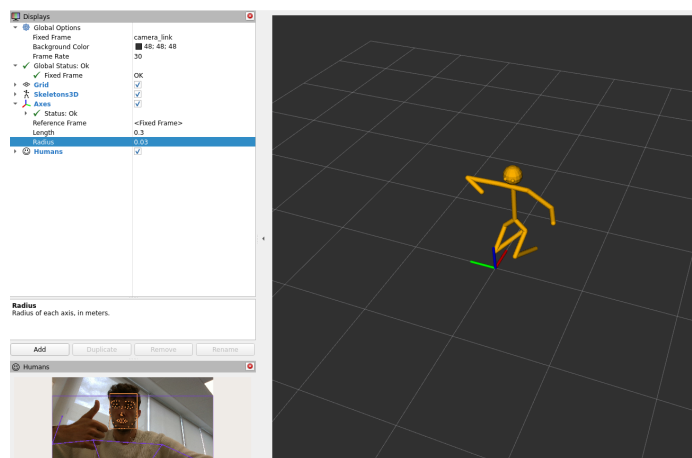


Figure 7: RViz's Skeletons3D plugin

### TF\_HRI

A plugin for visualising the human-related TF frames. These are highly dynamical, appearing and disappearing in a matter of seconds. Using the classic TF plugin would result in a crowded and chaotic frames visualisation. This plugin: - looks over the detected faces and bodies; - only displays the face and bodies TF frames for the currently detected bodies and faces; - readily remove the TF frames for those bodies and faces that are no more tracked, avoiding the disappearing phase observed in the original TF frame for the non-updated frames. It is possible to select which human frames to visualise among: - face frames; - gaze frames; - body frames.

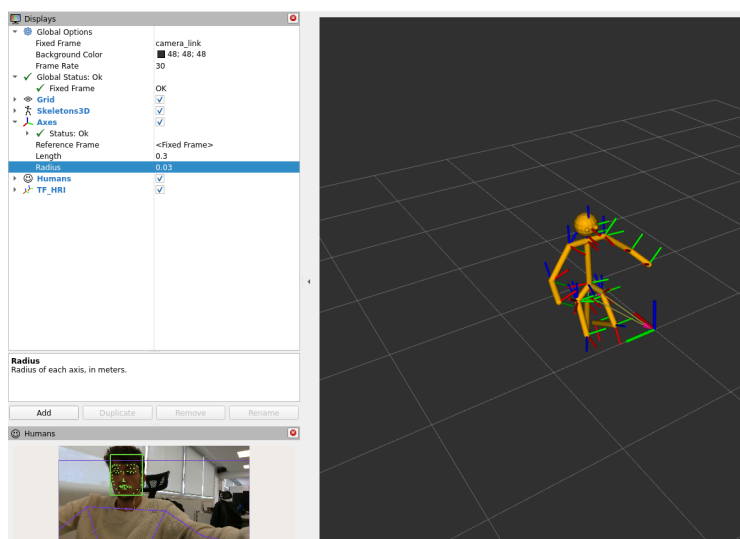


Figure 8: RViz's TF\_HRI plugin

### hri\_visualization

- Version 2.1.0
- Code repository: [https://github.com/ros4hri/hri\\_visualization](https://github.com/ros4hri/hri_visualization)
- Description: ROS4HRI-compatible node to visualize body and face detection results

- Details: The node generates image streams to visualise face/body detection results. These results must follow the ROS4HRI convention to be visualised by the hri\_visualization node. Full documentation available in the project's README.<sup>25</sup>

### human\_description

- *Version: 2.0.2*
- *Code repository: [https://github.com/ros4hri/human\\_description](https://github.com/ros4hri/human_description)*
- Description: This package contains a parametric kinematic description of humans. The files in this package are parsed and used by a variety of other components, notably in the context of human-robot interaction. Most users will not interact directly with this package.
- Details:

Parametric kinematic model of humans, in URDF format, for usage in robotics and HRI

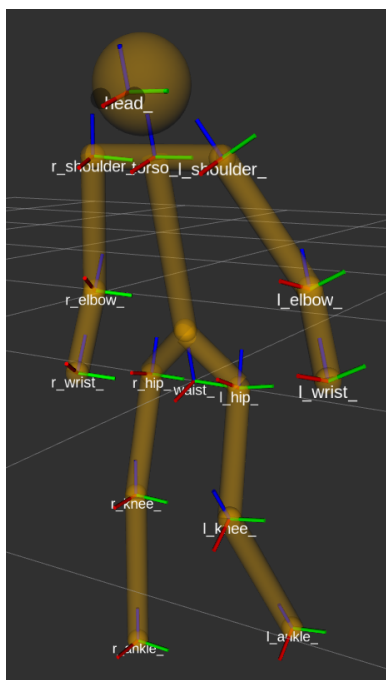


Figure 9: Human model, rendered in rviz

Part of the ROS4HRI project.

