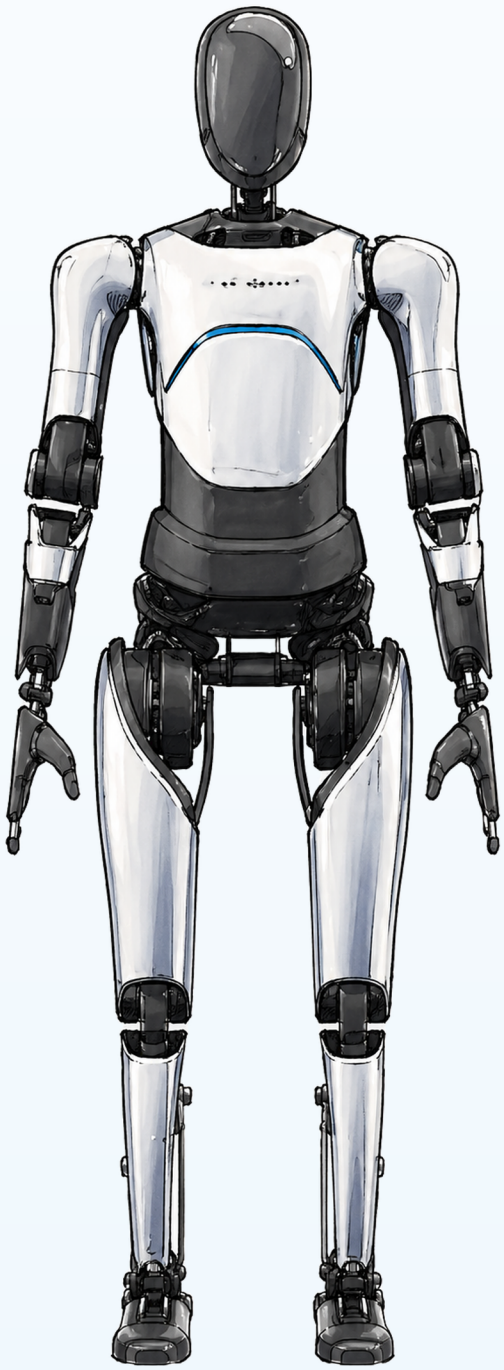
ILLEON
INTELLIGENCE IN MOTION

Agibot A2
User Manual



CONTACT

info@illeon.de
support@illeon.de
docs.illeon.de

Contents

1 Attention **3**

2 Disclaimer **4**

3 Powering the Robot On and Off **5**

3.1 Safety Prerequisites . . . 5
3.2 Power-On Procedure . . . 5
3.3 Power-Off Procedure . . . 6
3.4 Emergency Stop . . . 6
3.5 Quick Reference . . . 7

4 Quick Start **8**

4.1 SSH . . . 8
4.2 Webserver . . . 8
4.3 Webserver – Free A2 CPU Resources . . . 10
4.4 Webserver – Enable Internet . . . 10
4.5 Network Table . . . 11

5 Navigation **12**

5.1 Webserver Navi – SLAM . . . 12
5.2 Webserver Navi – Map Navigation . . . 13
5.3 Save a Map from the Command Line . . . 14
5.4 Agibot Service Diagnostics . . . 14

6 ROS 2 Package Reference **15**

6.1 A2-Specific Packages . . . 15
6.2 Generic Robot Packages . . . 18
6.3 Building the Workspace . . . 27

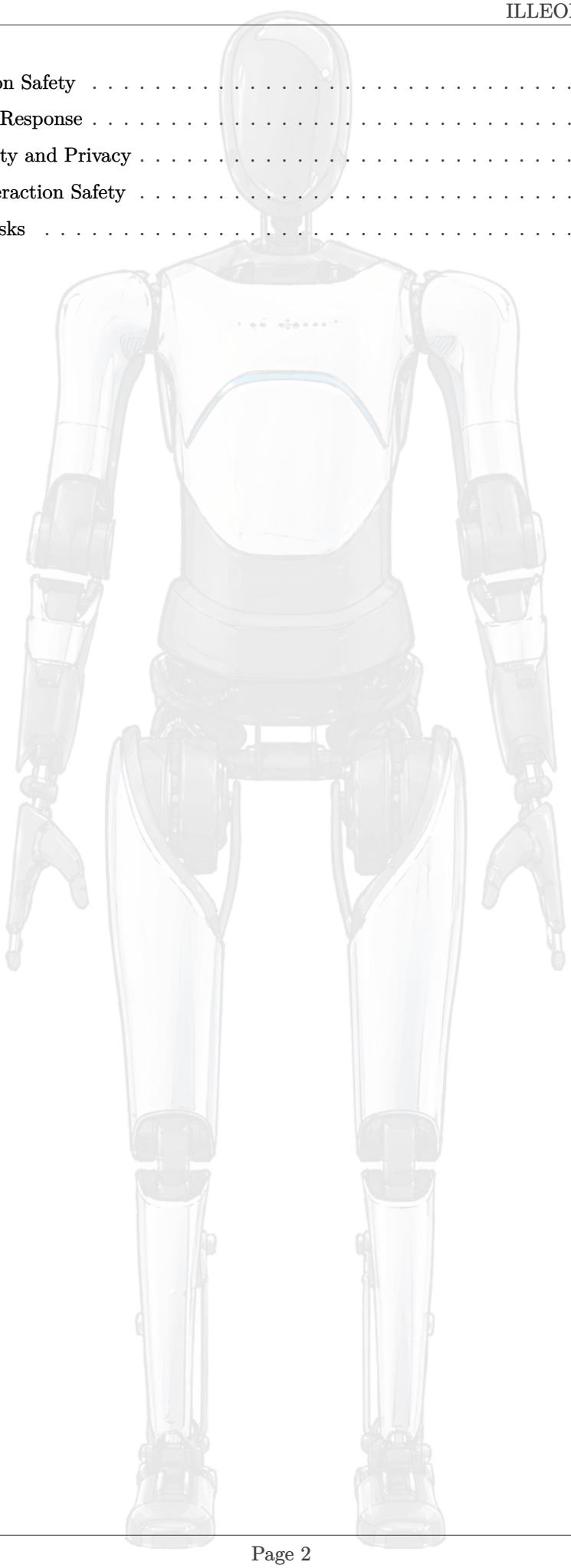
7 Fresh Installation **28**

7.1 Switch the A2 Identity . . . 28
7.2 Sync the Workspace . . . 28
7.3 Create the Workspace . . . 28
7.4 Configure WiFi . . . 28
7.5 Switch apt Sources to Official Mirrors . . . 28
7.6 Clone and Install . . . 29
7.7 Update `.bashrc` . . . 29
7.8 Build and Install ROS 2 Services . . . 29

8 Remote RViz (Laptop Viewing Robot Topics) **31**

8.1 Enable LAN Mode . . . 31
8.2 Launch RViz on the Laptop . . . 31
8.3 Revert the Whitelist Change on the Robot . . . 31

9.4 Manipulation Safety 34
9.5 Emergency Response 34
9.6 Data Security and Privacy 35
9.7 Human Interaction Safety 35
9.8 Residual Risks 35

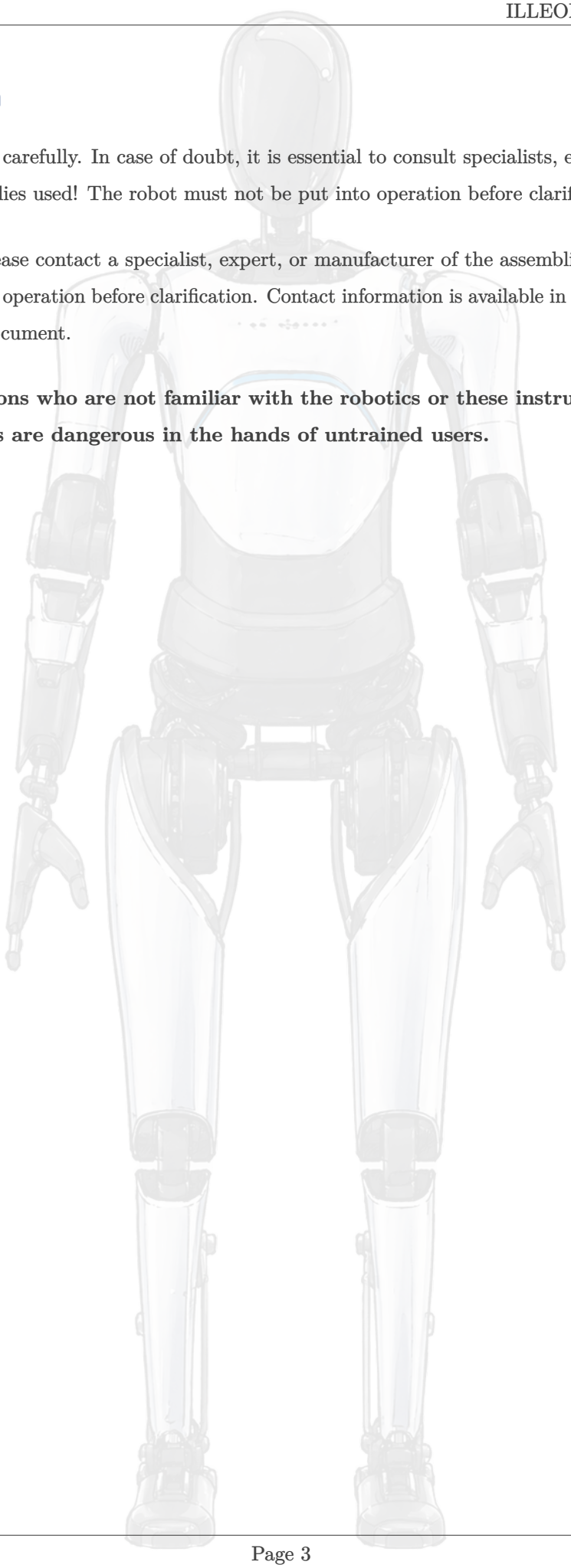


1 | Attention

Read this document carefully. In case of doubt, it is essential to consult specialists, experts, or manufacturers of the assemblies used! The robot must not be put into operation before clarification.

