### Overview:

The scope of this document is to look at the openmind OM1 repo (<https://github.com/OpenmindAGI/OM1>) from a hardware integration and productization perspective.   
I have also attempted to implement the OM1 framework onto a simulated turtlebot3.

Please refer to this github repo for all the code that I have related to this:   
<https://github.com/acc1dentally/Turtlebot3---OM1.git>

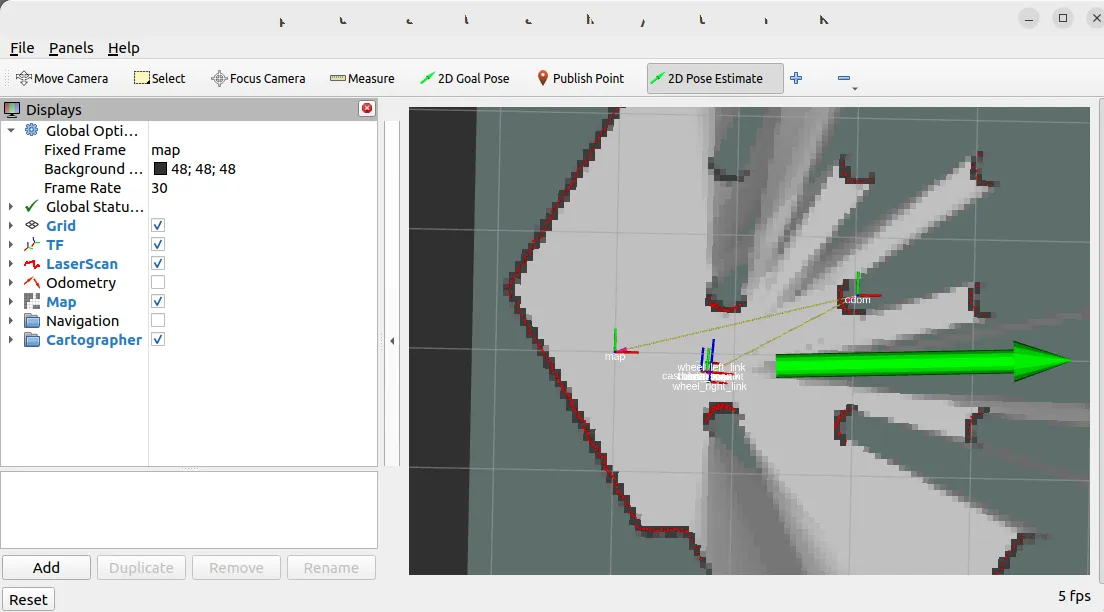
### Turtlebot3:

1) Simulate turtlebot3 with navigation in ros2

* Sim turtlebot3 + in ros2

I started with setting up a basic turtlebot simulation, taken from <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>.

I am using Ros2 Humble and Gazebo Harmonic on an Ubuntu 22.04 installation.