Full documentation available in the project's README.<sup>26</sup>

### kb\_msgs

- *Version: 1.1.0*

<sup>25</sup> [https://github.com/ros4hri/hri\\_visualization#readme](https://github.com/ros4hri/hri_visualization#readme)

<sup>26</sup> [https://github.com/ros4hri/human\\_description#readme](https://github.com/ros4hri/human_description#readme)

- Code repository: [https://github.com/pal-robotics/kb\\_msgs/](https://github.com/pal-robotics/kb_msgs/)
- Description: ROS services definition for knowledge-related tasks

knowledge\_core

- Version: 3.3.0
- Code repository: [https://github.com/severin-lemaignan/knowledge\\_core](https://github.com/severin-lemaignan/knowledge_core)
- Description: The KnowledgeCore knowledge base, including its service-based ROS API
- Details:

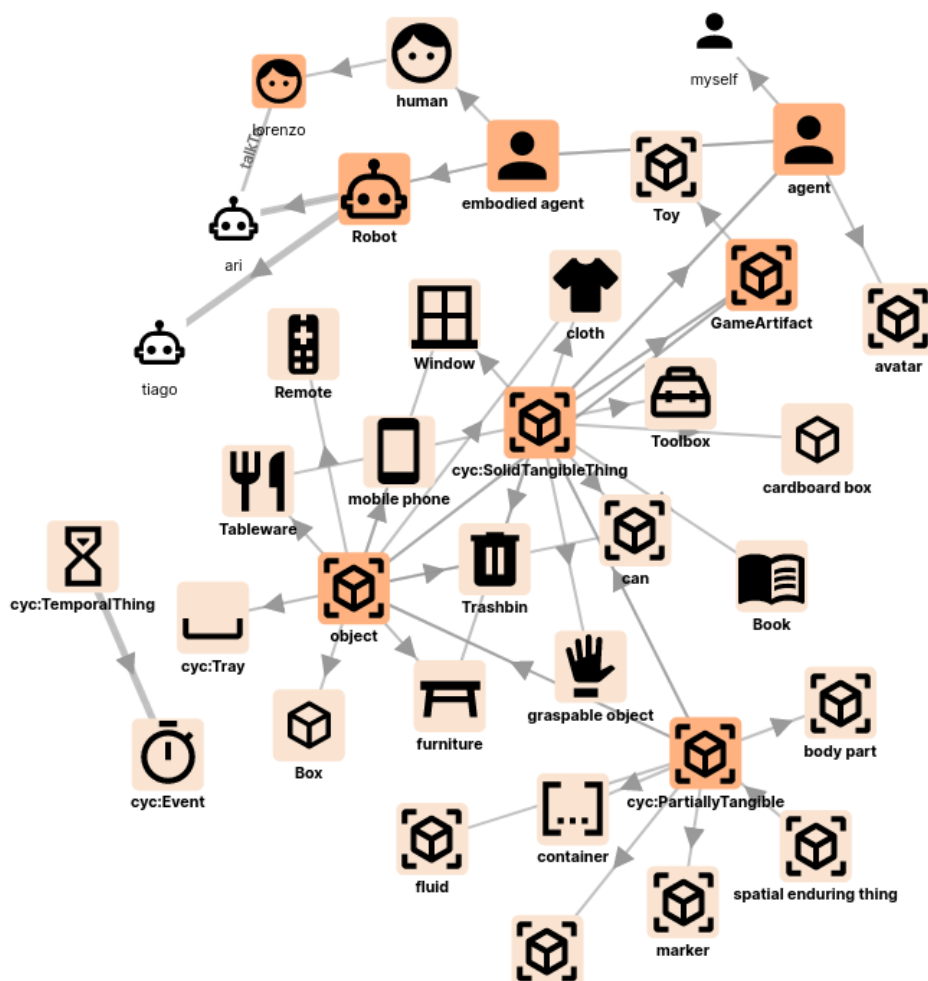


Figure 10: Screenshot of the included KnowledgeCore explorer

KnowledgeCore is a RDFlib-backed minimalistic knowledge base, initially designed for robots (in particular human-robot interaction or multi-robot interaction). It features full ROS <sup>27</sup>support.

