

Dharma

A robot-agnostic ROS 2 autonomy stack

Technical manual, version 0.2.0

I Wayan Agus Darmawan

Department of Computer Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia
PT Ezra Robotics Teknologi, Indonesia
darmawandevloper@gmail.com

Architecture, models, configuration, interfaces, the safety case and the development guide of the 0.2.0 stack.

Contents

1	Introduction	5
1.1	What changed in 0.2.0	5
1.2	Conventions of this manual	5
1.3	Repositories	6
2	Architecture	7
2.1	Layers	7
2.2	Data flow	8
2.3	Frames	11
2.4	Design principles	12
2.5	Launch composition	12
3	Configuration and robot profile	13
3.1	The robot profile	13
3.1.1	Derived quantities	14
3.1.2	Measuring the limits	14
3.2	Typed parameters	15
3.3	Parameter files and launch	16
4	Odometry	17
4.1	Estimators	17
4.2	Start gate (<code>wait_still</code>)	17
4.3	Monitor	18
4.4	Parameters	18
4.5	Launch	20
5	Localization	21
5.1	Gravity-aligned bridge	21
5.2	Estimator	21
5.2.1	Why	21
5.2.2	State and prediction	22
5.2.3	Update and gate	22
5.2.4	Sources	23
5.2.5	Publication without interpolation ramps	23
5.3	Fiducial tags	23
5.4	Interfaces	24
5.5	Parameters	24

6	Perception	27
6.1	Sector scan semantics	27
6.2	Pipeline	27
6.3	Limits	28
6.4	Nodes and parameters	28
7	Mapping and sites	31
7.1	Sites	31
7.2	Occupancy grid	31
7.2.1	Log-odds model	31
7.2.2	Which scans and points are integrated	31
7.2.3	Mapping sessions	32
7.3	Route teaching	32
7.4	Interfaces	33
7.5	Parameters	33
8	Navigation	36
8.1	One braking model	36
8.2	Planner	36
8.2.1	Costmap	36
8.2.2	Search	37
8.2.3	Post-processing and the footprint check	37
8.2.4	Goal session	37
8.3	Path follower	37
8.3.1	Route, start and progress	37
8.3.2	Speed profile	38
8.3.3	Controllers	38
8.3.4	Path shift	39
8.3.5	Reactive avoidance	39
8.3.6	Stop points	40
8.3.7	Supervisor, motion guard and command shaper	40
8.3.8	Pause reasons	41
8.4	Interfaces	41
8.5	Parameters	41
8.5.1	Controller parameters	46
9	Control and safety	48
9.1	Two-channel safety architecture	48
9.2	Pipeline	49
9.2.1	Arbitration	49
9.2.2	E-stop latch and re-arm	49
9.2.3	Envelope	49

9.2.4	Protective field	49
9.2.5	Scan validation and degraded sensing	50
9.2.6	Fail-safe behaviour	50
9.3	Safety case	50
9.4	Interfaces	53
9.5	Parameters	53
10	Health and diagnostics	55
10.1	Checks	55
10.2	TF latency budget	56
10.2.1	Causes, most likely first	56
10.2.2	Diagnosis in ten minutes	57
10.2.3	Fixes	57
10.3	Parameters	58
11	Simulation	61
11.1	Why the simulator models errors	61
11.2	Models	61
11.3	Robot-side contract	62
11.4	Scenarios and site fixture	62
11.5	Launch	63
11.6	Evaluation protocol	63
11.7	Parameters	63
12	Operator panels	65
12.1	Panels	65
12.2	Behaviour	65
13	Interfaces reference	67
13.1	Conventions	67
13.2	Topics and services by package	67
13.3	Messages	68
13.4	Services	73
14	Development	76
14.1	Package layout	76
14.2	Adding a plugin	76
14.2.1	Example: a controller package	77
14.3	Code style	79
14.4	Lint and format	80
14.5	Tests	80
14.6	Continuous integration	80

14.7 Releases	81
14.8 The template repository	81
14.9 Building this documentation	82
A Traceability: findings, fixes and tests	83
A.1 dharma_odometry	83
A.2 dharma_localization	84
A.3 dharma_perception	85
A.4 dharma_mapping	87
A.5 dharma_navigation	88
A.6 dharma_control	92
A.7 dharma_health	94
A.8 dharma_sim	95
A.9 dharma_rviz	96
A.10 dharma_interfaces and cross-stack findings	97

Chapter 1

Introduction

Dharma is a robot-agnostic autonomy stack for legged and wheeled service robots on ROS 2 Foxy, Humble and Jazzy. It provides LiDAR-inertial odometry with a start gate and a monitor, fiducial-anchored localization with an extended Kalman filter, occupancy mapping and sites, route teaching, path planning and following with avoidance around people, a control multiplexer with a protective field and a latched e-stop, health monitoring, RViz panels and a simulator. It contains no product features: points of interest, tours, narration and visitor tracking belong to products built on it (for example the Molly tour-guide robot).

This manual describes version 0.2.0. It is written for three readers:

- **integrators** who bring Dharma up on a new robot: Chapters 2 and 3 (architecture, robot profile and parameters), Chapter 9 (the safety case) and the interfaces reference (Chapter 13);
- **developers** who change or extend a package: the subsystem chapters (4–11) with their models and parameters, and Chapter 14 (layout, style, tests, CI, releases);
- **reviewers** who check that known defects are fixed: Appendix A maps every review finding to its fix and its regression test.

1.1 What changed in 0.2.0

Version 0.2.0 is a breaking clean-up of the 0.1 stack split out of a product. Two reviews modelled every subsystem and listed defects with priorities (P0 safety, P1 robustness, P2 maintainability). The main changes:

- **General only.** Points of interest, tours, narration, gestures, visitor tracking and the mission status were removed; goals are poses, sites hold a map, fiducials and routes.
- **One robot description.** Footprint, base height, kinematics, limits and frames come from the shared `robot.*` parameters (`RobotProfile`); no package declares its own geometry.
- **Structure.** Every package has a pure `core/` (no ROS imports), a `utils/` with conversions and thin nodes on `DharmaNode`; modules have at most 300 lines; families of algorithms are plugins.
- **Typed interfaces.** Status topics are typed messages with enum constants; safety actions are separate trigger services.
- **Safety.** One braking model for every speed limit; the protective field covers rotation and combined motion; blind or stale sensing is an obstacle; the e-stop latches reasons, has a hardware input and requires a re-arm; navigation never publishes the robot's `cmd_vel`.
- **Estimation.** Gravity-aligned odometry, an EKF over `map→odom` with covariance and chi-square gating, optional scan matching, mapping with jump and correction gates.

1.2 Conventions of this manual

Frames, topics, services and parameters are set in `typewriter`. Topic names are relative: a product namespaces the robot driver (`/<ns>/cmd_vel`). Units are SI; angles are radians internally and degrees only in human-facing configuration (parameter names end in `_deg`). A transform ${}^aT_b$ is the pose of frame b in frame a , i.e. the TF edge $a \rightarrow b$; it maps points from b to a . This edition describes the 0.2.0 code

of every repository; the parameter and interface tables are generated from it (`docs/manual/tools/reference.py`). Measured results are reported in the paper, not here.

1.3 Repositories

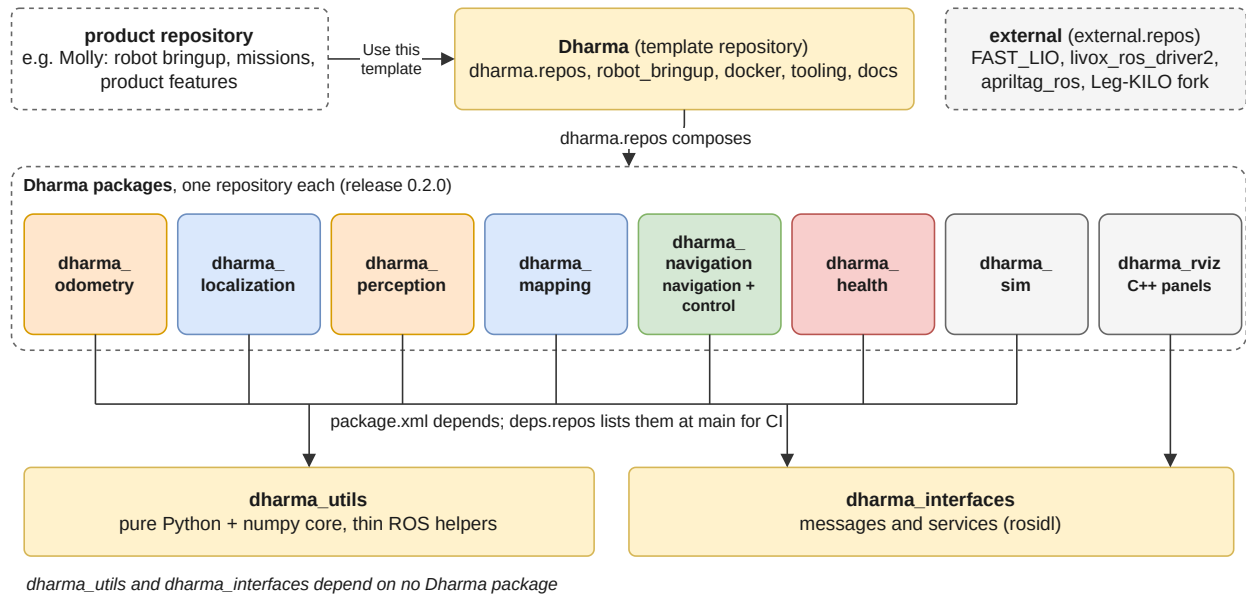


Figure 1.1: Repositories and their dependencies. Every package depends only on `dharma_utils` and `dharma_interfaces`; the Dharma template repository composes them, and a product is generated from the template.

Repository	Content
dharm_utils	math, typed configuration, robot profile, braking model, io, grid, plugins, logging, ROS node helpers, test helpers
dharm_interfaces	messages and services
dharm_odometry	estimator launch (FAST-LIO2, Leg-KILO or external), start gate, monitor
dharm_localization	bridge, EKF over map→odom, fiducial tags
dharm_perception	sector and safety scans from the LiDAR, camera throttle for fiducials
dharm_mapping	occupancy grid, sites, route recorder
dharm_navigation	packages dharm_navigation (planner, loader, follower, avoidance) and dharm_control (control mux)
dharm_health	health monitor and diagnostics
dharm_rviz	RViz panels (C++)
dharm_sim	simulator with drift, latency and dropout
Dharma	template repository: workspace, docker, bringup skeleton, tooling, documentation

Architecture

2.1 Layers

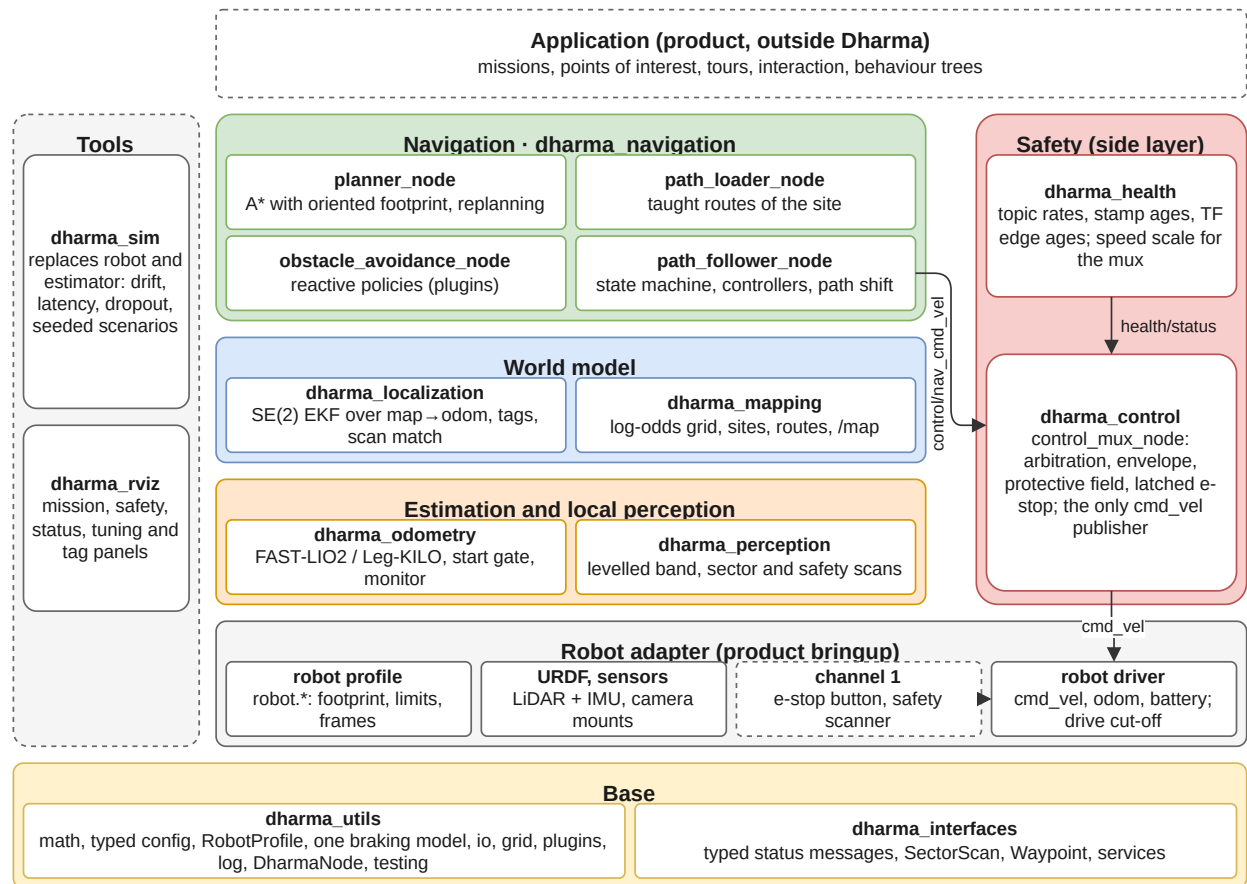


Figure 2.1: Layers of Dharma. Colours: estimation orange, world model blue, navigation green, safety red, base yellow, hardware grey.

