Honey Badger Robot Documentation (software v1.1.0)

MAB Robotics

Oct 21, 2026

CONTENTS

1 Getting Started **3**
- 1.1 Introduction . . . 3
- 1.2 Next steps . . . 4
- 1.3 Safety . . . 4
- 1.4 First startup instructions . . . 7
- 1.5 Setup and Startup . . . 10
- 1.6 Steam Deck Controller . . . 12

2 Mechanical **17**
- 2.1 General dimensions . . . 17
- 2.2 Robot Legs . . . 18
- 2.3 Connectors . . . 18

3 Software Overview **27**
- 3.1 Topics and Messages . . . 27
- 3.2 Software Platform . . . 30
- 3.3 Custom message types . . . 31
- 3.4 Network . . . 38
- 3.5 Geometry and Frames . . . 40
- 3.6 Software Update . . . 42

4 Payload **47**
- 4.1 Payload mount . . . 47
- 4.2 Spool . . . 47

5 FAQ **53**
- 5.1 Getting started . . . 53



General parameters

Parameter	Value	Unit
Robot model	Honey Badger 5.0 [hb50]	
Length (with legs)	725	mm
Length (stowed)	520	mm
Width	320	mm
Height (sitting)	145	mm
Height (standing)	400	mm
Weight (inc. battery)	16	kg
Payload capacity	3	kg
Power		
Battery capacity	276.5	Wh
Battery max voltage	50.4	V
Typical runtime*	1.5	h
Standby time	3+	h
Charger max power	250	W
Charging time (0 - 80%)	50	min
Charging time (to full)	2	h
Payload power (5V)	10	W
Payload power (Vbat)	up to 100	W
Environmental		
Ingress Protection (IP)	IP51	
Operating temperature	5 - 45	°C
Mobility		
Walking Speed (Stable)	0.35	m/s
Walking Speed (Walk)	0.6	m/s
Slopes	+-30	°
Connectivity		
WiFi 5 AP 802.11	up to 867 Mbps	
Ethernet	1x LAN, 1x WAN/LAN, up to 1 Gbps each	

*Operating time in nominal conditions - no payload, constant walking (`Walk` gait), with average speed of 0.5m/s, on mostly flat, dry concrete surface.

CHAPTER

ONE

GETTING STARTED



1.1 Introduction

This manual describes the basic operation, handling, and maintenance procedures for the quadruped mobile robot `Honey Badger 5.0`. It is intended for operators, engineers, and service personnel who need to safely prepare, start, control, stop, and inspect the robot.

Before operating the robot, read this manual carefully, especially the safety instructions and emergency stop procedures. The robot contains powerful electric actuators and can move suddenly if enabled incorrectly.

1.1.1 Overview

The robot is a four-legged mobile robotic platform designed for research, testing, inspection, and autonomous or remotely controlled operation. It is equipped with electric actuators, onboard computers, a battery power system, sensors, and communication interfaces. The robot can operate in several modes, including idle, standing, walking, teleoperation, and safety/recovery mode. Depending on the installed software and hardware configuration, the robot may support mapping, localization, or autonomous operation.

The main components of the robot are:

- quadruped mechanical body,
  - four articulated legs with electric actuators,
  - two onboard computers (LPC and APC) — see *Platform* for roles and network addresses,
- battery system,
- sensors,
- communication interfaces,

- operator control interface.

1.2 Next steps

For startup and operation, use the core procedure pages below:

- *Safety* — safety and emergency procedures.
- *FirstStartup* — step-by-step startup checklist.
- *ControlConsole* — Steam Deck control and operator console.
- *Setup* — package contents, network reference, and charging.

1.3 Safety

This section contains the main safety instructions for Honey Badger 5.0 operation, handling, and maintenance.

> Read this page before powering on or operating the robot.

Note

The `Honey Badger 5.0` is a powerful mobile robot equipped with electric actuators, batteries, sensors, and onboard computers. Incorrect operation may result in injury, damage to the robot or damage to the surrounding environment.

1.3.1 General safety rules

- Operate the robot only after reading this manual.
- Keep hands, tools, loose clothing, and cables away from the robot legs.
- Do not touch the legs or joints while the robot is powered on or enabled.
- Always keep distance of at least 1 m between the operating robot and people, animals, stairs, fragile objects, or moving objects.
- Do not leave the robot operating unattended; an operator should always be within line of sight of the operating robot.
- Do not exceed the specified payload, speed, or operating limits.
- Stop the robot immediately if unexpected behavior occurs.

1.3.2 Operator responsibilities

The operator is responsible for safe preparation, supervision, and shutdown of the robot.

Before operation, the operator must:

- inspect the robot for visible damage,
- check that the battery is properly installed,
- ensure that the operating area is clear,
- confirm that all required software and communication systems are working,
- make sure that all nearby people are aware that the robot may move.

During operation, the operator must:

- supervise the robot at all times,
- maintain a safe distance from the robot,
- be ready to activate the emergency stop,
- monitor the robot state and battery level,
- stop operation if the robot behaves unexpectedly.

1.3.3 Operating environment

Operate the robot only in environments suitable for its current configuration and software version.

Do not operate the robot:

- on unstable, slippery, or wet surfaces,
- near stairs, ledges, holes, or drop-offs,
- in public areas without proper supervision and safety measures,
- near moving vehicles or machinery,
- in rain, snow, or standing water unless the robot is explicitly configured for such conditions,
- in explosive or flammable environments,
- outside the specified temperature and humidity limits.

1.3.4 Mechanical hazards

The robot legs contain powerful moving joints. The legs may move suddenly during startup, calibration, recovery, or operation.

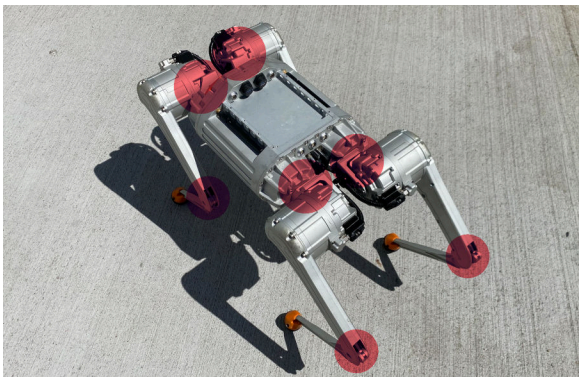
> **Warning**
>
> Keep hands, fingers, tools, and loose objects away from the legs and joints whenever the robot is powered on.

Do not:

- place hands between leg segments,
- lift or carry the robot by its legs,

- stand close to the robot during startup or recovery,
- attempt to stop leg motion by hand,
- perform mechanical adjustments while the robot is enabled.

Below is an image highlighting mechanical pinch points - avoid touching the robot near these at all times.



1.3.5 Electrical and battery safety

> **Warning**
>
> Honey Badger uses high capacity Lithium based battery pack. If operated incorrectly or outside guidelines specified in this manual, the battery can be a fire hazard.

The robot's battery is a specialized piece of equipment, unsuitable for use outside of the Honey Badger system. To avoid damage and hazardous conditions:

- use only approved batteries and chargers,
- do not use damaged, swollen, leaking, or overheated batteries,
- do not short-circuit battery terminals,
- do not connect or disconnect the battery if the robot is wet or water is present nearby,
- do not connect or disconnect the battery with wet hands,
- do not charge the battery unattended,
- do not operate the robot with damaged power cables or connectors,
- disconnect power before maintenance, unless the procedure explicitly requires power,
- after disconnecting the battery, secure the battery connector with the dedicated plug.

1.3.6 Safe handling and lifting

The robot is heavy and may be difficult to handle by one person.

- Power off the robot before lifting or transporting it.
- Disable the actuators before handling the robot.
- Carry the robot only by the main body or designated carrying points.
- Do not lift the robot by the legs, sensors, connectors, cables, or payload.
- Use two people or lifting equipment if required by the robot weight.
- Remove the battery before transport if required by the transport procedure.

Manual handling and repositioning

When moving or turning the robot by hand, always ensure it is powered off or in a safe idle state. The robot must not be handled while the actuators are enabled, as this can lead to unexpected motion or injury.

The illustration below shows a recommended belly-to-back repositioning move. Use this method only when the robot is powered off or safely idle.

Note

Always use an overhand grip when holding the robot legs. Keep your fingers away from gaps between mechanical parts, as they may cause crushing injuries.

Warning

Lifting the robot when Standing or Walking (using any gait) can result in unexpected and rapid legs movements. Never lift the robot while in Active balance mode!

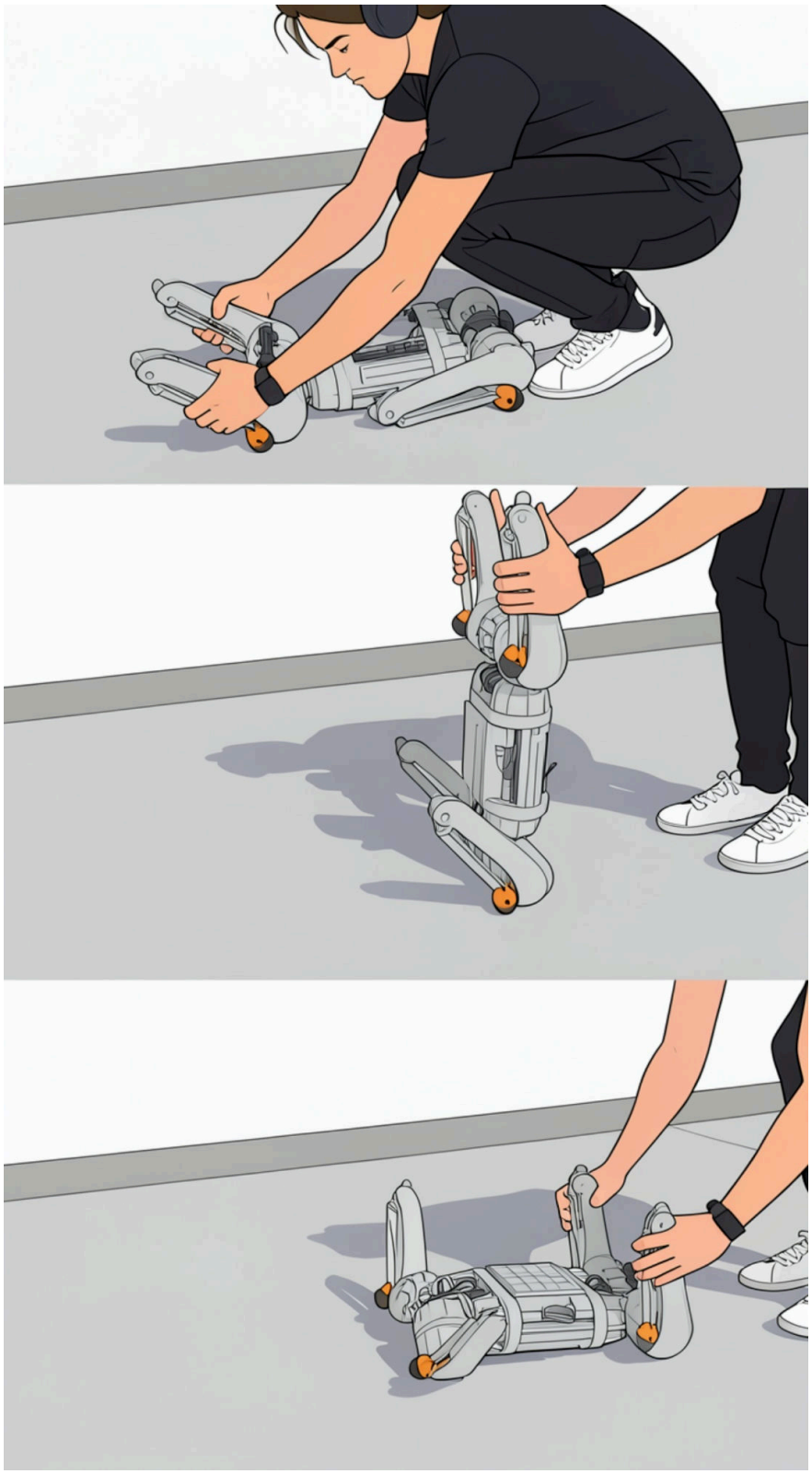
1.4 First startup instructions

This page provides a concise, safe checklist to power on, connect, and stand a new Honey Badger 5.0.

> Target audience: operators who need a short, reliable checklist to bring the robot to an operational standing state.

1.4.1 Quick checklist (read before starting)

- Read the Safety section in the main manual.
- Place robot on flat, stable ground with at least 1 m clearance around legs.
- Use two people to lift or position the robot if needed.
- Have the control console (Steam Deck) or laptop ready.



1.4.2 1 — Power on (approximately 90 seconds)

1. Place the robot on level ground and make sure nothing blocks leg motion.
2. Press and hold the robot power button for 2 seconds, or until LEDs start blinking purple. The purple blink lasts about 3 seconds and indicates the startup sequence has begun.
3. Wait for the power-up sequence to complete (approximately 90 seconds). Watch status LEDs.

