18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID # 350

AI4U, Digital Assistant for Operations

Denis Baron[a]*, Ambroise Marche[b]*, Gregory Navarro[c], Alexis Paillet[d]

[a] *Spaceship France Project, Subdirectorate Exploration and Human Space Flight, CNES (Centre National d'Etudes Spatiales), 18 avenue Edouard Belin 31400 Toulouse, France,* denis.baron@cnes.fr
[b] *Ambroise Marche, 24 rue du maréchal Niel 31100 Toulouse, France,* ambroise.marche@gmail.com
[c] *Spaceship France Project, Subdirectorate Exploration and Human Space Flight, CNES (Centre National d'Etudes Spatiales), 18 avenue Edouard Belin 31400 Toulouse, France,* gregory.navarro@cnes.fr
[d] *Spaceship France Project Manager, Subdirectorate Exploration and Human Space Flight, CNES (Centre National d'Etudes Spatiales), 18 avenue Edouard Belin 31400 Toulouse, France,* alexis.paillet@cnes.fr
* Corresponding Authors

Abstract

Spaceship FR is the French contribution to the European Space Agency's Spaceships program. Led by CNES, the French Space Agency, this project aims to test the technologies needed for future lunar and Mars bases. One of the main challenges facing astronauts is isolation, which will force us to rethink the way manned missions are managed from Earth. Because of communications latency, most missions will have to be conducted with greater autonomy and less control center support. These lunar and Martian outposts will be intelligent systems that will collect a vast amount of data, both about the base itself and the astronauts who will reside there. This data will need to be used locally to ensure the success of the missions. In this context, AI4U is an intelligent assistant that allows astronauts to interact with this knowledge base in natural language, correlating all the data it contains. Its main functions are to assist the crew during operations, to relieve them of repetitive and tedious tasks, and to act as a countermeasure to the long mission time that astronauts will experience. The latest developments in AI4U have been to improve the relevance of its answers and, above all, to increase the trust that can be placed in it, thanks to RAG (Retrieval Augmented Generation). We also worked on making AI4U more autonomous, i.e. independent of any Internet connection, to put it in a situation identical to what astronauts will encounter on lunar and Mars missions. Finally, we continued to work on the human factors aspect, evaluating the operational concept with the help of operators from CADMOS (Centre d'Aide au Développement des Activités en Micropesanteur et des Opérations Spatiales). The aim of this paper is to present these developments and show how AI4U interfaces with other systems to enable astronauts to interact with their environment.

Keywords: Intelligent agent, Large Language Models, Hybrid NLP, Astronaut operations, Spaceship FR, AMAIA, Retrieval-Augmented Generation (RAG), Model Context Protocol (MCP), Edge Computing, Digital Twin, Human-Agent Interaction.

Acronyms/Abbreviations

AI	Artificial Intelligence
AI4U	The digital assistant project
AMAIA	Astronaut Multipurpose AI Assistant ASR Automatic Speech Recognition
CADMOS	Centre d'Aide au Développement des Activités en Micropesanteur et des Opérations Spatiales
CNES	Centre National d'Études Spatiales
ESA	European Space Agency
GPU	Graphics Processing Unit
HA	Home Assistant
HR	Heart Rate
IoT	Internet of Things
JSON-RPC	JavaScript Object Notation Remote Procedure Call
LLM	Large Language Model
MDRS	Mars Desert Research Station
MCP	Model Context Protocol
MQTT	Message Queuing Telemetry Transport
NLP	Natural Language Processing
RAG	Retrieval-Augmented Generation
ReAct	Reasoning and Acting (LLM paradigm)
TTS	Text-to-Speech

1 Introduction

The next era of human space exploration, spearheaded by initiatives like NASA's Artemis programme and ESA's Terrae Novae 2030+ strategic roadmap, envisions establishing sustainable human presences on the Moon and eventually Mars. These ambitious undertakings involve extended stays far from Earth, fundamentally altering the operational landscape. Communication round-trip times, ranging from several seconds for lunar missions to potentially over 40 minutes for Mars missions, render traditional ground control centric operations impractical.

This latency necessitates a paradigm shift, empowering astronaut crews with unprecedented levels of autonomy and transferring significant operational responsibility from Mission Control Centers to the crew onboard the habitat or vehicle.

In this context, autonomous digital assistants emerge as a critical enabling technology. Such systems must be capable of seamlessly integrating and interpreting vast amounts of data from diverse sources – technical telemetry from habitat subsystems, environmental sensor readings, crew biometric data, procedural documents, and inputs from the crew themselves. They need to provide intuitive, reliable, and context-aware support to ensure mission safety and efficiency.

Building upon prior CNES research in astronaut digital assistants, initially presented under the name AI4U [1], the project has recently undergone a renaming.
For the remainder of this paper, the assistant will be referred to by its current designation, AMAIA (Astronaut Multipurpose AI Assistant), reflecting this change.

AMAIA is designed to deliver several key capabilities:

1. **Natural Interaction**: Provide conversational interaction capabilities, accepting both voice and text inputs in the crew's primary working language (initially French and English).
2. **Reliable Control**: Enable robust execution of commands related to habitat subsystems (e.g., lighting, life support) and robotic assets (e.g., rovers, manipulators).
3. **Intelligent Monitoring & Summarization**: Proactively monitor system health, identify anomalies, and provide concise summaries of complex situations or data trends upon request.
4. **Cognitive Support**: Offer access to onboard knowledge bases (manuals, procedures, scientific data) and potentially act as a cognitive aid, helping to mitigate the psychological effects of isolation and high workload.

A core design philosophy is ensuring trustworthiness and offline capability. All core functionalities, including complex language understanding and speech processing, must operate reliably without requiring a connection to Earth-based servers.

This paper details the progress and current maturity of the AMAIA system as of mid-2025. It reflects the advancements made following its initial deployment and evaluation during the Mars Desert Research Station (MDRS) Crew 311 analogue mission campaign. Section 2 provides the programmatic background within the Spaceship France initiative and outlines AMAIA's specific objectives. Section 3 delves into the technical details of the hybrid NLP architecture, the crucial role of the Model Context Protocol (MCP) for tool integration, and the offline speech pipeline. Section 4 describes the comprehensive digital simulation and validation environment created to test AMAIA in realistic scenarios. Section 5 discusses performance metrics, particularly latency, using feedback from analogue mission astronauts. Finally, Section 6 outlines the near-term development priorities and future work towards a fully autonomous assistant for the astronaut

2 Programme Context and Objectives

AMAIA is a technological component developed under the umbrella of Spaceship France [1]. This CNES-led initiative serves as the French contribution to ESA's broader Spaceship programme, which aims to federate European efforts in developing the technologies necessary for future lunar surface infrastructure and, eventually, Mars exploration. Spaceship France focuses on creating demonstrators and maturing technologies related to habitats, rovers, power systems, and the digital environment required to operate them autonomously.