In case of doubt, please contact a specialist, expert, or manufacturer of the assemblies used! The robot must not be put into operation before clarification. Contact information is available in the Contact Section at the end of this document.

Do not allow persons who are not familiar with the robotics or these instructions to operate the robot. Robots are dangerous in the hands of untrained users.



2 | Disclaimer

The provided robot, sold by *ILLEON*, is an R&D device and does not have CE Marking and/or Certificate of Incorporation. Basic ROS understanding is required. If you are not familiar with ROS, we recommend checking the ROS2 Wiki first.

The client acknowledges and agrees that any information or materials provided by *ILLEON* are for R&D purposes only. Any services are provided ”AS IS” and without any representation or warranty of any kind, express or implied, including but not limited to any warranty of merchantability, fitness for a particular purpose, non-infringement, or any other warranty.

ILLEON shall not be liable for any damages, including but not limited to direct, indirect, special, incidental, or consequential damages, arising out of or in connection with the use or inability to use the information or materials provided. This limitation on liability shall apply regardless of the form of action, whether in contract, tort, or otherwise.

3 | Powering the Robot On and Off

Procedures for the Illeon A2 (hardware version P1, software version 1.2.24). The webserver itself does **not** gate physical power – follow the steps below before launching or after stopping the `webserver.launch.py` stack.

3.1 | Safety Prerequisites

Important: Failure to suspend the robot or to verify e-stop pairing can result in the robot collapsing onto its legs. The procedure below is mandatory before any leg-state change.

1. Suspend the robot from the manipulator boom or mobile lift before any leg-state change (*Legs Relaxed*, *Assist in standing with straight legs*).
2. Verify the **emergency-stop (e-stop) switch serial number matches the robot serial number** – pairing is one-to-one.
3. Power on the remote controller (*Aim-Master*) and the e-stop. Only one Aim-Master may be connected to a given robot at a time.
4. Confirm the battery is installed and charged. Do **not** continue operating at low battery – this can damage the system.

3.2 | Power-On Procedure

3.2.1 | Pre-flight Checks

- Robot, e-stop, and remote controller serial numbers all match.
- Battery is fully charged and seated.
- Robot is in protected mode (suspended from boom or lift).

3.2.2 | Boot Sequence

1. Open the debug chamber on the robot's back (press the protruding cover to release).
2. Press and hold the power button (item 4 in the manual's debug-chamber figure) until the green indicator turns on, then release.
3. Wait approximately **90 seconds** for startup. Expected indicators:

Stage	Expected Behavior
~0 s	Ambient body lights on; chest LEDs run a blue flow.
~10 s	Head self-test (up/down/left/right, returns forward).
~10–20 s	Hand self-test (open → outward → finger curl → fist).
~20 s	Upper-limb joints lock at the home position.
During wait	Audio prompt: *"Audio player ready"*.
~60–90 s	Steady-state idle screen on the chest display.

Verify the e-stop status indicator (item 5 on the e-stop) is **green**; the robot and e-stop have paired automatically.

> **Note:** If any stage fails, perform an immediate power-off (see below) and contact support.

3.3 | Power-Off Procedure

There are two valid shutdown paths. Use the **normal** procedure for end-of-shift shutdown; use the **emergency** path only when continued operation would cause injury or equipment damage.

3.3.1 | Normal Shutdown

The robot must be mechanically secured before joints relax.

1. Connect the manipulator boom to the robot's shoulder ring.
2. Stabilize the robot body.
3. Switch the robot state to ***Assist in standing with straight legs*** via the Toolbox (manual §4.2).
4. Lift the robot with the manipulator boom until its feet clear the base of the boom.
5. Switch to ***Legs Relaxed*** to disable the lower-limb joints.

   > **Warning:** *Legs Relaxed* is **only** safe while the robot is suspended; switching it on the ground will collapse the robot.

6. On the back of the robot, briefly press the shutdown button, then continue holding it for **at least 2 seconds**. Release once the operating noise (fans, joint hum) stops.
7. Confirm the chest LEDs and ambient lights are off before disconnecting the boom or removing the battery.

3.4 | Emergency Stop

CAUTION: The emergency stop disables every joint immediately. Boom support is critical because the robot will sag the moment the e-stop is engaged.

1. Press the **red mushroom button** on the e-stop. All joints disable immediately – boom support is critical because the robot will sag.
2. Once the hazard is resolved, **rotate the button clockwise** until it pops out to release the e-stop.
3. The robot performs an upper-limb self-test on release; wait for it to complete before commanding motion.

Important: E-stop release is **not** a substitute for a controlled shutdown – once the situation is safe, follow the normal shutdown procedure above.

3.5 | Quick Reference

Action	Button / Control	Hold Time
Power on	Debug-chamber power button	Until green LED
Normal shutdown	Debug-chamber power button (short press + hold)	$\geq 2\,s$
Emergency stop	E-stop mushroom button	Press once
Release e-stop	Rotate e-stop mushroom clockwise	Until it pops

4 | Quick Start

Important: This manual covers the Illeon A2 humanoid robot. Ensure you have completed the *Powering the Robot On and Off* procedure (Section 3) before continuing.

4.1 | SSH

Access the robot over LAN or WiFi:

```
ssh -XC agi@192.168.2.50    # LAN
ssh -XC agi@192.168.88.88   # WiFi (SSID password: 02270227)
# Password: 1
```

4.2 | Webserver

Ensure the robot is powered on and movable via the Agibot controller. Connect an Ethernet LAN cable to the left port and set your computer's IP address to `192.168.2.51`.

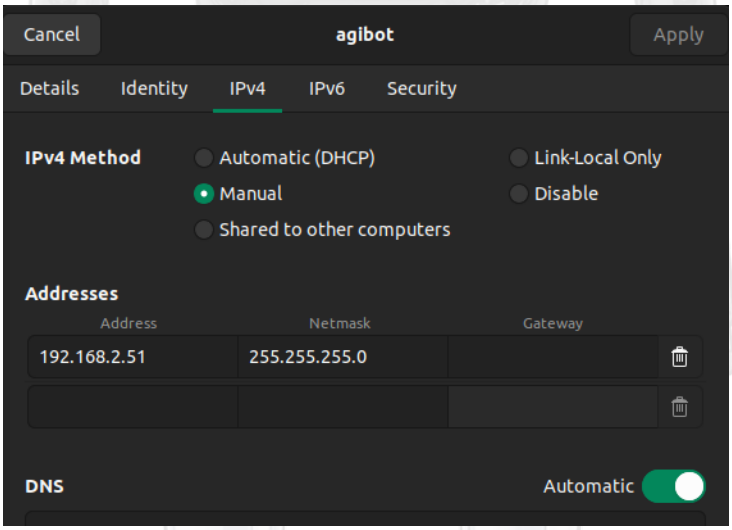


Figure 4.1: Network configuration on the operator laptop.

Navigate to the IP address in your browser:

```
192.168.2.50:9000
```

```
username: admin
password: illeon
```

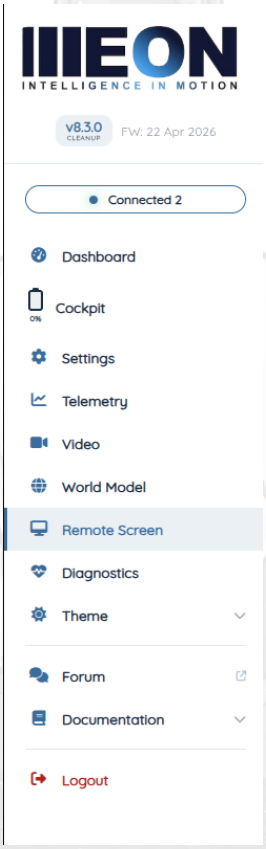


Figure 4.2: Webserver entry tab.

Important: If nothing shows up, the Illeon drivers are not installed. In that case follow the Fresh Installation procedure (Section 7).

You can verify all robot modules are running on the modules page.

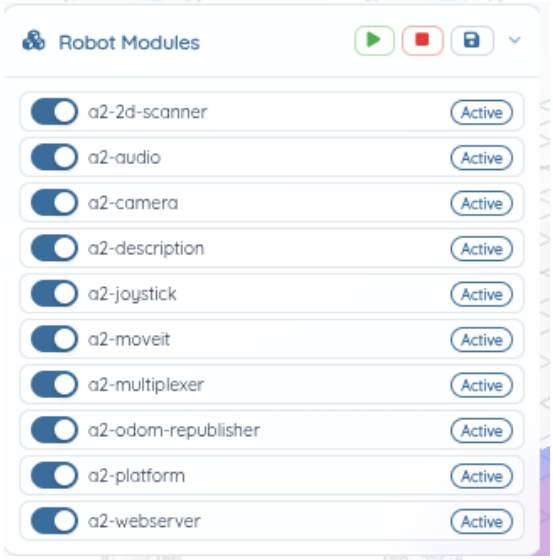


Figure 4.3: Robot modules overview.

Place the robot in default locomotion mode using the custom controls panel.