Dharma is organised in layers (Fig. 2.1); each depends only on the layers below and on the base packages.

Robot adapter (product). The driver bridge (velocity command in, robot odometry and battery out), the URDF with the sensor mounts, and the robot profile. A hardware e-stop and, where required, a safety-rated scanner act on the drive directly (channel 1, Section 9.1).

Estimation and local perception. `dharma_odometry` runs a LiDAR-inertial estimator and monitors it; `dharma_perception` turns the LiDAR into obstacle sectors around the robot and throttles the camera for the fiducial detector.

World model. `dharma_localization` keeps map→odom from fiducials, operator input and optional scan matching; `dharma_mapping` records occupancy maps and manages sites and routes.

Navigation. `dharma_navigation` plans on the map, loads taught routes, follows paths and runs reactive avoidance. Its velocity output is `control/nav_cmd_vel`, never the robot's `cmd_vel`.

Safety (side layer). `dharma_control` is the only publisher of `cmd_vel`: it arbitrates navigation and

operators, applies the envelope and the protective field and latches the e-stop. `dharma_health` watches rates, ages and system resources and requests a speed scale from the mux.

Base. `dharma_utils` and `dharma_interfaces`.

Tools. `dharma_sim` replaces the robot and the estimator in simulation; `dharma_rviz` provides operator panels.

2.2 Data flow

Fig. 2.2 shows the nodes and the main topics between them; the interfaces reference (Chapter 13) lists every message and service.

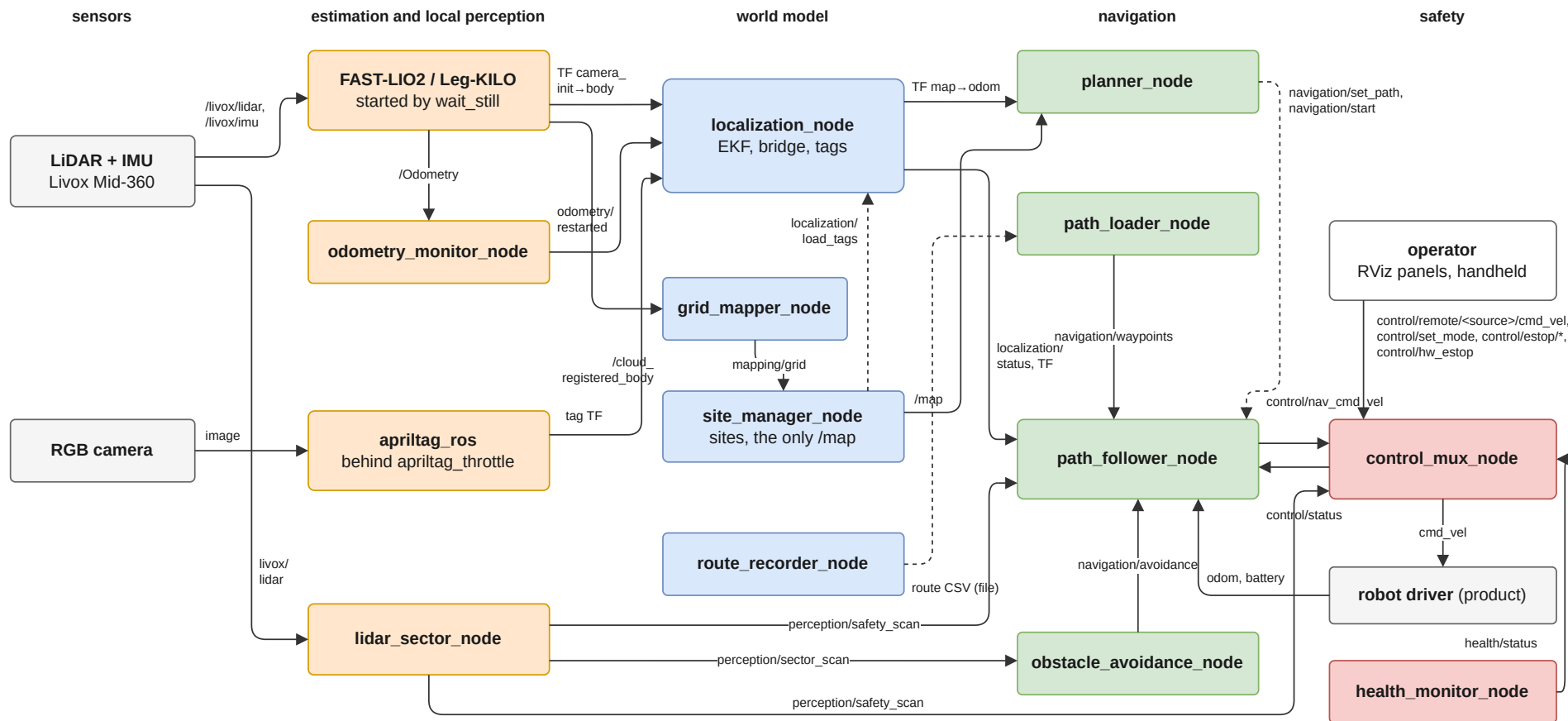


Figure 2.2: Nodes and main topics. Solid: topics and TF; dashed: services and files.

Package	Publishes	Subscribes	Services
odometry	odometry/status (OdometryStatus, latched), odometry/restarted (Empty)	/Odometry, leg odometry, IMU	–
localization	TF map→odom and the bridge (odom→camera_init, body→base_link), localization/status (latched), localization/pose (PoseWithCovarianceStamped), localization/tag_markers	odometry/restarted, /initialpose, IMU, /map and the registered cloud (scan matching)	localization/start, reset, assume_origin, record_tags, load_tags
perception	perception/sector_scan (front), perception/safety_scan (360°), perception/obstacles	Livox CustomMsg or PointCloud2, TF	–
mapping	/map (latched, only publisher), mapping/grid, mapping/status, site/status, route/saved	registered cloud, localization/status	site/list, site/load, mapping/start, mapping/stop, mapping/save, route/start, route/stop, route/mark_stop_point
navigation	planner/path, planner/status, planner/costmap, navigation/status, navigation/stop_point, navigation/path, navigation/waypoints, navigation/avoidance, control/nav_cmd_vel, markers	/map, perception/*_scan, control/status, localization/status, battery, odom	planner/go_to, planner/plan, planner/cancel, navigation/set_path, navigation/load_path, navigation/start, stop, pause, resume
control	cmd_vel (only publisher), control/status (latched)	control/nav_cmd_vel, control/remote/<source>/cmd_vel, perception/safety_scan, control/estop_request, control/hw_estop, health/status	control/set_mode, control/estop/engage, control/estop/release
health	health/status (latched), diagnostics	configured topics, status topics, TF	–

2.3 Frames

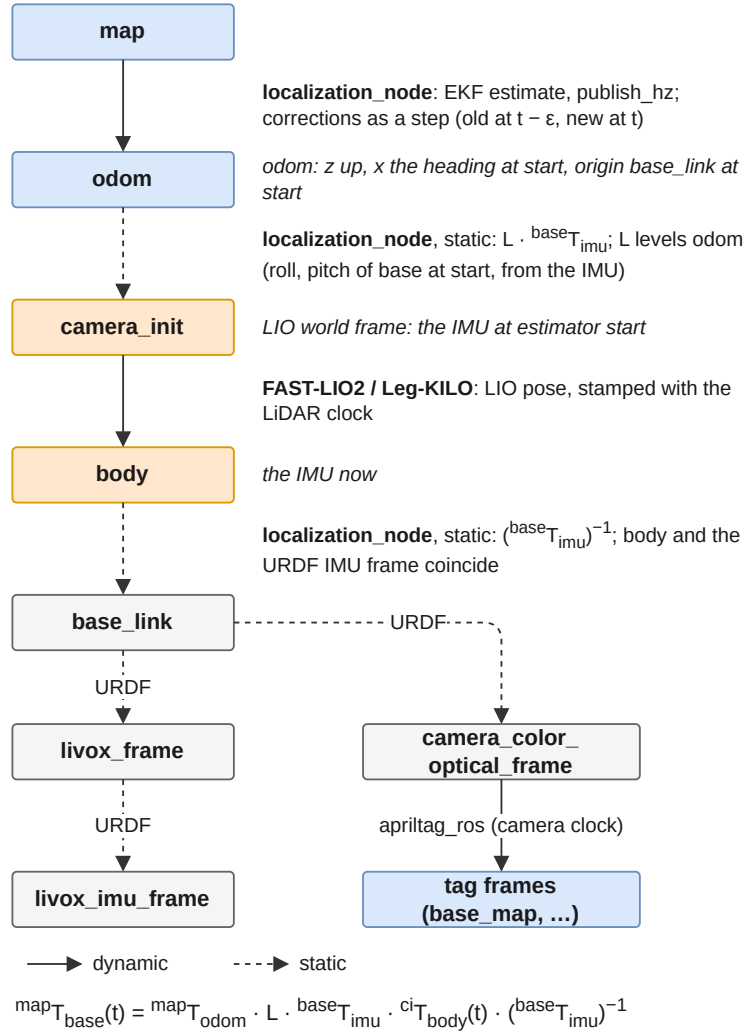


Figure 2.3: TF tree and who publishes each edge.

The tree follows REP-105 (Fig. 2.3). The LiDAR-inertial estimator publishes `camera_init`→`body`: `camera_init` is the IMU frame at estimator start and `body` the IMU now. Localization bridges it into the robot tree with two static transforms and publishes the map correction:

$$mapT_{base}(t) = mapT_{odom} \underbrace{L^{baseT_{imu}}}_{odomT_{ci}} ciT_{body}(t) \underbrace{(baseT_{imu})^{-1}}_{bodyT_{base}}. \quad (2.1)$$

$baseT_{imu}$ is the URDF mounting of the IMU and L the roll and pitch of the base at estimator start (Section 5.1). Consequently `odom` has z against gravity, its x axis is the heading at start and its origin is `base_link` at start. Frame names come from `robot.frames.*`; the LIO frames are parameters of localization.

The edges with time-varying values are `map`→`odom` (stamped by localization with the node clock) and `camera_init`→`body` (stamped by the estimator with the LiDAR clock); everything else is static. This matters for the TF latency budget (Section 10.2).

2.4 Design principles

1. **One source of truth for the robot.** Every geometric and dynamic assumption is derived from the robot profile (Chapter 3); derived values are computed, never copied.
2. **Pure core, ROS at the edge.** Algorithms are plain Python and numpy, unit-tested without ROS; nodes convert messages at the edge.
3. **Plugins instead of if/elif.** Controllers and avoidance policies register in a named registry, open to entry points.
4. **Uncertainty in the interfaces.** The robot pose is published with its covariance; statuses carry states, sigmas and reasons.
5. **Safety independent of autonomy.** The safety layer only reads typed inputs and can be verified on its own; navigation pauses itself on the control status.
6. **Fail safe.** Missing data is an obstacle; a failing callback stops the robot.
7. **Compatibility.** SemVer; every package of the stack moves together across a breaking change.

2.5 Launch composition

Each package has one launch file (`launch/<pkg>.launch.py`) with the arguments `params_file`, `robot_params`, `use_sim_time` and `log_level`; parameters are applied as `[package defaults, robot_params, params_file, non-empty overrides]`, so an empty argument never overrides a value from a file. No package launch file includes another one: the product bringup (generated from the template's `robot_bringup`) composes them. Nodes that can plausibly crash set `respawn=True`; the robot driver must stop on its own command timeout meanwhile. Foxy's launch cannot respawn: there a crashed node stays down and the health monitor reports it. Foxy also applies a node's own section of a parameter file after `/**` whatever the file order, so overrides go under the node name (`dharma_utils.ros.launch.node_overrides` for launch arguments).

Chapter 3

Configuration and robot profile

3.1 The robot profile

Every node reads the same `robot.*` parameters into an immutable `dharma_utils.robot.RobotProfile`. The product passes them once, in its robot params file, to every node through the launch argument `robot_params`:

```
/**:
  ros__parameters:
    robot:
      name: quadruped_a
      kinematics: omni          # omni: vx, vy, wz; diff: vx, wz
      base_height: 0.35         # m, base_link above the floor when standing
      footprint: {front: 0.40, rear: 0.40, half_width: 0.28}
      limits:
        max_vx: 0.8             # m/s
        max_vy: 0.4             # m/s (0 for diff)
        max_wz: 1.0             # rad/s
        max_accel: 0.5          # m/s^2
        max_decel: 0.6          # m/s^2, braking: measure it on the robot
        max_yaw_accel: 1.5      # rad/s^2, yaw braking
        reaction_time: 0.25     # s, command to braking (sensing + transport)
    )
    frames: {map: map, odom: odom, base: base_link}
```

Parameters marked [†] are read-only at runtime (`live=False`); every other one can be changed with `ros2 param set` and is validated before it applies.