Within this framework, AMAIA's primary goal is to provide a truly astronaut-centric interface that significantly enhances crew autonomy and efficiency, particularly under communication latency constraints. It moves beyond traditional dashboards and command lines by allowing astronauts to interact with complex systems using natural language, both spoken and typed. The core objectives derived from this goal are:

- **Intuitive Human-Agent Interaction**: Allow astronauts to issue commands, ask questions, and receive information using conversational language, minimizing the need for specialized training on complex interfaces. The system must understand the intent behind user requests, even when phrased imprecisely or conversationally.
- **Reliable System Control**: Provide a dependable mechanism for controlling habitat subsystems (e.g., lights, climate control, power distribution) and robotic assets (e.g., commanding rover movements, operating manipulators) via natural language commands. This requires robust intent recognition and mapping to specific system actions.
- **Context-Aware Information Access**: Enable querying of vast onboard knowledge bases, including operational procedures, technical manuals, system status logs, scientific data, and crew schedules. The system must be able to retrieve relevant information and synthesize it effectively, often correlating data from multiple sources (e.g., "What was the average power consumption of the greenhouse yesterday while the main lights were on?").
- **Proactive Assistance**: Move beyond simple question-answering to proactively monitor systems, detect potential anomalies or trends (e.g., rising CO2 levels, low battery on a sensor), and alert the crew or provide summaries without explicit requests.
- **Fully Offline Operation**: Ensure all core functionalities, including ASR, NLP, LLM reasoning, RAG, and TTS, can run entirely on local hardware within the habitat or vehicle ("at the edge"). This is non-negotiable for deep-space missions due to latency and potential communication blackouts.
- **Modularity and Extensibility**: Design the system architecture to be open and extensible, allowing easy integration of new data sources, control capabilities (tools), and updated AI models over the mission lifetime. This is achieved through adherence to the Model Context Protocol (MCP).

The specific features and architectural choices described in this paper, such as the hybrid NLP approach and the MCP standard, were refined based on requirements gathered during stakeholder workshops involving CNES operational (from CADMOS) and scientific experts. These refinements form the basis of the Version 1 (v1) AMAIA baseline. The focus is on demonstrating a robust, integrated system capable of handling representative operational tasks in a simulated environment.

3 System Architecture

3.1 Overview: A Hybrid Approach

The core of AMAIA is built upon a hybrid conversational architecture, illustrated in Figure 1. This design explicitly addresses the often-conflicting requirements of real-time responsiveness for simple, frequent commands and deep reasoning capabilities for complex, novel queries or tasks.

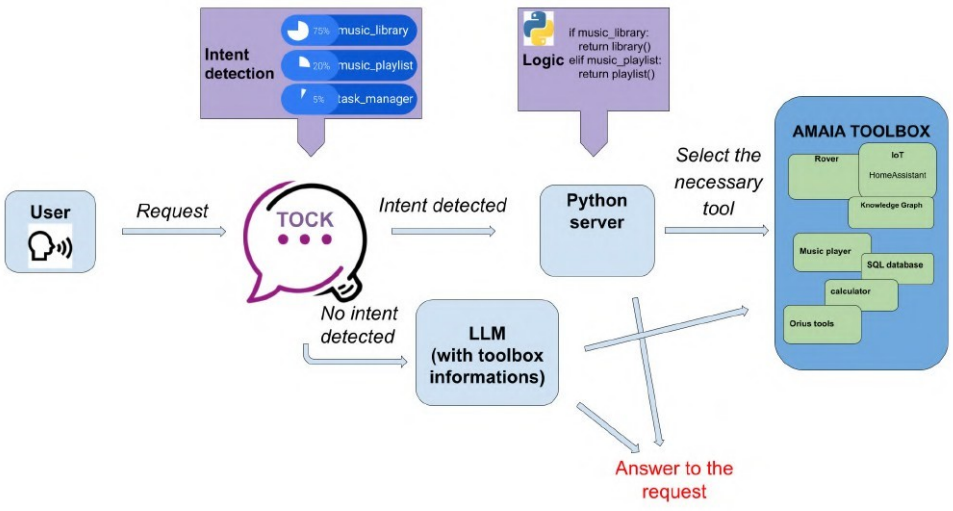

Fig. 1: AMAIA's hybrid conversational architecture, combining fast intent classification (Tock)

with flexible LLM-based reasoning (LangChain Agent using ReAct) and MCP-compliant tools.

The system functions as follows:

1. **Input Processing:** User input, whether received via voice (after ASR) or text chat, is first processed by a lightweight, highly optimized intent classifier. We currently utilize Tock [6], an open-source conversational AI platform known for its speed and efficiency.
2. **Intent Classification (Front-Gate):** Tock attempts to match the input utterance against a prede-fined set of common intents and entities relevant to spacecraft operations (e.g., `turn_on_light`,`get_temperature`, `check_rover_status`). These intents are trained on domain-specific examples. If an intent is recognized with high confidence (above a configurable threshold), Tock directly handles the request. This path is extremely fast, typically responding in under 200 milliseconds (see Section 5). Examples include:
   - User: " *Turn on the light in the greenhouse.*" (→ Tock recognizes `intent: turn_on_light`, `entity: device=light`, `entity: location=greenhouse`
   - User: "*What's the current CO2 level?*" → Tock recognizes `intent: get_sensor_reading`, `entity: sensor_type=CO2`.

   For intents requiring external action (like controlling a light or querying a sensor), Tock can directly invoke the appropriate MCP tool (see Section 3.2) or execute predefined logic.
3. **Escalation to LLM Agent:** If Tock cannot confidently classify the intent, or if the request is inherently complex, ambiguous, or requires multi-step reasoning or information synthesis beyond predefined patterns, the query is escalated to a more powerful Large Language Model (LLM) agent [4].
4. **LLM Agent Reasoning (ReAct Paradigm):** The LLM agent is built using the LangChain framework (`https://www.langchain.com/`) and employs the ReAct (Reasoning and Acting) paradigm [2]. ReAct enables the LLM to synergize reasoning and action-taking in an iterative loop. Given a complex query, the LLM:
   - **Reasons:** Analyzes the query and determines a plan or the next logical step. This often involves deciding which available tool would be most helpful.
   - **Acts:** Selects an appropriate tool (e.g., the RAG tool for document search, the Home Assistant tool for device control) and formulates the input required for that tool.
   - **Observes:** Receives the output from the tool execution.
   - **Repeats:** Uses the observation to refine its understanding and reasoning, potentially choosing another tool or formulating the final answer for the user.

   This iterative process allows the agent to tackle complex tasks like: "*Summarize the main maintenance tasks performed on the water recycling unit last week and check if its current efficiency is nominal according to the operations manual.*" This would involve querying a maintenance log database (potentially via RAG), querying the current system telemetry (via Home Assistant tool), retrieving nominal parameters from the manual (via RAG), comparing them, and synthesizing a response. We utilize an optimized version of a state-of-the-art instruction-following LLM (e.g., a quantized Mistral-24B variant) suitable for offline deployment on edge hardware.
5. **Response Generation:** The final response, whether generated directly by Tock or synthesized by the LLM agent, is presented to the user via the chat interface or converted to speech using the TTS module.