1.4.3 2 — Connect control console and network

- On the Steam Deck (or laptop): connect to the robot's AP and open the control GUI.
- Confirm the control console shows the robot as `connected`.
- For the AP name, password, and wired connection options, see *Setup*.

1.4.4 3 — Verify system status (basic checks)

- Check battery level is sufficient (> 30% recommended for initial testing).
- Confirm that `control_node` and `bridge_node` are `ACTIVE` in the left menu of the GUI.
- Look for any error or warning messages in the console/log window.

1.4.5 4 — Energize actuators (enable)

1. Ensure area is clear and everyone stands back.
2. From the control console select the `ENERGIZE` (enable) command.
3. If a hardware `E-stop` is present and active, reset it only when it is safe to do so.

For full safety guidance, see *Safety*.

1.4.6 5 — Stand up sequence

1. With actuators energized, use the `STAND UP` command in the GUI.
2. Wait until the robot reaches a stable standing posture.
3. Switch to `STAND` or a walking mode (`WALK`, `STABLE WALK`) as required by your workflow. With the *adaptive locomotion mode* on, a walking robot stands still when there is no joystick input.
4. If the robot does not stand or joints twitch, immediately command `IDLE` and disable actuators.

1.4.7 6 — Basic drive test (low risk)

- Select a slow gait (e.g., `STABLE WALK`) and command a small forward step.
- Observe leg motion and listen for abnormal noises.
- If behavior is abnormal: immediately stop, select `IDLE`, and disable actuators.

1.4.8 7 — Shutdown

1. Stop all motion and select `IDLE` in the GUI.
2. Wait until the robot is stable and logs show no active motion.
3. Press and hold the power button for 2 seconds to power off, or use the `POWER COMMANDS` menu and select `SHUTDOWN`.

1.4.9 Troubleshooting (quick)

- No Wi-Fi: check that AP SSID is broadcasting.
- Control console says disconnected: reboot the console app and re-check Wi-Fi.
- Robot fails to energize: check battery voltage and error logs in the console.
- Overheating: disable actuators and inspect hardware.

1.4.10 Next steps / Links

- For full setup and charging procedures see *Setup*.
- For detailed control console usage see *ControlConsole*.
- For safety and emergency procedures see *Safety*.

1.5 Setup and Startup



1.5.1 What is included

The standard robot package may include the following items:

Item	Description	Standard	Pro
Quadruped robot	Main robotic platform	✓	✓
Battery pack	Power source for the robot	✓	✓
Battery charger	Charger compatible with the supplied battery	✓	✓
Operator controller	Dedicated remote controller	✓	✓
Documentation	Operating manual, safety instructions, and configuration notes	✓	✓
Ethernet cable	Used for direct network connection	optional	✓
Transport case	Protective case for storage and transport	optional	✓

The exact contents may depend on the purchased configuration. Check the packing list supplied with the robot.

1.5.2 Before first use

This page describes package contents, robot network setup, and charging. For the step-by-step startup sequence, follow *FirstStartup* and *Safety*.

1.5.3 Robot network reference

The robot provides a Wi-Fi access point for the control console and other devices.

- **hb50_XXXX** — 2.4 GHz network
- **hb50_XXXX_5G** — 5 GHz network

`XXXX` is the robot serial code shortened to 4 hexadecimal characters, for example `hb50_AFAB`.

Default Wi-Fi password: `milkaorzechowa`

For Steam Deck setup and controller-specific connection guidance, see *ControlConsole*.

1.5.4 Charging the robot

Use only the approved charger supplied with the Honey Badger 5.0 robot.

Before charging

- Place the robot on stable, flat ground.
- Ensure the robot is stopped and in a safe state.
- Check that the charger, cable, and connector are clean, dry, and undamaged.
- Keep the charging area dry and away from flammable materials.

Charging procedure

1. Connect the charger to the robot.
2. Connect the charger to mains power.
3. Confirm the power LED shows the expected charging state.
4. When charging completes, disconnect mains power.
5. Disconnect the charger from the robot.

Warning

Do not charge the robot unattended. Stop charging immediately if abnormal heat, smell, smoke, or noise is detected.

Note

If you operate the robot while charging, protect the cable from pinch points and avoid moving legs near the charger cable.

For detailed startup, stand-up, and shutdown guidance, use *FirstStartup*.

1.6 Steam Deck Controller

Remote control of HB robots is best done using the MAB-supplied Steam Deck controller with custom software and drivers. The controller makes it easy to start up, change gaits, and control the robot using the `HB50 Remote` application.

1.6.1 Network Setup

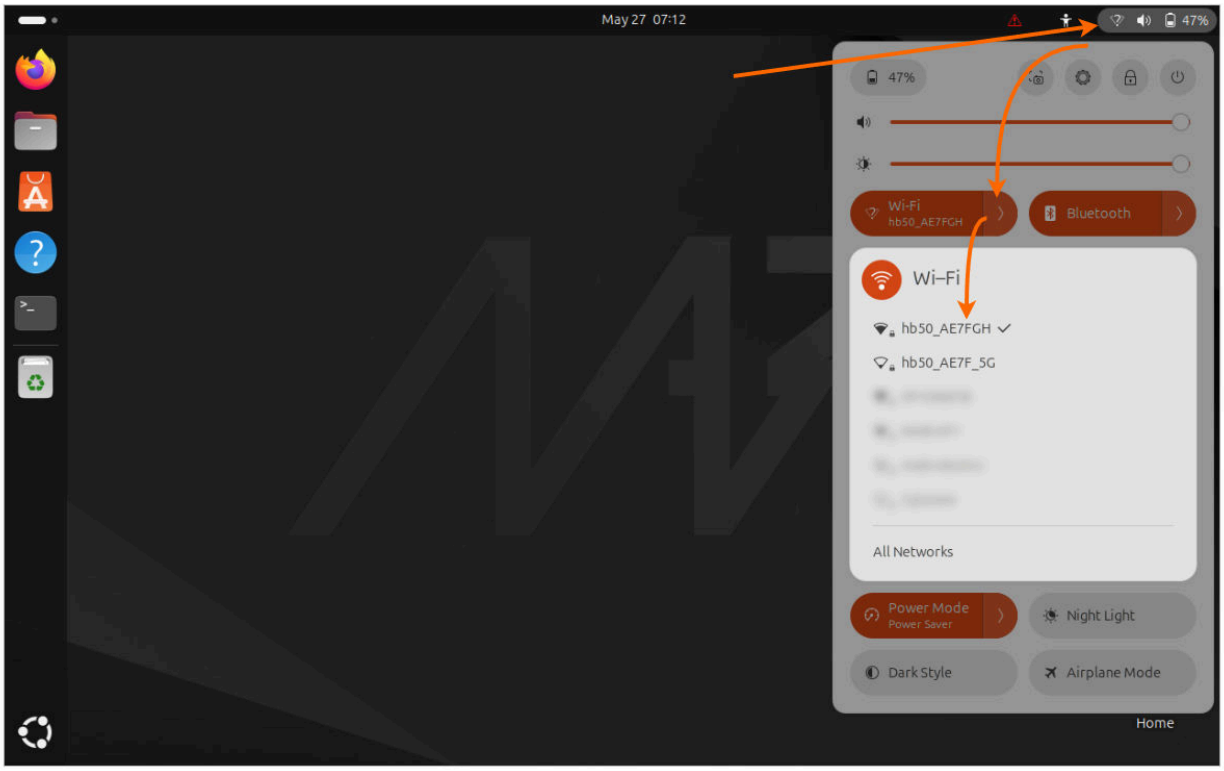
For robot Wi-Fi SSID and password details, see *Setup*.

Note

You can use Wi-Fi to connect to other networks (for example internet access). We recommend disabling automatic reconnection to avoid confusion when the robot's network becomes unavailable.

Wired connection

Connecting the Steam Deck to the robot is also possible via wired connection. This can be done through the device USB-C port and an Ethernet-to-USB dongle (not included). This setup can be extended to support IP traffic over other media such as optical fiber or long-range radios.

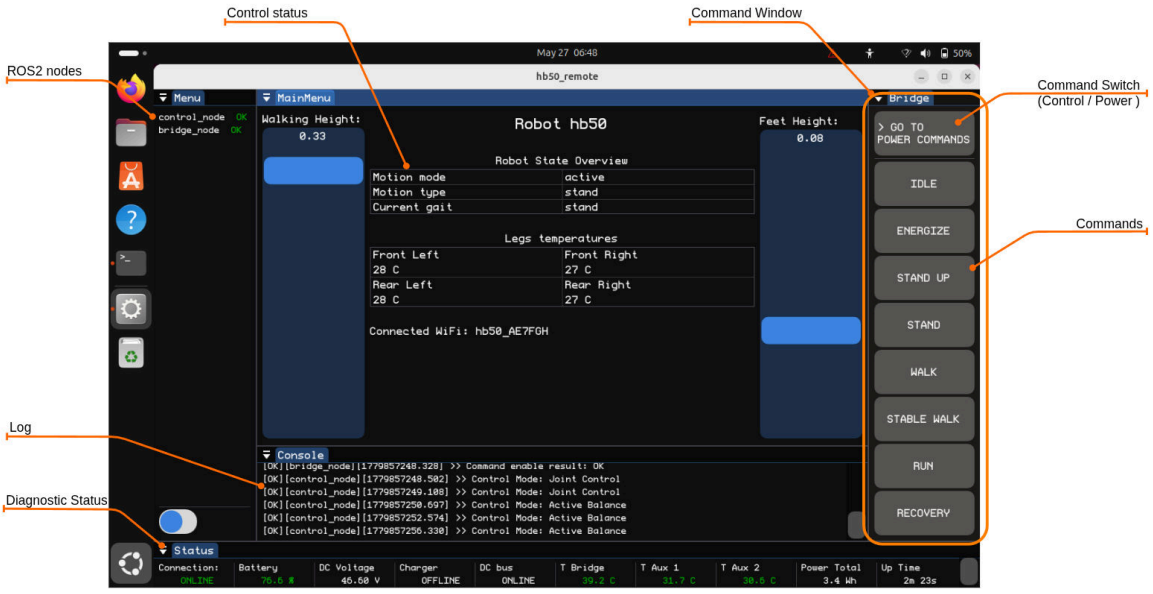


1.6.2 User Interface

On the desktop, there is an `HB50 Remote` application icon that starts the `remote_node` application for controlling Honey Badger robots. The application provides a quick diagnostic overview and enables typical robot operation tasks.

The application window consists of:

- **Status** (bottom) — connection state, battery level, DC voltage, charger and DC bus state, temperatures, energy used and up time,
- **Menu** (left) — state of `control_node` and `bridge_node`: `ACTIVE`, `INACTIVE`, `UNCONFIGURED` or `OFF`. During normal operation both nodes are `ACTIVE`. The `DEV MODE` toggle at the bottom is PIN-protected and intended for MAB service,
- **MainMenu** (center) — walking height and feet height sliders, robot state overview (motion mode, motion type and current gait), leg temperatures, connected Wi-Fi network and the *adaptive locomotion mode* toggle,
- **Console** — status messages from the robot and the application,
- **Bridge** (right) — robot commands. The Command Switch button at the top (`> GO TO POWER COMMANDS` / `> GO TO CONTROL COMMANDS`) switches between control and power commands.



Control commands

Command	Function
`IDLE`	Stops all motion and disables the actuators (on HB50W after 3 s).
`ENERGIZE`	Enables the actuators.
`STAND UP`	Stands the robot up. Requires energized actuators.
`STAND`	Standing with active balance.
`WALK`	Walking gait.
`STABLE WALK`	Slow, stable walking gait.
`WHEELS`	Driving on wheels — HB50W only, available directly after `STAND UP`. Use `STAND UP` to leave the wheel mode.
`RECOVERY`	Moves the legs back to the starting position after a fall. Available after `ENERGIZE` or `STAND UP`. If the robot lies on its back, the application asks to which side it should roll over — check for obstacles and choose the safe side.

Power commands

Each power command asks for confirmation before it is executed.