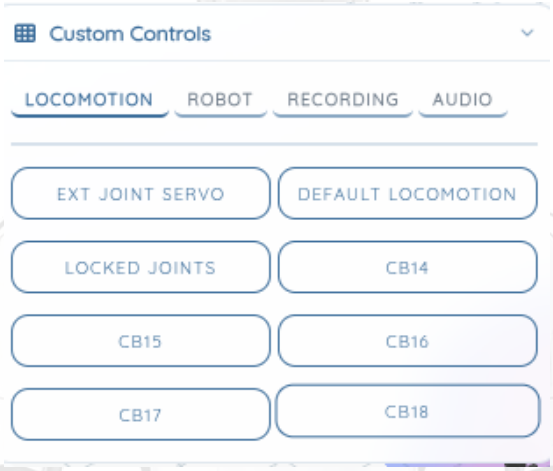


Figure 4.4: Custom controls – default locomotion.

4.3 | Webserver – Free A2 CPU Resources

Ensure you are connected via **LAN**. Open the Remote Screen tab.

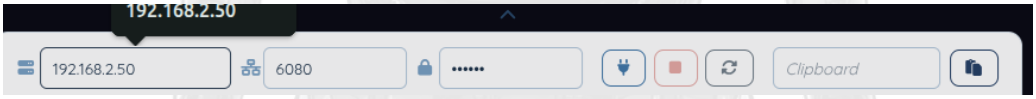


Figure 4.5: Remote screen (VNC) tab strip.

Double-click the **Agibot Agent Killer**. The password is **1**.

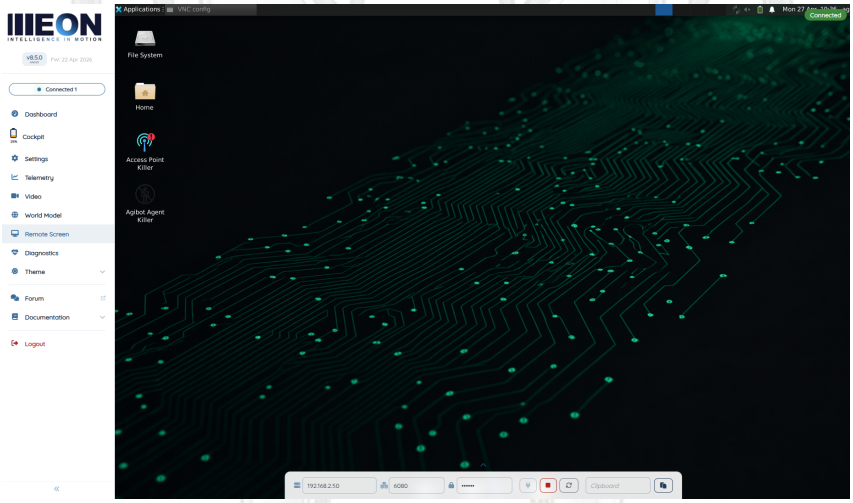


Figure 4.6: Agibot Agent Killer prompt.

4.4 | Webserver – Enable Internet

Ensure you are connected via **LAN**. Open the Remote Screen tab and double-click the Access Point Killer (password **1**). After a few seconds connect to WiFi from the top-right tray.

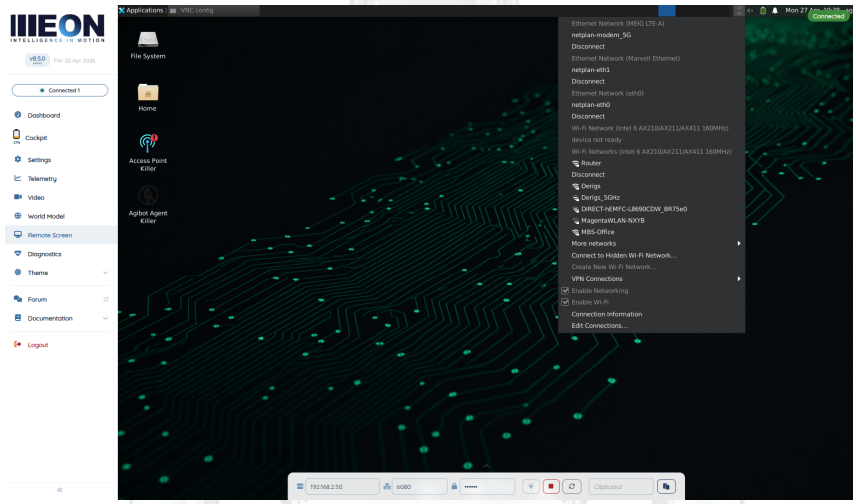


Figure 4.7: WiFi tray on the robot's desktop.

Right-click and open a terminal, then type `ifconfig` to view the new IP (e.g. **192.168.179.9**). Connect your local PC to the same network. You can then SSH and reach the webserver at the new IP, e.g. **192.168.179.9:9000**.

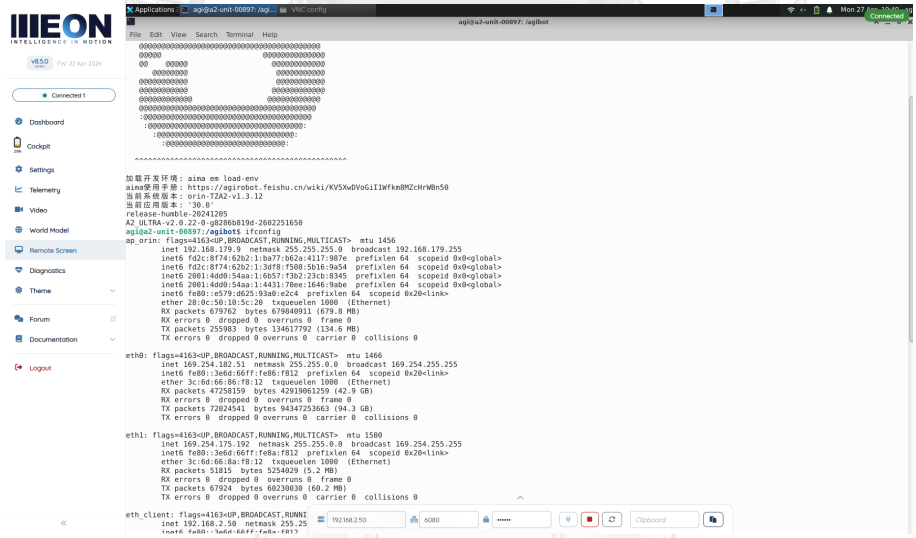


Figure 4.8: Resolving the WiFi IP via `ifconfig`.

4.5 | Network Table

IP / Endpoint	Device	Username	Password
192.168.2.50	A2 Onboard PC (LAN)	agi	1
192.168.88.88	A2 Onboard PC (WiFi)	agi	1
192.168.2.50:9000	Webserver	admin	illeon
192.168.2.51	Operator laptop (static)	-	-

5 | Navigation

The Illeon A2 supports SLAM-based mapping and map navigation directly from the cockpit web interface.

5.1 | Webserver Navi – SLAM

Open the cockpit tab and select the indoor sub-tab.

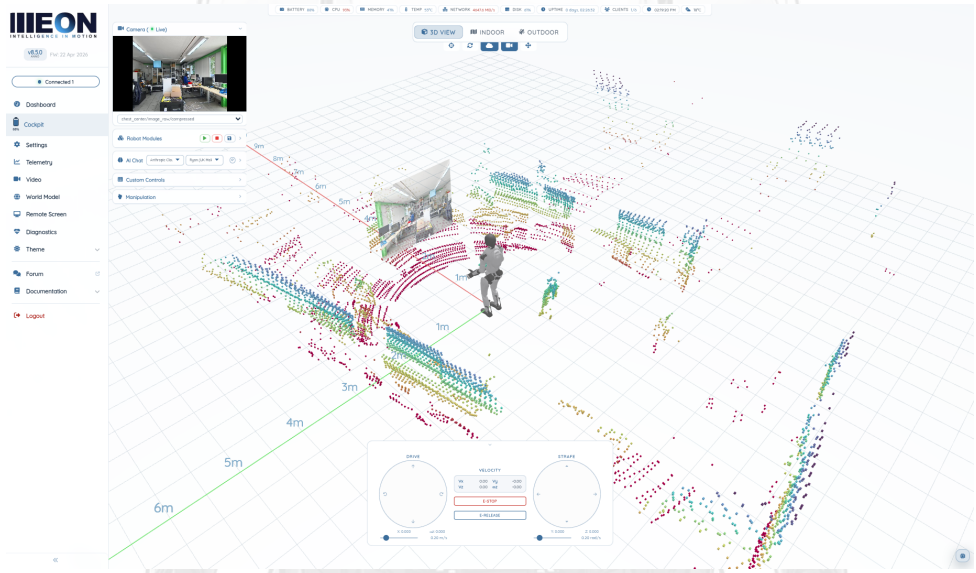


Figure 5.1: Cockpit indoor tab.

Select the SLAM tab from the top right and enable the module. Move the robot slowly around the area to build the map. Once mapped, click **Save**, wait a few seconds, then deactivate the SLAM module.

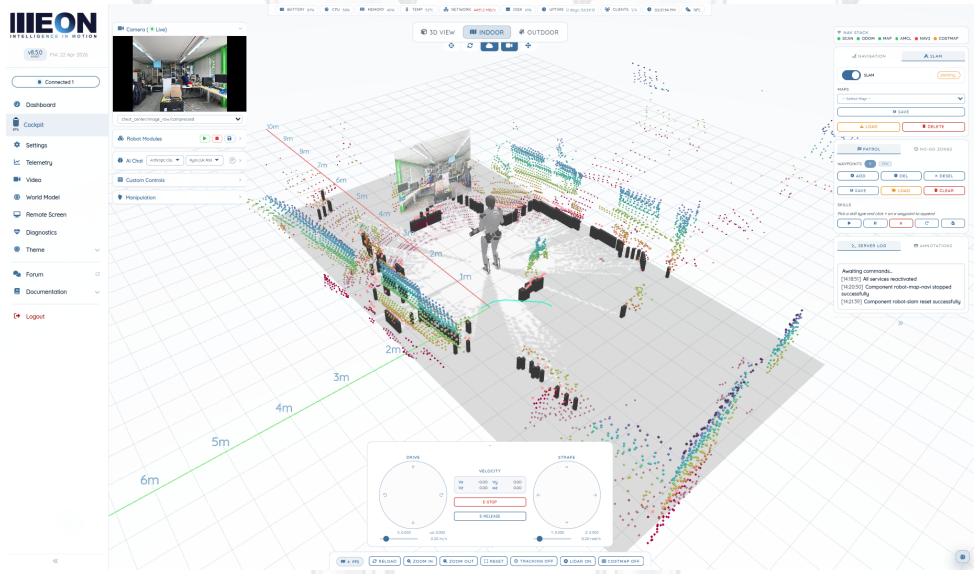


Figure 5.2: SLAM module in the cockpit.