This hybrid approach provides the best of both worlds: near-instantaneous response for frequent, simple commands via the Tock "front-gate," ensuring a smooth user experience, while retaining

the flexibility and deep reasoning power of an LLM agent for handling the long tail of complex and unforeseen requests. The entire execution flow, especially the LLM agent's reasoning steps and tool usage, is designed to be transparent and auditable for operational safety and debugging.

3.2 MCP Tooling Ecosystem

A cornerstone of AMAIA's flexibility and extensibility is its adherence to the Model Context Protocol (MCP). MCP is a lightweight, open standard proposed by Anthropic to facilitate the interoperability between LLM agents and various domain-specific tools, sensors, or actuators in a space operations context.

It defines a simple yet effective way for tools to advertise their capabilities and for the LLM agent to discover and invoke them dynamically at runtime.

MCP Standard Definition: Each MCP-compliant tool must provide metadata describing its function, inputs, and outputs. This metadata typically includes:

- `name`: A unique identifier for the tool (e.g., `home_assistant_get_state`, `rover_move_forward`).
- `description`: A natural language description of what the tool does, designed to be understood by the LLM. This is crucial for the agent's ability to select the correct tool based on the user's request. Example: "'Retrieves the current state or sensor value of a specified device managed by Home Assistant. Use this to get temperature, humidity, light status, etc.'"
- `inputs`: A specification of the arguments the tool expects, often defined using a schema like JSON Schema. This tells the agent what information it needs to provide when calling the tool. Example: `"{""device_id"": ""string"", ""attribute"": ""string (optional)""}"`.
- `outputs`: A specification of the data the tool returns upon successful execution. Example: `"{""state"": ""string"", ""value"": ""number/string"", ""unit"": ""string (optional)""}"`.

The interaction itself typically uses a standard remote procedure call mechanism like JSON-RPC over a message bus (e.g., MQTT) or a direct API call. When the LLM agent decides to use a tool (based on its reasoning and the tool's description), it formats the required input parameters according to the schema and invokes the tool's endpoint. The tool executes its specific logic (e.g., querying a database, sending a command to hardware) and returns the result, which the LLM then uses in its reasoning loop.

Current AMAIA MCP Tool Adapters: AMAIA's current MCP ecosystem includes adapters for key functionalities required in the Spaceship FR context:

- **Habitat Monitoring & Actuation via Home Assistant (HA):** Home Assistant (`https://www. home-assistant.io/`) acts as a central hub, integrating various sensors and actuators within the simulated habitat using protocols like MQTT. The HA MCP tool allows AMAIA to:
  - Query the state of any entity managed by HA (e.g., "*What is the temperature in the greenhouse?*" → Agent calls HA tool with `device_id: sensor.greenhouse_temperature`).
  - Control actuators (e.g., "*Turn off the main cabin lights.*" → Agent calls HA tool with `service: light.turn_off`, `target: light.main_cabin`).
  - Retrieve historical data (e.g., "*Plot the CO2 trend over the last 24 hours.*").
- **Robotic Control of the *Scout* Rover:** This tool interfaces with the control system of the *Scout* rover prototype via MQTT. Specific MCP actions allow AMAIA to translate natural language commands into rover actions:

  - Locomotion: `rover_move(direction, distance)`, `rover_turn(direction, angle)`.
  - Teleoperation assist: `rover_follow_target(target_id)`.
  - Manipulator control: `rover_arm_action(action_type, target_object)` (e.g., `action_type: pick_up`, `target_object: sample_vial_3`).

- **Biomedical Data Streams:** This tool integrates with data collected from Samsung Galaxy® smartwatches worn by the crew. Data (HR, SpO_2, skin temperature, step count) is relayed (typically via Bluetooth to a local device, then MQTT) and made available through Home Assistant or a dedicated MCP tool. This allows queries like: "*What is my current heart rate?*" or potentially alerts based on predefined thresholds.
- **Task Scheduling Tool:** This tool allows AMAIA to schedule actions for future execution, based on user requests like "*Remind me to check the water filter in 1 hour*" or "*Turn off the greenhouse lights tonight at 22:00*".
  - It accepts the details of the action to be performed (often involving calls to other MCP tools like the Home Assistant tool) and the desired timing (relative delay or absolute time).
  - A backend persistent scheduler service (e.g., using libraries like APScheduler) manages the queue of scheduled tasks.
  - When the scheduled time arrives, the service triggers the execution of the corresponding action.
  - This capability is crucial for managing routine tasks, reminders, and automating operations based on time-dependent conditions. Example usage: User says "*Dans 20 minutes éteins le ventilateur de la baie technique*" (In 20 minutes, turn off the fan in the engineering bay) → LLM Agent calls `schedule_task` with action details (condition: time=20, room_name: engineering bay, action: turn_off_fan()).
- **Knowledge Retrieval and Memory Management (RAG):** This essential tool implements the Retrieval-Augmented Generation (RAG) pipeline, providing the LLM agent with relevant context to generate factually grounded and contextually aware responses. Crucially, this mechanism retrieves information from **three distinct sources** to construct the context provided to the LLM:

  1. **Document Knowledge Base:** This consists of a collection of textual documents (e.g., procedures, manuals, scientific reports, past mission logs). When the LLM agent requires information potentially contained within these documents, it invokes the RAG tool with its query. The typical RAG process for this base involves:
     - 1. Embedding the query into a vector representation.
     - 2. Performing a similarity search against a pre-indexed vector database (e.g., using FAISS or ChromaDB) containing chunks of the onboard documents.
     - 3. Retrieving the most relevant document chunks.
     - 4. Providing these chunks back to the LLM agent as context.

     The LLM then uses this retrieved context, along with its internal knowledge and the original query, to generate a factually grounded answer, significantly reducing the risk of hallucination. This access is essential for answering questions like "What is the procedure for recalibrating the oxygen sensor?".
  2. **Foundational Knowledge Graph:** This structured knowledge base, organized as a graph, contains key facts and relationships specifically concerning the **space exploration domain**. This includes information such as **satellite specifications, orbital parameters, spacecraft system architectures, properties of celestial bodies,** technical details of onboard equipment, and other factual elements

relevant to the mission context. Its purpose is to provide AMAIA with a reliable source of core, domain-specific knowledge, complementing the information found in documents or conversation history [5][9]. When a user's query pertains to this structured knowledge, the RAG tool can query this graph to:

- Identify relevant entities or concepts (nodes) within the graph (e.g., a specific satellite, a component type, an orbital concept).
- Retrieve associated facts (node attributes, such as a satellite's mass or a component's operational range) and key relationships (edges connecting nodes, like 'component X is part of subsystem Y', or 'satellite Z is in orbit A').
- Provide this structured information as enriched context to the LLM, enabling it to reason about complex technical relationships or specific domain facts more accurately.

This graph ensures the consistency and accuracy of the fundamental, domain-specific factual information AMAIA relies upon.