Parameter	Default	Unit	Range	Description
every node				
<code>robot.name</code>	<code>robot</code>			robot model, for logs
<code>robot.kinematics</code>	<code>omni</code>		<code>omni, diff</code>	<code>omni</code> : vx, vy, wz; <code>diff</code> : vx, wz
<code>robot.base_height</code>	<code>0.35</code>	<code>m</code>	<code>0.01..3</code>	<code>base_link</code> above the floor when standing
<code>robot.footprint.front</code>	<code>0.4</code>	<code>m</code>	<code>0.01..5</code>	<code>base_link</code> to the front edge
<code>robot.footprint.rear</code>	<code>0.4</code>	<code>m</code>	<code>0.01..5</code>	<code>base_link</code> to the rear edge
<code>robot.footprint.half_width</code>	<code>0.28</code>	<code>m</code>	<code>0.01..5</code>	half of the body width
<code>robot.limits.max_vx</code>	<code>0.8</code>	<code>m/s</code>	<code>0..10</code>	forward / backward speed
<code>robot.limits.max_vy</code>	<code>0.4</code>	<code>m/s</code>	<code>0..10</code>	sideways speed (<code>omni</code> only)
<code>robot.limits.max_wz</code>	<code>1</code>	<code>rad/s</code>	<code>0..10</code>	yaw rate
<code>robot.limits.max_accel</code>	<code>0.5</code>	<code>m/s²</code>	<code>0.01..20</code>	linear acceleration
<code>robot.limits.max_decel</code>	<code>0.6</code>	<code>m/s²</code>	<code>0.05..20</code>	braking deceleration

Parameter	Default	Unit	Range	Description
<code>robot.limits.max_yaw_accel</code>	1.5	rad/s ²	0.05..50	yaw braking
<code>robot.limits.reaction_time</code>	0.25	s	0..2	command to braking delay (sensing + transport)
<code>robot.frames.map</code>	map			global, fixed frame
<code>robot.frames.odom</code>	odom			continuous odometry frame
<code>robot.frames.base</code>	base_link			robot body frame

3.1.1 Derived quantities

Nothing derived from the robot is configured separately:

Quantity	Derivation and users
inscribed radius	$r_{in} = \min(f, b, w)$: lethal radius of the planner (plus margin), inner edge of the rotation ring
circumscribed radius	$r_c = \sqrt{\max(f, b)^2 + w^2}$: oriented checks of the planner, outer edge of the ring swept when turning, creep cap in REMOTE
self filter	footprint grown by a margin: perception, mapping
floor	<code>base_link</code> minus <code>robot.base_height</code> : height bands of perception, mapping, scan matching
lateral speed	$v_y = 0$ for <code>diff</code> : envelope, controllers, approach
braking model	$D(v) = vt_r + v^2/(2a)$ with <code>max_decel</code> and <code>reaction_time</code> : speed profile, path shift, motion guard, protective field
jump limits	<code>max_vx</code> , <code>max_wz</code> : odometry monitor, mapping jump gate

For the default footprint $r_{in} = 0.28$ m and $r_c = 0.49$ m. The 0.1 planner used a radius of 0.35 m for this footprint (finding N1): a path it accepted could bring a corner of the body about 14 cm into a wall when the robot turned or moved sideways in a narrow corridor.

The profile validates cross-field consistency: `max_vy` must be 0 for `diff` kinematics and the three frames must differ.

3.1.2 Measuring the limits

`max_decel`, `max_yaw_accel` and `reaction_time` bound every speed limit in the stack through $D(v) = vt_r + v^2/(2a)$. The defaults (0.6 m/s², 1.5 rad/s², 0.25 s) are placeholders: measure them on the robot before it drives on its own, in the worst case (slippery floor, full payload, low battery). `docs/manual/tools/brake_test.py` sends step commands through the mux and analyses the recorded speed.

1. **Prepare.** A flat, open area of at least three stopping distances at `max_vx` in the driving direction, people kept away, the hardware e-stop at hand (channel 1). Start the bringup with the default (conservative) limits.
2. **Switch the mux to REMOTE** with the source the tool uses:

```
ros2 service call /control/set_mode dharma_interfaces/srv/SetMode \
  "{mode: remote, source: dashboard}"
```

3. **Record step runs**, at least three per speed, at several speeds up to `max_vx`, and the same for yaw up to `max_wz`:

```
python3 brake_test.py drive --speed 0.5 --odom /<ns>/odom --out lin_0.5_1.csv
python3 brake_test.py drive --axis angular --speed 1.0 --odom /<ns>/odom --out yaw_1.0_1.csv
```

The tool sends zeros for 1 s (the mux re-arms on neutral commands), the speed for 3 s, then zeros, on control/remote/<source>/cmd_vel at 50 Hz, and records the twist of an Odometry topic with the exact times of the command changes. Use the estimator's /Odometry instead of the driver's when the driver twist is unreliable (legged robots).

4. **Analyse.** `python3 brake_test.py analyze lin_*.csv` prints, per run, the speed before the step v_0 , the reaction time, the deceleration and the measured and modelled stopping distances, then the worst values. The deceleration is a line fitted between 80 % and 20 % of v_0 ; the reaction time runs from the step command to where that line meets v_0 , so it contains the mux, the transport, the driver and the odometry delay.
5. **Configure** in the robot profile: `max_decel` and `max_yaw_accel` the smallest measured decelerations, and `reaction_time` the largest measured reaction *plus* the sensing delay the step test does not see: the mean of `ros2 topic delay /perception/safety_scan` and one LiDAR frame period (an obstacle that appears just after a frame starts is published with the next one). Then add margin.
6. **Verify** with the new profile: repeat the fastest runs; every measured stopping distance must stay below $D(v_0)$ of the configured values.

Example. The demo subcommand records the same step on `dharma_sim`'s body model with known values (deceleration 0.6 m/s^2 , command latency 0.12 s, speed noise 1 cm/s), so the analysis can be checked; these are simulated runs, not a robot measurement:

```
$ for v in 0.3 0.5 0.8; do brake_test.py demo --speed $v --out run_${v}.csv; done
$ brake_test.py analyze run_*.csv
run          v0  t_r [s]      a  D meas  D model
run_0.3.csv   0.298  0.099  0.588  0.105  0.105
run_0.5.csv   0.498  0.115  0.622  0.257  0.257
run_0.8.csv   0.798  0.090  0.590  0.613  0.611
worst of 3 runs: reaction 0.115 s, deceleration 0.588
```

The body applies a new command over the 20 ms step that ends when it becomes active, so its deceleration starts at 0.100 s; noise-free runs give exactly 0.100 s and 0.600 m/s^2 . With 1 cm/s of noise single runs scatter by about $\pm 0.015 \text{ s}$ and $\pm 4 \%$, which is why the worst of several runs is configured. In this example the profile would get `max_decel` below 0.588 and `reaction_time` above 0.115 s plus the sensing delay.

3.2 Typed parameters

Each node declares its configuration as frozen dataclasses with `dharma_utils.config.param` fields (default, doc, unit, min, max, choices, live) and `nested(...)` sections (dotted names such as `speed.cruise`). `DharmaNode`:

- declares every field as a ROS parameter with a descriptor (description, unit, range, read-only for `live=False`): `ros2 param describe <node> <name>` shows it;
- validates all values at start (ranges, choices, types and the cross-field `validate()` of the dataclass) and refuses to start on a safety value that fails;

- validates every `ros2 param set` the same way and rejects an invalid value with the reason, keeping the old one;
- reports parameters in the YAML files that no dataclass declares (typos).

Safety values (margins, timeouts, limits) are read-only at runtime. Defaults live in the dataclasses; the package's `config/<pkg>.yaml` contains only what differs and documentation. Each subsystem chapter lists its parameters, generated from the dataclasses by `docs/manual/tools/reference.py`.

3.3 Parameter files and launch

A product layers its files over the package defaults:

1. the package defaults `config/<pkg>.yaml`;
2. the robot profile (`robot_params`), shared by every node;
3. a stack or site file (`params_file`) with per-node overrides;
4. non-empty launch arguments (an empty argument never overrides a file; a file argument that is set but missing fails the launch).

Paths accept `~` and `$VAR`. Paths received through services (tags files, routes, sites) must resolve inside `sites_dir` (`dharma_utils.io.resolve_inside`); a request that escapes it is refused.

Chapter 4

Odometry

`dharma_odometry` starts the LiDAR-inertial estimator behind one launch file, holds its start until the IMU is still and level, and monitors its output. The estimator publishes `camera_init→body` and `/Odometry`, starting from wherever the robot is at launch; it drifts slowly and `dharma_localization` keeps `map→odom` on top of it. The defaults are for a Livox Mid-360 indoors.

4.1 Estimators

method:=fast_lio (default). FAST-LIO2 [1], an iterated error-state Kalman filter with state $(\mathbf{R}, p, v, b_\omega, b_a, g)$: the IMU propagates, point-to-plane residuals of each LiDAR scan correct. The initial gravity g_0 is the mean acceleration of the first samples and `camera_init` is the IMU frame at that time. It runs with `extrinsic_est_en: false` since the Mid-360 IMU is built in and factory calibrated (review finding #8).

method:=leg_kilo Leg-KILO [2] through the DarmawanRobotics fork (headless viewer, the same `odom/body` frames as FAST-LIO, the `/cloud_registered_body` output). It runs in LIO mode with `init_type: 2` (gravity alignment: the start rotation aligns the mean acceleration with gravity). Its KILO mode needs per-foot contact forces that most drivers do not publish.

method:=external another publisher of `/Odometry` (the simulator); only the monitor runs.

4.2 Start gate (`wait_still`)

Both estimators take gravity and the IMU biases from their first samples, so they must start standing still. `wait_still` is the launch prefix of the estimator: it holds the start, and every respawn, until a window of `still_sec` (2 s) of IMU samples satisfies

$$\max_k \|\omega_k\| < \omega_{\max}, \quad \frac{\text{std}_k \|f_k\|}{\text{mean}_k \|f_k\|} < \sigma_{\text{rel}}, \quad \theta = \angle(\bar{f}, z) \leq \theta_{\max}, \quad (4.1)$$

with $\omega_{\max} = 0.03$ rad/s, $\sigma_{\text{rel}} = 1\%$ and $\theta_{\max} = 3^\circ$ (it was 8° , finding O1). The relative spread is unit-free, so an IMU reporting g (the Mid-360) or m/s^2 is handled alike. After `timeout_sec` it starts anyway with an error.

Why the tilt matters. If the estimator starts tilted by θ , its world frame is tilted and a horizontal distance s appears as a height change and a shorter planar distance:

$$\Delta z \approx s \sin \theta, \quad \frac{\Delta s_{xy}}{s} = 1 - \cos \theta. \quad (4.2)$$

θ	Δz per metre	distance to a 0.5 m z error	planar scale error
2°	3.5 cm	14.3 m	0.06 %
5°	8.7 cm	5.7 m	0.38 %
8° (0.1 limit)	13.9 cm	3.6 m	0.97 %

Mapping and perception filter obstacles in a height band above the floor. With an 8° start the floor 5 m ahead lies 0.7 m above the band of a level frame. Version 0.2.0 therefore lowers the gate limit to 3° and levels `odom` in the localization bridge (Section 5.1).

4.3 Monitor

`odometry_monitor_node` publishes `odometry/status` (`OdometryStatus`, `latched`) and `odometry/restarted`.

State	When
INITIALIZING	no /Odometry yet (the gate is holding the estimator)
LOST	no /Odometry for <code>stale_sec</code> ; a jump or a restart, for <code>hold_sec</code>
DEGRADED	$ z $ above <code>max_z_drift_m</code> on a flat floor; odom tilted against gravity by more than <code>tilt.max_deg</code> while still; the speed test in alarm
OK	otherwise

Jump. A step between consecutive poses is a jump when it is longer than `jump.min_step_m` and faster than `jump.speed_factor` times the robot speed limit (`robot.limits`).

Restart (review finding #7). A fresh estimator starts at the identity pose. Being near the origin is not enough: a robot turning in place next to its start pose gets there too. A restart is a pose at the origin (within `restart.origin_radius_m` and `restart.origin_yaw_deg`) after a message gap longer than `restart.gap_sec` (`respawn`, `gate`) or after a translation jump back to the origin, or a stamp going back (bag loop, simulator reset). Yaw alone never counts. On a restart the monitor publishes `odometry/restarted`: localization forgets `map→odom` and levels again.

Speed residual test. With the robot's own odometry (`leg_odom_topic`), the planar speeds are compared as a residual test with a one-sided CUSUM [3]:

$$r_k = v_k^{\text{lio}} - v_k^{\text{leg}}, \quad \chi_k^2 = \frac{r_k^2}{\sigma_{\text{lio}}^2 + \sigma_{\text{leg}}^2}, \quad S_k = \max(0, S_{k-1} + \chi_k^2 - \kappa), \quad (4.3)$$

alarm while $S_k > h$. Under agreement χ_k^2 follows $\chi^2(1)$ (mean 1), so $\kappa = 3.84$ (its 95 % quantile) lets the sum decay, while a lasting disagreement (a degenerate corridor, slipping feet, a stuck estimator) accumulates. The thresholds come from the sensor standard deviations, not from per-robot speeds. S is bounded at $2h$ for a fast recovery. The test is off without leg odometry or while it is older than `speed.leg_timeout_sec`.

Tilt while still. While the IMU is still, the gravity it measures, rotated into `odom`, gives the tilt of `odom` itself; above `tilt.max_deg` the estimator started tilted and z will drift with distance. A robot standing on a ramp tilts its body, not `odom`, and stays OK. This assumes the IMU axes are the /Odometry body axes (true for FAST-LIO and Leg-KILO).

4.4 Parameters

Parameters marked [†] are read-only at runtime (`live=False`); every other one can be changed with `ros2 param set` and is validated before it applies.