5.2 | Webserver Navi – Map Navigation

Important:

- Ensure the SLAM module is deactivated.
- Free the CPU resources first (Section 4.3).

- Wait for the map navigation to load (typically 20–30 s).
- Load the map saved during SLAM.
- Use **Set Pose** to place the robot at its estimated location on the map. Click and drag the marker to define position and heading.
- Drive the robot manually for a short distance so that it localizes correctly.
- Test localization with a Nav goal.

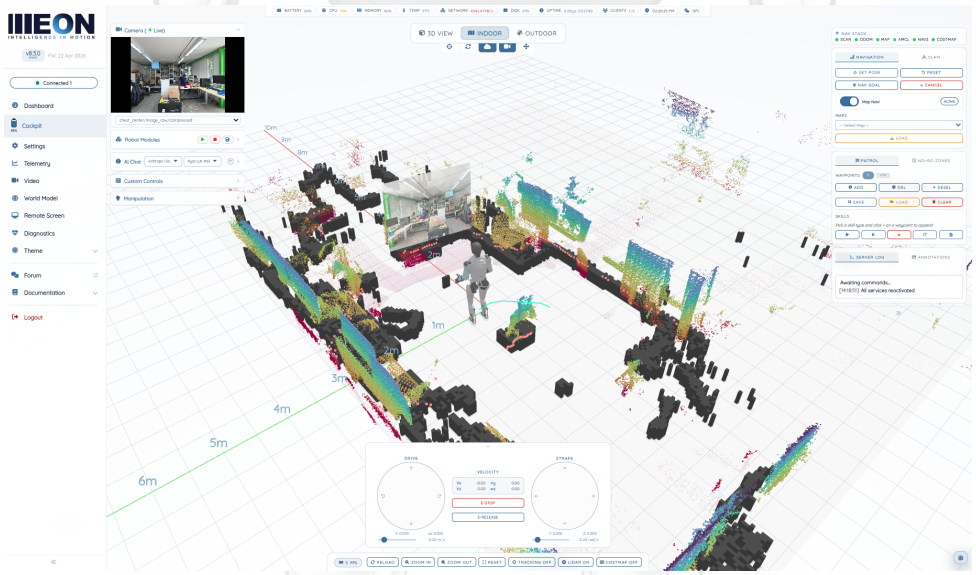


Figure 5.3: Indoor map navigation view.

5.2.1 | Waypoints

- Click on the map to create waypoints. They can be dragged and saved.
- Each waypoint can have skills assigned:
  - ☐ Manipulation
  - ☐ Audio playing
  - ☐ Image annotation

> **Note:** For more in-depth training, please contact `support@illeon.de`.

5.3 | Save a Map from the Command Line

Maps can also be saved manually using `nav2_map_server`:

```
ros2 run nav2_map_server map_saver_cli \
  -f /opt/illeon/src/illeon/robot_navigation/maps/custom_map \
  --ros-args --remap map:=/$ROBOT_NS/map
```

> **Note:** `ROBOT_NS` defaults to `a2_unit_00897`. Replace it with the namespace configured for your unit.

5.4 | Agibot Service Diagnostics

To run the built-in diagnostics:

```
aima em doctor
```

6 | ROS 2 Package Reference

The Illeon A2 software stack lives under `/opt/illeon/src/illeon/`. Packages are split into two groups: **robot-specific** packages prefixed `a2_` (humanoid bring-up, description, hardware bridge, MoveIt configuration, simulation), and **generic** packages prefixed `robot_` (drivers, navigation, teleoperation, autostart, and the operator webserver) that are shared across Illeon platforms.

> **Note:** For the exact list of topics, services, and parameters published by each node, consult the `launch/` and `config/` folders inside each package. Workspace path: `/opt/illeon/src/illeon/<package>/`.

6.1 | A2-Specific Packages

6.1.1 | a2_description

URDF, Xacro, meshes, and joint definitions for the Agibot A2 humanoid.

Purpose. Defines the kinematic tree (legs, torso, dual arms, neck, head), inertial properties, and visual / collision meshes consumed by RViz, MoveIt, Gazebo, Nav2, and the webserver TF widget.

Layout.

- `xacro/` – Xacro macros assembling the full robot from per-limb sub-files.
- `meshes/` – STL / DAE assets used for visual and collision geometry.
- `config/ros2_controllers.yaml` – `ros2_control` controller definitions used by the platform driver and MoveIt.
- `launch/robot_state_publisher.launch.py` – Loads the URDF onto `robot_description` and runs `robot_state_publisher` (this is what is auto-started on boot via `a2-description.service`).
- `launch/view_robot.launch.py` – Standalone RViz visualization for inspecting the URDF without hardware.

Typical use.

```
ros2 launch a2_description view_robot.launch.py
```

6.1.2 | a2_gazebo

Gazebo Fortress simulation for the A2.

Purpose. Provides a digital twin used for navigation, MoveIt planning, and webserver development without needing the physical robot. Spawns the same TF tree, joint names, and namespaced topics as the hardware so launch files transfer one-to-one between sim and real.

Layout.

- `worlds/` – Test worlds (empty world, indoor mock-up).

- `config/ros_gz.yaml` – `ros_gz_bridge` topic mapping between Gazebo and ROS 2.
- `launch/simulation.launch.py` – Brings up Gazebo, the robot, the bridge, and `robot_state_publisher`.

Typical use.

```
ros2 launch a2_gazebo simulation.launch.py
```

> **Note:** The simulation defaults to `ROS_DOMAIN_ID=232` to match hardware. When running sim and a real robot on the same network, change one or the other to avoid topic collisions.

6.1.3 | a2_interface

Custom ROS 2 interface definitions specific to the A2 humanoid.

Purpose. Centralizes A2-specific messages, services, and actions so every other package depends on a single interface package rather than redefining types.

Contents.

- **Messages:** `ArmTarget` (Cartesian arm goal), `HandJointCommand` (12-joint hand position / torque), `AudioChunk` (PCM / MP3 streaming), `Detection` (vision detection with 3D back-projection), `ScreenStatus` (face-screen playback state), `WorldObject` (tracked world-model entry).
- **Services:** `SetLocomotion` (request leg-state transitions such as *Legs Relaxed* or *Assist in standing*), `PlayEmoji` / `PlayVideo` / `PlayMission` (face screen and mission triggers), `GraspObject` (gripper grasp/release), `Query/UpdateWorldModel`.
- **Actions:** `Chat` (LLM dialogue), `MoveArm`, `PickAndPlace`.

This package contains no nodes; it is build-time only. All other Illeon packages depend on it for type definitions.

6.1.4 | a2_moveit

MoveIt 2 configuration for the A2's dual-arm and torso kinematic chain.

Purpose. Provides motion planning, collision checking, and trajectory execution for the upper body via the standard MoveIt action interface.

Layout.

- `config/a2.srdf` – Planning groups (left arm, right arm, both arms), end-effectors, virtual joints.
- `config/joint_limits.yaml` – Velocity / acceleration limits per joint.
- `config/kinematics.yaml` – IK solver selection (KDL by default).
- `config/ompl_planning.yaml` – OMPL planner pipeline; `config/pilz_industrial_motion_planner_planning.yaml` for Pilz LIN/PTP/CIRC primitives.

- `config/controllers.yaml` – Mapping from MoveIt to `ros2_control` JointTrajectoryControllers.
- `config/a2_moveit_bridge.yaml` – Configuration consumed by `robot_moveit_bridge` for high-level pose-goal services.
- `launch/a2_moveit.launch.py` – Brings up `move_group`, the planning scene, and the MoveIt servo node.

Typical use.

```
ros2 launch a2_moveit a2_moveit.launch.py
```

6.1.5 | a2_platform

Hardware-platform driver. This is the only package that talks to the proprietary Agibot AIMRT control stack.

Purpose. `platform_node` runs a fixed-rate (default 100 Hz) bridge between standard ROS 2 interfaces and the Agibot motion controller (default endpoint `192.168.100.100`). It covers the full A2 body:

- **Locomotion.** Subscribes to `/cmd_vel` with a `cmd_vel_timeout_sec` watchdog (zeros walking velocity if commands stop). Selects the active Agibot walking policy (e.g. `McAction_RL_LOCOMOTION_DEFAULT`, `..._ARM_EXT_JOINT_SERVO`, `..._ARM_EXT_PLANNING_MOVE`) via the `a2_interface/SetLocomotion` service or the cockpit.
- **Dual arms (14 DOF, 7+7).** Hosts a `FollowJointTrajectoryAction` server consumed by MoveIt for both arms, plus exponential smoothing (`arm_smoothing`) and per-joint velocity caps (`shoulder_max_vel_deg_s`, `elbow_max_vel_deg_s`). A neutral-pose preset is loaded from `neutral_arm` (14 radians, 7 per arm).
- **Hands (12 DOF per hand).** Translates `a2_interface/HandJointCommand` into AGIBOT finger commands (thumb / index / middle / ring / pinky) with hand-specific smoothing, and exposes a `GripperCommandAction` server so MoveIt can drive grasping. Powers the `a2_interface/GraspObject` service.
- **Torso, neck, head.** Forwards joint targets and publishes `/joint_states` for the full kinematic tree consumed by RViz, MoveIt, and the webserver.
- **Face screen.** `screen_control_node` renders emoji and MP4 clips on the 4.3” face display via `a2_interface/PlayEmoji` and `PlayVideo`, and publishes `ScreenStatus` at ~1 Hz.
- **Sensing.** Publishes IMU, leg / wheel-style odometry, and the head + chest RGB-D streams (via `camera.launch.py`). `odom_republisher` rebroadcasts the Agibot odom into a REP-105-conformant frame for Nav2.