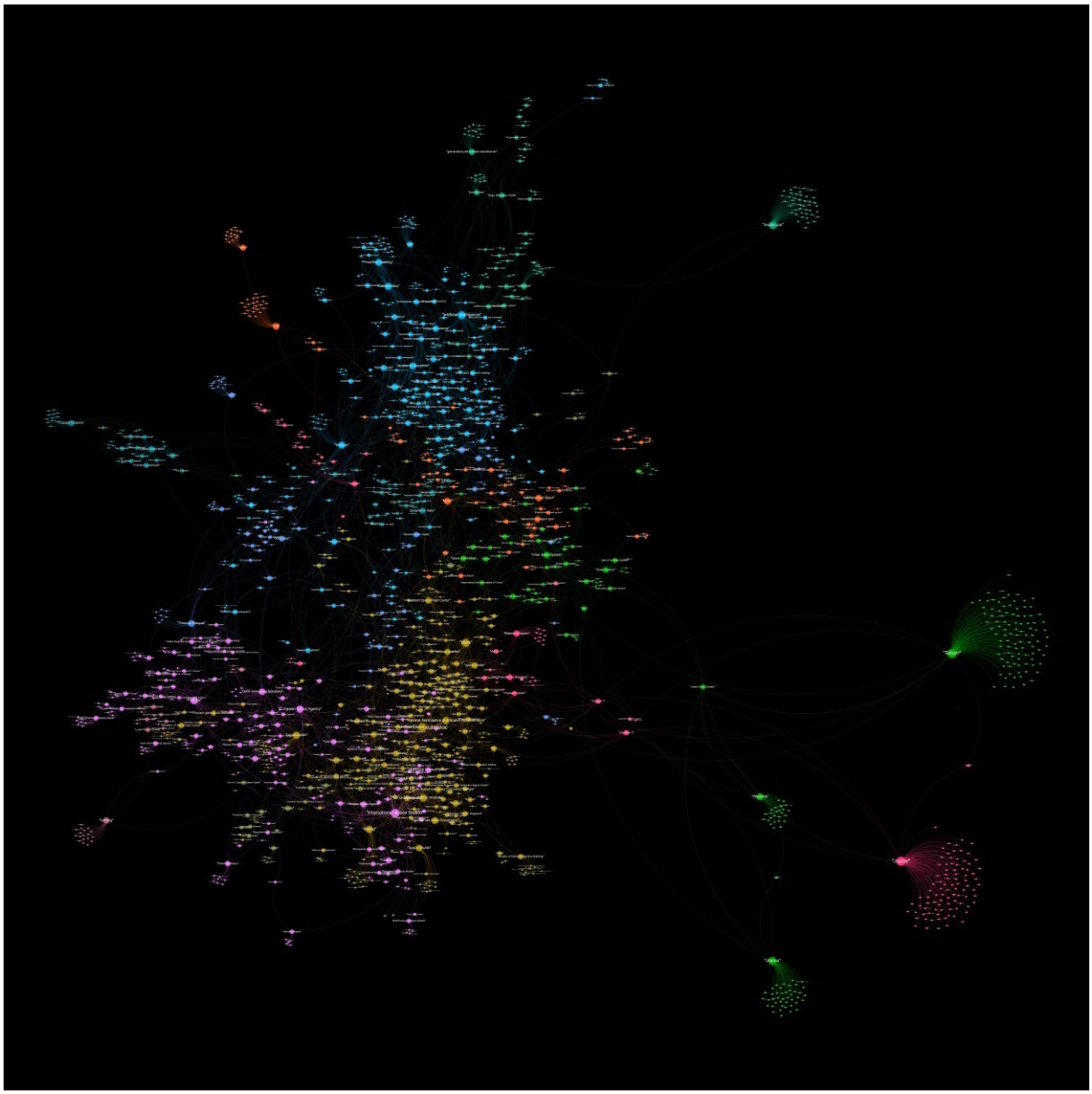

Fig. 2: Visualization of the full integrated graph structure underlying AMAIA's knowledge