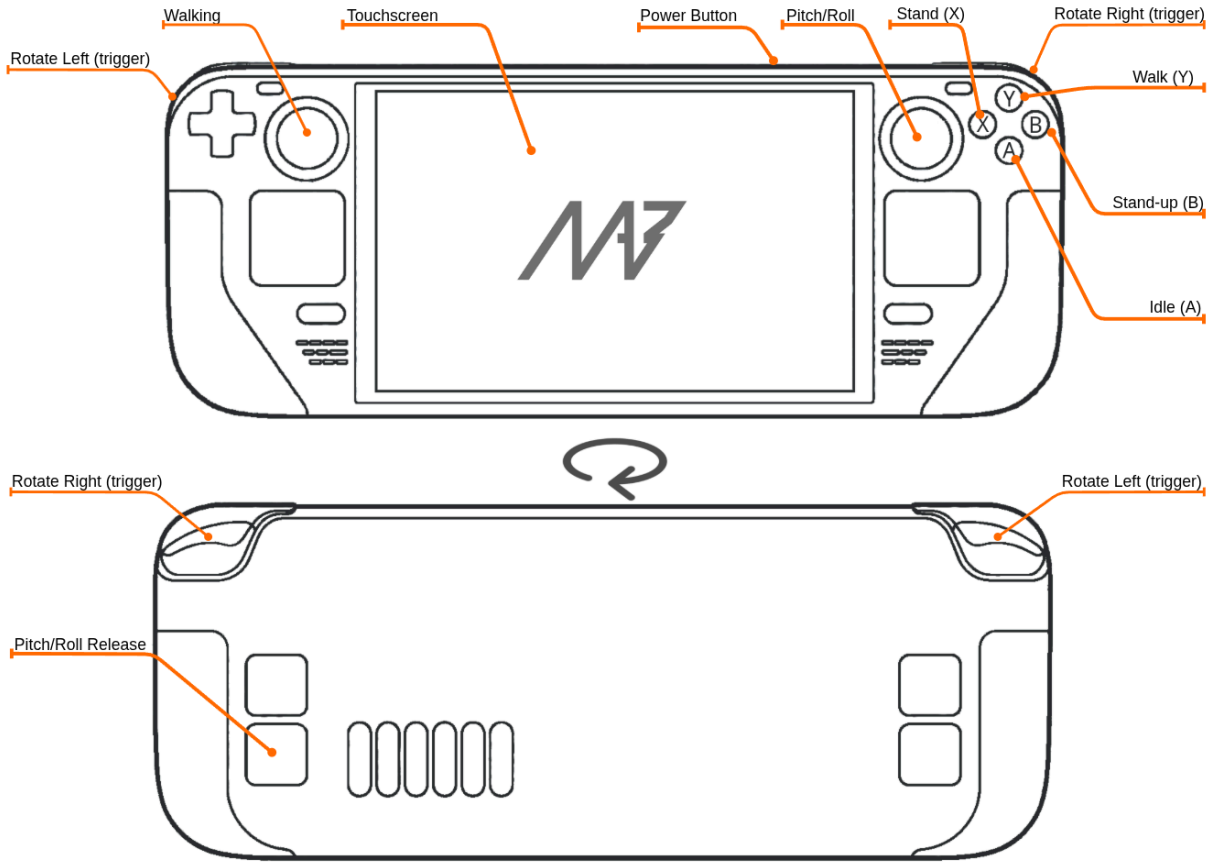
Command	Function
`SHUTDOWN`	Shuts down the robot.
`ENABLE BUS`	Powers on the actuators bus.
`DISABLE BUS`	Powers off the actuators bus.
`HEALTH CHECK`	Checks communication with all actuators and their positions. The result is shown in the console.
`RESET CONTROL`	Restarts `control_node`.

Adaptive locomotion mode

When the adaptive locomotion mode is on (default), a walking robot stops stepping and stands after 1 s without joystick input. When the joystick is moved again, the robot returns to the selected gait (`WALK` or `STABLE WALK`) and ramps up the velocity over 0.5 s. When the mode is off, the robot keeps stepping in place.

1.6.3 Controls Layout

Alongside `remote_node` there is also another node `teleop_node`, which captures joystick button inputs and converts them to robot commands. Below is a list of supported commands and actions:

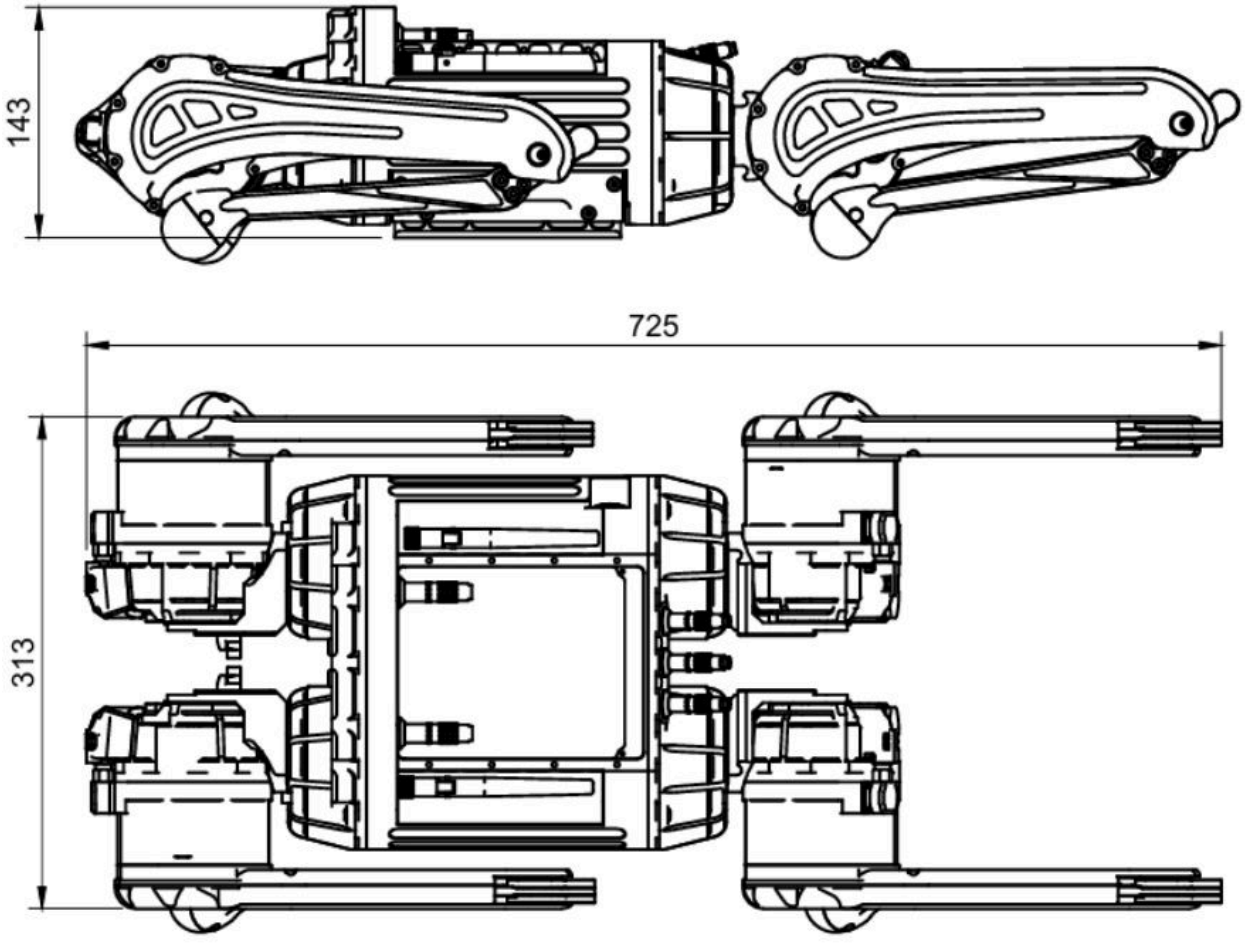


CHAPTER

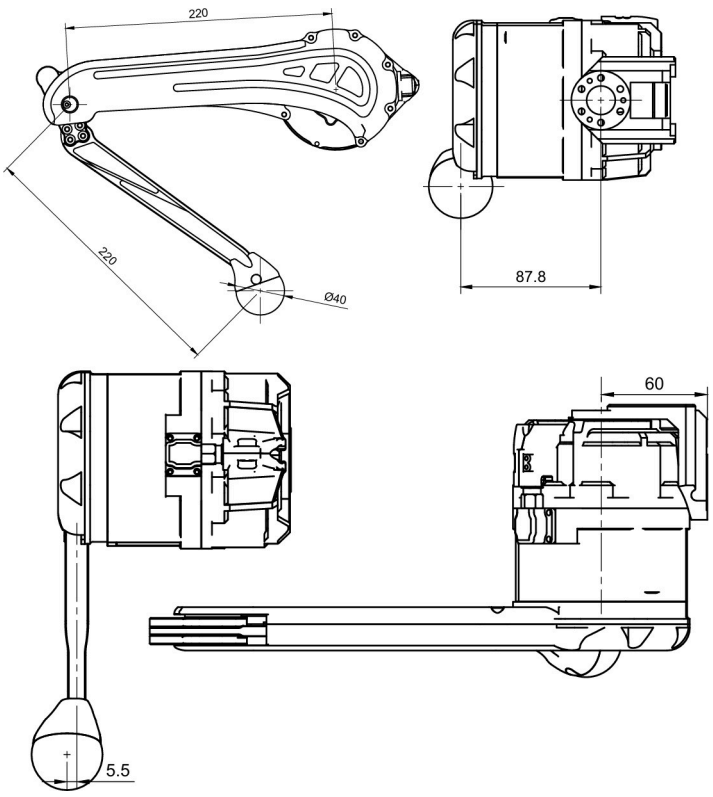
TWO

MECHANICAL

2.1 General dimensions



2.2 Robot Legs



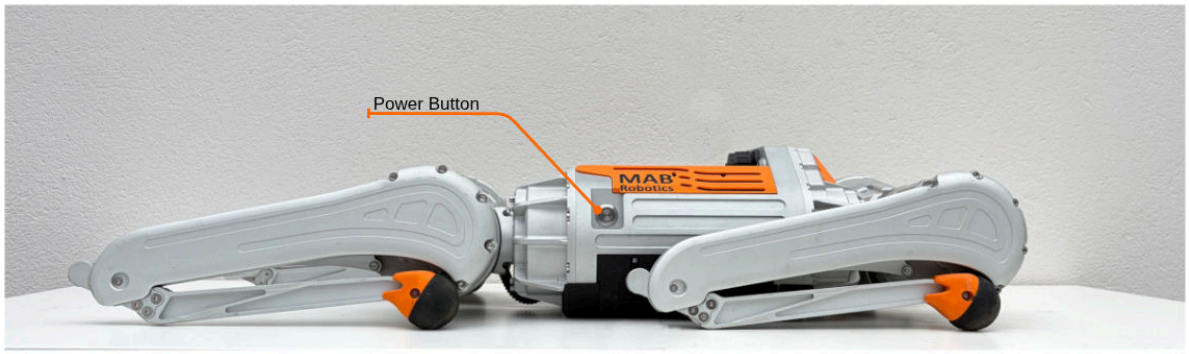
2.2.1 Joint Motion Range

Joint no.	Location	Range
0	Front/Back	150° (90°+60°)
1	Front	180° (90°+90°)
1	Back	270° (90°+180°)
2	Front/Back	140° (90°+50°)

2.3 Connectors

2.3.1 Power & Status button

Power button is located on the right side of the robots' body. It can be used to power on, and off the robot, and displays current status of the robots' internal systems.



Status of the robot

The robot status is indicated by the RGB LED ring around the power button.

Battery charge level

The LED color represents the battery charge level:

LED indication	Battery charge level
Green	More than 60%
Yellow	30–60%
Red	10–30%
Blinking red	Less than 10%

When the robot is off, briefly pressing the power button displays the battery charge level for 10 seconds.

Robot off

When the robot is powered off:

Button action	Robot behavior
Brief press	Displays battery charge level for 10 seconds
Hold for 2 seconds	Starts the power-on sequence. The power button LED blinks purple for about 3 seconds to indicate startup has begun.

Robot on

When the robot is powered on, the power button LED shows the current battery charge level.

Button action	Robot behavior
Hold for 2 seconds	Starts the shutdown sequence

Charging

When the charger is connected:

Robot state	LED indication
Robot off	Slow pulsing between off and the color based on battery charge level
Robot on	Slow pulsing between cyan and the color based on battery charge level

Emergency stop

When the emergency stop is active and the robot leg actuators are disabled electrically, the power button LED blinks between red and yellow.

Do not attempt to operate the robot while the emergency stop is active. Remove the cause of the emergency stop before resetting the system.

Firmware update

When the power button LED blinks in a red-blue (police) pattern, the robot's internal mainboard is in firmware update mode. This state is intended for service or firmware update procedures and should not be active during normal operation.

2.3.2 Connectors

The robot includes several connectors, a power button, and status LEDs. Their location may vary depending on the hardware version.

To plug the cable to the connector:

- match connector and cable orientation by aligning red markers,
- push the cable into the connector, until audible and/or haptic click.

To unplug the connector:

- firmly grab cable connectors body,
- slightly pull the connector housing, internal sliding mechanism will be actuated (refer to schematic),
- pull the cable from the socket - it should come out without significant force requirement.
- for battery connector, always plug battery connector plug immediately after disconnecting from the robot.

Connector number	Function
1	Power Button
2	Charger
3	AUX 1
4	AUX 2 (e-stop)
5	LPC USB 3.0
6	APC (Orin) USB 3.0
7	LAN
8	WAN/LAN
9	WiFi Antenna
10	WiFi Antenna

For the canonical description of the onboard computers (LPC / APC), their roles and network addresses, see *Platform*.