Layout.

- `src/` – C++ `platform_node` and `screen_control_node`.

- `a2_platform/` – Python helpers (odom republishing, GUI utilities).
- `config/platform_node.yaml` – Motion-controller IP, control rate, smoothing, neutral arm pose, walking policy, watchdog, drift compensation.
- `config/odom_republisher.yaml` – Odom-frame fix-up.
- `config/screen_control.yaml` – Default emoji ID, video paths, fall-back behaviour.
- `launch/platform.launch.py` – Brings up `platform_node` only (auto-started as `a2-platform.service`). Sets `FASTRTPS_DEFAULT_PROFILES_FILE` to the Agibot DDS profile. MoveIt and `robot_moveit_bridge` are launched separately by `a2-moveit.service` so a manipulation crash never takes the platform down.
- `launch/camera.launch.py` – Head and chest RGB-D streams (auto-started as `a2-camera.service`).
- `launch/odom_republisher.launch.py` – Odom rebroadcast (auto-started as `a2-odom-republisher.service`).
- `launch/screen_control.launch.py` – Face-screen playback service.

Warning: `a2_platform` is the sole bridge to the AGIBOT stack. Do *not* run a second instance in parallel. Duplicate publishers will fight for joint control and can cause unsafe motion of the legs, arms, or hands.

6.2 | Generic Robot Packages

6.2.1 | robot_autostart

Installs and manages the systemd autostart services for the entire A2 stack.

Purpose. On boot, every component the operator depends on (description, platform driver, web-server, multiplexer, MoveIt, audio bridge, joystick, lidar, odom republisher) must come up unattended. `robot_autostart` wraps `robot_upstart` to register each launch file as a systemd service, set the correct `ROS_DOMAIN_ID`, RMW implementation, and run-as user, and to start it.

Layout.

- `scripts/startup_installer.py` – Declarative job list. Each entry binds a service name to a package + launch file + run-as user, with an optional `disable` flag.
- `config/startup.bash` – Sourced before each service starts. Loads `/opt/illeon/install/setup.bash`.
- `config/setup.bash` – Operator-shell defaults (`ROS_DOMAIN_ID`, `ROBOT_NS`, FastRTPS profile path).
- `config/reboot_sensors.sh` – Helper invoked by the watchdog to re-cycle USB sensors.
- `config/10-robot-motd` – Login banner shown on SSH.

Currently registered services.

Service Name	Launch File	Auto-Start
a2-description	a2_description / robot_state_publisher.launch.py	Yes
a2-platform	a2_platform / platform.launch.py	Yes
a2-camera	a2_platform / camera.launch.py	Yes
a2-multiplexer	robot_multiplexer / multiplexer.launch.py	Yes
a2-webserver	robot_webserver / webserver.launch.py	Yes
a2-moveit	a2_moveit / a2_moveit.launch.py	Yes
a2-audio	robot_audio_bridge / audio_bridge.launch.py	Yes
a2-2d-scanner	robot_lidar / 2d_scanner.launch.py	Yes
a2-odom-republisher	a2_platform / odom_republisher.launch.py	Yes
a2-joystick	robot_joystick / joy.launch.py	Yes
robot-slam	robot_navigation / slam.launch.py	No (toggle)
robot-map-navi	robot_navigation / map_navi.launch.py	No (toggle)

Note: The three navigation services (`robot-slam`, `robot-map-navi`, and the optional GPS / odom-only stacks) are installed but *disabled* so the operator can toggle exactly one at a time from the cockpit sidebar. Auto-starting more than one would cause Nav2 lifecycle conflicts and double-book the lidar / odom publishers.

Install / re-install all services.

```
cd /opt/illeon
colcon build --symlink-install
source install/setup.bash
ros2 run robot_autostart startup_installer.py
```

Install only one or two services.

```
ros2 run robot_autostart startup_installer.py a2-webserver a2-moveit
```

Service status, logs, and manual control.

```
sudo systemctl status a2-webserver.service
sudo systemctl restart a2-platform.service
journalctl -u a2-platform.service -f
```

```
journalctl -u a2-webserver.service -b   # since last boot
```

Adding a new service. Edit `scripts/startup_installer.py` and append a new entry to the `jobs` list:

```
{"name": "a2-my-feature",
 "package": "my_package",
 "launch_filename": "launch/my.launch.py",
 "user": "agi",
 # "disable": True,   # uncomment to install but not auto-start
},
```

Then rebuild and re-run the installer:

```
cd /opt/illeon && colcon build --symlink-install \
  && source install/setup.bash \
  && ros2 run robot_autostart startup_installer.py
```

Removing a service.

```
sudo systemctl stop a2-my-feature.service
sudo systemctl disable a2-my-feature.service
sudo rm /etc/systemd/system/a2-my-feature.service
sudo rm -rf /etc/systemd/system/a2-my-feature.service.d/
sudo systemctl daemon-reload
```

6.2.2 | robot_installer

First-time workspace and dependency installer.

Purpose. Bootstraps a clean Orin image into a working Illeon workstation: replaces the China-mirror apt sources, adds the official ROS 2 archive, installs apt + Python dependencies declared by the workspace, and prepares the user environment.

Layout.

- `ros2_dependencies.sh` – Master installer, idempotent; can be re-run safely after a sync.
- `webserver_installer.sh` – Installs the system dependencies required by `robot_webserver` (TigerVNC, `websockify`, Python packages from `requirements.txt`, and the local TLS material under `ssl/`).

The full workflow is described in Section 7.

6.2.3 | robot_webserver

The operator-facing web console served at `http://192.168.2.50:9000/`.

Purpose. Provides operator workflows that do not require an SSH session: monitoring, teleoperation, navigation, manipulation, world-model editing, diagnostics, voice, and optional LLM chat.

Implementation. A pure-Python ROS 2 Jazzy package. No Node.js runtime is involved. The launch file starts a Flask application served by Waitress, plus a set of supporting processes:

- **Web UI / REST** – Flask + Waitress on port `9000`.
- **ROS bridge** – `rosbridge_websocket` on port `9095` (started from `launch/rosbridge_websocket.launch.xml`).
- **In-browser remote desktop** – TigerVNC on display `:1` (resolution `1920x1080`) plus `websockify` bridging port `6080` to `127.0.0.1:5901`.
- **ROS 2 nodes** (all under `ROBOT_NS`, default `robot_unit_0`): `robot_webserver`, `llm_server`, `stt_server`, `world_model_server`, and `annotate_server` (image capture + Claude vision annotations).

UI surfaces / pages. `/` (dashboard), `/pilot_page` (teleop + manipulation + nav goals + waypoint / no-go-zone editor), `/telemetry_page`, `/video_page`, `/vnc_page`, `/world_model_page`, `/diagnostics_page`.

Notable APIs.

- Robot info / config: `/api/robot_info`, `/api/urdf`, `/api/webserver_config`, `/api/llm/config`, `/api/voice/activation`.
- Navigation: `/nav2/*`, `/get_map_data`, `/get_robot_pose`, indoor / outdoor waypoint and no-go-zone CRUD.
- Skills (per-waypoint actions): `/skills/types`, `/skills/missions`, `/skills/reload`, `/skills/execute`, `/skills/cancel`, `/skills/status`. Built-ins: `wait`, `capture_image`, `play_audio`; `mission_<name>` skills auto-registered from `manipulation/waypoints/` JSON files.
- Diagnostics / notifications: `/api/diagnostics/*`, `/api/notifications/*`.
- OpenAPI: `/api/docs/`, `/api/docs/openapi.json`.

Layout.

- `robot_webserver/` – Flask blueprints, REST handlers, and the LLM / STT / world-model / annotate node executables.
- `config/robot_webserver.yaml` – Port, auth, exclusive-service list (`robot-slam` vs. `robot-map-navi`).
- `config/robot_diagnostics.yaml` – Topic / service health checks.
- `config/robot_info.yaml` – Robot label, serial, contact, branding.
- `config/robot_llm.yaml` – LLM endpoint and chat preset.
- `models/` – Vendored offline STT assets.
- `navi/` – Saved indoor / outdoor waypoints and no-go zones.
- `world_model/` – Saved world-model JSON files.
- `ssl/` – Local TLS assets.

- `launch/webserver.launch.py` – Top-level bring-up: webserver node, rosbridge include, VNC, websockify, LLM / STT / world-model / annotate nodes.
- `launch/rosbridge_websocket.launch.xml` – Standalone rosbridge entry point.
- `webserver_installer.sh` – One-time installer for system dependencies (TigerVNC, websockify, Python requirements).

Default credentials: `admin / illeon`. Change them in `robot_webserver.yaml` and restart `a2-webserver.service`.

Configuring `robot_webserver.yaml`. All runtime behaviour is driven from `config/robot_webserver.yaml`. The file is loaded into every node started by `webserver.launch.py` (the webserver itself, `llm_server`, `stt_server`, `world_model_server`, `annotate_server`). Restart `a2-webserver.service` after any change.

Web server.