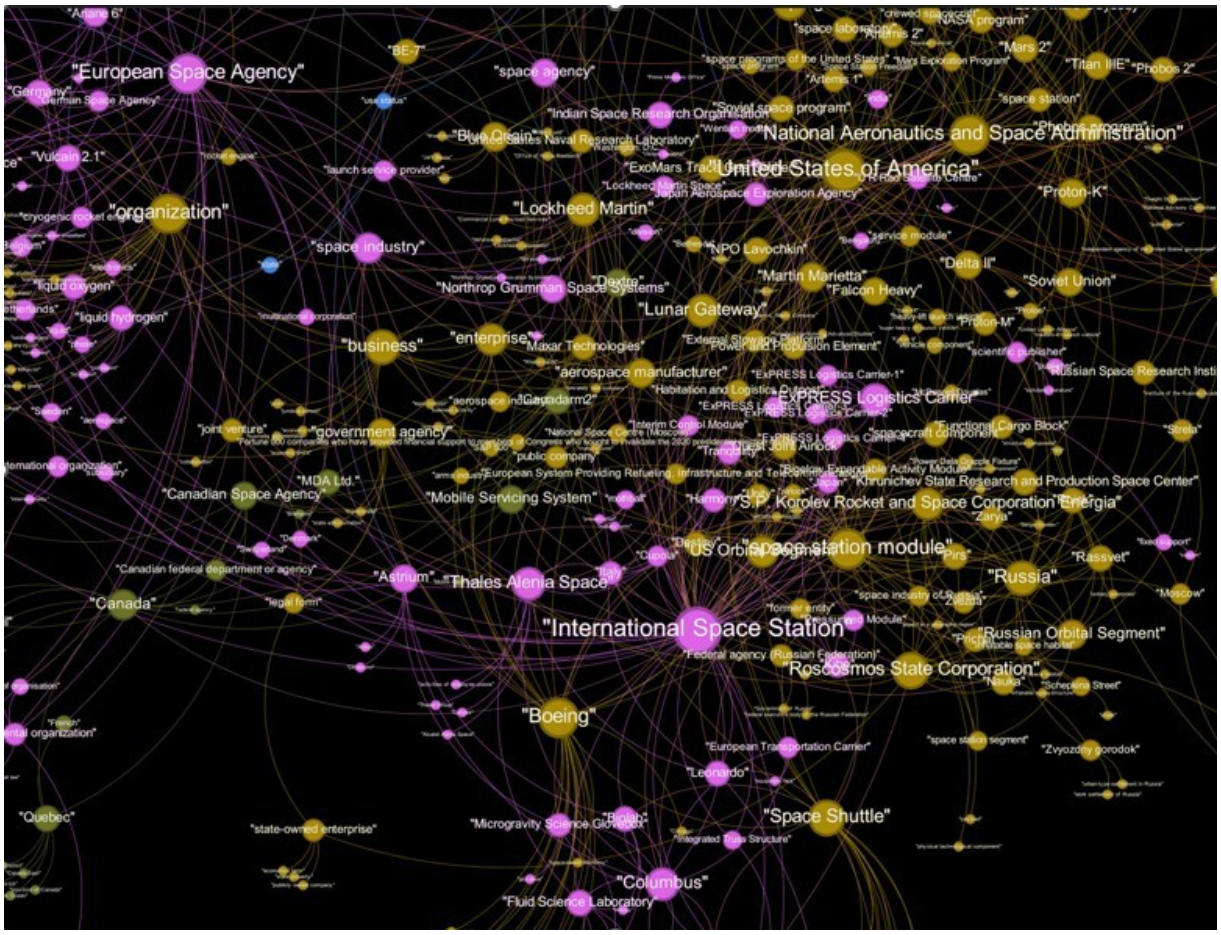

Fig. 3: Zoom in around the international space station

3. **Long-Term Conversational Memory Graph:** To provide AMAIA with persistent memory and enhance long-term contextual understanding, this base structures the history of past interactions as a graph. Instead of a simple chronological list, key elements of conversations (dialogue turns, discussed topics, decisions made, expressed user preferences) are represented as interconnected nodes. The RAG process is applied *directly to this memory graph itself*:
   - 1. The user's query (or the current dialogue turn) is embedded into a vector representation.
   - 2. A similarity search is performed against the nodes of the memory graph (representing past conversation segments or concepts) to identify the node(s) most semantically relevant (primary matches).
   - 3. Crucially, the tool then retrieves not only the primary matching node(s) but also retrieves the nodes directly linked to them within the graph structure (i.e., their neighbors). This step leverages the explicit relationships captured in the memory graph to provide essential relational context from past dialogue.
   - 4. The textual content or structured data associated with both the primary and the linked (neighboring) nodes is compiled into a cohesive block of context.
   - 5. This memory-derived context is then passed to the LLM agent.

   This graph-centric RAG approach applied to memory allows AMAIA to handle follow-up questions effectively, recall past discussions (e.g., User: "*What were we discussing about the CO2 scrubbers yesterday?*" → RAG retrieves relevant dialogue nodes from the previous day), and maintain conversational coherence over longer periods by leveraging the relational structure of the memory.

These three sources, while distinct in their content and potentially their structure, are queried by the RAG mechanism to collectively provide the LLM with rich, multi-faceted context, thereby improving the relevance, factuality, and coherence of its responses.

3.3 Speech Pipeline : Enabling Offline Voice Interaction

A key requirement for AMAIA is fully offline operation, extending to its voice interaction capabilities. Relying on cloud-based services for Automatic Speech Recognition (ASR) or Text-to-Speech (TTS) is not viable due to latency and connectivity constraints in deep space. Therefore, AMAIA incorporates a dedicated, locally hosted speech pipeline.

Deployment: All speech processing modules are packaged as Docker containers, facilitating deployment and management. They currently run on a dedicated machine equipped with an NVIDIA RTX-4090 GPU located within the simulation environment ("on-site"). The GPU provides the necessary computational power for running state-of-the-art speech models with acceptable latency.

Automatic Speech Recognition (ASR): We utilize OpenAI's Whisper models [8] (e.g., 'medium' or 'large' variant, depending on resource constraints vs. accuracy needs) for ASR. Whisper is chosen for its high accuracy across various accents and languages, robustness to background noise, and its open availability, allowing for local deployment. The ASR module receives the raw audio stream from the user's microphone, transcribes it into text, and passes the transcription to the core AMAIA NLU engine (Tock or the LLM agent).

Text-to-Speech (TTS): For generating spoken responses, we employ Coqui TTS (https://coqui. ai/), an open-source TTS framework. Coqui offers several advantages:

- **High Quality:** It provides access to state-of-the-art TTS models capable of generating natural-sounding speech.
- **Fine-tuning:** Crucially, Coqui's framework allows for fine-tuning TTS models on custom voice data. We have leveraged this capability significantly. By training a custom French TTS model on pre-recorded voice samples specific to the mission context (potentially using recordings from astronauts or operational personnel), we achieve a voice that is not only natural but also consistent and contextually appropriate.
- **Offline Operation:** Models can be fully downloaded and run locally.

This fine-tuning process proved particularly valuable. During the MDRS-311 campaign, initial tests with generic pre-trained TTS models sometimes exhibited minor artefacts like pitch variations or "hallucinations" in prosody. Our custom French Coqui TTS model, trained on dedicated recordings, has effectively eliminated these issues, resulting in a much more reliable and pleasant auditory output for the crew.

The complete voice interaction loop (User speaks → Mic captures audio → ASR transcribes → AMAIA processes text → AMAIA generates text response → TTS synthesizes audio → Speaker plays audio) operates entirely locally, ensuring responsiveness and independence from external networks. Optimizing this loop for minimal latency remains an ongoing effort.

4 System Architecture

4.1 Habitat Simulation Framework

Effective validation of the AMAIA agent requires testing within an environment that closely mimics the complexities of a real operational setting, such as a future lunar or Martian habitat. To this end, a dedicated simulation framework has been developed and is currently employed.

This framework generates realistic, dynamic data streams corresponding to a wide array of sensors and subsystems typically found in such habitats (e.g., environmental controls, power systems, life support parameters). These data streams are processed via standard protocols like MQTT and integrated into a central Home Assistant instance [7]. Home Assistant serves as the interface layer, presenting the simulated hardware state just as real devices would. AMAIA interacts with this simulated reality exclusively through its Home Assistant MCP tool, enabling it to query sensor values and command device actions using natural language.

A key component of the framework is its visual representation layer. Different visualisations of the simulated habitat can be configured – ranging from schematic dashboards to more detailed layouts representing the base

structure (examples shown in Figures 4 and 5). Regardless of the specific visual style, these interfaces display the live status of the simulated systems. Crucially, they also provide **immediate visual feedback** when AMAIA executes a command; for instance, a light icon changes state on the layout when AMAIA successfully processes an "turn on light" command.

The primary purpose of this simulation environment, with its large number of simulated sensors and actuators distributed across a representative habitat layout, is to rigorously evaluate AMAIA's performance. It allows for testing its scalability in handling a complex state space, its robustness in understanding commands within context, and its overall effectiveness in providing assistance within an environment designed to approximate operational realism. This framework is therefore essential for identifying potential issues and refining AMAIA's capabilities before deployment in physical testbeds or analogue missions.