Parameter	Default	Unit	Range	Description
odometry_gate (wait_still)				
<code>imu_topic</code> [†]	<code>/livox/imu</code>			IMU topic (sensor_msgs/Imu)
<code>still_sec</code>	2	s	0.2..60	this long without motion before the start

Parameter	Default	Unit	Range	Description
max_gyro_rad_s	0.03	rad/s	0..10	largest angular rate in the window
max_acc_rel_std	0.01		0..1	std of laccl / mean laccl (unit-free: m/s ² or g)
max_tilt_deg	3	deg	0..90	gravity against the IMU z axis; 0: off
min_samples	20		2..10000	IMU samples needed in the window
timeout_sec	60	s	0..3600	start anyway (with an error); 0: wait forever
odometry_monitor_node				
odom_topic [†]	/Odometry			estimator odometry (nav_msgs/Odometry)
leg_odom_topic [†]	" "			robot leg / wheel odometry for the speed test; empty: off
imu_topic [†]	/livox/imu			IMU with the axes of the odometry body; empty: no tilt
status_period_sec [†]	0.2	s	0.01..10	odometry/status period
stale_sec	0.5	s	0.05..60	no odometry this long: LOST
hold_sec	5	s	0..600	a jump or restart stays LOST this long
max_z_drift_m	0.5	m	0..100	z above this is DEGRADED (flat floor); 0: off
restart.gap_sec	2	s	0.1..60	a message gap this long followed by the origin
restart.origin_radius_m	0.05	m	0..1	a fresh estimator starts this close to the origin
restart.origin_yaw_deg	1	deg	0..180	and with this little yaw
jump.speed_factor	3		1..100	a step faster than this times the robot speed limit
jump.min_step_m	0.3	m	0..100	and longer than this is a jump
speed.window_sec	0.5	s	0.05..10	estimator speed from the poses of this window
speed.sigma_lio_mps	0.05	m/s	0.001..10	std of the estimator speed
speed.sigma_leg_mps	0.15	m/s	0.001..10	std of the leg / wheel odometry speed
speed.cusum_drift	3.84		0..1000	kappa, subtracted from every chi ² (chi ² (1) 95 %: 3.84)
speed.cusum_threshold	20		0.1..10000	h: the CUSUM above this is a mismatch
speed.leg_timeout_sec	0.3	s	0.01..10	older leg odometry is not compared
tilt.max_deg	5	deg	0..90	odom z against gravity above this; 0: off
tilt.still_sec	1	s	0.2..60	IMU window
tilt.max_gyro_rad_s	0.03	rad/s	0..10	still: rate
tilt.max_acc_rel_std	0.01		0..1	still: std of laccl / mean

Parameter	Default	Unit	Range	Description
<code>tilt.min_samples</code>	20		2..10000	IMU samples needed in the window

4.5 Launch

```

ros2 launch dharma_odometry odometry.launch.py # FAST-LIO2
ros2 launch dharma_odometry odometry.launch.py method:=leg_kilo save_map:=
  true
ros2 launch dharma_odometry odometry.launch.py method:=external # simulator

```

Arguments: `method`, `params_file`, `robot_params`, `site_config`, `use_sim_time`, `log_level`, `save_map`, `wait_still`, `respawn`, `leg_odom_topic`, `fast_lio_params_file`, `leg_kilo_config_file`, `leg_kilo_viewer`. A file argument that is set but missing fails the launch (finding #17). The package no longer includes the localization launch file nor records routes (moved to `dharma_mapping`): odometry must not depend on a layer above it.

Chapter 5

Localization

`dharma_localization` bridges the LiDAR-inertial odometry into the REP-105 tree and keeps `map→odom` with an extended Kalman filter. A map's frame is the frame its fiducials were recorded in: a tag in view puts the robot back on that map wherever it starts. Any marker detector works as long as it puts the marker frames under the camera optical frame on TF (`apriltag_ros` today [4,5]).

5.1 Gravity-aligned bridge

The node publishes two static transforms:

$${}^{odom}\mathbf{T}_{ci} = L {}^{base}\mathbf{T}_{imu}, \quad {}^{body}\mathbf{T}_{base} = ({}^{base}\mathbf{T}_{imu})^{-1}, \quad (5.1)$$

so that ${}^{odom}\mathbf{T}_{base} = L {}^{base}\mathbf{T}_{imu} {}^{ci}\mathbf{T}_{body}(t) {}^{imu}\mathbf{T}_{base}$ equals L at start (where ${}^{ci}\mathbf{T}_{body} = \mathbf{I}$).

Levelling. Standing still, the accelerometer measures the reaction to gravity, a vector pointing up in the IMU axes. Over a still window (1 s, angular rate below 3°/s, spread of the specific force below 3 %) the mean specific force $\tilde{f}$ is expressed in `base_link` at estimator start,

$$u = \frac{\mathbf{R}_{imu}^{base} \mathbf{R}_{body}^{ci}(t) \tilde{f}}{\|\mathbf{R}_{imu}^{base} \mathbf{R}_{body}^{ci}(t) \tilde{f}\|}, \quad (5.2)$$

(the rotation since start is undone, so the robot may already have moved). L is the rotation aligning u with z , with its yaw removed so that the x axis of `odom` stays the heading at start. A start tilt above 20° is refused as implausible (slope or wrong IMU frame); after `leveling.timeout_sec` the bridge goes up unlevelled with a warning. After `odometry/restarted` the window is cleared and the new start is levelled again. Only the direction of $\tilde{f}$ is used, so the accelerometer units do not matter.

5.2 Estimator

5.2.1 Why

In 0.1 one tag detection replaced `map→odom` directly (finding L1). There was no covariance, no fusion of detections and no correction between tags; detections were rejected when the correction itself exceeded 0.5 m or 15° (review #1), which rejected good fixes far from the odometry origin and accepted small rotations that moved the robot metres. The errors of a single detection propagate as

Case	Parameters	Error
PnP of a 16 cm tag, position $\approx d \delta$	$d = 3 \text{ m}, \delta = 3^\circ$	15.7 cm
same	$d = 5 \text{ m}, \delta = 2^\circ$	17.5 cm
heading of <code>map→odom</code> , $e_\perp \approx L \delta \psi$	$L = 10 \text{ m}, \delta \psi = 1^\circ$	17.5 cm
same	$L = 30 \text{ m}, \delta \psi = 1^\circ$	52.4 cm

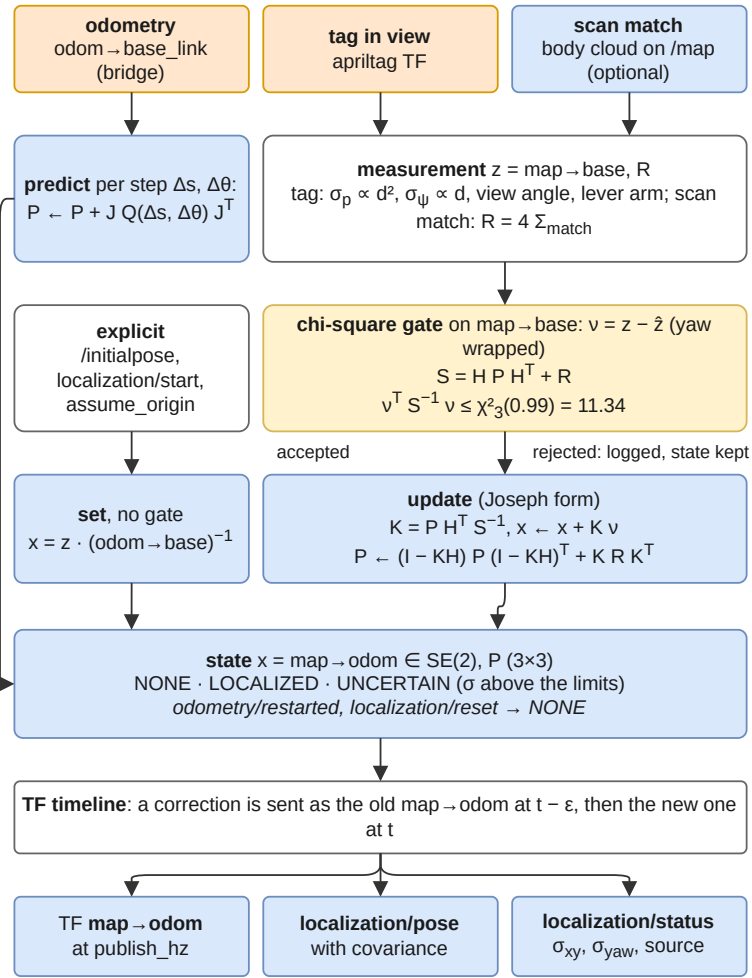


Figure 5.1: EKF over $\text{map} \rightarrow \text{odom}$: prediction from odometry motion, gated automatic corrections, explicit operator input and the TF timeline.

5.2.2 State and prediction

The state is $x = (x, y, \psi)$ of ${}^{\text{map}}\mathbf{T}_{\text{odom}}$ with a 3×3 covariance $\mathbf{P}$. Given the odometry pose $o = {}^{\text{odom}}\mathbf{T}_{\text{base}}$, the robot pose is $h(x) = x \oplus o = (x_b, y_b, \psi_b)$ and

$$\mathbf{H} = \frac{\partial h}{\partial x} = \begin{bmatrix} 1 & 0 & -(y_b - y) \\ 0 & 1 & x_b - x \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.3)$$

The third column is the lever arm: a rotation of $\text{map} \rightarrow \text{odom}$ moves a robot far from the odom origin by a lot. Odometry motion is counted in steps (a robot standing still never grows uncertain); for a step of distance Δs and rotation $\Delta \theta$ the drift of the robot pose

$$\mathbf{Q} = \text{diag}(\sigma_s^2 \Delta s, \sigma_s^2 \Delta s, \sigma_{\psi s}^2 \Delta s + \sigma_{\psi \theta}^2 \Delta \theta) \quad (5.4)$$

is mapped onto the state, $\mathbf{P} \leftarrow \mathbf{P} + \mathbf{H}^{-1} \mathbf{Q} \mathbf{H}^{-\top}$. With $\sigma_s = 0.03 \text{ m}/\sqrt{\text{m}}$ the position uncertainty grows to about 10 cm after 10 m.

5.2.3 Update and gate

A measurement z of $\text{map} \rightarrow \text{base_link}$ with covariance $\mathbf{R}$, taken while the odometry read o , gives

$$v = z - h(x) \text{ (yaw wrapped)}, \quad \mathbf{S} = \mathbf{H} \mathbf{P} \mathbf{H}^{\top} + \mathbf{R}, \quad d^2 = v^{\top} \mathbf{S}^{-1} v. \quad (5.5)$$

Automatic sources pass only if $d^2 \leq \chi_3^2(p)$ ($p = \text{estimator.gate_confidence}$, 0.99: 11.34) [6]. The gate opens as the uncertainty grows after a long drive and is tight after a correction. An accepted measurement updates

$$\mathbf{K} = \mathbf{P}\mathbf{H}^\top \mathbf{S}^{-1}, \quad \mathbf{x} \leftarrow \mathbf{x} + \mathbf{K}\mathbf{v}, \quad \mathbf{P} \leftarrow (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}(\mathbf{I} - \mathbf{K}\mathbf{H})^\top + \mathbf{K}\mathbf{R}\mathbf{K}^\top \quad (5.6)$$

(Joseph form, symmetric by construction), followed by a floor on the robot pose covariance (`estimator.min_sigma_xy`, `estimator.min_sigma_yaw_deg`) against overconfidence. The reported sigmas are those of the robot pose, $\mathbf{H}\mathbf{P}\mathbf{H}^\top$: σ_{xy} is the square root of the largest eigenvalue of its position block.

States. NONE before the first fix (sigmas infinite), LOCALIZED, UNCERTAIN when σ_{xy} exceeds `estimator.max_sigma_xy` or σ_ψ exceeds `estimator.max_sigma_yaw_deg`.

5.2.4 Sources

Tag. A detection ${}^{cam}\mathbf{T}_{tag}$ of a tag with known ${}^{map}\mathbf{T}_{tag}$ gives

$${}^{map}\mathbf{T}_{base} = {}^{map}\mathbf{T}_{tag} ({}^{base}\mathbf{T}_{cam} {}^{cam}\mathbf{T}_{tag})^{-1}, \quad (5.7)$$

computed in SE(3) because the optical frame (z forward, x right, y down) is never planar, then flattened to (x, y, ψ) . Detections farther than `tag.max_distance_m` or seen more obliquely than `tag.max_view_angle_deg` are ignored. The noise grows with distance d (at least 0.3 m) and view angle α : $\sigma_p = \sigma_{p,1} d^2 / \cos \alpha$, $\sigma_\psi = \sigma_{\psi,1} d / \cos \alpha$. A heading error rotates the robot about the tag: with $\mathbf{v}$ the vector from the robot to the tag and $\mathbf{l} = (v_y, -v_x)$,

$$\mathbf{R} = \begin{bmatrix} \sigma_p^2 \mathbf{I} + \sigma_\psi^2 \mathbf{l} \mathbf{l}^\top & \sigma_\psi^2 \mathbf{l} \\ \sigma_\psi^2 \mathbf{l}^\top & \sigma_\psi^2 \end{bmatrix}. \quad (5.8)$$

Far tags therefore count little, and a test with the real optical rotation $(-\pi/2, 0, -\pi/2)$ and non-zero yaws checks the geometry (review #15).

Pose estimate. `/initialpose` (RViz 2D Pose Estimate). RViz sends a fixed UI covariance whatever the operator meant, so the operator is trusted (5 cm, 2°); a tighter covariance from another client (a simulator) is used as given. Applied without gate.

Scan match (optional, `scan_match.enabled`). The registered body cloud is put into the level `odom` with its full 3D pose, a height band above the floor is kept, the points are expressed in the planar base frame and matched against `/map` with the correlative matcher of `dharma_utils` [7] (coarse to fine on a likelihood field); the covariance of the score surface, scaled by 4, is $\mathbf{R}$.

Explicit. `localization/start` (localize from a tag in view), `localization/assume_origin` (`map→odom` = identity, for a new map) and a pose estimate set the state without gate, so a far but consistent tag that the gate rejects (after a kidnapping or a wrong pose estimate) still relocates on request.

5.2.5 Publication without interpolation ramps

`tf2` interpolates between the two samples around a lookup time. If `map→odom` changed between two periodic samples, lookups in that interval would see a blend of the old and the new value, mapping scans to poses that never existed (findings T1/T3, review #14). A correction is therefore published as the previous value at $t - \varepsilon$ followed by the new value at t ($\varepsilon = 1$ ms), and periodic samples are strictly increasing in time.