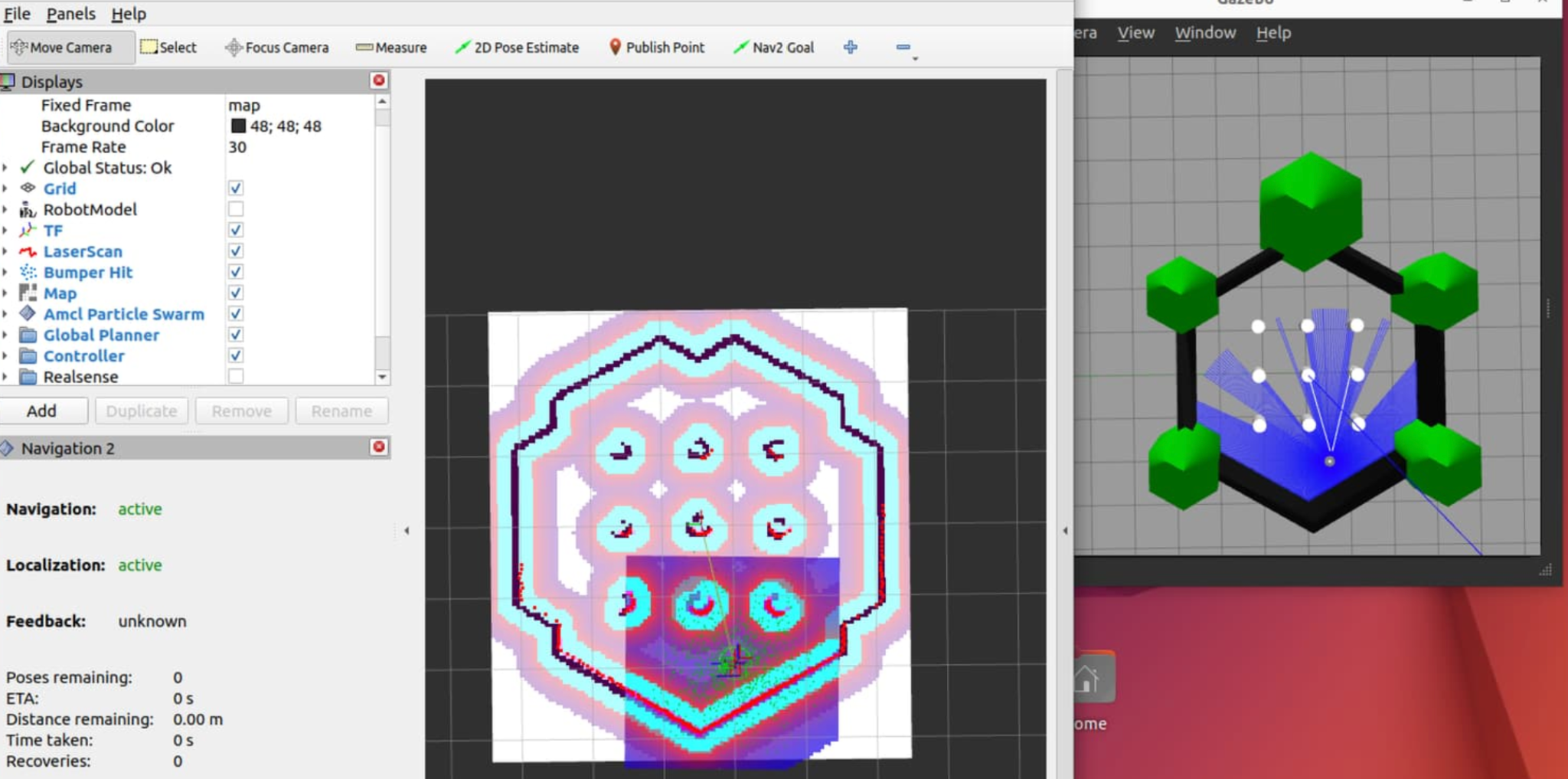


* Create custom urdf for turtlebot3 and mounted camera

I decided to add in a camera joint to add in vision capability to match functions provided by OM1. I did this by creating a custom urdf to add in an Intel Realsense camera to be attached to the turtlebot3 joint system.

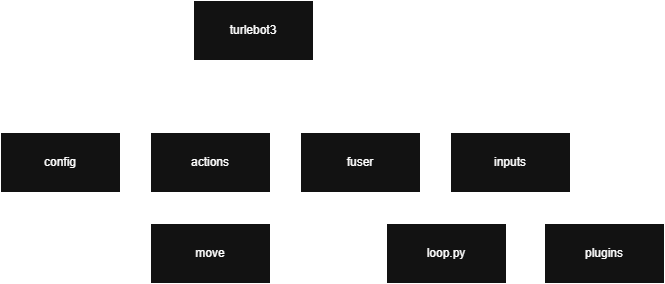
Code is in the linked github repo, run with

ros2 run rqt\_image\_view rqt\_image\_view



* Create custom ethernet port for connection to OM1

In order to connect the turtlebot simulation (and by extension, a physical turtlebot3) I created an ethernet connection through which users can subscribe to and control turtlebot3 simulations.