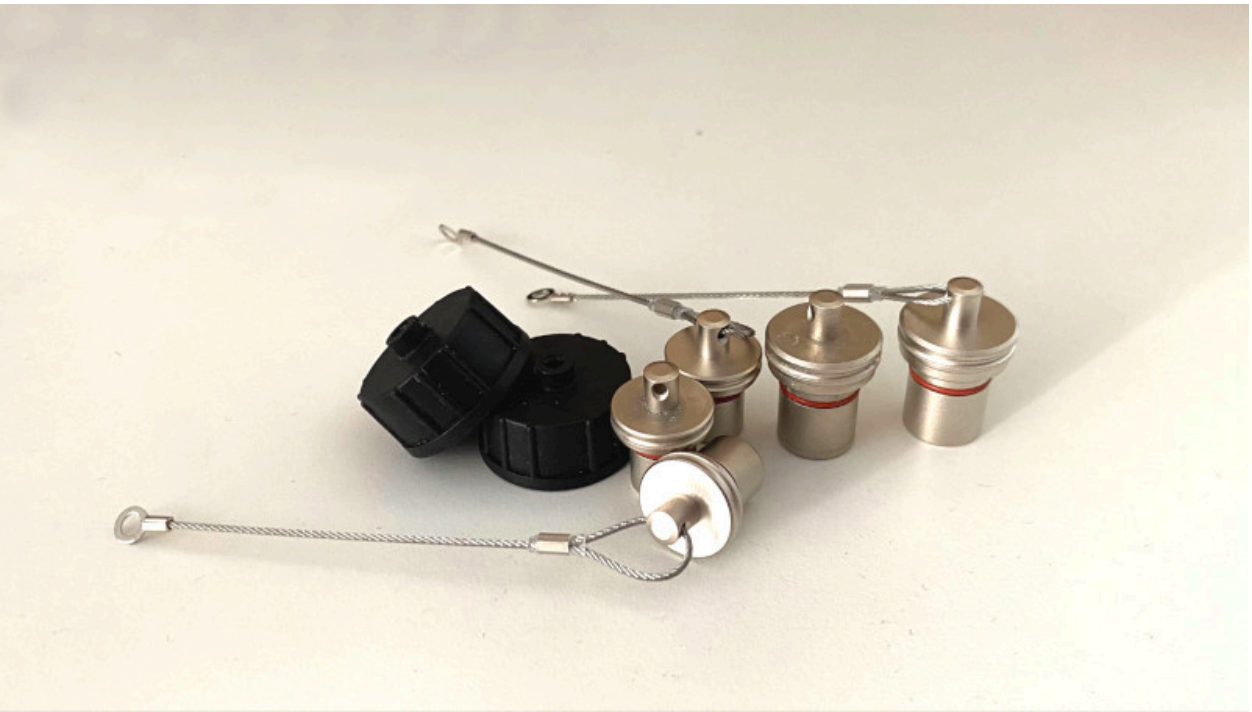
Environmental safety

While all robots connectors were selected and rated for outdoor use, they must remain plugged at all times during operation.

> **Warning**
>
> The ports without plugged connector (or matching plug) are not suitable for use near water, dust, dirt or in high humidity scenarios!

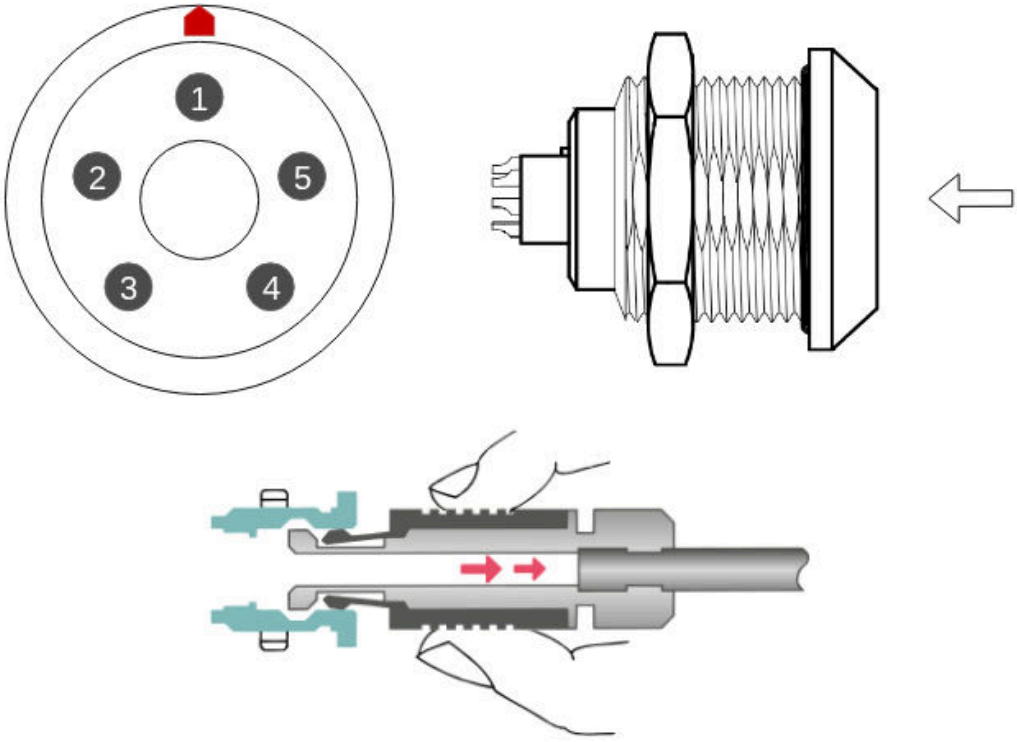
There are 3 types of plugs for the robot:

- 2x plastic USB port cap,
- 2x 12mm plug (LAN, WAN/LAN),
- 3x 10mm plug (AUX1, AUX2, Charger)



AUX1

AUX1 connector hosts payload power connector and dedicated FDCAN lines.



Pinout is as follows:

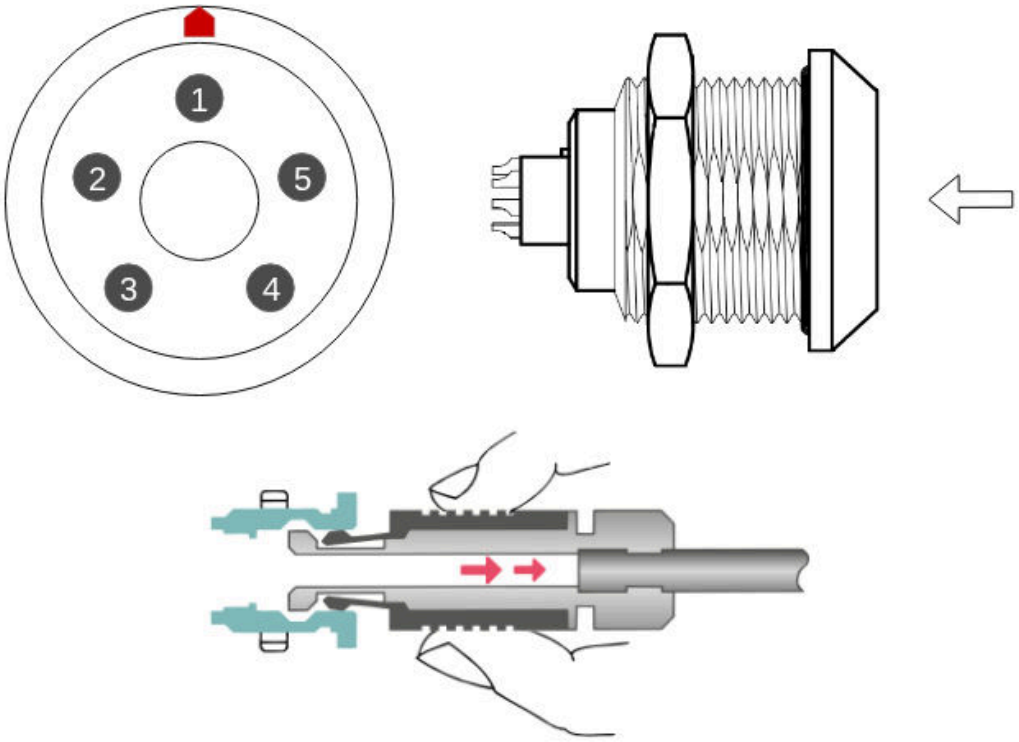
Pin no.	Function
1	Not Connected
2	VCC 36V-50.4V, up to 50W
3	GND
4	FDCAN - CAN L
5	FDCAN - CAN H

AUX2

AUX2 connectors hosts isolated 5V supply, and Emergency Stop inputs.

Pinout is as follows:

Pin no.	Function
1	Not Connected
2	5V (Isolated), up to 10W
3	GND (Isolated)
4	E-stop
5	E-stop

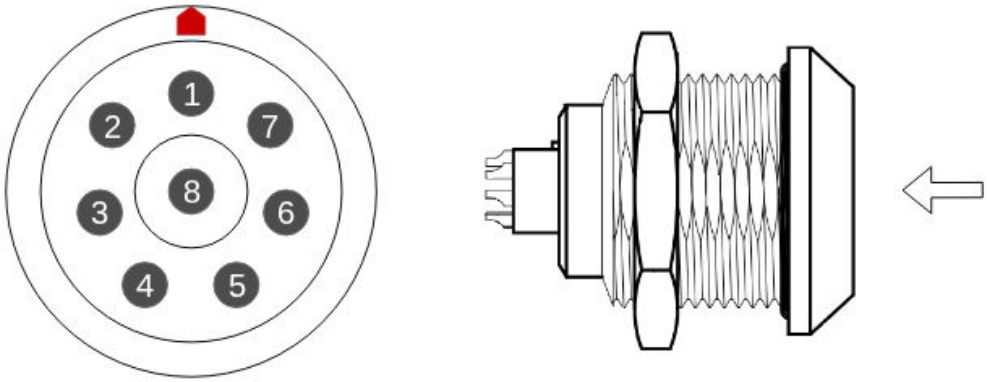


Warning

This connector offers isolated output - its outputs should not be shorted to regular GND (in AUX1) nor to the robots' chassis. Shorting ant of the connections to the batteries GND, may lead to damage to internal electronics.

LAN

The LAN connector (located on the left side of the *rear panel*) provides standard wired Ethernet connectivity.



Pinout is as follows:

Pin no.	Color
1	white-orange
2	orange
3	white-green
4	green
5	white-brown
6	brown
7	white-blue
8	blue

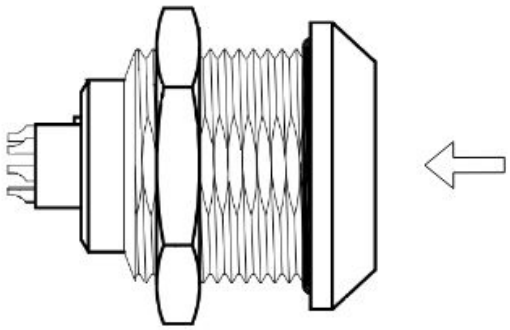
Warning

The socket pinout corresponds to the RJ45 T568B wiring standard.

WAN/LAN

The WAN/LAN connector (located on the right side of the *rear panel*) provides standard wired Ethernet connectivity.





Pinout is as follows:

Pin no.	Color
1	white-orange
2	orange
3	white-green
4	green
5	white-brown
6	brown
7	white-blue
8	blue

Warning

The socket pinout corresponds to the RJ45 T568B wiring standard.

E-stop

If E-stop button is used, it should be normally-closed, pressing the button should disconnect the circuit.

There are two input pins connected to the robots E-stop mechanism. These pins should be shorted to each other during normal operation. The circuit operates in normally-closed manner. The E-stop circuit has direct hardware connection to robots power system, bypassing software for always-ready emergency use.

> **Note**
>
> This is 'hard' E-stop mechanism, using it will immediately cut the power to robots' legs, leading to a fall. This may, lead to unintended damage to the robot and its surrounding. Additionally sudden power cutoff may lead to voltage surges that can damage electronics, including exposed payload power port. **Avoid using E-stop, except in emergency**

Charger port

Charger port is only suitable to work with MAB supplied charger.

CHAPTER

THREE

SOFTWARE OVERVIEW

Note

For the canonical description of the onboard computers (LPC / APC), their roles and network addresses, see the *Platform* chapter.

The robot uses ROS 2 Jazzy (on Ubuntu 24.04 LTS), as its system architecture. There are a few main nodes running on the robot:

- **bridge_node** - node responsible for low-level hardware handling,
- **control_node** - motion and locomotion control
- state_publisher_node - handles TF messages,
- config_node - global parameter server.