5.3 Fiducial tags

`record_tags` averages the map pose of the visible `tag.frames` over `tag.record_duration_sec` and merges them into the tags file. It writes the anchors of a site, so it refuses unless the state is

LOCALIZED and never records the tag the current `map→odom` was derived from (circular, review #2). Recordings with fewer than `tag.record_min_samples` samples or a larger spread than the limits are refused. It runs as a session over the node's timers in its own callback group (no sleep in a callback, finding L2), so `map→odom` keeps flowing at `publish_hz` while it records. Reads are tolerant: a corrupt tags file is reported (`success: false`) and left untouched (review #3); every write keeps a `.bak`. `load_tags` accepts only paths that resolve inside `sites_dir` (review #12); `~` is expanded (review #11).

5.4 Interfaces

Interface	Type	
<code>localization/start</code>	Trigger	localize from a tag in view, explicitly
<code>localization/reset</code>	Trigger	forget <code>map→odom</code>
<code>localization/assume_origin</code>	Trigger	<code>map→odom = identity</code>
<code>localization/record_tags</code>	Trigger	record the visible tags
<code>localization/load_tags</code>	LoadFile	switch the tags file (inside <code>sites_dir</code>)
<code>localization/status</code>	LocalizationStatus	latched: state, source, sigmas, time since the last correction
<code>localization/pose</code>	PoseWithCovarianceStamped	robot pose with covariance
<code>localization/tag_markers</code>	MarkerArray	known tags
in: <code>odometry/restarted</code> , <code>/initialpose</code> , <code>IMU</code> , <code>/map</code> , <code>/cloud_registered_body</code>		

5.5 Parameters

Parameters marked [†] are read-only at runtime (`live=False`); every other one can be changed with `ros2 param set` and is validated before it applies.

Parameter	Default	Unit	Range	Description
localization_node				
<code>sites_dir</code> [†]	<code>~/dharma/map</code>			<code>load_tags</code> paths must be inside
<code>tags_file</code> [†]	<code>" "</code>			tags file at start (empty: none until <code>load_tags</code>)
<code>lio_odom_frame</code> [†]	<code>camera_init</code>			LIO world frame
<code>lio_body_frame</code> [†]	<code>body</code>			LIO body frame (the IMU)
<code>imu_frame</code> [†]	<code>livox_imu_frame</code>			URDF frame of the LIO body
<code>publish_hz</code> [†]	20	Hz	1..200	<code>map→odom</code> on <code>/tf</code>
<code>pose_hz</code> [†]	10	Hz	0..100	<code>localization/pose</code> ; 0: off

Parameter	Default	Unit	Range	Description
assume_start_origin [†]	false			the odom origin is the map origin at start (no tag needed)
estimator.sigma_xy_per_sqrt_m	0.03	m/sqrt(m)	0..1	position drift: 1-sigma after 1 m of travel
estimator.sigma_yaw_deg_per_sqrt_m	0.1	deg/sqrt(m)	0..10	heading drift per metre of travel
estimator.sigma_yaw_deg_per_sqrt_rad	0.5	deg/sqrt(rad)	0..10	heading drift per radian turned
estimator.gate_confidence	0.99		0.95, 0.99, 0.999	chi-square gate (3 dof) of automatic corrections
estimator.max_sigma_xy	0.5	m	0.01..100	above: state UNCERTAIN (position)
estimator.max_sigma_yaw_deg	10	deg	0.1..180	above: state UNCERTAIN (heading)
estimator.min_sigma_xy	0.01	m	0..1	floor after a correction (no overconfidence)
estimator.min_sigma_yaw_deg	0.25	deg	0..10	heading floor after a correction
estimator.motion_step_m	0.05	m	0.001..1	odometry motion counted in steps (jitter-free)
estimator.motion_step_deg	2	deg	0.01..45	odometry rotation counted in steps
tag.camera_frame [†]	camera_color_optical_frame			frame the detector puts the tags under
tag.frames	[base_map, charging_dock]			tag TF frames to record (the detector tag frames)
tag.max_age_sec	0.5	s	0.01..10	older detections ignored
tag.max_distance_m	4	m	0.1..50	farther tags ignored
tag.max_view_angle_deg	60	deg	1..89	tags seen more obliquely are ignored
tag.sigma_xy_at_1m	0.02	m	0.001..1	position sigma at 1 m, grows with distance ²
tag.sigma_yaw_deg_at_1m	1	deg	0.01..30	heading sigma at 1 m, grows with distance
tag.update_hz [†]	10	Hz	0.5..50	tag checks
tag.min_interval_sec	0.5	s	0..60	per tag, between two fused detections
tag.auto_localize	true			while not localized: localize from any visible tag
tag.auto_correct	true			while localized: fuse visible tags through the gate
tag.record_duration_sec	2	s	0.2..60	record_tags
tag.record_min_samples	5		1..1000	detections needed per tag
tag.record_max_std_m	0.05	m	0.001..1	noisier recordings are refused
tag.record_max_std_deg	3	deg	0.01..45	noisier recordings are refused (heading)
tag.marker_size	0.15	m	0.01..2	tag plate in RViz

Parameter	Default	Unit	Range	Description
scan_match.enabled [†]	false			match the registered cloud against /map
scan_match.cloud_topic [†]	/cloud_registered_body			deskewed cloud in the LIO body frame
scan_match.rate_hz	1	Hz	0.05..10	matches per second, at most
scan_match.min_height_m	0.1	m	-1..5	band above the floor
scan_match.max_height_m	1.5	m	0..10	band above the floor
scan_match.max_range_m	15	m	0.5..200	farther points ignored
scan_match.xy_window_m	0.3	m	0.05..5	search window
scan_match.yaw_window_deg	6	deg	0.5..45	search window
scan_match.min_score	0.5		0..1	mean likelihood needed
scan_match.covariance_scale	4		1..1000	inflates the match covariance (correlated points)
scan_match.occupied_threshold	65		1..100	occupancy value of an obstacle
leveling.enabled [†]	true			level odom with the IMU (off: odom = base)
leveling.imu_topic [†]	/livox/imu			IMU in the LIO body frame axes
leveling.window_sec	1	s	0.1..10	still window
leveling.max_rate_deg_s	3	deg/s	0.1..90	still: gyro rate below this
leveling.max_accel_spread	0.03		0.001..0.5	still: accelerometer std / mean norm below this
leveling.max_tilt_deg	20	deg	1..60	larger start tilts are refused (wrong frame?)
leveling.timeout_sec	15	s	0..600	then the bridge goes up unlevelled; 0: wait

Chapter 6

Perception

`dharmaperception` turns the LiDAR into two sector scans in `robot.frames.base`: a front scan for reactive avoidance (`perception/sector_scan`) and a fail-safe 360° scan for the protective field and the follower's motion guard (`perception/safety_scan`). It also throttles the camera for the fiducial detector.

6.1 Sector scan semantics

A `SectorScan` holds the nearest obstacle per angular sector; sector i is centred at $\text{angle_min} + (i + \frac{1}{2}) \text{angle_increment}$. `ranges[i] >= range_max` means nothing within range; `ranges[i] <= BLOCKED (0)` or `NaN` means blocked or unknown, and *every consumer treats it as an obstacle at the robot*. Every sector is `BLOCKED` when:

- the LiDAR is blind: fewer than `blind.min_points` points off the robot body, or (with `blind.min_region_points`) one of `blind.regions` azimuth regions empty;
- the sensor mount transform or, with `filter.level_source` `tf`, the roll and pitch of the base (`odom`→`base_link`) is missing;
- processing raised an exception, or no usable cloud arrived for `input.timeout_sec` (driver down, only malformed messages).

In 0.1 an all-zero scan was read as free by the protective field (review B1) and `NaN` sectors as free by the follower (F1); this convention removes both.

6.2 Pipeline

1. **Parse.** The serialized Livox `CustomMsg` is read with `numpy` (no per-point Python objects; every length validated, both endiannesses). `PointCloud2` goes through `dharmautils.ros.pointcloud.read_xyz`. A malformed message is dropped with a throttled error (review #4).
2. **Livox noise.** The tag byte marks each point per field (bits 0–1 adhesion between objects, 2–3 rain, fog or dust, 4–5 other; 0 high, 1 medium, 2 low confidence). Points at `filter.noise_level` in the enabled fields are dropped (review #12).
3. **Self filter.** Points inside `robot.footprint` grown by `filter.self_margin`.
4. **Level.** Points in `base_link` are rotated by the roll and pitch of the base in `odom` at the cloud stamp, so the band is relative to the floor while the gait pitches the body (finding P1).
5. **Height band.** `filter.height_min` to `filter.height_max` above the floor, the floor being `robot.base_height` below `base_link`. Optional `filter.ground_segmentation` estimates the floor per cell (lowest point within a maximum step) for ramps and thresholds.
6. **Sectors.** Each range is the k -th nearest point of the sector (`front.k`, `safety.k`, default 2): a single dust return does not stop the robot; a sector with fewer than k points is free.
7. **Obstacle memory.** Confirmed safety points are kept in `odom` for `memory.memory_sec` and merged into the safety scan while they are hidden below the lowest beam; an odometry jump clears the memory (review B4).

Why levelling matters. For a flat floor and `base_link` at height h pitched by ϑ , a floor point at horizontal range r appears at $z \approx -h + r \sin \vartheta$ and enters the band at

$$r > r_{\text{false}} = \frac{h + z_{\min}}{\sin \vartheta}. \quad (6.1)$$

Band margin $h + z_{\min}$	$\vartheta = 1^\circ$	2°	3°	5°
10 cm	5.7 m	2.9 m	1.9 m	1.15 m
5 cm	2.9 m	1.4 m	0.96 m	0.57 m

A trotting legged robot pitches by a few degrees, so in `base_link` the floor became an obstacle within the 4 m safety range (false stops), and pitching up hid low obstacles (missed detections). The 0.1 configuration also assumed $h = 0.35$ m in one file and 0.30 m in another; both now come from the profile.

6.3 Limits

- **Near-field blind zone.** A Mid-360 sees down to -7° : mounted 0.53 m above the floor it sees a 10 cm object only beyond about 3.5 m. The memory keeps objects seen earlier, but an object that appears inside the zone (placed there, or the robot switched on next to it) is not seen. Use a pitched mount or a near-field sensor (bumper, ToF, 2D scanner) where this matters.
- **Negative obstacles** (stairs down, holes, platform edges) are not detected: missing floor is not an obstacle.
- **Motion distortion.** The scan is not deskewed: a Mid-360 frame spans 100 ms, so a point is up to $v \cdot 0.1$ m off (8 cm at 0.8 m/s) and turning at ω moves a point at range r by up to $0.1 \omega r$. The latency from the first point to the published scan is the frame (100 ms), driver and transport (about 5–10 ms) and processing (about 10–15 ms for 20k points on one desktop core); `robot.limits.reaction_time` must stay above it and the field margins above the distortion.
- **Quantization.** The nearest point of a sector stands for the whole sector: 72 safety sectors are 5° wide (lateral error up to $r\Delta\theta/2$: 8.7 cm at 2 m, 17.5 cm at 4 m). Raise `safety.num_sectors` for a finer field.
- This is a software channel; it does not replace a safety-rated scanner (Section 9.1).

6.4 Nodes and parameters

Node	Role
<code>lidar_sector_node</code>	Livox CustomMsg (raw bytes, default) or PointCloud2 in; perception/sector_scan, perception/safety_scan, perception/obstacles out
<code>apriltag_throttle</code>	camera image and info in, image at rate_hz out
<code>apriltag</code> (apriltag_ ros)	tag frames on TF for localization

Parameters marked [†] are read-only at runtime (`live=False`); every other one can be changed with `ros2 param set` and is validated before it applies.

Parameter	Default	Unit	Range	Description
lidar_sector_node				
input.type [†]	livox		livox, pointcloud2	livox: raw Livox CustomMsg bytes; pointcloud2: any PointCloud2
input.livox_topic [†]	livox/lidar			Livox CustomMsg topic
input.pointcloud_topic [†]	/cloud_registered_body			PointCloud2 topic (frame fixed to the robot)
input.frame [†]	livox_frame			sensor frame when the header has none
input.timeout_sec [†]	0.5	s	0.05..10	no usable cloud for this long: every sector is published BLOCKED
front.num_sectors [†]	32		1..1440	sectors over the fov
front.fov_deg [†]	180	deg	1..360	centred ahead
front.range_min [†]	0.1	m	0..10	closer: ignored
front.range_max [†]	8	m	0.1..100	free beyond
front.k [†]	2		1..100	a sector needs k points; its range is the k-th nearest
safety.enabled [†]	true			publish perception/safety_scan
safety.num_sectors [†]	72		4..1440	sectors over 360 deg
safety.range_min [†]	0.1	m	0..10	closer: ignored
safety.range_max [†]	4	m	0.1..100	free beyond
safety.k [†]	2		1..100	a sector needs k points; its range is the k-th nearest
filter.self_margin [†]	0.05	m	0..1	footprint grown by this is the robot
filter.level_source [†]	tf		tf, none	tf: roll/pitch of base in odom (unknown -> BLOCKED); none: base assumed level
filter.tilt_max_age [†]	0.2	s	0.01..5	oldest odom->base usable for leveling
filter.height_min [†]	0.1	m	-1..3	band bottom above the floor
filter.height_max [†]	1.6	m	0..5	band top above the floor
filter.ground_segmentation [†]	false			estimate the floor height per cell (ramps, uneven floors)
filter.ground_cell [†]	0.3	m	0.05..5	ground segmentation cell size
filter.ground_max_step [†]	0.05	m	0..0.5	largest local floor offset accepted
filter.drop_adhesion_noise [†]	true			Livox tag bits 0-1 (glue points)
filter.drop_particle_noise [†]	true			Livox tag bits 2-3 (rain, dust)
filter.drop_other_noise [†]	false			Livox tag bits 4-5 (other)
filter.noise_level [†]	2		1..2	drop at this tag confidence: 1 medium, 2 low
memory.enabled [†]	true			remember obstacles that left the view

Parameter	Default	Unit	Range	Description
<code>memory.memory_sec[†]</code>	10	s	0..60	remembered points decay after this
<code>memory.radius[†]</code>	5	m	0.1..20	points this close are remembered (merged only while below the lowest beam)
<code>memory.resolution[†]</code>	0.05	m	0.01..1	memory cell size
<code>memory.max_cells[†]</code>	20000		100..1e+06	memory size bound
<code>memory.lower_fov_deg[†]</code>	-7	deg	-90..0	lowest beam elevation of the sensor (Mid-360: -7), for the near-field blind zone
<code>memory.max_jump[†]</code>	0.3	m	0..10	odom jump beyond the motion: forget
<code>blind.min_points[†]</code>	100		0..1e+06	fewer valid points in a cloud: blind
<code>blind.regions[†]</code>	4		1..36	azimuth regions around the sensor z axis
<code>blind.min_region_points[†]</code>	0		0..100000	fewer points in any region: blind (0: off)
<code>publish_markers[†]</code>	true			perception/obstacles for RViz

Chapter 7

Mapping and sites

`dharma_mapping` records occupancy maps from the LiDAR-inertial odometry, manages sites (one directory per mapped place with its map, the fiducials that anchor it and its routes) and teaches routes.