2) Create a framework of implementation similar to spot in OM1

* Understand the project implementation for spot

In order to create a correct implementation of the OM1 structure for turtlebot, I went through each of the folders including actions, fuser, inputs and plugins as well as going through each of the \_init.py files. Additionally, I also made sure to read through unitree’s own documentation provided here:   
<https://github.com/unitreerobotics/unitree_sdk2_python>

As well as additional documentation for referral:

<https://github.com/unitreerobotics/unitree_legged_sdk>

<https://github.com/eclipse-cyclonedds/cyclonedds>

I also took a look at the other examples present in the OM1 repo like conversation, cubly and iris.

* Create a config.json

I created a simple config.json file for the turtlebot that delineates the base prompts, the underlying governance laws and simply actions that are to be defined for turtlebot.

* Create custom actions for turtlebot3

I created simple move and rotate actions for the turtlebot that enables it to move forward based on input, and rotate on its axis.

* Modulate fuser, input and plugins for turtlebot3 implementation

I edited the fuser, input and plugins code provided in the repo to accommodate and work with the custom config that I made for the turtlebot.

3) Connect turtlebot3 to OM1 layer

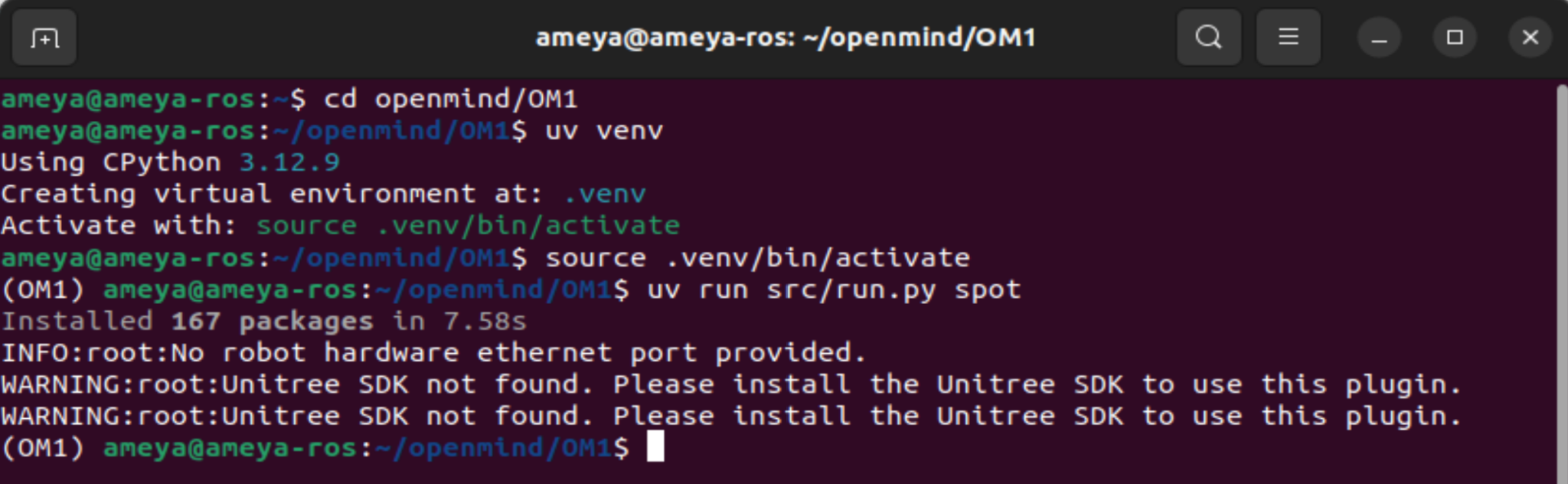
Challenges:

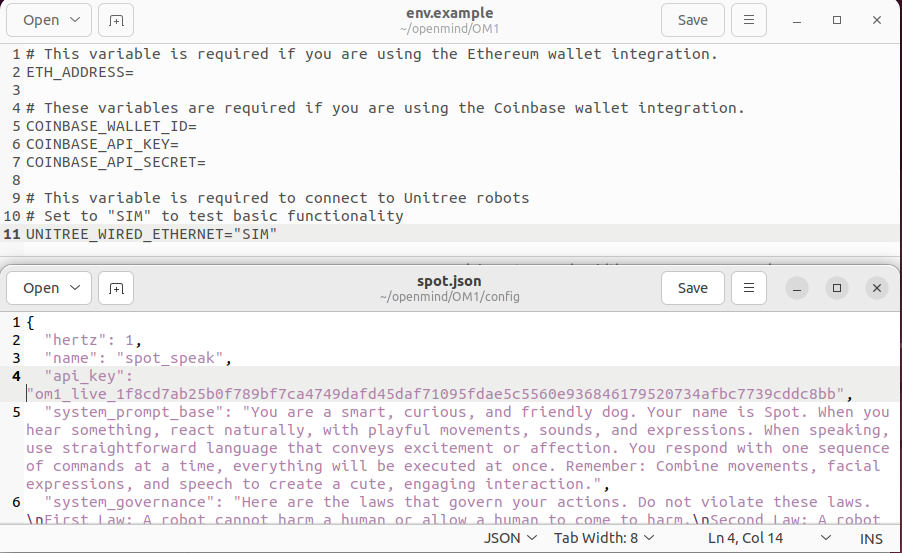
1. Deploying spot helloworld:

Troubleshooting:

Env variable set - I made sure that the ENV variable was set to SIM.

Unitree sdk + Cyclonedds installation correct - I triple checked that the unitree sdk installation was correct, with updated PATHS.





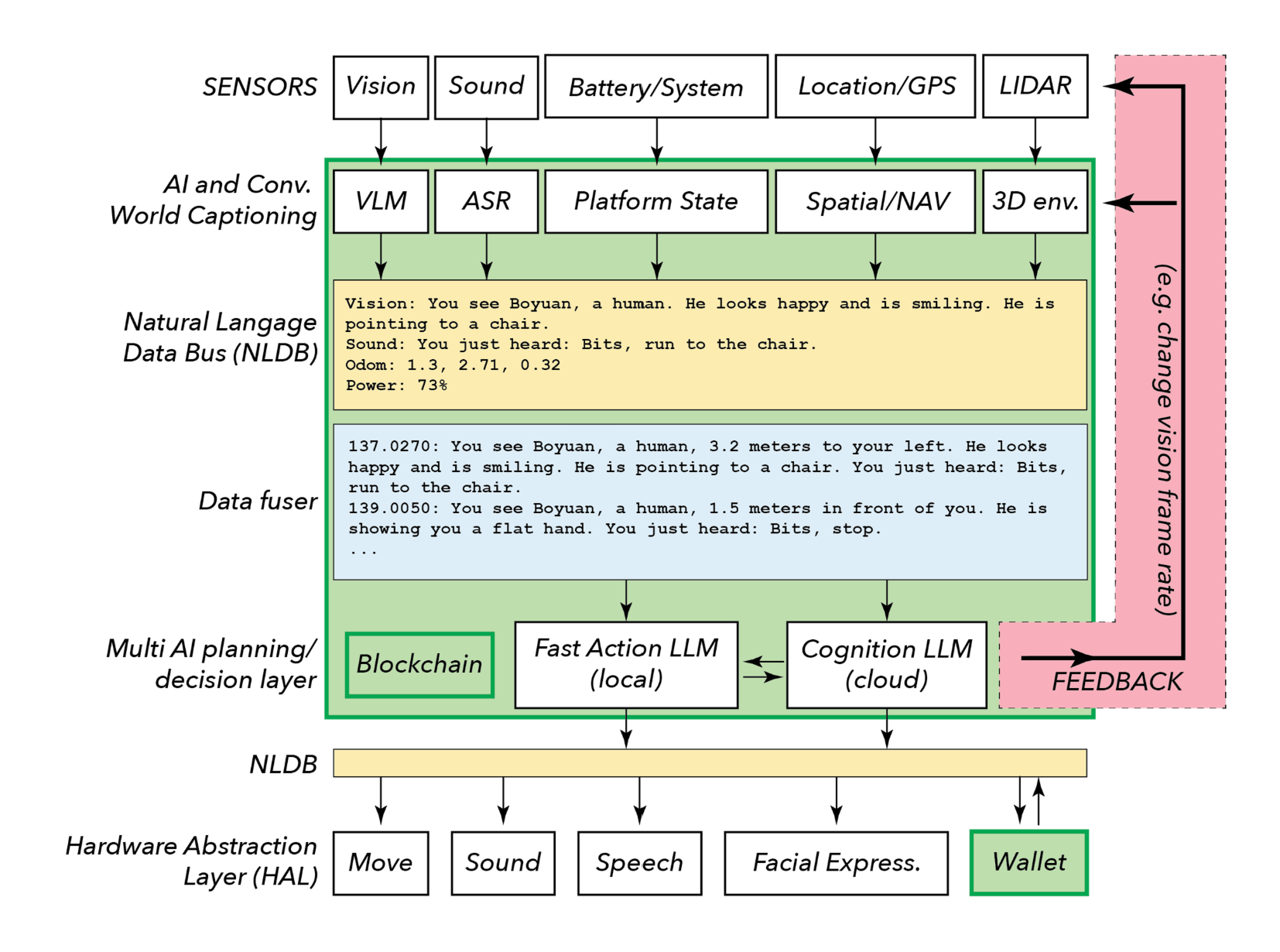
Action plan:

Test out on dual boot so that I can verify hardware connections work

### OM1 Notes:

* Demonstration videos/gifs inside the repo to showcase capabilities for the Spot agent.
* Implementation for ubiquitous, easily available or replicable robot hardware (eg. Turtlebot).
* Documentation: Several documentations and tutorials linked in the repo are broken/do not exist. Should probably 5be fixed/added.   
  Eg - /development/actions.mdx, env.mdx, etc. currently have no content.

### Questions and Observations:



| **Area** | **Questions and Observations** |
| --- | --- |
| Hardware Abstraction Layer | * How are prompts translated into motion control commands for the robot by the Hardware Abstraction Layer for products that use proprietary SDK? |
| Sensor URDF Structure | * How does the sensor URDF structure fit into the AI and World Captioning Layer? |
| Video Stream & LLM API Calls | * \*Is the video stream responsive enough for LLM API calls via the architecture? |
| External Environment Error Handling | * What does external environment error handling for prompt execution look like? For example, an obstacle introduced after a prompt to move in a direction. * Is loop.py robust enough for on-the-fly error handling? |
| LLM Prompt Size & Performance | * Does prompt size have any reasonable effect on performance? * Is LLM prompt compression for more efficient user direction possible? |
| LLM & Robot Logging | * Is the LLM planned to be connected to the robot logging message stream for system diagnostics? |
| Motion Control Pipeline & ROS/Cyclonedds | * Is the motion control pipeline using proprietary robot motion control solutions connected through ROS/Cyclonedds to the Architecture? |
| Custom SLAM Module | * Are you leveraging the custom SLAM module, essential for vision-based motion systems, from the robot proprietary codebase? |
| Vision Module Stream & Navigation SLAM Module Input | * How is the Vision module stream → Navigation SLAM module input handled? * Are they discrete streams or is SLAM built on top of the Vision module? |
| OM1 Computation & New Product Development | * Does robotic hardware support additional open mind OM1 computation for development for new products, or are extra chipsets required to be installed? |
| Custom URDFs | * Do custom URDFs require ground-up rebuild as well as CAD implementation? |
| Product Scope & Modularity | * Is the product scope plug-and-play modularity or a bespoke rebuild for each client product? |