It stores triples (like RDF/OWL triples), and provides an API <sup>28</sup>accessible via a simple socket protocol or a ROS wrapper.

<sup>27</sup> <https://www.ros.org>

<sup>28</sup> <http://doc/api.md>

pykb<sup>29</sup> provides an idiomatic Python binding over the socket interface, making it easy to integrate the knowledge base in your application. A similar API wrapper exists for ROS as well (see example below).

It integrates with the reasonable<sup>30</sup> OWL2 RL reasoner to provide OWL2 semantics and fast knowledge materialisation.

Full documentation available in the project's README.<sup>31</sup>

#### oro

- *Version: 2.2.1*
- *Code repository: <https://github.com/severin-lemaignan/openrobots-ontology/>*
- *Description: The OpenRobots ontology*
- *Details: Full documentation available in the project's README.<sup>32</sup>*

#### people\_facts

- *Version: 2.0.1*
- *Code repository: [https://github.com/pal-robotics/people\\_facts](https://github.com/pal-robotics/people_facts)*
- *Description: The people\_facts package*
- *Details: This node listens to various ROS4HRI topics and stores semantic information about humans into a knowledge base. Full documentation available in the project's README.<sup>33</sup>*

#### pyhri

- *Version 2.6.2*
- *Code repository: <https://github.com/ros4hri/libhri>*
- *Description: A python wrapper around the libhri C++ library*
- *Details: Full documentation available in the project's README.<sup>34</sup>*

#### rqt\_chat

- *Version: 1.0.0*
- *Code repository: [https://github.com/pal-robotics/rqt\\_chat](https://github.com/pal-robotics/rqt_chat)*
- *Description: A RQt chat interface compatible with ROS4HRI*
- *Details:*

---

<sup>29</sup> <https://github.com/severin-lemaignan/pykb>

<sup>30</sup> <https://github.com/gtfierro/reasonable>

<sup>31</sup> [https://github.com/severin-lemaignan/knowledge\\_core#readme](https://github.com/severin-lemaignan/knowledge_core#readme)

<sup>32</sup> <https://github.com/severin-lemaignan/openrobots-ontology/#readme>

<sup>33</sup> [https://github.com/pal-robotics/people\\_facts#readme](https://github.com/pal-robotics/people_facts#readme)

<sup>34</sup> <https://github.com/ros4hri/libhri#readme>

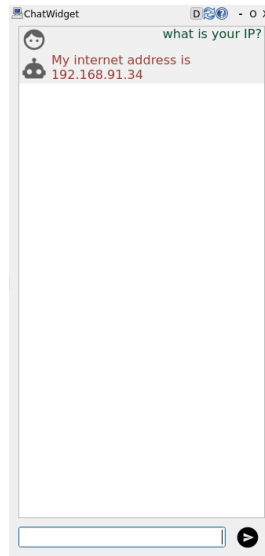


Figure 11: *rqt\_chat* screenshot

A simple chat plugin for rqt, compatible with ROS4HRI.

User messages are published as `hri_msgs/msg/LiveSpeech` messages on the `/humans/voices/anonymous_speaker/speech` topic.

TTS action call to `/tts_engine/tts` (`tts_msgs/action/TTS`) are then displayed as robot's messages.

Full documentation available in the project's README.<sup>35</sup>

### **rqt\_human\_radar**

- *Version: 2.2.1*
- *Code repository: [https://github.com/ros4hri/rqt\\_human\\_radar](https://github.com/ros4hri/rqt_human_radar)*
- Description: A radar-like visualization for humans in the scene, representing their position, orientation, engagement level
- Details: An rqt plugin showing a radar view of the objects and people in field of view of the robot. When talking about people, the plugin represents their engagement status as well. Full documentation available in the project's README.<sup>36</sup>

### **tts\_msgs**

- *Version: 1.1.0*
- *Code repository: [https://github.com/pal-robotics/pal\\_tts\\_msgs](https://github.com/pal-robotics/pal_tts_msgs)*
- Description: PAL text-to-speech message definition

<sup>35</sup> [https://github.com/pal-robotics/rqt\\_chat#readme](https://github.com/pal-robotics/rqt_chat#readme)

<sup>36</sup> [https://github.com/ros4hri/rqt\\_human\\_radar#readme](https://github.com/ros4hri/rqt_human_radar#readme)