7.1 Sites

<code><sites_dir>/<site>/</code>	
<code>map.yaml, map.pgm</code>	occupancy map (map_server format)
<code>map.yaml.bak, map.pgm.bak</code>	the previous map (a loadable pair)
<code>tags.json</code>	fiducial poses in the map frame (localization
<code>/record_tags)</code>	
<code>routes/<date>_<time>.csv</code>	taught routes (<code>x, y, yaw_deg, dwell_sec > 0:</code>
<code>stop point)</code>	
<code>backup-<time>[-n]/</code>	what a mapping reset archived

A site's map frame is the frame its fiducials were recorded in. Site names match `^[a-z0-9][a-z0-9_]{0,47}$`; every path is derived from `sites_dir` and a site name, never taken from a request (review #12). The map pair is written crash safe: a temporary directory, `fsync`, `rename`; the YAML records the image size and a CRC, so a torn pair is refused on load and the `.bak` pair is used (review #5). A reset archives the old site into a unique `backup-<time>[-n]` directory, after all preconditions were checked (review #13). Broken site files are notes in `site/status`, never a crash (review #7).

7.2 Occupancy grid

7.2.1 Log-odds model

For a cell with occupancy probability p the grid stores $l = \log \frac{p}{1-p}$:

$$l_t = \text{clip} \left(l_{t-1} + \begin{cases} l_{\text{hit}} = 0.85 & \text{cell hit} \\ l_{\text{miss}} = -0.4 & \text{cell crossed by a ray} \end{cases}, [-2.0, 3.5] \right). \quad (7.1)$$

A cell is occupied above $\text{logit}(0.65) = 0.619$ (a single hit suffices) and free below $\text{logit}(0.35) = -0.619$ (two rays). A saturated cell becomes free only after $\lceil (3.5 + 0.619)/0.4 \rceil = 11$ crossing rays, about 5.5 s of observation at 2 Hz, so a person standing still for long while mapping stays in the map. Free space is carved only up to the nearest hit per angular bin (`scan.angular_bins, 720`): glass and open doors are not cleared. This is conservative by design.

7.2.2 Which scans and points are integrated

- **Height band.** Points between `scan.height_min` and `scan.height_max` above the floor, measured in the map frame (floor = `base_link` height in the map minus `robot.base_height`). This needs a level map frame: localization levels `odom` with the IMU, so `map` and `odom` are gravity-aligned and a pitching body never turns the floor into walls (finding O1).
- **Self filter.** Points inside the footprint grown by `scan.self_margin`.

- **Correction gate.** A scan is skipped when `map→odom` changes within `gate.correction_window_sec` around it by more than `gate.correction_tol_m` or `gate.correction_tol_deg`: its TF pose may be the old, the new or a blend of both (review #14).
- **Jump gate.** A sensor that moved more than $d_{\max} + v_{\max}\Delta t$ (or turned more than $\psi_{\max} + \omega_{\max}\Delta t$) since the previous scan jumped (odometry reset, bad correction). Integration then holds until a scan agrees with the map at its pose again (score above `gate.resume_score`), so a jump never smears the map or grows the grid (finding M1, review #8).
- **Growth cap.** The grid grows in steps of `grid.grow_m` and never beyond `grid.max_size_m` per side; hits beyond are dropped. In 0.1 a 300 m odometry jump grew the grid to 109 MB.
- **Optional refinement** (`refine.enabled`, off by default). The TF pose is matched on the current grid before integrating, gated on score and ambiguity. It sharpens walls under small pose errors, but its ambiguity gate separates a corridor from a room only by a factor of two, so it stays off until a test shows it helps.

There is no loop closure: map quality equals odometry plus localization quality. A pose graph over scan matches, with tags as landmarks, is future work.

7.2.3 Mapping sessions

`mapping/start` on a new site (or with `reset`) maps in the current map frame, or takes the current pose as origin when not localized. Continuing a site that already has a map or tags needs the robot localized in *its* frame: on one of its tags or by a pose estimate on its map; an `origin` localization at boot does not count (review #9). The session remembers its site and refuses a save into another one; a stop that the mapper does not confirm is a failure (review #4).

7.3 Route teaching

`route_recorder_node` (moved here from `dharma_odometry`: routes are site data) samples `map→base_link` while `route/start` is active and writes `<site>/routes/<date>_<time>.csv` incrementally and atomically (`dharma_utils.io.write_route`). It records only while `localization/status` is `LOCALIZED` (a TF that exists is not a localization); `route/mark_stop_point` makes the current pose a stop point with `dwel_sec`; file errors make the service fail instead of killing the node.

7.4 Interfaces

Interface	Type	
/map	OccupancyGrid, latched	the site map, or the live grid while mapping (only publisher: site manager)
mapping/grid	OccupancyGrid, latched	the live grid
mapping/status	MappingStatus, latched	scans, rejected, size, known area
site/status	SiteStatus, latched	site, sites, map, mapping, tags, routes, notes
route/saved	String	path of a saved route
site/list, site/load	ListSites, LoadSite	another site resets localization
mapping/start	StartMapping	reset archives the old site
mapping/stop, mapping/save	Trigger	
route/start, route/stop	Trigger	refused unless LOCALIZED
route/mark_stop_point	MarkStopPoint	

Used: `localization/status`, `localization/load_tags` (a path inside `sites_dir`), `localization/start`, `reset`, `assume_origin`. Internal: `grid_mapper/start|stop|save|load` (site names only).

7.5 Parameters

Parameters marked [†] are read-only at runtime (`live=False`); every other one can be changed with `ros2 param set` and is validated before it applies.

Parameter	Default	Unit	Range	Description
grid_mapper_node				
<code>sites_dir</code> [†]	<code>~/dharma/map</code>			one directory per site
<code>cloud_topic</code> [†]	<code>/cloud_registered_body</code>			deskewed scan in a frame fixed to the robot
<code>rate_hz</code> [†]	2	Hz	0.1..50	scans integrated
<code>publish_hz</code> [†]	1	Hz	0.1..10	mapping/grid while mapping
<code>max_cloud_age_sec</code> [†]	1	s	0.1..30	scans without TF this long: dropped
<code>save_margin_m</code> [†]	1	m	0..20	unknown border kept when saving
<code>grid.resolution</code> [†]	0.05	m	0.01..1	cell size
<code>grid.initial_size_m</code> [†]	20	m	1..1000	side of the first grid
<code>grid.grow_m</code> [†]	5	m	0.1..100	extra border added when it grows
<code>grid.max_size_m</code> [†]	150	m	1..2000	largest side; hits beyond are dropped (memory: (size / resolution) ² * 4 bytes)
<code>grid.log_odds_hit</code> [†]	0.85		0..10	added per hit
<code>grid.log_odds_miss</code> [†]	-0.4		-10..0	added per crossing

Parameter	Default	Unit	Range	Description
grid.log_odds_min [†]	-2		-50..0	clamp, free side
grid.log_odds_max [†]	3.5		0..50	clamp, occupied side
grid.occupied_threshold [†]	0.65		0.5..1	probability above which a cell is occupied
grid.free_threshold [†]	0.35		0..0.5	probability below which a cell is free
scan.voxel [†]	0.05	m	0..1	downsampling, 0: off
scan.min_range [†]	0.3	m	0..10	from the sensor
scan.max_range [†]	15	m	0.5..200	hits and free space beyond: ignored
scan.angular_bins [†]	720		8..7200	free space is carved to the nearest hit per bin
scan.height_min [†]	0.05	m	-1..3	band bottom above the floor
scan.height_max [†]	1.5	m	0..10	band top above the floor
scan.self_margin [†]	0.05	m	0..1	footprint grown by this is the robot
gate.max_jump_m [†]	0.5	m	0.01..100	sensor motion between scans beyond this plus max speed * dt: a jump
gate.max_jump_deg [†]	30	deg	1..180	yaw beyond this plus max_wz * dt
gate.correction_window_sec [†]	0.5	s	0..10	map->odom compared this long before the scan and at the newest time
gate.correction_tol_m [†]	0.02	m	0..1	map->odom change counted as correction
gate.correction_tol_deg [†]	0.5	deg	0..10	map->odom yaw change, same
gate.resume_score [†]	0.6		0..1	after a jump, a scan agreeing this well with the map resumes integration
refine.enabled [†]	false			match each scan on the current grid
refine.xy_window [†]	0.2	m	0.01..2	search window around the TF pose
refine.yaw_window_deg [†]	3	deg	0.1..30	yaw search window
refine.min_score [†]	0.5		0..1	mean likelihood needed to accept a match
refine.max_sigma_m [†]	0.045	m	0.001..1	largest accepted position sigma
refine.min_occupied [†]	200		10..1e+06	occupied cells near the robot needed to match
site_manager_node				
sites_dir [†]	~/dharma/map			one directory per site
site [†]	" "			site loaded at start; empty: the one of the last run
state_file [†]	~/dharma/site			remembers the active site
start_mapping [†]	false			map 'site' at start when it is a new site (mapping mode)
service_timeout_sec [†]	5	s	0.5..60	waiting for localization and mapper

Parameter	Default	Unit	Range	Description
refresh_sec [†]	2	s	0.5..60	site/status refresh (tags, routes)
route_recorder_node				
sites_dir [†]	~/dharma/map			routes go to <sites_dir>/<site>/routes
output_path [†]	" "			directory for routes while no site is active; empty: refuse
min_spacing_m	0.2	m	0.01..10	a new waypoint after this much motion
record_rate_hz [†]	10	Hz	0.5..100	pose sampling while recording
flush_sec	2	s	0.1..60	the route file is rewritten at most this often
max_tf_age_sec	0.5	s	0.01..10	older map->base: no sample

Chapter 8

Navigation

`dharma_navigation` plans paths on the site map, loads taught routes, follows paths around people and runs reactive avoidance. Its velocity output is `control/nav_cmd_vel` (the launch default; never the robot's `cmd_vel`, review F14/B2): the control mux decides what reaches the robot.

Node	Role
<code>planner_node</code>	<code>planner/go_to</code> : plan on <code>/map</code> , hand the path to the follower, replan around blockages
<code>path_loader_node</code>	<code>navigation/load_path</code> : read a route CSV inside <code>sites_dir</code> , re-sample and smooth it (stop points never move), publish it latched
<code>path_follower_node</code>	join, follow, final approach, stop points, pause reasons, motion guard; publishes <code>control/nav_cmd_vel</code>
<code>obstacle_avoidance_node</code>	a reactive policy on <code>perception/sector_scan</code> → <code>navigation/avoidance</code>

8.1 One braking model

Every speed limit in the stack uses the model of `dharma_utils.safety`: from speed v , with reaction time t_r and deceleration a , the robot needs

$$D(v) = vt_r + \frac{v^2}{2a}, \quad \text{and the fastest speed that stops within } d \text{ is } v^*(d) = a \left(-t_r + \sqrt{t_r^2 + \frac{2d}{a}} \right), \quad (8.1)$$

with $v^*(d) = 0$ for $d \leq 0$; the yaw-rate version uses the yaw deceleration and an angle. In 0.1 the path shift used $v = \sqrt{2ad}$ without the reaction time while the protective field used $D(v)$ (finding N3); at 1.0 m of room the former allowed 0.78 m/s where 0.66 m/s was right. The speed profile, the path shift, the motion guard and the protective field now share (8.1).

8.2 Planner

8.2.1 Costmap

`/map` becomes a costmap at `costmap.resolution`: occupied cells (and unknown ones with `costmap.unknown_is_obstacle`) are max-pooled onto the planning grid and live obstacles of the safety scan can be stamped in. With the distance d of a cell to the nearest obstacle and the radii from the footprint (finding N1),

$$r_\ell = r_{\text{in}} + m, \quad r'_c = r_c + m, \quad R = \max(R_{\text{infl}}, r'_c), \quad (8.2)$$

a cell is lethal for every heading when $d \leq r_\ell$; between r_ℓ and r'_c a collision depends on the heading and is checked with the oriented footprint; up to R the cost decays,

$$c(d) = \begin{cases} \infty & d \leq r_\ell \\ 1 + W \frac{e^{-k(d-r_\ell)} - e^{-k(R-r_\ell)}}{1 - e^{-k(R-r_\ell)}} & r_\ell < d < R, \\ 1 & d \geq R \end{cases} \quad W = 5, k = \text{costmap.cost_scaling}. \quad (8.3)$$

8.2.2 Search

A* [8], 8-connected without corner cutting; the step cost is the step length times the cost of the target cell and $f = g + wh$ with the octile heuristic $h = \max(\Delta_r, \Delta_c) + (\sqrt{2} - 1) \min(\Delta_r, \Delta_c)$. Since $c \geq 1$, h is admissible for $w = 1$ (optimal paths); for $w > 1$ a path costs at most w times the optimum. A robot inside the inflation walks out of it (`search.start_escape`); a goal inside an obstacle moves to the nearest cell within `search.goal_tolerance` where the robot fits at the goal heading, and arrival is tested against that snapped goal (review F8).