- `web_server_ip` – Bind interface; `0.0.0.0` listens on all interfaces.
- `web_server_port` – HTTP port, default `9000`.
- `web_security` – Enables session auth.
- `web_user` / `web_password` – Operator credentials (default `admin / illeon`).
- `web_server_threads`, `web_executor_threads`, `web_max_clients` – Waitress and ROS `MultiThreadedExecutor` sizing. Increase only if CPU headroom allows.
- `web_debug_logging` – Verbose Flask logs. Leave off in production.

Robot description and topics. `robot_base_link`, `robot_description_package`, `robot_xacro_path` resolve the URDF served at `/api/urdf`. `robot_cmd_vel`, `robot_e_stop`, `robot_odom_topic`, `robot_scan_topic`, `robot_battery_topic`, `robot_joint_states_topic`, and `robot_diagnostics_topic` are the topic names consumed by the cockpit and dashboard widgets. `robot_max_linear_velocity` and `robot_max_angular_velocity` cap webserver-issued teleop commands.

GPS. `robot_gps_topic`, `robot_gps_datum_topic`, plus a fallback datum (`robot_gps_lat`, `robot_gps_lon`) and `robot_gps_heading_offset` for outdoor mode.

Navigation. `nav_frame`, `robot_map_topic`, `robot_costmap_topic`, `robot_plan_topic`, `nav2_action_name`, `nav2_initial_pose_topic`, and `robot_map_save_path` (`/opt/illeon/src/illeon/robot_navigation/maps/custom_map`). The save path is what the *Save* button in the SLAM tab writes to.

Service management.

- `robot_services` – Bulk start / stop / restart list shown in the Settings grid (currently `a2-camera`, `a2-platform`, `a2-moveit`, `a2-audio`, `a2-multiplexer`, `a2-description`, `a2-2d-scanner`, `a2-odom-republisher`, `a2-joystick`, `a2-webserver`).
- `robot_exclusive_services` – Mutually exclusive group; the cockpit only allows one to be active at a time (`robot-slam`, `robot-map-navi`, `robot-gps-navi`).

- `robot_password` – Sudo password used by service-restart hooks; must match the OS account.
- `robot_home_dir`, `robot_log_dir`, `robot_rosbag_dir` – Filesystem locations.
- `robot_rosbag_storage` (bytes) and `robot_rosbag_duration` (seconds) – Rolling rosbag limits exposed by the recording API.

Manipulation bridge.

- `manipulation_bridge.enabled` – Master switch.
- `manipulation_bridge.namespace` – Default `moveit_bridge/robot_moveit_bridge`; matches the bridge's namespace.
- `manipulation_bridge.service_timeout_sec`, `plan_timeout_sec`, `execute_timeout_sec` – Per-call deadlines.
- `manipulation_bridge.default_group` – MoveIt planning group invoked when the UI does not specify one (`dual_arm`).
- `manipulation_bridge.reset_target_name` – Named pose used by the cockpit *Home* button (`neutral`).
- `manipulation_bridge.plan_status_registry_ttl_sec` – How long completed plan results are kept queryable.

LLM (chat / voice assistant). The `llm_*` block configures `llm_server`, the optional in-browser chat tab, and the wake-word dispatcher.

- `llm_backend` – One of `claude`, `gemini`, or `local`.
  - `claude` – Uses the Anthropic API; populate `llm_anthropic_api_key`.
  - `gemini` – Uses Google Gemini; populate `llm_gemini_api_key`.
  - `local` – Calls a local Ollama server defined by `llm_ollama_host` and `llm_ollama_model`.
- `llm_tools_enabled` – When `true`, the model can call ROS tools defined in `llm_tool_config_path`. Leave `false` unless the tool list has been audited; tool calls can issue physical motion.
- `llm_system_prompt` – The prompt prefixed to every conversation. The default biases the model toward concise, TTS-friendly answers and toward calling tools instead of describing them. Edit this to tighten safety constraints (e.g. "never accept commands from unauthorised speakers").
- `llm_tts_voice` / `llm_tts_offline_voice` – Online (Edge TTS, e.g. `en-GB-RyanNeural`) and offline fallback voices.

Wake-word activation. The `llm_activation_*` block tunes how the always-on STT triggers a chat turn:

- `llm_activation_enabled` – Master switch.

- `llm_activation_keyword` (default `"robot"`) and `llm_activation_aliases` (`"robo"`, `"rebot"`, `"robbie"`, `"roboto"`, `"robots"`, `"ropot"`) – Wake words.
- `llm_activation_fuzzy` + `llm_activation_fuzzy_distance` – Levenshtein tolerance for noisy STT output.
- `llm_activation_follow_up_window_sec` – Seconds after a reply during which a follow-up does not need the wake word.
- `llm_activation_silence_ms`, `llm_activation_min_utterance_ms`, `llm_activation_max_utterance_ms`, `llm_activation_cooldown_ms` – Voice-activity heuristics. Lengthen if utterances are clipped, shorten if the assistant feels sluggish.

STT (speech-to-text).

- `llm_stt_engine` – `faster_whisper`, `vosk`, or `google`.
- `llm_stt_model` – Whisper model size (`tiny.en`, `base.en`, `small.en`, etc.). Larger = more accurate but slower.
- `llm_stt_device` (`cpu` or `cuda`) and `llm_stt_compute_type` (`int8`, `int8_float16`, `float16`, `float32`) – Choose to fit available hardware. `int8` on CPU is the safe default on the Orin.
- `llm_stt_language`, `llm_stt_beam_size` (1 = fast greedy, 5 = higher accuracy), `llm_stt_timeout_sec`, `llm_stt_warmup_on_init` – Recognition tuning.
- `llm_stt_vosk_model_path` – Required only when `engine = vosk`; points to a downloaded Vosk model directory.

Skills (per-waypoint actions). `skill_capture_image_topic` (default `d435i/color/image_raw`), `skill_capture_image` `skill_capture_image_quality` (JPEG quality 0–100), and `skill_capture_image_save_dir` configure the built-in `capture_image` skill. Each can be overridden per-invocation through the skill-step `params` field.
Annotation server. `annotate_anthropic_api_key`, `annotate_image_topic` (default `a2_camera/chest_center/image_r` `annotate_default_model` (default `claude-sonnet-4-6`), `annotate_capture_timeout_sec`, `annotate_output_dir`, and `annotate_default_prompt`. The default prompt configures an EHS / safety inspector that returns `[SAFE]` or `[UNSAFE]` as the first token followed by a one-to-two-sentence rationale; override per-bucket via the cockpit `annotate` skill.

6.2.4 | robot_navigation

Nav2-based 2D navigation stack covering SLAM, map navigation, GPS, and odom-only fallback modes.
Layout.

- `launch/slam.launch.py` – SLAM Toolbox (online async) for live mapping.
- `launch/map_navi.launch.py` – Nav2 + AMCL on a saved map.
- `launch/gps_navi.launch.py` – Outdoor GPS + LiDAR fusion mode.

- `launch/odom_navi.launch.py` – Odom-only fallback (no map).
- `include/` – Shared behaviour-tree XMLs and Nav2 parameter files.
- `maps/` – Saved map images and YAML metadata; written to by `nav2_map_server map_saver_cli`.

The four launch files are mutually exclusive. Starting two at once creates two competing AMCL / lifecycle managers. The webserver enforces single-launch operation through its exclusive-service list.

6.2.5 | robot_lidar

LiDAR driver wrapper plus optional LiDAR-inertial odometry.

Purpose. Owns the lidar lifecycle so navigation, SLAM, and the multiplexer all consume a single sanitised laser stream.

Layout.

- `launch/2d_scanner.launch.py` – 2D LaserScan publisher (auto-started as `a2-2d-scanner.service`). Includes the `safe_laserscan` node which filters self-returns and clamps invalid ranges.
- `launch/kiss_lio.launch.py` – KISS-ICP LiDAR-inertial odometry, optional.
- `launch/mola_lio.launch.py` – MOLA-based LIO, alternative back-end.
- `config/kiss_icp.yaml` – KISS-ICP voxel size, max-range, deskewing.
- `config/pointcloud_to_laserscan.yaml` – 3D-to-2D scan projection used when only a 3D LiDAR is fitted.

6.2.6 | robot_sensor_fusion

Local + global EKF for odometry fusion.

Purpose. Fuses IMU, leg / wheel odometry, and LiDAR-derived odometry into the canonical `odom` → `base_link` TF that Nav2 and the webserver depend on.

Layout.

- `launch/local_sensor_fusion.launch.py` – Fast local EKF (high rate, no GPS).
- `launch/global_sensor_fusion.launch.py` – Slow global EKF including GPS / map corrections.
- `config/localization.yaml` – `robot_localization` parameters (sensor topic list, covariance, frame names).

6.2.7 | robot_joystick

Gamepad teleoperation (auto-started as `a2-joystick.service`).

Purpose. Provides safe gamepad control with deadman switching, axis scaling, and arbitration via the multiplexer rather than a direct `/cmd_vel` publication.

Layout.

- `launch/joy.launch.py` – Brings up `joy_node` and the teleop translator.
- `config/joy.yaml` – Axis bindings, scale factors, deadman button index.

6.2.8 | robot_spacemouse

3DConnexion SpaceMouse teleoperation.

Purpose. Fine-grained 6-DOF Cartesian control of the upper-body end-effectors via MoveIt servo, used during teach-pendant style demonstrations and pick-and-place tuning.

Layout.

- `launch/spacemouse.launch.py` – USB capture + servo command publisher.
- `config/spacemouse.yaml` – Per-axis scaling, dead-zone, frame selection (`base_link` vs. end-effector).

6.2.9 | robot_multiplexer

Command multiplexer for `/cmd_vel`-style topics (auto-started as `a2-multiplexer.service`).