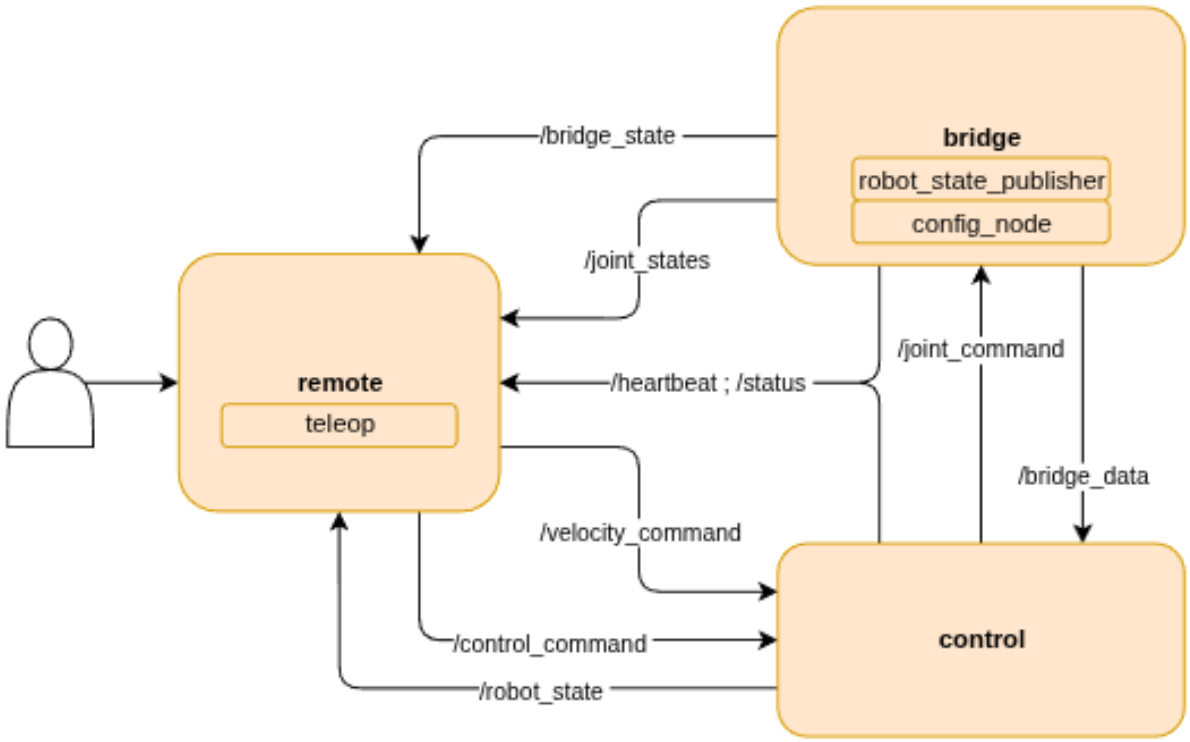
`bridge_node` and `control_node` are ROS 2 lifecycle nodes. In normal operation both are in the `active` state, which can be checked with `ros2 lifecycle get /hb50/control_node`.

Note

All ROS 2 components (nodes, topics, services, etc.) use namespaces that match the robot name. For example, a Honey Badger robot in the default configuration uses the namespace `hb50`, and the robot with wheels uses `hb50w`. The namespace appears as a prefix to all names of the ROS 2 components.

3.1 Topics and Messages

The Honey Badger software stack follows a typical ROS 2 architecture where applicable. Certain nodes and topics should be self-explanatory for users who already worked with ROS-based robots.



3.1.1 Topics

Here is a list of relevant topics published by the robot:

Topic	Message Type	QoS	Publish Rate
/hb50/bridge_data	hb50_commons/msg/BridgeData	mabRT	500 hz
/hb50/bridge_state	hb50_commons/msg/BridgeState	mabRT	500 hz
/hb50/bridge_state_10hz	hb50_commons/msg/BridgeState	Reliable	10 hz
/hb50/control_command	std_msgs/msg/String	Reliable	as needed
/hb50/heartbeat	hb50_commons/msg/Status	Reliable	1 hz
/hb50/imu_data	sensor_msgs/msg/Imu	Reliable	500 Hz
/hb50/joint_command	hb50_commons/msg/JointCommand	mabRT	500 Hz
/hb50/joint_states	sensor_msgs/msg/JointState	Reliable	10 Hz
/hb50/joy	sensor_msgs/msg/Joy	Reliable	as needed
/hb50/robot_description	std_msgs/msg/String	Reliable	as needed
/hb50/robot_state	hb50_commons/msg/RobotState	mabRT	500 Hz
/hb50/robot_state_10hz	hb50_commons/msg/RobotState	Reliable	10 hz
/hb50/status	hb50_commons/msg/Status	Reliable	as needed
/hb50/velocity_command	geometry_msgs/msg/Twist	Reliable	as needed
/hb50/version_data/robot	hb50_commons/msg/VersionData	Reliable, transient local	once at startup
/hb50/version_data/deck	hb50_commons/msg/VersionData	Reliable, transient local	once at startup
/tf	tf2_msgs/msg/TFMessage	Reliable	10 hz
/tf_static	tf2_msgs/msg/TFMessage	Reliable	as needed

/hb50/control_command and /hb50/velocity_command are the high-level command inputs of control_node, see *Control command* and *Velocity command*.

3.1.2 Services

Service	Type	Provider	Description
/hb50/bridge_contr	hb50_commons/srv/EnableFeature	bridge	Hardware commands used by the power commands of the remote (enable, disable, enable_leg_bus, disable_leg_bus, healthcheck, shutdown, ...)
/hb50/maneuver_con	hb50_commons/srv/EnableFeature	control_r	Loading, playing and recording of maneuvers
/hb50/maneuvers_fr	hb50_commons/srv/GetRobotData	bridge	Names of the maneuvers saved on the robot
/hb50/config_save	std_srvs/srv/Trigger	config_nc	Saves the current parameters to the robot configuration file
/hb50/config_resto	std_srvs/srv/Trigger	config_nc	Restores the default configuration

3.1.3 Quality of Service

There are two types of QoS used in the robot:

- mabRT - depth: 1, best_effort, durability: volatile;
- Reliable - depth: 10, reliable, durability: volatile;

`version_data` topics use depth 1, reliable, durability: transient local, so a late subscriber still receives the last message.

3.1.4 Messages

The robot uses a mix of built-in ROS 2 messages and custom types defined in the `hb50_commons` package. For full, canonical field-level documentation of custom messages, see *Messages*.

3.2 Software Platform

The Honey Badger robot has two onboard computers with separate roles:

- **LPC (Locomotion PC)** — UP Squared Pro 7000
- **APC (Application PC)** — Nvidia Jetson Orin Nano Super

This separation helps keep real-time locomotion control isolated from user applications and high-level compute workloads.

3.2.1 LPC (Locomotion PC)

The LPC is the primary real-time control computer for the robot. It typically runs:

- `bridge_node` — low-level interface to the mainboard and actuators
- `control_node` — locomotion and motion control stack
- `state_publisher_node` — ROS 2 transform publisher
- `config_node` — configuration and parameter server

The LPC is responsible for safe, deterministic robot motion and direct hardware handling.

The nodes are started automatically at power-up by the `hb50` systemd service, which runs the `bringup.launch.py` launch file as user `hb`. The output of the current and the previous run is stored in `~/.hb50.log` and `~/.hb50_previous.log`. To restart the robot software, connect over SSH and run:

```
sudo systemctl restart hb50
```

The installed software version is stored in `/var/run/hb/version.json` and published on the `/hb50/version_data/robot` topic.

3.2.2 APC (Application PC)

The APC is the user-facing application computer. It is intended for:

- perception and mapping
- autonomous behavior and planning
- custom user applications
- data logging and analysis

By default, MAB-supplied ROS 2 nodes are not launched on the APC. This keeps the APC available for additional compute workloads without affecting the locomotion stack.

3.2.3 Network roles

Both computers are connected to the robot's internal network:

- `LPC` — static IP `10.11.0.10`, hostname `hb50.lan`
- `APC` — static IP `10.11.0.11`, hostname `hb50-orin.lan`

The robot also provides a Wi-Fi access point for external controllers, such as the Steam Deck.

3.2.4 Why this matters

Separating the locomotion computer from the application computer delivers several benefits:

- improved stability for low-level control
- reduced jitter when running compute-intensive user code
- clearer operational boundaries for users and developers
- easier troubleshooting and system monitoring

For network and connectivity details, see *Networking*.

3.3 Custom message types

This page documents the custom ROS 2 message types used by Honey Badger 5.0. All custom messages are defined in the `hb50_commons` package.

3.3.1 BridgeData — high-rate hardware frame

Message type: `hb50_commons/msg/BridgeData`
Producer: `bridge_node`
Rate: 500 Hz
QoS: mabRT
Use case: Control applications requiring raw actuator data

Contains the high-frequency mainboard update with kinematics and IMU data. This is the primary topic for control loops and state estimation.

Fields

Field	Type	Description
`header`	`std_msgs/Header`	Timestamp and frame ID
`orientation`	`geometry_msgs Quaternion`	IMU orientation (quaternion)
`angular_velocity`	`geometry_msgs Vector3`	Body angular velocity (rad/s)
`linear_accele`	`geometry_msgs Vector3`	Body linear acceleration (m/s^2)
`joint_name`	`string[]`	Names of actuators (e.g., `["fr_j0", "fr_j1", ...]`). HB50W reports 16 actuators: the 12 leg joints followed by the wheels `fr_w`, `fl_w`, `rl_w`, `rr_w`
`joint_posi`	`float32[]`	Current position of each joint (rad)
`joint_velo`	`float32[]`	Current velocity of each joint (rad/s)
`joint_effo`	`float32[]`	Estimated torque of each joint (N·m)
`joint_temp`	`uint8[]`	Temperature of each actuator (°C)
`joint_stat`	`uint16[]`	Status flags for each joint

3.3.2 BridgeState — power and diagnostic state

Message type: `hb50_commons/msg/BridgeState`
Producer: `bridge_node`
Rate: 500 Hz (standard), 10 Hz (`/bridge_state_10hz`)
QoS: mabRT (500 Hz), Reliable (10 Hz)
Use case: Diagnostics, UI, monitoring

Contains power subsystem state, temperatures and hardware status. A low-frequency variant (`/bridge_state_10hz`) is provided for visualization and lossy networks.

Fields

Field	Type	Description
`battery`	`float32`	Battery charge state (0–100 %)
`bus_v`	`float32`	Bus voltage (V)
`bus_i`	`float32`	Bus current (A)
`bus_p`	`float32`	Instantaneous bus power (W)
`bus_pt`	`float32`	Cumulative bus power transfer (kWh)
`legs_v`	`float32`	Leg actuator supply voltage (V)
`charger_v`	`float32`	Charger input voltage (V)
`bridge_t`	`float32`	Bridge board temperature (°C)
`ext1_t`	`float32`	Auxiliary sensor 1 temperature (°C)
`ext2_t`	`float32`	Auxiliary sensor 2 temperature (°C)
`drives_enabled`	`bool`	Whether actuators are energized
`status[]`	`string[]`	Hardware status strings (warnings, errors)
`up_time`	`int32`	System uptime (s)

3.3.3 JointCommand — actuator control commands

Message type: `hb50_commons/msg/JointCommand`
Producer: `control_node`, user nodes
Rate: 500 Hz (as needed)
QoS: mabRT
Use case: Control applications commanding joint motion

Commands the actuators with target position, velocity, torque and PD gains. **Critical:** actuators must be commanded in strict order defined in the robot configuration.

> `bridge_node` subscribes only to `/hb50/joint_command` topic. Custom control programs can publish `JointCommand` messages directly to this topic; if the custom publisher maintains at least 10 Hz, `bridge_node` will prioritize it over the default `control_node` publisher.

Fields

Field	Type	Description
`header`	`std_msgs/Header`	Timestamp and source
`source_node`	`string`	Name of commanding node (e.g., `control_node`)
`name[]`	`string[]`	Actuator identifiers (e.g., `["fr_j0", "fr_j1", ...]`)
`kp[]`	`float32[]`	Position gain for each joint
`ki[]`	`float32[]`	(Reserved) Integral gain
`kd[]`	`float32[]`	Velocity damping gain for each joint
`t_pos[]`	`float32[]`	Target position (rad)
`t_vel[]`	`float32[]`	Target velocity (rad/s)
`t_trq[]`	`float32[]`	Target torque (N·m)
`brake[]`	`bool[]`	Enable brake for each joint

Important: Array lengths must be at least the number of actuators (12, or 16 for HB50W), otherwise `bridge_node` drops the message. Actuators are commanded by array index, not by `name[]`, so the order must correspond to the actuator ordering in `BridgeData`.

3.3.4 RobotState — estimated state

Message type: `hb50_commons/msg/RobotState`
Producer: `control_node`
Rate: 500 Hz (standard), 10 Hz (`/robot_state_10hz`)
QoS: mabRT (500 Hz), Reliable (10 Hz)
Use case: State machine, planning, visualization

Contains state estimation outputs, body kinematics and dynamics. A low-frequency variant (`/robot_state_10hz`) is provided for visualization and UI applications.