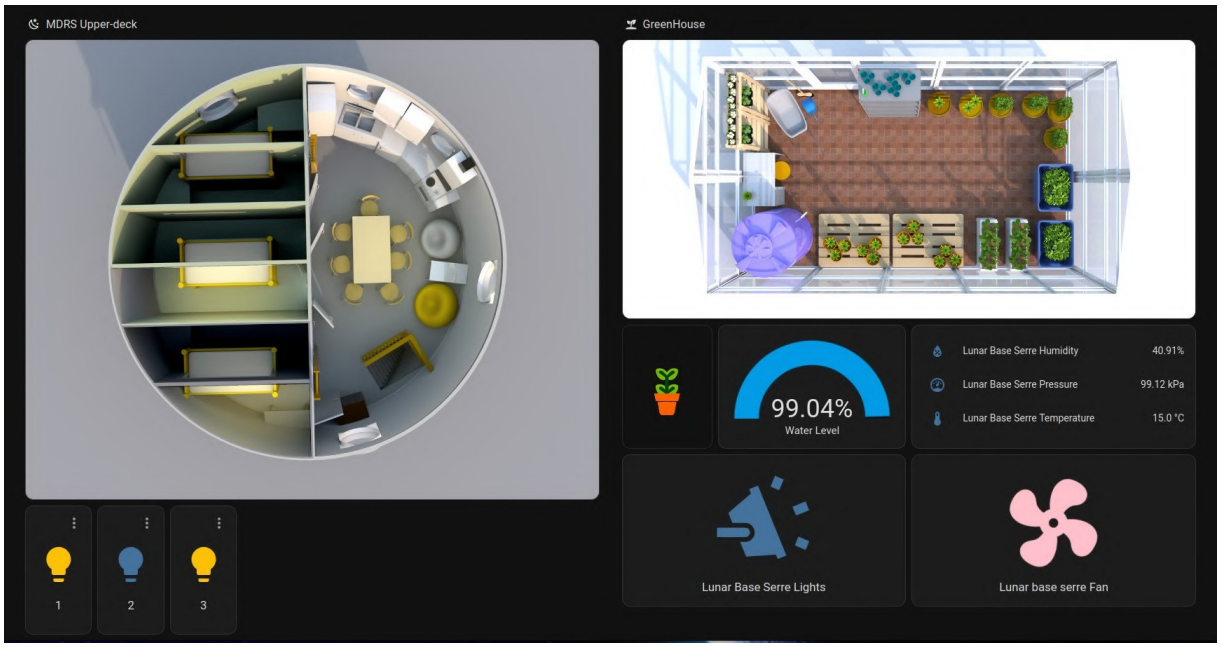

Fig. 4: Example visual interface (dashboard style) connected to the habitat simulation framework, displaying sensor data and reflecting commanded actions.

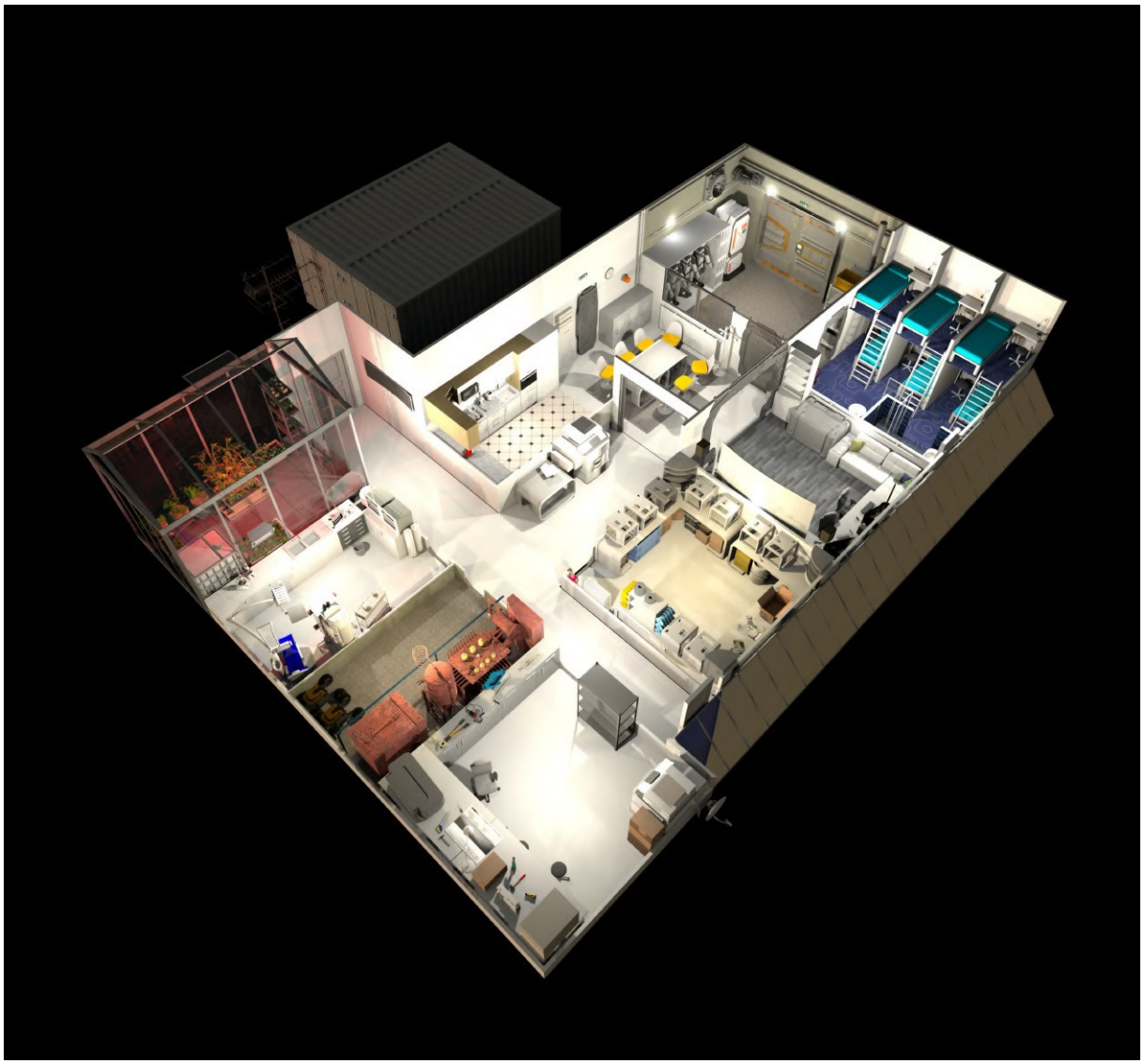

Fig. 5: Example of a more detailed visual layout representing the simulated habitat, also providing visual feedback for AMAIA's interactions.

4.2 Rover Voice Control

Validating interaction with mobile robotic assets is another key objective. AMAIA's integration with the *Scout* rover is tested, with control over robotic arms and rover itself.

Interface Mechanism: As described in Section 3.2, communication relies on publishing commands to specific MQTT topics (e.g., `/scout/cmd/move`, `/scout/cmd/turn`, `/scout/cmd/arm`) which the rover's control system subscribes to. The rover, in turn, publishes its status (position, battery level, task completion) to other topics (e.g., `/scout/status/pose`, `/scout/status/battery`) which can be monitored by AMAIA via Home Assistant or a dedicated rover status tool.

Voice Command Examples: AMAIA's NLP pipeline translates natural voice commands into these MQTT messages. Supported commands include:

- *Locomotion:*
  - "*Scout, move forward two meters.*" →
    MQTT: `"{""distance"": 2.0}"` to `/scout/cmd/move/forward`
  - "*Scout, avancer de 3 mètres.*" →

(Same as above after NLU)

- "*Scout, turn left 45 degrees.*" →
  MQTT: `"{""angle"": 45.0}"` to `/scout/cmd/turn/left`
- "*Scout, pivoter à droite de 90 degrés.*" →
  MQTT: `"{""angle"": 90.0}"` to `/scout/cmd/turn/right`
- "*Scout, stop.*" →
  MQTT: `"{}"` to `/scout/cmd/stop`

Performance: End-to-end latency tests, measuring the time from the end of the user's utterance to the rover beginning execution (or simulator acknowledging the command), are consistently performed. For typical motion commands, this latency – encompassing ASR, NLU (Tock or LLM path), MCP tool invocation, MQTT transmission, and rover command processing – remains under 5 seconds in the current prototype setup. This is generally considered acceptable for interactive control, though further optimization is ongoing.