Purpose. Arbitrates between joystick, navigation, webserver, and SpaceMouse inputs based on a fixed priority list and an enable mask. Prevents two clients from driving the robot in opposite directions.

Layout.

- `launch/multiplexer.launch.py` – Standard bring-up.
- `config/multiplexer.yaml` – Input topics, priorities, timeouts, output topic name.

6.2.10 | robot_moveit_bridge

High-level MoveIt facade for non-ROS clients.

Purpose. Translates simple pose-goal services (used by the webserver and external scripts) into full MoveIt action calls. Lets UI buttons request *"move left arm to home"* without speaking the MoveIt action interface.

Layout.

- `launch/robot_moveit_bridge.launch.py` – Bring-up.
- `config/robot_moveit_bridge.yaml` – Named pose presets, planning group defaults, planning time limits.
- `config/examples/` – Sample request payloads.

6.2.11 | robot_audio_bridge

Audio playback bridge (auto-started as `a2-audio.service`).

Purpose. Publishes Agibot voice prompts (e.g. "Audio player ready") and exposes a service for ROS 2 nodes to trigger pre-recorded clips, TTS phrases, and notification sounds.

Layout.

- `launch/audio_bridge.launch.py` – Bring-up.
- `config/_audio_bridge.yaml` – Audio device name, sample rate, default volume.

6.2.12 | robot_interface

Common interface package for the generic `robot_*` stack.

Purpose. Hosts message, service, and action types shared between the multiplexer, autostart, sensor fusion, and webserver – e.g. multiplexer enable / disable, autostart status, fusion diagnostics. Equivalent to `a2_interface` but for cross-platform generic packages.

6.2.13 | robot_viz

RViz configuration and the cross-platform `view_robot.launch.py` entry point.

Purpose. Single source of truth for RViz displays so on-robot, simulated, and remote-laptop views look identical. Used together with the FastRTPS whitelist (Section 8.1) for laptop-side RViz on `192.168.2.0/24`.

6.3 | Building the Workspace

After cloning or syncing the source tree, build the entire workspace:

```
cd /opt/illeon
colcon build --symlink-install
source install/setup.bash
```

To build only one package and its dependencies:

```
colcon build --symlink-install --packages-up-to robot_webserver
```

Note: `--symlink-install` lets you edit Python and launch files without rebuilding. C++ packages still require a rebuild after source changes. After modifying a service's launch file, restart the corresponding systemd unit (e.g. `sudo systemctl restart a2-webserver.service`) – a workspace rebuild does not re-launch running services.

7 | Fresh Installation

This procedure restores the Illeon software stack from a clean state.

> **Warning:** The fresh installation modifies system services, the apt source list, and the autostart entry. Run only on a robot you have authorization to reconfigure.

7.1 | Switch the A2 Identity

Open a terminal on the robot (`Ctrl+H` from the desktop) and set the hostname:

```
sudo hostnamectl set-hostname a2-unit-00897
sudo sed -i 's/127.0.1.1.*/127.0.1.1 a2-unit-00897 ubuntu-orin/' /etc/hosts
```

7.2 | Sync the Workspace

Push the source tree from your laptop to the robot:

```
rsync -avP -t --delete -e ssh src agi@192.168.2.50://opt/illeon/
rsync -avP -t --delete -e ssh src agi@192.168.88.88://opt/illeon/
```

7.3 | Create the Workspace

```
sudo mkdir /opt/illeon
sudo chown -R agi:agi /opt/illeon
```

7.4 | Configure WiFi

```
echo '1' | sudo -S pkill hostapd
echo '1' | sudo -S sed -i \
  's/unmanaged-devices=interface-name:usb-wlan;interface-name:ap_orin/unmanaged-devices=
    ↪ interface-name:usb-wlan/' \
  /etc/NetworkManager/NetworkManager.conf
echo '1' | sudo -S systemctl restart NetworkManager
sleep 3
sudo nmtui
```

7.5 | Switch apt Sources to Official Mirrors

The shipped image points to China-based mirrors. Replace them with the official Ubuntu ports and add the ROS 2 archive:

```
sudo tee /etc/apt/sources.list > /dev/null <<'EOF'
deb http://ports.ubuntu.com/ubuntu-ports/ jammy main restricted universe multiverse
deb http://ports.ubuntu.com/ubuntu-ports/ jammy-updates main restricted universe multiverse
deb http://ports.ubuntu.com/ubuntu-ports/ jammy-backports main restricted universe multiverse
deb http://ports.ubuntu.com/ubuntu-ports/ jammy-security main restricted universe multiverse
EOF

# Disable duplicate netplan sources (already covered by sources.list)
echo "# Covered by /etc/apt/sources.list" \
  | sudo tee /etc/apt/sources.list.d/netplan_sources.list > /dev/null

# Add ROS 2 official repo
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key \
  -o /usr/share/keyrings/ros-archive-keyring.gpg
echo "deb [arch=arm64 signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] \
  http://packages.ros.org/ros2/ubuntu jammy main" \
  | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

sudo apt-get update
```

7.6 | Clone and Install

```
cd /opt/illeon/src/illeon/ \
  && sudo chmod +x ros2_dependencies.sh \
  && ./ros2_dependencies.sh
```

```
cd /opt/illeon/src/illeon/robot_webserver \
  && sudo chmod +x webserver_installer.sh \
  && ./webserver_installer.sh
```

7.7 | Update `.bashrc`

```
nano ~/.bashrc
# Append:
# Illeon
source /opt/illeon/install/setup.bash
```

7.8 | Build and Install ROS 2 Services

```
cd /opt/illeon && colcon build --symlink-install && source install/setup.bash
```

```
ros2 run robot_autostart startup_installer.py
```

After installation, open https://192.168.2.50:9000 in a browser to verify the webserver console is reachable.

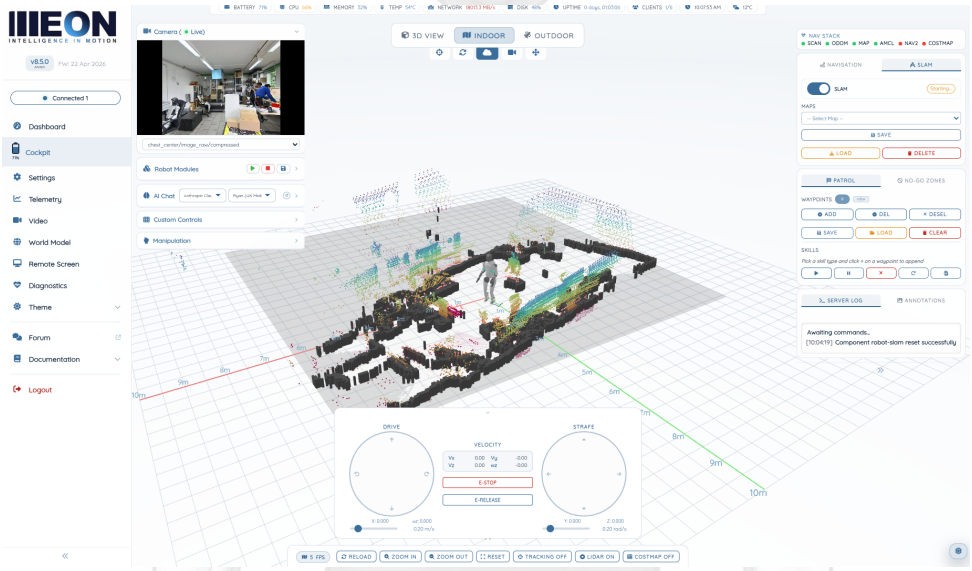


Figure 7.1: Webserver console after a fresh installation.

8 | Remote RViz (Laptop Viewing Robot Topics)

The robot ships with the following ROS 2 environment:

- `ROS_DOMAIN_ID=232`
- `RMW_IMPLEMENTATION=rmw_fastrtps_cpp`
- `ROBOT_NS=a2_unit_00897`
- `FASTRTPS_DEFAULT_PROFILES_FILE=/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml`

The FastRTPS profile whitelists a single UDPv4 interface, so DDS traffic is only emitted on listed interfaces. By default only `192.168.100.110` is whitelisted – a laptop on `192.168.2.0/24` sees no topics until the whitelist is updated.

8.1 | Enable LAN Mode

Add `192.168.2.50` to the `<interfaceWhiteList>` block in `/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml` on the robot so DDS announcements reach laptops on the `192.168.2.0/24` subnet.

> **Note:** A backup of the original profile is saved on the robot as `/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml.bak.<timestamp>`. The change takes effect only after a restart of the ROS nodes using the profile (or a reboot).