Fields

Field	Type	Description
header	std_msgs/Header	Timestamp and frame ID
leg[]	LegState[]	Array of leg states (one entry per leg)
pose	geometry_msgs/Pose	Estimated body pose (position and orientation)
body_vel	geometry_msgs/Twist	Body velocity in body frame (m/s, rad/s)
world_vel	geometry_msgs/Twist	Body velocity in world frame (m/s, rad/s)
motion_mode	byte	Current motion mode (enum)
motion_type	byte	Motion type classification (enum)
motion_gait	byte	Current gait (enum)

Enums

motion_mode:

Value	Name
0	preop
1	joint
2	active
3	wheels (HB50W)

motion_type:

Value	Name
0	idle
1	stand
2	walk
3	safety
4	maneuver
5	wheels (HB50W)

motion_gait:

Value	Name
0	idle
1	stand
2	trott (WALK)
8	crawl (STABLE WALK)

joint mode is used for lying, standing up and maneuvers, active mode for balancing and walking (STAND, WALK, STABLE WALK).

3.3.5 LegState — per-leg status

Message type: `hb50_commons/msg/LegState` **Nested in:** `RobotState` **Use case:** Leg-specific diagnostics and control

Describes the estimated state of a single leg, including foot contact and forces.

Fields

Field	Type	Description
`leg_name`	`string`	Leg identifier (e.g., `"FR"`, `"RL"`)
`contact`	`bool`	Whether foot is in contact with ground
`foot_pos_body`	`geometry_msgs/Vector3`	Foot position in body frame (m)
`foot_vel`	`geometry_msgs/Vector3`	Foot velocity (m/s)
`foot_force_est`	`geometry_msgs/Vector3`	Estimated ground reaction force (N)

3.3.6 Status — event and status messages

Message type: `hb50_commons/msg/Status` **Producers:** All nodes (by heartbeat sub-node), `bridge_node` (hardware events), `control_node` (control events) **Rate:** 1 Hz (heartbeat), as needed (events) **QoS:** Reliable **Use case:** Monitoring, logging, alarm handling

General-purpose status message for reporting important events, warnings and errors.

Fields

Field	Type	Description
`header`	`std_msgs/Header`	Timestamp and source frame
`level`	`byte`	Severity level: `OK=1`, `WARN=2`, `ERROR=3`, `STALE=4`
`reporting_node`	`string`	Name of the node sending the status
`status`	`string`	Human-readable message
`data[]`	`byte[]`	Optional structured data (node-specific)

Level enum

- `OK=1` — normal operation
- `WARN=2` — warning condition
- `ERROR=3` — error condition
- `STALE=4` — stale/timeout condition

All nodes should publish a heartbeat Status message with `level=OK` at least every 1 second. Missing heartbeats indicate communication loss or node crash.

3.3.7 VersionData — software version

Message type: `hb50_commons/msg/VersionData`
Producers: `bridge_node` (`/hb50/version_data/robot`), `remote_node` (`/hb50/version_data/deck`)
Rate: once at startup
QoS: Reliable, transient local
Use case: Checking the software version of the robot and the Steam Deck

Field	Type	Description
`version_major`	`uint8`	Major version
`version_minor`	`uint8`	Minor version
`version_rev`	`uint8`	Revision
`version_tag`	`string`	Version tag
`git_hash`	`string`	Git commit of the build
`git_branch`	`string`	Git branch of the build
`build_date`	`uint32`	Build date
`robot_uid`	`string`	Unique device ID (`/etc/machine-id`)

```
ros2 topic echo --once /hb50/version_data/robot
```

3.3.8 Control command

`/hb50/control_command` (`std_msgs/msg/String`, Reliable QoS) changes the robot motion mode. It is the topic used by the control commands of the *HB50 Remote* application.

Command	Remote button	Description
`idle`	`IDLE`	Stops all motion and disables the actuators
`enable`	`ENERGIZE`	Enables the actuators
`stand_up`	`STAND UP`	Stands the robot up, requires enabled actuators
`stand`	`STAND`	Standing with active balance
`walk`	`WALK`	Walking gait (`trott`)
`stable`	`STABLE WALK`	Stable walking gait (`crawl`)
`wheel_ctrl`	`WHEELS`	Driving on wheels, HB50W only, accepted directly after `stand_up`
`adaptive_enable` / `adaptive_disable`	Adaptive locomotion toggle	Turns the *adaptive locomotion mode* on or off

`stand`, `walk` and `stable` are accepted only after `stand_up`. The result is reported on `/hb50/status`.

```
ros2 topic pub --once /hb50/control_command std_msgs/msg/String "{data: stand_up}"
```

3.3.9 Velocity command interpretation

The standard ROS geometry message `/hb50/velocity_command` (`geometry_msgs/msg/Twist`, Reliable QoS) is used for high-level gait commands. Components are interpreted as:

Linear velocity

- `x` — forward velocity (normalized to `[-1, 1]`)
- `y` — lateral velocity (normalized to `[-1, 1]`)
- `z` — reference walking height (normalized to `1`)

Angular velocity

- `x` — roll command (normalized to `[-1, 1]`)
- `y` — pitch command (normalized to `[-1, 1]`)
- `z` — yaw rate (normalized to `[-1, 1]`)

Normalization: All normalized commands are scaled by configuration parameters (`cmd_vel.lin`, `cmd_vel.ang`) to produce physical values in SI units (m/s, rad/s).

Timeout: The command must be published continuously (the Steam Deck publishes at 10 Hz). If no command arrives for 250 ms, `control_node` sets the velocities to zero and the height to the default.

3.3.10 Message usage guide

For control loop development

- **Subscribe to:** `/hb50/bridge_data` (500 Hz) for raw actuator feedback
- **Publish to:** `/hb50/joint_command` (500 Hz) for actuator commands
- **Monitor:** `/hb50/robot_state` (500 Hz) for state estimation
- **Check:** `/hb50/heartbeat` (1 Hz) to detect stale nodes

For UI and visualization

- **Subscribe to:** `/hb50/robot_state_10hz` and `/hb50/bridge_state_10hz` (10 Hz, Reliable QoS)
- **Monitor:** `/hb50/status` for warnings and errors
- **Listen to:** `/tf` and `/tf_static` for frame transforms

For diagnostics and logging

- **Monitor:** `/hb50/bridge_state` for power and temperature trends
- **Watch:** `/hb50/status` and `/hb50/heartbeat` for node health
- **Check:** `joint_temperature` and `joint_status` in `/hb50/bridge_data` for actuator issues

3.4 Network

The robot is equipped with an on-board Wi-Fi router, allowing for easy access, *software updates*, and management of multiple connected devices.

By default, the robot is configured with an independent Wi-Fi access point. SSIDs follow a pattern:

- **hb50_XXXX** — for the 2.4 GHz network,
- **hb50_XXXX_5G** — for the 5 GHz network.

Where XXXX is robot serial code shortened to 4 hex symbols - e.g. "hb50_AFAB" or "hb50_D71C_5G". Default password to the network is: `milkaorzechowa`.

The router also hosts DNS server, allowing for easy SSH connections.

Note

Due to ROS 2 constraints, when a node (such as `remote_node`) is launched *before* the network connection is established (IP address resolved), it usually fails to properly discover and detect other ROS 2 nodes, topics, and services.

Before starting any node that will interoperate with Honey Badger, make sure you are connected to the correct network.

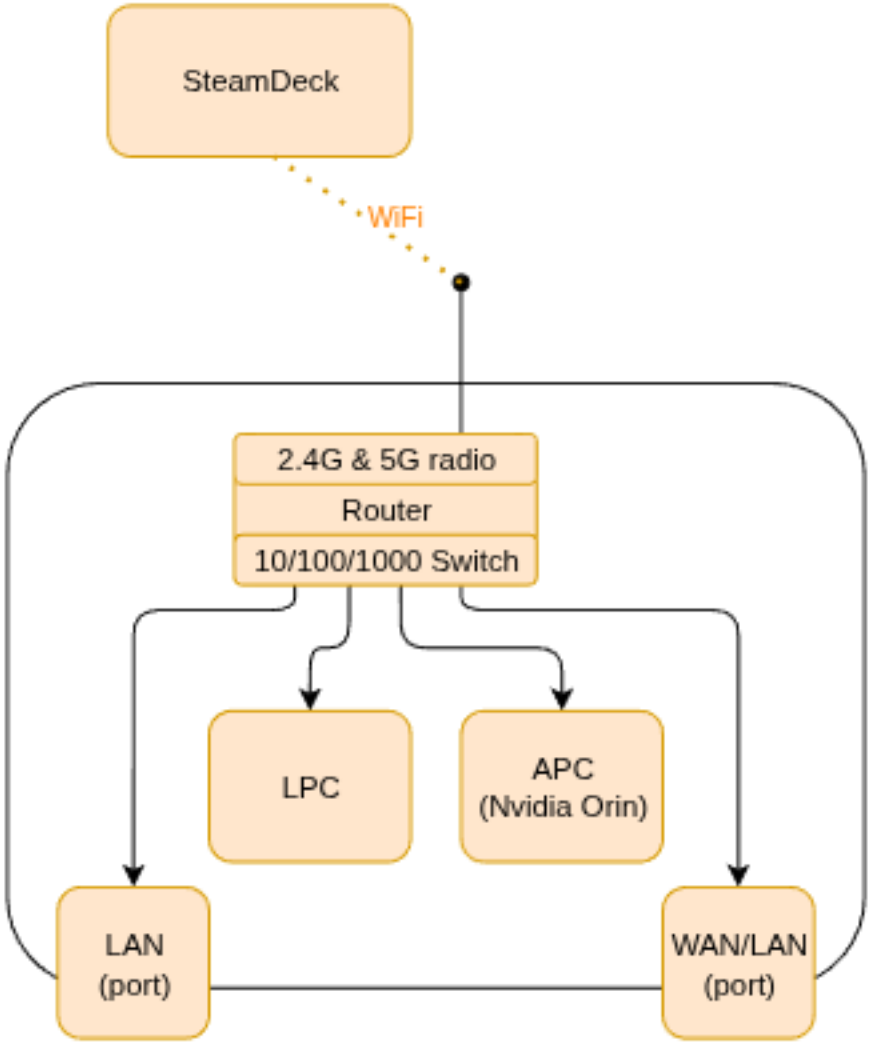
3.4.1 ROS 2 and SSH

Both of the robot's onboard computers can be accessed via ROS 2 or directly with SSH.

SSH

The robot's DNS server uses 10.11.0.1 as the base address (also the router address), and both LPC and APC have static IP addresses:

Device	Static Address	Hostname	Username
LPC	10.11.0.10	hb50.lan	hb
APC (Orin)	10.11.0.11	hb50-orin.lan	hb-orin
Steam Deck	Dynamic IP	deck.lan	mab


SteamDeck
WiFi
2.4G & 5G radio
Router
10/100/1000 Switch
LPC
APC
(Nvidia Orin)
LAN
(port)
WAN/LAN
(port)

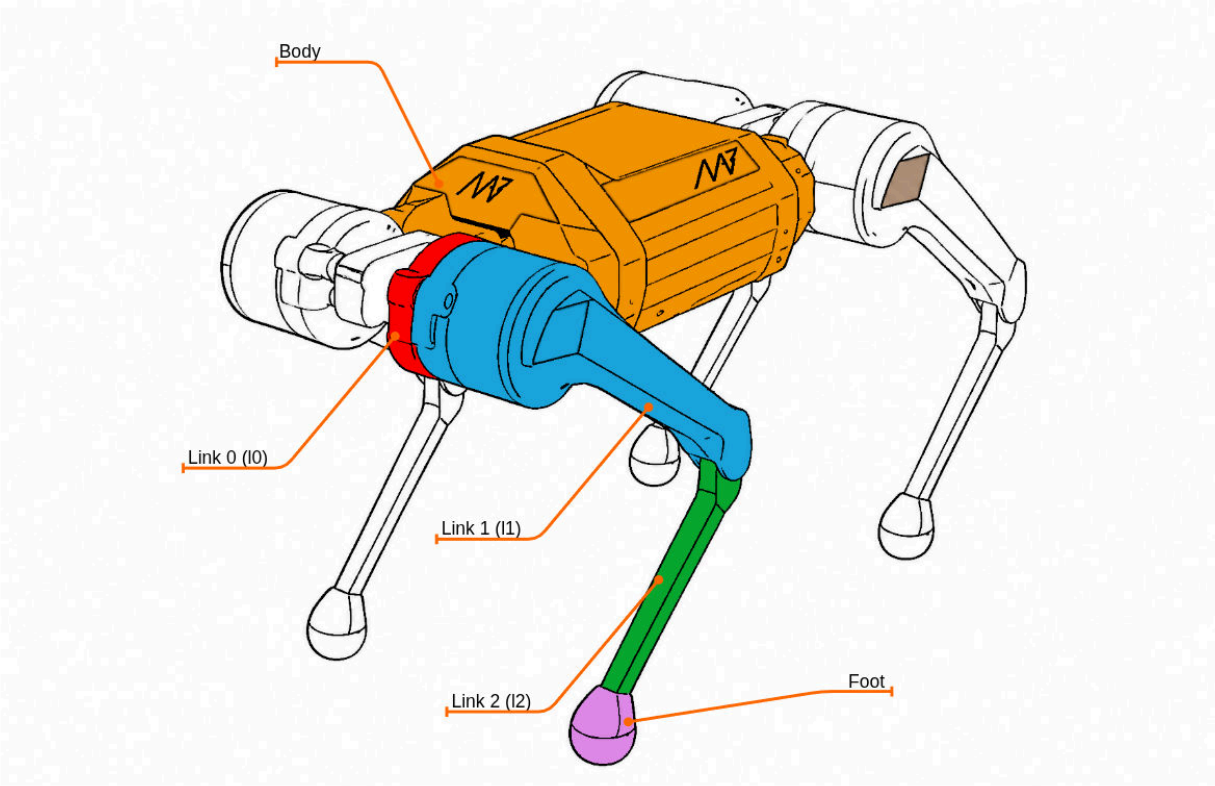
ROS 2

By default, the robot uses `ROS_DOMAIN_ID` 69, so the robot's topics should be visible shortly after power-up. Ensure your client uses the same `ROS_DOMAIN_ID` when interacting with the robot.