4.3 MDRS-311 Operational Feedback

AMAIA underwent operational testing during the one-month MDRS-311 analogue mission. This mission environment simulated key aspects of deep-space operations, including a communication protocol that restricted the crew's contact with external Mission Support to a constrained daily window (e.g., one hour), thereby enforcing crew autonomy.

Critically, AMAIA's architecture, designed for fully offline "on-premise" operation, ensured its continuous availability to the crew throughout the mission, independent of these communication constraints.

The six crewmembers interacted with AMAIA via its text-based chat interface to support various operational scenarios during their simulation. Post-mission feedback was positive, with the crew particularly praising the chat interface for its clarity, responsiveness, and ease of use.

This real-world deployment was also invaluable for identifying practical issues, leading to the resolution of some residual challenges with Automatic Speech Recognition (ASR) under operational conditions and other minor bugs, ultimately contributing to improved system stability in subsequent software versions.

5 Performance Evaluation

Evaluating the performance of a conversational agent like AMAIA involves multiple dimensions, including task success rate, response accuracy, robustness, and user satisfaction. However, a critical metric, especially for interactive use, is latency – the time it takes for the system to respond to a user request. We conducted targeted tests to measure end-to-end latency for different types of commands, reflecting the performance of the hybrid architecture.

The tests were performed on the prototype hardware configuration: AMAIA software running in Docker containers on a server equipped with an Intel Core i9 CPU, 64GB RAM, and an NVIDIA RTX-4090 GPU for LLM inference and speech processing. Latency was measured from the moment the user's input (end of speech utterance for voice, or text submission for chat) was received by AMAIA to the moment the response began to be output (start of TTS audio playback, or first token appearing in chat). French ASR and TTS models were used where applicable.

Table 1 summarizes the approximate latency ranges observed for representative command types, distinguishing between the fast-path intent classifier (Tock) and the more complex LLM agent path (LangChain/ReAct).

Command Type / Complexity	Classifier Path (Tock)	LLM Agent Path (ReAct)
Simple Intent + Tool Invocation (HA Query) (e.g., "*What's the temperature in the greenhouse?*", "*What time is it?*")	≈ 0.2 s	≈ 5 s (incl. LLM tool selection step)
Simple Intent + Tool Invocation (HA Action) (e.g., "*turn on the light.*","*Turn off the water pump.*")	≈ 0.2 s	≈ 5 s (incl. LLM tool selection step)
Complex Query / Reasoning (RAG Invocation) (e.g., "*What's the procedure for...?*")	-	≈ 6–8 s (incl. RAG retrieval + LLM synthesis)
Multi-Step Task Execution (e.g., "*Verify pressure and humidity in my room, and then...*")	-	≈ 8–12 s (multiple tool calls/ReAct cycles)

Table 1: Approximate end-to-end latencies on the AMAIA prototype (Hardware: Intel i9 / 64GB RAM / RTX-4090; Offline French ASR/TTS models).

Note: Latencies include ASR/TTS processing time when applicable (approx. 0.5-1.5s total). LLM path latency depends heavily on query complexity and the number of ReAct steps required.

Discussion of Results: The results clearly demonstrate the effectiveness of the hybrid architecture:

- **Classifier Path Speed:** Simple, frequent commands handled directly by Tock exhibit sub-second latency (typically < 200ms for NLU + action dispatch). This ensures a highly responsive feel for basic interactions like turning devices on/off or asking for simple status values.
- **LLM Agent Path Latency:** Queries requiring the LLM agent inevitably incur higher latency. This is primarily due to:
  - **LLM Inference Time:** Running inference on the local LLM (even an optimized one) takes a few seconds on the RTX-4090.
  - **Tool Execution:** If the agent needs to call tools (like RAG or Home Assistant), the execution time of these tools adds to the overall latency. RAG retrieval, in particular, can take 1-2 seconds depending on the database size and search complexity.
  - **ReAct Cycles:** Complex queries may require multiple reasoning-acting steps by the LLM, further increasing the total time.

  While latencies of 5-12 seconds are acceptable for complex information retrieval or multi-step tasks, they highlight the importance of handling common commands via the fast path. Reducing LLM path latency through model optimization, quantization, and efficient tool design remains a key area for future work, especially for deployment on more constrained edge hardware.
- **Qualitative Observations and Future Work on Accuracy:** While quantitative metrics for task success rate, intent recognition accuracy, and overall robustness across a wide range of inputs are subject to upcoming formal evaluation campaigns, qualitative observations during development testing provide initial positive indications. The system generally behaved as expected for tasks falling within the scope of the implemented tools and the current knowledge base. Specifically, the

RAG mechanism was observed to noticeably improve the factual grounding and relevance of responses for knowledge-intensive queries compared to relying solely on the base LLM's parametric memory.

Furthermore, the hybrid architecture's design inherently addresses robustness in one aspect: common, well-defined commands handled by the deterministic classifier path are less susceptible to the variability sometimes seen with purely LLM-based approaches. However, formally quantifying the accuracy, robustness to paraphrasing or noise, and overall task completion rates across the entire system (including both paths) remains a key objective for future work (see Section 6).

Overall, the performance evaluation conducted so far, focusing primarily on latency (Table 1), demonstrates the viability of the hybrid architecture in balancing responsiveness for simple interactions with the potential for deeper reasoning. This provides a solid foundation, while acknowledging that rigorous assessment of accuracy and robustness is the necessary next step towards operational deployment.

6 Roadmap and Future Work

Following the successful validation milestones achieved, including the MDRS-311 deployment and the development of the v1 baseline architecture, the immediate focus is on consolidating features and hardening the system. Beyond that, several key development axes are planned to enhance AMAIA's capabilities and prepare it for more integrated demonstrations.

Immediate Priorities:

- **Consolidating Shared Crew Memory**: Currently, interactions are largely stateless or rely on short-term context within a single query. We are implementing a persistent "shared crew memory" vault. This will allow AMAIA to maintain context across multiple turns in a dialogue and, crucially, across interactions with different crew members working on related tasks. For example, if one crew member reports an anomaly, another crew member asking about system status later should receive information reflecting that report. This involves designing mechanisms for storing, retrieving, and managing relevant contextual information (e.g., recent events, ongoing tasks, crew locations) in a structured way accessible to the LLM agent.