8.2.3 Post-processing and the footprint check

Line-of-sight shortcuts are accepted only when the largest cost along the line does not exceed that of the segment they replace; a line touching a cell corner counts both side cells, so it never slips through a diagonal pinch (review F17). Waypoints are placed every `search.spacing` and smoothed by gradient steps $y_i += \alpha(x_i - y_i) + \beta(y_{i-1} + y_{i+1} - 2y_i)$, discarded when they touch a lethal cell. Along the result, the oriented footprint is checked at the heading the follower will use (the path direction, the goal heading at the end); where it does not fit, the cell is blocked and the search runs again, up to `search.footprint_retries` times. For non-holonomic robots a lattice or hybrid A* planner can be added behind the same interface.

8.2.4 Goal session

`planner/go_to` plans, calls `navigation/set_path` and `navigation/start`, and monitors the follower: `BLOCKED` for `monitor.replan_after_sec` or `OFF_PATH` plans again with the obstacles of the safety scan, up to `monitor.max_replans`. A blind or stale safety scan is never read as clear (review F1): the goal fails. Every failure after `set_path` stops the follower (review F9). `planner/cancel` has its own callback group and a cancel flag checked between planning, `set_path` and `start` (review F10). The pose is used only when TF is younger than `tf_timeout_sec`; a parameter change rebuilds the costmap (review F15).

8.3 Path follower

The follower is a pure state machine, `Follower.tick(inputs) -> Decision`: the node feeds it the pose, the scans and the avoidance once per control tick and publishes the decision (command, status, stop events). The states are the `NavigationStatus` constants and an explicit table lists the allowed transitions (Fig. 8.1); a transition outside the table is a bug that raises in the tests.

8.3.1 Route, start and progress

`navigation/set_path` (planner) or the latched `navigation/waypoints` (loader) give a route of at least two points (review F17); a running follower starts over on a new route from where the robot is. `navigation/start` keeps the progress when the robot is still near it, otherwise it locates the robot on the route, weighing the heading so the leg it walks along wins over a parallel one. Beside the route (up to `join.max_distance`) it joins first (`JOINING`); further away the start is refused. Start hooks (`hooks.start`) are called asynchronously and the robot moves only when all succeeded.

Progress is searched in a short window ahead of the current one, against the line the robot is meant to be on (route plus path offset), and its advance per tick is capped: a parallel leg never captures it while the robot is shifted (review F5).

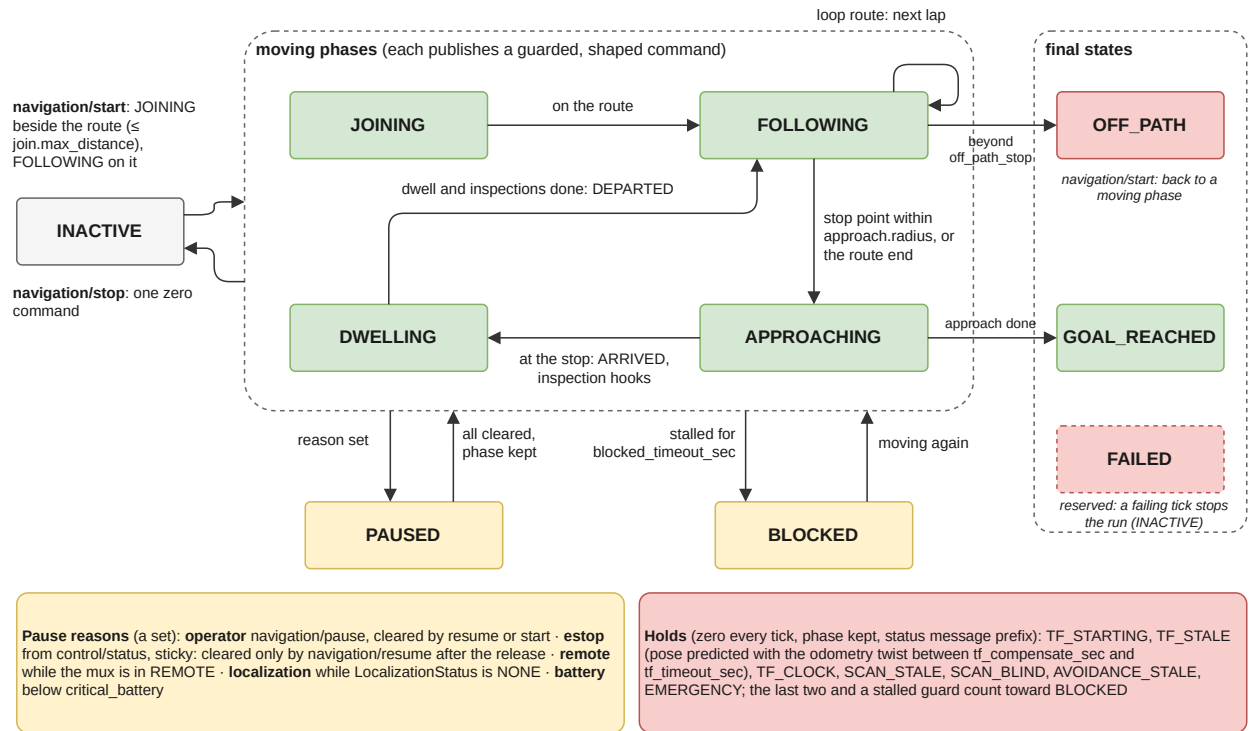


Figure 8.1: States of the path follower, pause reasons and holds.

8.3.2 Speed profile

At each route point i with curvature κ_i :

1. curvature limit $v_i \leq \sqrt{a_{\text{lat}}/|\kappa_i|}$, clipped to $[v_{\text{min}}, v_{\text{cruise}}]$;
2. approach speed at stop points and at the end;
3. backward pass $v_i \leq \sqrt{v_{i+1}^2 + 2a_{\text{dec}} \Delta s_i}$;
4. the braking model (8.1), reaction time included, before every stop;
5. forward pass $v_{i+1} \leq \sqrt{v_i^2 + 2a_{\text{acc}} \Delta s_i}$ out of the start and every stop (new in 0.2.0: the profile is consistent and predicts remaining time).

Speeds and accelerations are capped by `robot.limits`. The speed regulator integrates a signed error and resets on new paths; in 0.1 it integrated $|\omega| \geq 0$, so a positive gain would have driven the speed to its minimum on long routes (finding N2).

8.3.3 Controllers

Controllers implement `PathController` and register in the `dharma.controllers` registry; their parameters appear as `<name>.*` on the follower. Every controller applies the avoidance the same way (finding N4): `heading_offset` (rad) rotates the commanded direction of travel, `yaw_rate_bias` (rad/s) adds to the yaw rate.

Controller	Law
holonomic (default)	translation toward a carrot at arc length $s + L$ with $v = \min(v^*, \ c - p\)$; heading toward the route direction a <code>heading_lookahead</code> ahead with a proportional, saturated yaw rate. The cross-track error obeys $\dot{e} = -ve/\sqrt{L^2 + e^2} \approx -(v/L)e$, so $e(t) = e_0 e^{-t/\tau}$ with $\tau = L/v$: 95 % after $3L$ of travel, independent of the speed. Lateral and heading loops are decoupled, which is why it does not oscillate on legged robots.
pure_pursuit [9]	curvature $2y/d^2$ toward the carrot at the requested speed
pid	PID on the heading error toward the carrot, with a clamped integral (anti-windup, finding N2)
lqr	error model $x = [e_y, e_\psi]^\top$, $x_{k+1} = \begin{bmatrix} 1 & v\Delta t \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} u_k$, gain from the discrete Riccati equation per speed, $\omega = v\kappa - Kx$
stanley [10]	$\omega = v\kappa - k_h(e_\psi + \arctan \frac{ke}{v+k_s})$ toward the offset route
mppi [11]	sampled yaw-rate sequences around the previous plan (warm start), unicycle roll-outs, cost from route distance, carrot distance, intrusion into the clearance around the safety-scan points and effort, average $u^* = \sum_i w_i u_i$ with $w_i \propto \exp(-(J_i - J_{\min})/\lambda)$; seeded RNG (finding N2)

8.3.4 Path shift

With `avoidance_mode path_shift` (omnidirectional robots), obstacle points of the 360° safety scan are expressed in route (Frenet) coordinates (s, d) [12]: arc length along the route and signed distance to its left. A lateral offset o is clear when no point lies in the corridor

$$|d - o| < w + c, \quad s_{\text{robot}} - b \leq s \leq s_{\text{robot}} + f + \ell, \quad (8.4)$$

(w half width, c clearance, b, f rear and front of the footprint, ℓ look-ahead). Points beside the body block an offset outright; points ahead limit how far the robot can drive. Among the candidates $o \in \{-o_{\max}, \dots, o_{\max}\}$,

$$o^* = \arg \min_{o \text{ clear}} |o| + \lambda |o - o_{\text{prev}}|, \quad (8.5)$$

with a switch margin for other offsets than the current one; a move back toward the route is taken only after it stayed proposed for `path_shift.return_delay_sec`. Without a clear offset the robot keeps the one with the largest free distance (nearest the current one on ties) and slows. The commanded offset moves toward the target at `path_shift.offset_rate`. The speed limit is $v^*(d_{\text{free}} - m)$ where d_{free} is the smallest free distance over the target offset, the commanded offset *and the measured lateral offset* (review F4), so a robot still sliding is not sped up by the offset it has not reached. A blind scan stops the robot. Near a stop point it stays on the route and waits for an obstacle instead of passing it (review F6).

8.3.5 Reactive avoidance

With `reactive`, `obstacle_avoidance_node` runs one policy of the `dharma.avoidance` registry on `perception/sector_scan` and publishes an `AvoidanceCommand`: heading offset (rad), yaw-rate bias (rad/s), lateral bias (m/s, omni), speed scale and an emergency flag (an obstacle closer than `shared.emergency_distance` in the cone $\pm$ `shared.emergency_cone_deg`). Blocked sectors are obstacles.

Policy	Principle
braitenberg	danger left and right steers, danger ahead or beside slows
vfh [13] (default)	only when the frontal cone is blocked, aim at the nearest edge of a wide enough gap
potential_field [14]	obstacles inside the safe distance repel, $F = -\sum_j \eta \left(\frac{1}{r_j} - \frac{1}{r_0} \right) \frac{1}{r_j^2} \hat{u}_j$; the sideways part steers and pushes, danger ahead slows
follow_gap [15]	bubble the nearest obstacle, aim at the centre of the widest gap

The outputs are smoothed asymmetrically: the speed scale drops at once and only rises through the filter, the emergency flag is never filtered (review F13).

Is the emergency cone enough? With $t_r = 0.25$ s the stopping distances are

v	$a = 0.5$	$a = 0.6$	0.35 m of room?
0.35 m/s (cruise in 0.1)	0.21 m	0.19 m	yes
0.60 m/s	0.51 m	0.45 m	no
1.0 m/s	1.25 m	1.08 m	no

so the cone (0.75 m from `base_link`, about 0.35 m from the nose) is a second line; the motion guard and the protective field limit the speed before.

8.3.6 Stop points

Stop points (`dwell_sec > 0`) are detected by `progress` (within `approach.radius` along the route) and by distance minus the path offset; a stop on a later leg that passes close never triggers early, and one is skipped only once the `progress` passed it by `approach.skip_margin` (review F6). At a stop: final x , y , yaw `approach`, `ARRIVED`, inspection hooks (asynchronous; answers for an earlier stop or route are dropped), `dwel`, `DEPARTED`. Route processing keeps stop points closer than 3 cm apart (review F12).

8.3.7 Supervisor, motion guard and command shaper

- **Pose.** `TF_STARTING` without a pose; `TF_STALE` above `safety.tf_timeout_sec`; between `safety.tf_compensate_sec` and the timeout the pose is predicted forward with the odometry twist instead of stopping; `TF_CLOCK` for a pose from the future (clock mismatch).
- **Scans and avoidance.** `SCAN_STALE`, `SCAN_BLIND` (blocked sectors), `AVOIDANCE_STALE`, `EMERGENCY` (held for `safety.emergency_release_sec`).
- **Blocked and off path.** No progress for `safety.blocked_timeout_sec` is `BLOCKED`; more than `safety.off_path_stop` from the route beyond the path offset is `OFF_PATH` (the robot slows from `safety.off_path_slow`).
- **Motion guard.** Every command, in every phase (following, joining, final approach), is limited along its direction by the 360° safety scan with the footprint and the braking model, and the yaw rate by the free turning angle (review F7; 0.1 checked joining and the final approach with the front cone only).
- **Command shaper.** Robot envelope (x and y scaled together, $v_y = 0$ for diff robots), smoothing of yaw rate and sideways speed, slew limits from `robot.limits`, non-finite values to zero (review F11).

Zero commands are published once on entering `INACTIVE` or a final state, every tick while paused or held, on stop, on shutdown (a hook) and when a control tick raises (review F2). All state-changing callbacks

and the control loop share one mutually exclusive callback group and a lock (review F3); timers (dwell, approach) are frozen while paused (review F7).