3.5 Geometry and Frames

3.5.1 Robot structure

The main robot parts are illustrated below. Use the naming and frame conventions from this page consistently in logs, diagnostics and documentation.



Leg naming

Abbreviation	Meaning
`FR`	Front-right leg
`FL`	Front-left leg
`RL`	Rear-left leg
`RR`	Rear-right leg

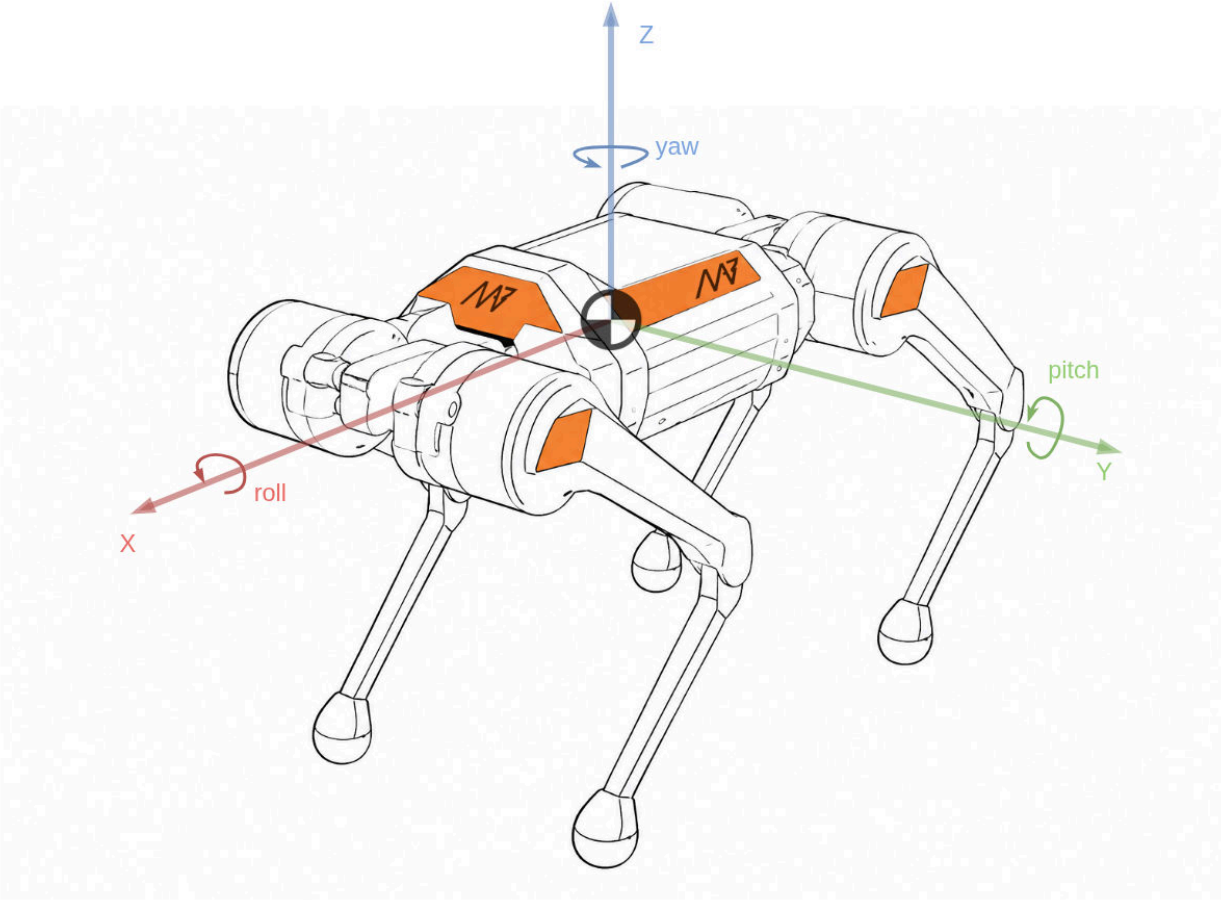
Local coordinate frame

The robot uses a right-hand coordinate system with origin at the chassis center (approximately the center of mass). Axes and rotations are defined as:

- X — forward (positive toward the robot front)
- Y — left (positive toward the robot left)
- Z — up (positive opposite gravity)

Rotations:

- Roll — rotation around X
- Pitch — rotation around Y
- Yaw — rotation around Z



Use these conventions when interpreting TFs, URDF links and joint commands.

3.6 Software Update

The robot software can be updated in the field using a signed update bundle supplied by MAB. The update is started from the *Steam Deck controller* and updates both the robot and the Steam Deck. It does not require rebuilding anything.

> **Note**
>
> This chapter covers the robot's ROS 2 software stack running on the LPC and the `HB50 Remote` application on the Steam Deck. It is not related to the mainboard firmware update mode described in *Button & Connectors*.

3.6.1 Update bundle

Updates are delivered as a single file named `hb50-<target>-v<version>.tar`, for example `hb50-all-v1.2.0.tar`. Target `all` means the bundle is valid for both the robot and the Steam Deck. The file can be sent by email, downloaded or copied from a USB stick — do not unpack or rename it.

Every bundle is signed by MAB. Before installing, both the Steam Deck and the robot verify:

- the signature against the MAB vendor key built into the software,
- the integrity (SHA-256) of the software package inside the bundle,
- that the bundle is intended for the device (target `robot`, `steamdeck` or `all`),
- that the bundle version is newer than the installed one.

A bundle that fails any of the first three checks is rejected and nothing is changed.

> **Note**
>
> Robots and Steam Decks running software older than v1.2.0 must be updated once with the *first update* package before they can use the procedure below.

3.6.2 Updating the robot

1. Copy the bundle to the Steam Deck

Copy the bundle into the `updates` folder in the home directory of the Steam Deck (`/home/mab/updates/`), for example from a USB stick in the Steam Deck desktop mode, or from a laptop connected to the robot's network (see *Network*):

```
scp hb50-all-v1.2.0.tar mab@deck.lan:~/updates/
```

If there are several bundles in the folder, the one with the newest version is used.

2. Put the robot in a safe state

Connect the Steam Deck to the robot's network. Updates are only accepted when the robot is in the `idle` motion mode. Lay the robot down in a stable position and make sure it stays powered on for the whole procedure (connect the charger if the battery is low).

3. Check the update

In the `HB50 Remote` application, use the *Command Switch* to go to the Power section and press `CHECK SOFTWARE UPDATE`.

The Steam Deck verifies the bundle, copies it to the robot (`/home/hb/updates/`) and asks the robot to verify it as well. The result is shown in the status console, with the `Remote:` part for the Steam Deck and the `Robot:` part for the robot:

- white (OK) — the bundle is verified and ready to install, the console shows its version and git revision,
- yellow (warning) — the bundle is valid, but it is not newer than the installed version; applying it will reinstall or downgrade the software,
- red (error) — the bundle was rejected, see *Status codes*.

4. Apply the update

Press `APPLY SOFTWARE UPDATE`. This button only works after a successful check.

The robot is updated first:

1. it unpacks the new software next to the current one (`/home/hb/install-<version>-<id>/`),
2. stops the robot software (`hb50` service),
3. switches `/home/hb/install` to the new version,
4. starts the robot software again and waits up to 60 s for the first control cycle, which shows that the configuration, the hardware communication and the controller all work.

If the new software starts correctly, the bundle is removed from `/home/hb/updates/` and older software versions are deleted, keeping only the current and the previous one. If it does not start, the robot automatically rolls back to the previous version.

The Steam Deck waits up to 3 minutes for the robot to come back with the new version and then installs the same version on itself, in the same way. This takes about 30 seconds in total.

> **Warning**
>
> The robot software restarts during the update, so the Steam Deck loses connection to the robot for a moment. Do not power off the robot or close the `HB50 Remote` application until the update has finished.

5. Restart the application

When the status console shows `restart the app to finish the update`, close the `HB50 Remote` application and start it again.

If the console shows `Robot did not come back with v<version> - remote was NOT updated, check the robot`, the robot update failed and the Steam Deck was not changed. The updater log can be checked on the robot with:

```
journalctl -u hb50-updater
```

To confirm the installed version, connect over SSH to the robot or the Steam Deck and run:

```
cat ~/install/.hb50_version
```

3.6.3 Status codes

Status codes returned by `CHECK SOFTWARE UPDATE` (the Steam Deck reports only the message, in the `Remote:` part):

Code	Name	Meaning
0	`OK`	Bundle verified and ready to install
1	`ERR_BUNDLE_NOT_FOUND`	No `hb50-<target>-v<version>.tar` file for the device in `~/updates/`
2	`ERR_EXTRACT_FAILED`	The file is not a valid bundle
3	`ERR_MISSING_FILES`	The bundle is incomplete or corrupted
4	`ERR_INVALID_SIGNATURE`	The bundle is not signed by MAB — refused
5	`ERR_MANIFEST_PARSE`	The bundle metadata could not be read
6	`ERR_WRONG_TARGET`	The bundle is not intended for the device
7	`ERR_PAYLOAD_HASH_MISMAT`	The software package was modified or damaged — refused
8	`WARN_NOT_NEWER_VERSION`	The bundle is not newer than the installed version
9	`ERR_ROBOT_NOT_IDLE`	The robot is not in `idle` motion mode

If there is no bundle on the Steam Deck, only the robot is checked, using a bundle already present in `/home/hb/updates/` on the robot. The Steam Deck is then not updated.

If the console shows `Could not copy <bundle> to the robot`, the Steam Deck could not reach the robot over SSH. Check the network connection. The Steam Deck logs in to the robot with an SSH key set up by the *first update*; if the key is missing, run `ssh-copy-id hb@10.11.0.10` on the Steam Deck.

Status codes returned by `APPLY SOFTWARE UPDATE`:

Code	Name	Meaning
0	`OK`	Update handed over to the updater, the robot software will restart
1	`ERR_NO_VERIFIED_UPDATE`	No successful check was run before applying
2	`ERR_UNPACK_FAILED`	The new software could not be unpacked (e.g. no disk space)
3	`ERR_DEPLOY_FAILED`	The updater could not be started
4	`ERR_ROBOT_NOT_IDLE`	The robot is not in `idle` motion mode

3.6.4 First update

The first update uses the package `hb50-first-update-v<version>.tar` supplied by MAB, which contains the update bundle and the scripts that install it on both devices.

1. Unpack the package on the Steam Deck.
2. Put the robot in `idle` mode, connect the Steam Deck to the robot's network and close the `HB50 Remote` application.
3. Open a terminal on the Steam Deck in the unpacked folder and run:

   ```
   ./first_update.sh hb50-all-v<version>.tar
   ```

   If the robot does not use the default address `10.11.0.10`, add it at the end, for example `./first_update.sh hb50-all-v<version>.tar hb@<robot-ip>`.
4. Type the robot password when asked. You may be asked twice: once to set up the SSH key, once to install the software. If the script asks for the robot name, type the model: `hb40`, `hb50`, or `hb50w` for the robot with wheels.
5. Wait until the script prints `=== Done: robot and remote run v<version>. Start the HB50 Remote app again.` and start the `HB50 Remote` application.

The script can be run again safely: the robot or the Steam Deck is skipped if it already runs this version. If the robot is not reachable, only the Steam Deck is updated and the script ends with `robot was NOT updated` — connect the robot and run the same command again. If something fails, the script stops and prints the reason; if it stops before `Stopping hb50...`, nothing on the robot was changed.

3.6.5 ROS 2 services

The robot update can also be triggered without the Steam Deck, using the services provided by `control_node`. This updates only the robot, from a bundle copied to `/home/hb/updates/` on the robot:

Service	Type	Description
`/hb50/check_for_upd`	`hb50_commons/srv/CheckForUpdate`	Verifies a bundle. `bundle_path` selects the bundle file or the directory to search (empty — `/home/hb/updates/`)
`/hb50/apply_update`	`hb50_commons/srv/ApplyUpdate`	Installs the bundle verified by the last check

```
ros2 service call /hb50/check_for_update hb50_commons/srv/CheckForUpdate "{bundle_
↪path: ''}"
ros2 service call /hb50/apply_update hb50_commons/srv/ApplyUpdate
```

3.6.6 Manual rollback

To go back to the previous version on the robot, connect over SSH and run:

```
ls -d ~/install-*                          # list installed versions
sudo systemctl stop hb50
ln -sfn ~/install-<previous_version> ~/install
sudo systemctl start hb50
```

On the Steam Deck, run the same `ls` and `ln` commands, then restart the `HB50 Remote` application. Keep the robot and the Steam Deck on the same version.