- **Extending the MCP Tool Catalog**: Expanding the range of systems AMAIA can interact with is key to increasing its utility. Near-term efforts focus on integrating tools relevant to crew well-being and operational planning:
    - *Integrated Nutrition and Activity Planning Tool*: Development of an MCP tool designed to assist with crew meal planning by considering individual nutritional needs (potentially informed by health monitoring tools), available food inventory, and even correlating dietary plans with planned or logged physical activity levels. This could enable AMAIA to provide recommendations for optimizing nutrient intake or suggest adjustments to exercise routines based on dietary choices.
    - *Crew Schedule Management Tool*: Consolidating and deploying the previously conceptualized task scheduling capability into a robust MCP tool for managing the crew's shared operational schedule. This involves functionalities for adding, querying, modifying, and receiving reminders for events, tasks derived from procedures, personal appointments, and potentially resource allocation linked to the schedule, thereby enhancing overall mission time management and coordination.
    - *Procedure Execution Assistance Tool*: Introducing a capability specifically designed to guide astronauts through complex operational or maintenance procedures step-by-step. This tool would leverage AMAIA's access to procedural documents (via RAG) and potentially system state information (via other MCP tools like Home Assistant) to: provide clear instructions for each step, display relevant diagrams or warnings, track the astronaut's progress through the procedure, answer contextual questions about specific steps, and potentially assist in verifying step completion or logging outcomes. The goal is to reduce cognitive load, improve adherence, and enhance safety during critical task execution.

  - *Automatic Report Generation Tool*: Implementing a tool that automates the compilation and drafting of routine reports required during missions. This tool would leverage AMAIA's access to the shared crew memory (for significant events, anomalies), data from other MCP tools (e.g., system status updates, procedure outcomes, resource usage data),relevant media (like photos taken during experiments) and discussion transcripts with AMAIA during tasks. It could generate summaries like:
    - Daily/Weekly Activity Logs: Consolidating completed tasks, significant events, and system status changes.
    - Anomaly Reports: Automatically drafting initial reports when anomalies are detected or logged, pulling in relevant system data and timestamps.
    - Experiment Status Summaries: Compiling progress based on logged procedure steps or data inputs related to specific experiments.
    - Resource Consumption Reports: Tracking usage of consumables or power based on available data streams.

  The goal is to significantly reduce the manual effort required for documentation, ensure consistency in reporting, and allow crew members to quickly review and finalize system-generated drafts.

- **Hardening Offline RAG Pipelines**: Enhancing the robustness and scope of the offline Retrieval Augmented Generation capability is critical for trustworthy information access. This involves:
  - *Enriching AMAIA's Knowledge Graph with CNES Expertise*: Investigating how to distill and integrate the extensive domain know-how and specialized knowledge from CNES in the space sector (regarding systems, operations, procedures, etc.) directly into AMAIA's own knowledge graph. The goal is to structure this internal expertise to enhance AMAIA's relevance, accuracy, and contextual reasoning capabilities, going beyond simple information retrieval from documents.
  - *Improving Indexing and Retrieval Strategies*: Experimenting with more advanced indexing techniques (e.g., hybrid search combining keyword and semantic search) and retrieval strategies (e.g., re-ranking retrieved documents for relevance) to improve the quality of context provided to the LLM.
  - *Expanding the Knowledge Corpus*: Ingesting a wider range of relevant documentation (e.g., detailed schematics, more extensive mission logs, scientific background papers).

- **Preparing for Edge Compute Deployment**: The current prototype relies on relatively high-power GPU hardware (RTX-4090). A critical next step is migrating the entire AMAIA system, including the LLM and speech models, to space-representative edge computing hardware, such as NVIDIA Jetson Xavier or similar platforms. This involves significant optimization efforts:
  - *Model Quantization*: Reducing the precision of model weights (e.g., to INT8 or INT4) to decrease memory footprint and computational requirements, while minimizing accuracy loss.
  - *Model Pruning/Distillation*: Exploring techniques to run smaller, faster models specifically tailored to the required tasks.
  - *Optimized Inference Engines*: Utilizing inference engines like TensorRT optimized for the target hardware.

The roadmap positions AMAIA as an evolving capability, progressively increasing its autonomy, intelligence, and integration within the broader ecosystem of tools supporting future human space exploration.
Crucially, the project stands to benefit synergistically from the ever-improving landscape of hardware technology, where gains in computational power, data acquisition capabilities, and connectivity create fertile ground for increasingly sophisticated and capable AI assistants like AMAIA.

7 Conclusion

The development of AMAIA presented in this paper demonstrates significant progress towards an intelligent, autonomous digital assistant capable of supporting astronaut crews during future long-duration, deep-space missions characterized by communication latency. We have shown that a hybrid conversational architecture, combining a fast, deterministic intent classifier (Tock) with a flexible, reasoning capable LLM agent (LangChain/ReAct), can effectively reconcile the requirements for both sub-second responsiveness on routine tasks and deep reasoning for complex queries and procedures.

By adhering to open standards like MCP and prioritizing fully offline operation, AMAIA establishes a pathway towards trustworthy and reliable autonomous digital assistance.

The ongoing work on shared crew memory and edge compute deployment aims to further enhance AMAIA's capabilities, paving the way for its use as an indispensable tool for crews operating autonomously on the Moon, Mars, and beyond. AMAIA represents a key contribution of the Spaceship France programme to enabling the next generation of human space exploration

References

[1] Navarro, G., "Spaceship FR Project: How to Interface with the Different Actors to Contribute to Space Exploration and Human Spaceflight?" *53rd International Conference on Environmental Systems (ICES)*, Louisville, KY, USA, July 2024, ICES-2024-332.

[2] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y., "ReAct: Synergizing Reasoning and Acting in Language Models," *International Conference on Learning Representations (ICLR)*, 2023. Also available as *arXiv*:2210.03629 [cs.CL], 2022.

[3] Chase, H., "LangChain: Framework for Developing Applications Powered by Language Models," GitHub Repository, https://github.com/langchain-ai/langchain, 2022–Present. [Online; accessed May 15, 2024].

[4] Weng, L., "LLM-powered Autonomous Agents," Blog Post, *Lil'Log*, June 2023. https://lilianweng.github.io/posts/2023-06-23-agent/ [Online; accessed May 15, 2024].

[5] Huzaifa, M., Sriram, K., Sharma, S., Sarkar, A., Wang, B., and Dey, D., "KnowAug: Knowledge Augmentation for LLM-based Embodied Agents using Scene Knowledge Graphs," *arXiv*:2410.16804 [cs.RO], 2024.

[6] The Open Conversation Kit, "Tock: The Open Conversation Kit," GitHub Repository, https://github.com/theopenconversationkit/tock, 2016–Present. [Online; accessed May 15, 2024].

[7] Home Assistant Contributors, "Home Assistant: Open source home automation that puts local control and privacy first," Software Project, https://www.home-assistant.io/, 2013–Present. [Online; accessed May 15, 2024].

[8] Radford, A. et al., "Robust Speech Recognition via Large-Scale Weak Supervision," *arXiv*:2212.04356 [eess.AS], 2023.

[9] Zhao, W., Zou, Y., Feng, Z., Wang, Y., Zhao, Z., and Yan, R., "GFM-RAG: A Graph

Foundation Model for Retrieval-Augmented Generation on Graphs," *arXiv*:2405.12988 [cs.AI], 2024.