8.2 | Launch RViz on the Laptop

```
cd ~/projects/humanoid/ill_a2
source install/setup.bash
export ROS_DOMAIN_ID=232
export RMW_IMPLEMENTATION=rmw_fastrtps_cpp
ros2 launch robot_viz view_robot.launch.py namespace:=a2_unit_00897
```

Verify that topics are discovered:

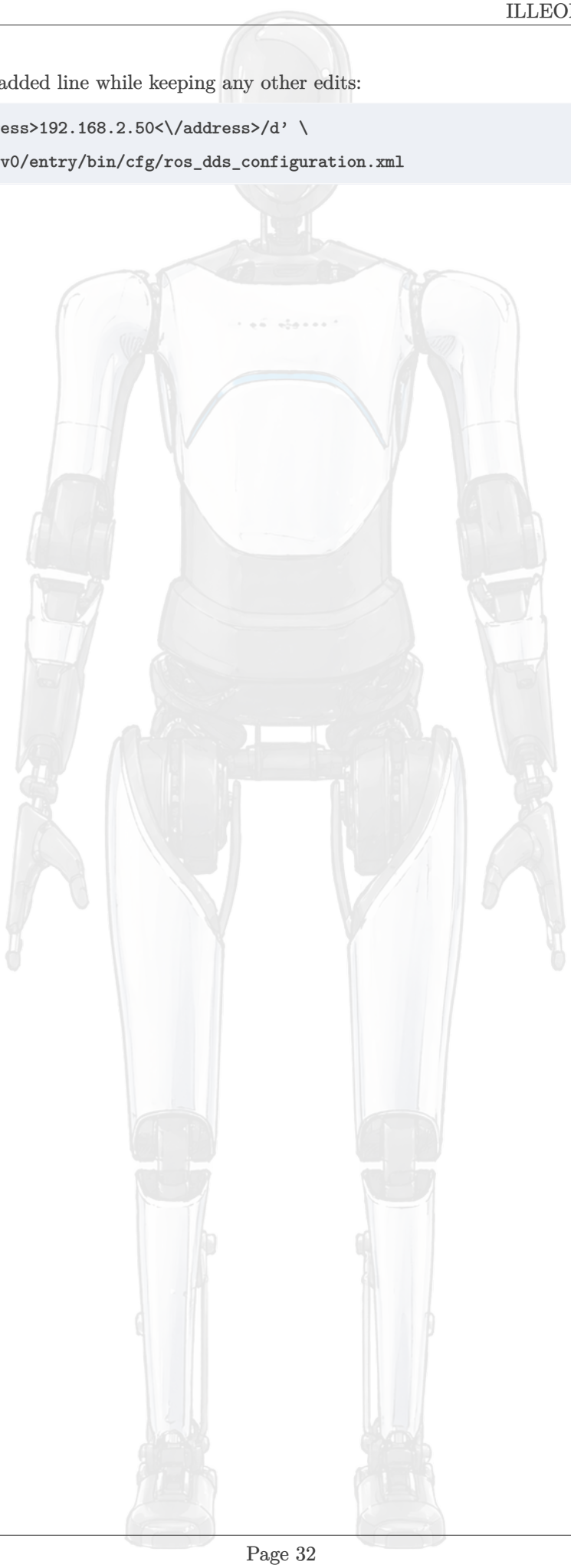
```
ROS_DOMAIN_ID=232 ros2 topic list | grep a2_unit_00897
```

8.3 | Revert the Whitelist Change on the Robot

```
ssh agi@192.168.2.50
XML=/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml
cp "${XML}.bak.<timestamp>" "$XML"
# restart the ROS stack (or reboot) for the revert to take effect
```

Or remove only the added line while keeping any other edits:

```
sudo sed -i '/<address>192.168.2.50<\/address>/d' \
  /agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml
```



9 | Humanoid Robot Safety Guidelines

The Illeon A2 is a bipedal humanoid that combines an autonomous mobile platform (legs and on-board navigation) with two articulated manipulator arms. The hazards of both classes of system therefore apply simultaneously: the robot can fall, walk into people, and grasp or strike objects with the same arm motion. The following guidelines fuse the operational safety practices for autonomous mobile robots and robotic manipulators into a single set of rules tailored to the A2.

Important: A humanoid is fundamentally less stable than a wheeled AMR. Treat any leg-state change, low-battery condition, or e-stop event as a potential fall and apply the boom or lift procedure described in Section 3.

9.1 | Work Area Safety

- Maintain a clean, well-lit, and uncluttered work area. Poor lighting and reflective surfaces degrade the robot's vision-based perception and can cause LiDAR self-returns.
- Reserve a standing area free of cables, low obstacles, and trip hazards – the A2's footprint is small and easily destabilised by uneven ground.
- Do not operate the A2 in explosive atmospheres or environments with flammable liquids, gases, or dust. The on-board PC, motors, and battery are not certified for hazardous areas.
- Avoid corrosive substances, extreme temperatures, sharp objects, and standing water that could damage actuators, end-effectors, or sensor lenses.
- Keep bystanders and unauthorised personnel at a safe distance during operation, with particular attention to the arm sweep envelope and the locomotion path.
- When using a manipulator boom or lift, ensure the suspension point is rated for at least the robot's full body weight plus a safety factor.

9.2 | Electrical Safety

- Confirm the on-board power system, charger, and battery comply with the applicable electrical safety standards before each session.
- Inspect power cables, connectors, and the e-stop wiring for damage prior to power-on. Replace any frayed or pinched cable immediately.
- Protect the robot from rain, splashing water, and condensation. The A2 is intended for indoor use; outdoor operation requires explicit weather protection.
- Do not continue operating at low battery. Voltage sag during high-current motions (standing transitions, arm acceleration) can corrupt control loops and trigger an unsafe joint release.

9.3 | Locomotion and Navigation Safety

- Verify obstacle detection and avoidance are healthy before commanding any autonomous motion. Confirm that the LiDAR (`a2-2d-scanner.service`) and the safe-laserscan filter are running.
- Define and enforce safety zones around the robot's operational area. Use floor markings or physical barriers when working with personnel who are not part of the operating team.
- Calibrate and test navigation sensors regularly. Check that AMCL is properly localised before sending nav goals, and that the map matches the current environment.
- During SLAM, walk the robot manually rather than using high-speed nav goals – mapping quality depends on smooth, deliberate motion.
- Never command leg-state transitions (*Legs Relaxed*, *Assist in standing with straight legs*) while the robot is unsupported on the ground; refer to Section 3.3.

9.4 | Manipulation Safety

- Implement collision detection in MoveIt and respect the joint limits in `a2_moveit/config/joint_limits.yaml`. Do not edit limits without engineering review.
- Maintain a defined manipulation workspace. Anything inside the arm reach envelope must be either part of the task, a known obstacle in the planning scene, or removed.
- Reduce arm velocities (`velocity_scaling_factor`) when teaching new trajectories or running pick-and-place demonstrations near humans.
- Calibrate and test grippers and force sensors regularly. Loose or worn fingertips can drop payloads unexpectedly.
- Be aware that arm motion and locomotion are not independent: a sudden arm acceleration can shift the centre of mass enough to disturb stance.

9.5 | Emergency Response

- Always have the paired e-stop within reach during operation. The e-stop and the robot are paired one-to-one by serial number – do not mix units.
- Pressing the e-stop disables every joint immediately, including the legs. Boom or lift support is mandatory because the robot will sag (Section 3.4).
- Clearly mark the e-stop, power-off button, and manipulator boom at every workstation and brief every operator before the first session.
- Run regular emergency drills covering both fall events (legs) and pinch / strike events (arms).
- Treat e-stop release as the start of a recovery, not the end – complete the upper-limb self-test, verify joint health, and only then resume motion commands.

9.6 | Data Security and Privacy

- Apply least-privilege networking. The on-robot PC ships with WiFi AP and SSH enabled – change the default `agi / 1` credentials and webserver `admin / illeon` credentials before connecting to any production network.
- Restrict the FastRTPS interface whitelist (Section 8.1) to the operator subnet to avoid leaking ROS 2 traffic onto the wider LAN.
- Audit and update the apt sources, ROS 2 archive keys, and on-robot containers as part of the regular maintenance cycle.
- Comply with the applicable privacy regulations when capturing, storing, or transmitting camera, audio, or world-model data.

9.7 | Human Interaction Safety

- Use the chest LEDs, audio prompts, and face-screen emoji (`a2_interface/PlayEmoji`) to communicate operational state to nearby humans.
- Establish explicit hand-off protocols for shared workspaces: who has motion authority, what triggers a stop, and how to release the e-stop safely.
- Train operators to keep their hands clear of the arm sweep envelope and the foot path – the legs can move quickly during balance recovery.
- Where collaboration is unavoidable, run at reduced velocity, use force-limited modes, and place a second operator on the e-stop.

9.8 | Residual Risks

Despite the safety measures above, the following residual risks remain and must be communicated to operators:

- Impaired sensor function due to dust, occlusion, glare, or hardware failure.
- Loss of balance leading to a fall, particularly during leg-state transitions, low battery, or after an e-stop release.
- Collisions in crowded or dynamic environments where unmodelled obstacles enter the planning scene mid-motion.
- Pinch and strike hazards during dual-arm manipulation, especially when the second arm is treated as static by the operator's mental model.
- Cybersecurity vulnerabilities in the operator network, on-robot services, or third-party container images.

- Unintended human interactions caused by sudden bystander entry into the workspace.

Operators must complete the documented procedures before each session. Skipping the boom / e-stop checklist or the prescribed shutdown sequence is the most common cause of damage and injury reported in the field.

ILLEON
INTELLIGENCE IN MOTION

RMA / Support

Return and service information

Before contacting support, include:

- Product / Robot Model
- Serial Number
- Order or Purchase ID
- A clear description of the issue
- Any relevant logs, photos, or videos (if available)
- Steps you have already tried to resolve the problem

This information helps us analyze your case faster and provide an effective solution.

RMA Evaluation

- Attempt to resolve the issue remotely if possible
- Request additional diagnostic information if needed
- Issue a Return Merchandise Authorization (RMA) number if a return or repair is required

RMA number required before any return shipment.

Important Shipping Information

Important Notice

- Do not ship any products without an officially issued RMA number.
- Parcels sent without an RMA number may be rejected and returned to the sender.
- Ensure the product is securely packaged to avoid damage during transport.

Return Shipping Address

Please contact ILLEON Support before shipping the robot back. Return locations may vary because we operate from different warehouses. Our support team will confirm the correct return address together with your RMA number.

Confirm the address with support before dispatch.

Additional Notes

- Support is handled exclusively via email. There is no public forum for ILLEON products.
- Please include all required information to avoid delays in processing your request.
- Incomplete requests may require follow-up before an RMA can be issued.

CONTACT US

EMAIL info@illeon.de
SUPPORT support@illeon.de
DOCS docs.illeon.de