CHAPTER

FOUR

PAYLOAD

Configuration of the Honey Badger robot can be extended by mounting additional payload to its back. It allows user for upgrading Honey Badger capabilities.

4.1 Payload mount

Payload can be attached on the robots' back, using mounting points:

For proper robot operations, the payload mass should be evenly distributed on the robot, along X and Y axes. And constrained by the approximate payload area.

Note

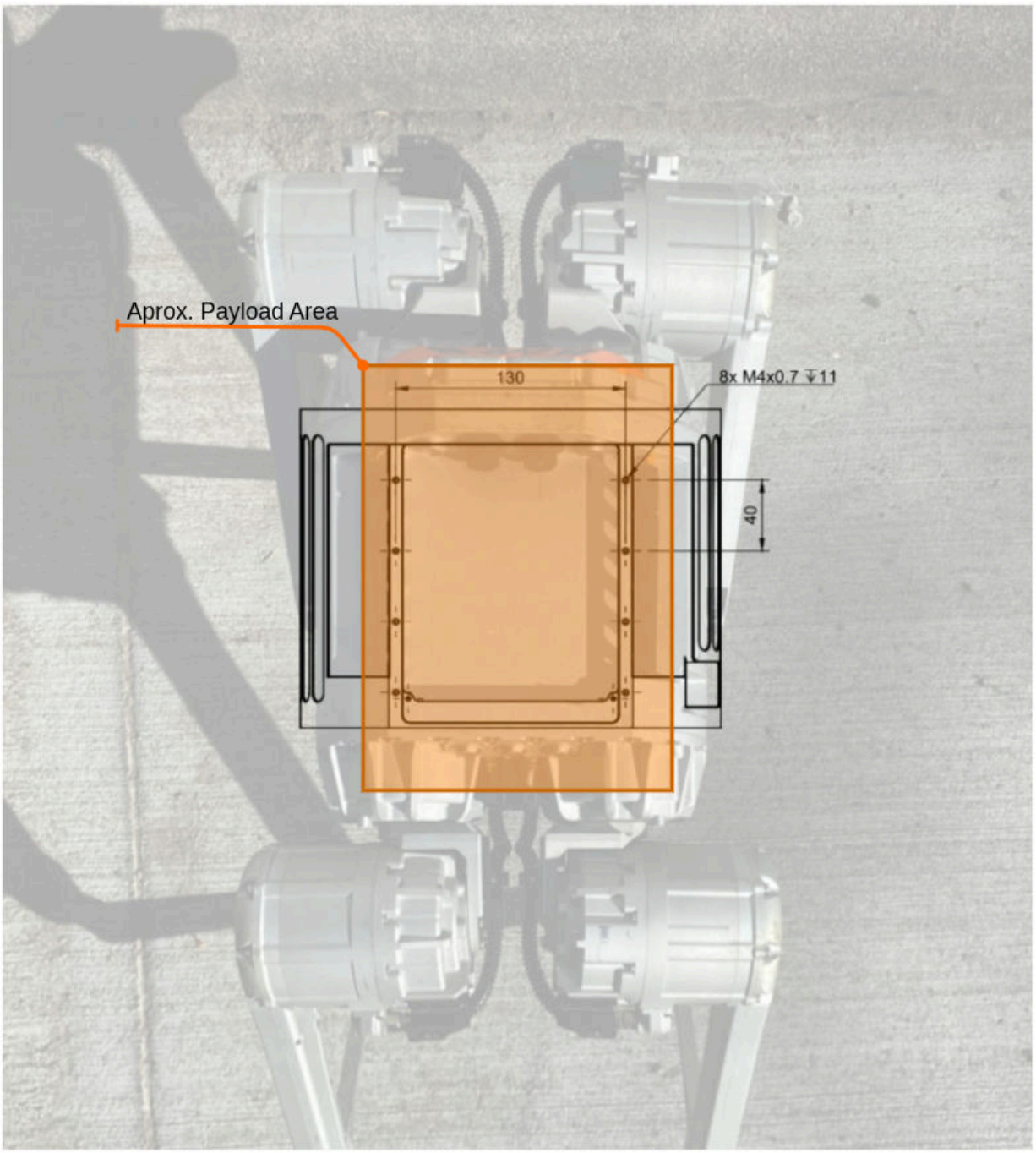
In some scenarios, mainly during recovery procedures (flipping from the back), legs can reach over the nominal payload area. When designing payloads that reach or extend beyond boundaries of approximate payload area, contact MAB team to avoid collisions between legs and payload.

4.2 Spool

The Honey Badger robot can be equipped with an optical-fiber spool that allows the operator to establish a wired connection to the robot. This solution is useful in long-range environments where wireless communication is impossible due to interference or limited Wi-Fi range.

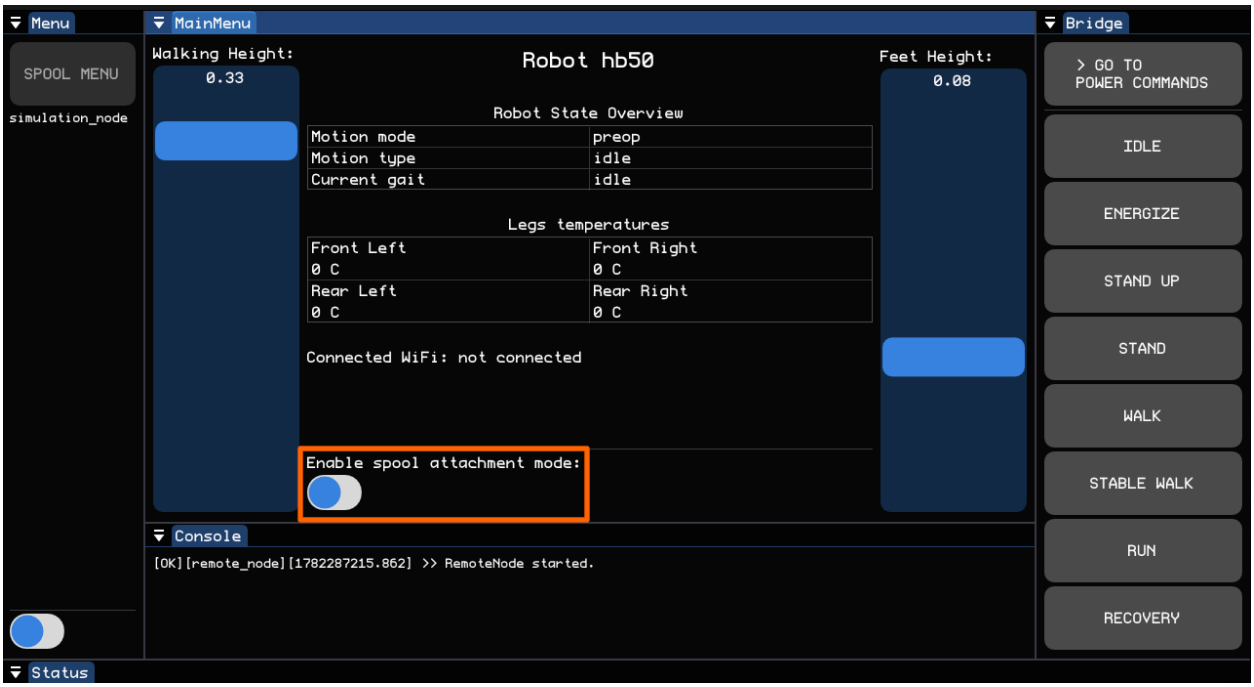
4.2.1 Setup

Communication with the spool driver is established through the ROS 2 `spool_node`. Powering up the platform initializes the spool driver, which in turn automatically starts the node.


Aprox. Payload Area
130
8x M4x0.7 ↧11
40

4.2.2 Spool remote controller interface

> **Warning**
>
> The spool attachment is disabled by default. Always verify the spool is enabled before operating the robot with the spool attached.
>
> If the spool is not enabled and the robot moves, the spool or fiber can be damaged. Enabling the spool also updates the robot motion controller with the spool mass and inertial parameters.
>
> Make sure the `Enable spool attachment mode` toggle shown below is active before using spool functions.



Spool control has a separate interface built into the remote controller. It allows the operator to handle the spool safely. To begin, the operator should press the `SPOOL MENU` button (number 1) to open the spool control view.

Number	Name	Function
1	`SPOOL MENU`	Opens menu containing whole functionality of spool.
2	`ENABLE SPOOL`	Enables spool motors.
3	`DISABLE SPOOL`	Disables spool motors.
4	`ZERO SPOOL`	Sets the current position of unwound optical fiber as the zero point.
5	`Operating Mode Switch`	Enables/Disables manual mode.
6	`-/+ Direction Control`	− stands for winding spool and + for unwinding.
7	`-/+ Velocity Control`	− decreases linear velocity of winding by -0.1 [m/s] and + for increasing linear velocity of winding by 0.1 [m/s].

Operating mode

The spool platform can operate in two separate modes: automatic and manual.

Automatic mode

Automatic mode is used for synchronous operation with the Honey Badger robot. In this mode, the spool platform receives velocity values from the *hb50_commons/msg/RobotState* topic, which publishes the robot's estimated state and actual linear velocity.

Manual mode

Manual mode allows the operator to directly add a velocity command to the spool while the robot is operating. The manual command is added to the automatic robot velocity, so the resulting spool velocity is the sum of both contributions.

To enable manual mode with the remote controller, the user should unlock the safety switch (*button number 5*).

By default, the system is operated using the remote controller. However, if the operator prefers manual mode via the terminal, they should publish a safe velocity to the `/hb50/payload/spool_unroll_speed` topic.

Example of use:

```
ros2 topic pub -r 10 /hb50/payload/spool_unroll_speed std_msgs/msg/Float32 "{data: 0.
↪1}"
```

The command unwinds the spool at a linear velocity of 0.1 m/s in addition to the current robot velocity. The manual command must be published continuously — if no command arrives for 250 ms, the manual velocity is set to zero.

> **Warning**
>
> Passing a negative float value winds the optical fiber. Always ensure there is enough spare fiber available before winding.
>
> Do not allow the fiber to become taut or caught on obstacles while the spool is active.

The spool velocity from automatic mode and the manual velocity command are summed together. The total spool velocity is limited to a maximum of 1 m/s, so if the combined robot and manual velocities exceed 1 m/s, the spool speed is capped at 1 m/s.

Spool topics

`spool_node` subscribes to the following topics:

Topic	Message Type	QoS	Publish Rate
`/hb50/payload/spool_unroll_speed`	`std_msgs/msg/Float32`	Reliable	10 hz
`/hb50/robot_state_10hz`	*`hb50_commons/msg/RobotState`*	Reliable	10 hz

If `/hb50/robot_state_10hz` is not received for 1 s while the spool is enabled, the automatic velocity is set to zero and an error is reported on `/hb50/status`.

And publishes to the topics:

Topic	Message Type	QoS	Publish Rate
`/hb50/payload/spool_unroll_length`	`std_msgs/msg/Float32`	Reliable	10 hz
`/hb50/torque`	`std_msgs/msg/Float32`	Reliable	10 hz

> **Note**
>
> For precise definition of QoS check: *Quality of Service*.

Spool services

Spool uses a single service to enable different features.

Service	Interface Type	QoS
`/hb50/payload/spool_control`	`hb50_commons::srv::EnableFeature`	Reliable

The interface type defined as `hb50_commons::srv::EnableFeature` has the following parameters:

Type	Name of parameter	Function
`string`	`feature`	Describes function that should be applied.
`bool`	`state`	Represents logic value of 0 or 1 that stands for disabling or enabling feature.
`uint8[]`	`data`	Additional data, not used by the spool.
`bool`	`success`	Informs about result of applied service.

The operator has two feature names available for calibration or safety control. The argument is passed as a `string` and can be one of the following:

`feature` name	Function
`zero_length`	Sets the current position of unwound optical fiber as the zero point.
`enable_motors`	If `1` is passed to the `state`, enables motors and allows them to be ready to work; if `0` is passed, disables motors.

CHAPTER

FIVE

FAQ

This page collects common questions and answers in a simple, scannable format. Each question is shown as a heading, and the answer is placed in a collapsible section so readers can expand only the topics they need.

5.1 Getting started

5.1.1 What if after the startup robot doesn't stand up straight?

Answer

The issue is caused by edge case behavior. While we are working on a fix, please follow this workaround:

In the remote, use *Command Switch* to get into the Power section, then:

- click `DISABLE BUS`
- set up the robot in the starting *position lying down*, like on the technical drawings
- click `ENABLE BUS`
- click `HEALTH CHECK`

Then go back to the Control Section and follow the normal starting routine.

If the error is still there, redo the procedure, but instead of laying the robot flat, rotate the problematic joint by 10 degrees in any direction.