8.3.8 Pause reasons

A set of reasons, each with its own way of clearing (review F16):

Reason	Set and cleared
operator	navigation/pause; cleared by navigation/resume or start
estop	control/status reports the e-stop; sticky: cleared only by navigation/resume after the release, so the robot never drives off on its own after an e-stop
remote	while the mux is in REMOTE
localization	while localization/status is NONE; cleared when localized again
battery	below critical_battery; below low_battery a loop route finishes its lap and stops

Start is refused while estop or localization hold.

8.4 Interfaces

Interface	Type
planner/go_to, planner/plan	GoTo, PlanPath
planner/cancel	Trigger
planner/status, planner/path, planner/costmap	PlannerStatus, Path, OccupancyGrid
navigation/set_path, navigation/load_path	SetPath, LoadPath
navigation/start, stop, pause, resume	Trigger
navigation/status	NavigationStatus, latched
navigation/stop_point	StopPointEvent
navigation/waypoints, navigation/path	WaypointPath, Path
navigation/avoidance	AvoidanceCommand
control/nav_cmd_vel	Twist
in: /map, perception/safety_scan, perception/sector_scan, control/status, localization/status, battery, odom	state, pause reasons, progress loaded route avoidance → follower to the control mux

8.5 Parameters

Parameters marked [†] are read-only at runtime (live=False); every other one can be changed with `ros2 param set` and is validated before it applies.

Parameter	Default	Unit	Range	Description
planner_node				
costmap.resolution	0.1	m	0.01..1	planning cell size, the map is max-pooled to it
costmap.margin	0.1	m	0..1	clearance added to the footprint (keep >= half a cell)
costmap.inflation_radius	0.8	m	0..5	paths keep away from walls up to here
costmap.cost_scaling	3	1/m	0..50	how fast the wall cost falls off
costmap.unknown_is_obstacle	true			never plan through unmapped space
costmap.occupied_threshold	50		1..100	map values from here are obstacles
search.goal_tolerance	0.5	m	0..5	a goal in an obstacle moves to free space this close
search.start_escape	1	m	0..5	a robot inside the inflation walks out up to here
search.heuristic_weight	1		1..5	> 1: faster search, slightly longer paths
search.timeout_sec	5	s	0.1..60	search time limit
search.spacing	0.1	m	0.02..1	distance between waypoints
search.smooth_weight_data	0.1		0..1	pull toward the search path
search.smooth_weight_smooth	0.3		0..0.49	pull toward the neighbours
search.footprint_retries	5		0..20	searches around poses where the oriented footprint collides
monitor.arrival_tolerance	0.35	m	0.01..5	GOAL_REACHED counts this close to the plan goal
monitor.stop_on_arrival	true			stop the follower at the goal
monitor.replan_after_sec	3	s	0.1..300	follower BLOCKED this long: plan again
monitor.max_replans	5		0..100	per goal, then failed
monitor.cancel_after_sec	1	s	0.1..60	follower INACTIVE this long: cancelled
use_live_obstacles	true			replans see the obstacles of the safety scan
scan_timeout_sec [†]	1	s	0.05..10	older safety scans are ignored
tf_timeout_sec [†]	0.5	s	0.05..10	an older robot pose is unknown
publish_costmap	false			planner/costmap for RViz (debug)
status_hz	2	Hz	0.1..50	planner/status while active
service_timeout_sec	5	s	0.1..60	calls to the follower
path_loader_node				
sites_dir [†]	~/dharma/map			routes are loaded from inside this directory only

Parameter	Default	Unit	Range	Description
route [†]	" "			route CSV loaded at start (inside sites_dir)
remember_file [†]	~/ .dharma/ last_route			last loaded route, reloaded at start ("": off)
process	true			resample and smooth routes between stop points
processing.spacing	0.1	m	0.02..1	point spacing after resampling
processing.smooth_weight_data	0.05		0..1	pull toward the recording
processing.smooth_weight_smooth	0.4		0..0.49	pull toward neighbours
processing.max_deviation	0.3	m	0..5	warn when smoothing moves the route further
republish_sec	2	s	0..60	navigation/path republish for RViz (0: off)
path_follower_node				
control_rate [†]	20	Hz	1..200	loop rate
controller	holonomic			path controller (dharma.controllers)
avoidance_mode	path_shift		path_shift, reactive, off	path_shift: slide sideways (holonomic)
loop_route	false			restart a closed route at its end (patrol)
loop_close_distance	1	m	0..20	end this close to the start: a closed route
require_localization	true			pause while localization/status is NONE
low_battery	0.2		0..1	below: finish the lap, no new one
critical_battery	0.1		0..1	below: pause (reason battery)
battery_hysteresis	0.05		0..0.5	recharge above + this
speed.cruise	0.4	m/s	0.01..5	cruise speed
speed.min_speed	0.1	m/s	0..5	floor of the curve limit
speed.approach_speed	0.15	m/s	0.01..5	speed at stop points and the route end
speed.lateral_accel	0.3	m/s ²	0.01..10	curve limit $v = \sqrt{a / \text{curvature}}$
speed.accel	0.3	m/s ²	0.01..20	comfortable acceleration
speed.decel	0.3	m/s ²	0.01..20	comfortable braking
speed.smoothing	0.5		0.05..1	weight of a new yaw rate / sideways command (1: off)
safety.stop_margin [†]	0.3	m	0..5	gap kept to obstacles when stopping
safety.yaw_margin_deg [†]	5	deg	0..90	turning room kept to obstacles
safety.scan_timeout_sec [†]	0.5	s	0.05..5	older safety scans stop the robot
safety.avoidance_timeout_sec [†]	0.5	s	0.05..5	older avoidance (reactive) stops it
safety.tf_timeout_sec [†]	0.5	s	0.05..5	an older pose stops the robot

Parameter	Default	Unit	Range	Description
safety.tf_compensate_sec [†]	0.1	s	0..5	older poses are predicted forward
safety.clock_tolerance_sec [†]	0.1	s	0..5	a pose from further in the future: clock
safety.emergency_release_sec	1	s	0..30	an emergency stop holds this long after it clears
safety.blocked_timeout_sec	10	s	0.5..600	no progress this long: BLOCKED
safety.off_path_slow	0.6	m	0.05..10	further from the route: slow to the floor speed
safety.off_path_stop	1.5	m	0.1..20	further from the route: OFF_PATH, stopped
path_shift.max_offset	0.8	m	0..5	furthest from the route
path_shift.step	0.1	m	0.02..1	candidate offset spacing
path_shift.look_ahead	2	m	0.1..20	route ahead of the front edge that must be free
path_shift.clearance	0.1	m	0..2	gap kept beside the body
path_shift.offset_rate	0.3	m/s	0.01..5	how fast it slides onto a new offset
path_shift.switch_cost	0.5		0..10	keeps its side (hysteresis)
path_shift.switch_margin	0.05	m	0..1	extra clearance a new offset needs
path_shift.return_delay_sec	0.5	s	0..30	a way back must stay proposed this long
path_shift.stop_margin	0.4	m	0..5	stop this far before an obstacle (front edge)
approach.radius	0.5	m	0.05..5	this close (along the route) the approach starts
approach.skip_margin	0.5	m	0.05..10	progress this far past a stop: skipped
approach.timeout_sec	20	s	1..600	give up converging (paused time not counted)
approach.kp_xy	1.2	1/s	0..10	gain on the position error
approach.kp_yaw	1.5	1/s	0..10	gain on the yaw error
approach.max_speed	0.2	m/s	0.01..2	approach speed cap
approach.max_yaw_rate	0.6	rad/s	0.01..5	yaw rate cap
approach.min_speed	0.04	m/s	0..1	floor to beat the robot deadband
approach.position_tolerance	0.05	m	0.005..1	arrived within
approach.yaw_tolerance_deg	3	deg	0.1..45	aligned within
join.max_distance	3	m	0..50	a robot this far walks onto the route (0: off)
join.timeout_sec	60	s	1..600	give up and follow from where it is
join.on_route	0.15	m	0.01..2	joined within this
hooks.start [†]	[]			called on start; the robot waits for every success

Parameter	Default	Unit	Range	Description
<code>hooks.stop[†]</code>	<code>[]</code>			called on stop (not waited for)
<code>hooks.inspection[†]</code>	<code>[]</code>			called at every stop point
<code>hooks.timeout_sec[†]</code>	5	s	0.1..120	start and stop hook timeout
<code>hooks.inspection_timeout_sec</code>	20	s	0.1..600	a stop waits this long for inspections
<code>regulator.enabled</code>	false			scale the speed down when turning hard
<code>regulator.reference_yaw_rate</code>	0.3	rad/s	0..5	turning slower than this needs no slow-down
<code>regulator.kp</code>	0.5	s/rad	0..10	scale per rad/s above it
<code>regulator.ki</code>	0	1/rad	0..10	scale per integrated rad
<code>regulator.integral_limit</code>	1	rad	0..10	anti-windup clamp
<code>regulator.min_scale</code>	0.3		0..1	never slower than this fraction
<code>processing.spacing</code>	0.1	m	0.02..1	point spacing after resampling
<code>processing.smooth_weight_data</code>	0.05		0..1	pull toward the recording
<code>processing.smooth_weight_smooth</code>	0.4		0..0.49	pull toward neighbours
<code>processing.max_deviation</code>	0.3	m	0..5	warn when smoothing moves the route further
<code>markers</code>	true			publish RViz markers
obstacle_avoidance_node				
<code>policy</code>	vfh			avoidance policy (dharma.avoidance)
<code>shared.safe_distance</code>	1	m	0.1..20	obstacles closer than this shape the command
<code>shared.emergency_distance</code>	0.75	m	0..10	obstacle in the cone closer than this: stop
<code>shared.emergency_cone_deg</code>	60	deg	1..360	frontal cone of the emergency stop
<code>shared.max_heading_offset_deg</code>	60	deg	0..180	largest turn of the direction of travel
<code>shared.smoothing</code>	0.8		0..0.99	0..1 filter on steering and speeding up (never on braking)
<code>braitenberg.steering_gain</code>	1	rad	0..10	turn per unit danger
<code>braitenberg.velocity_gain</code>	1		0..10	slow-down per unit danger
<code>braitenberg.steering_zone_center_deg</code>	45	deg	0..180	
<code>braitenberg.steering_zone_width_deg</code>	60	deg	1..360	
<code>braitenberg.velocity_zone_center_deg</code>	60	deg	0..180	

Parameter	Default	Unit	Range	Description
braitenberg.velocity_zone_width_deg	90	deg	1..360	
braitenberg.following_zone_width_deg	30	deg	1..360	
vfh.min_gap_width_deg	20	deg	1..180	narrowest passable gap
vfh.steering_gain	1		0..5	fraction of the gap heading taken
vfh.velocity_gain	1		0..10	slow-down per unit frontal danger
potential_field.repulsion_gain	0.5		0..10	strength of the repulsion
potential_field.steering_gain	0.8	rad	0..5	turn per atan(force)
potential_field.lateral_gain	0.3		0..5	sideways m/s per unit force
potential_field.max_lateral	0.2	m/s	0..2	cap on the sideways push
potential_field.velocity_gain	1		0..10	slow-down per unit frontal danger
follow_gap.bubble_radius	0.45	m	0..5	safety bubble around the nearest obstacle
follow_gap.steering_gain	1		0..5	fraction of the gap heading taken
follow_gap.velocity_gain	1		0..10	slow-down per unit frontal danger
markers	true			publish RViz markers

8.5.1 Controller parameters

Parameter	Default	Unit	Range	Description
path_follower_node: holonomic				
holonomic.lookahead	0.6	m	0.1..5	carrot ahead along the route; longer = gentler
holonomic.heading_lookahead	0.6	m	0.1..5	the heading follows the route this far ahead
holonomic.kp_yaw	1.5	1/s	0..10	yaw rate per rad of error
holonomic.max_lateral_speed	0.25	m/s	0..5	sideways speed cap (0: never strafe)
holonomic.max_reverse_speed	0.15	m/s	0..5	backward speed cap when the route is behind
holonomic.min_alignment_scale	0.25		0..1	translation factor while the heading is far off
path_follower_node: pure_pursuit				
pure_pursuit.lookahead	1	m	0.1..5	carrot ahead along the route
path_follower_node: pid				
pid.lookahead	1	m	0.1..5	carrot ahead along the route

Parameter	Default	Unit	Range	Description
pid.kp	1.5	1/s	0..20	proportional gain
pid.ki	0	1/s ²	0..20	integral gain
pid.kd	0.2		0..20	derivative gain
pid.integral_limit	0.5	rad s	0..10	anti-windup clamp of the error integral
path_follower_node: lqr				
lqr.q_lateral	1		0..100	weight on the cross-track error
lqr.q_heading	0.5		0..100	weight on the heading error
lqr.r_yaw_rate	0.5		0.001..100	weight on the yaw rate (higher = gentler)
path_follower_node: stanley				
stanley.k_cross_track	1		0..20	cross-track gain
stanley.k_soft	0.2	m/s	0.01..5	softening so low speeds do not over-steer
stanley.k_heading	1.5	1/s	0..20	steering to yaw rate gain
path_follower_node: mppi				
mppi.horizon_steps	15		2..100	steps per rollout
mppi.step_sec	0.1	s	0.01..1	rollout step
mppi.samples	200		10..5000	rollouts per tick
mppi.yaw_rate_std	1	rad/s	0.01..5	sampling noise
mppi.temperature	0.05		0.001..100	softmax sharpness of the normalized cost
mppi.window	4	m	0.5..20	route ahead in the cost
mppi.obstacle_weight	5		0..1000	cost per m of intrusion
mppi.clearance	0.5	m	0..5	rollouts closer to an obstacle pay for it
mppi.seed [†]	0		0..	random seed (reproducible)

Chapter 9

Control and safety

`dharma_control` contains `control_mux_node`, the **only** publisher of the robot's `cmd_vel`. It arbitrates navigation (AUTO) and one operator source (REMOTE), keeps every command inside the robot's envelope, limits it with a protective field on the 360° safety scan and holds a latched e-stop with a hardware input and re-arm.

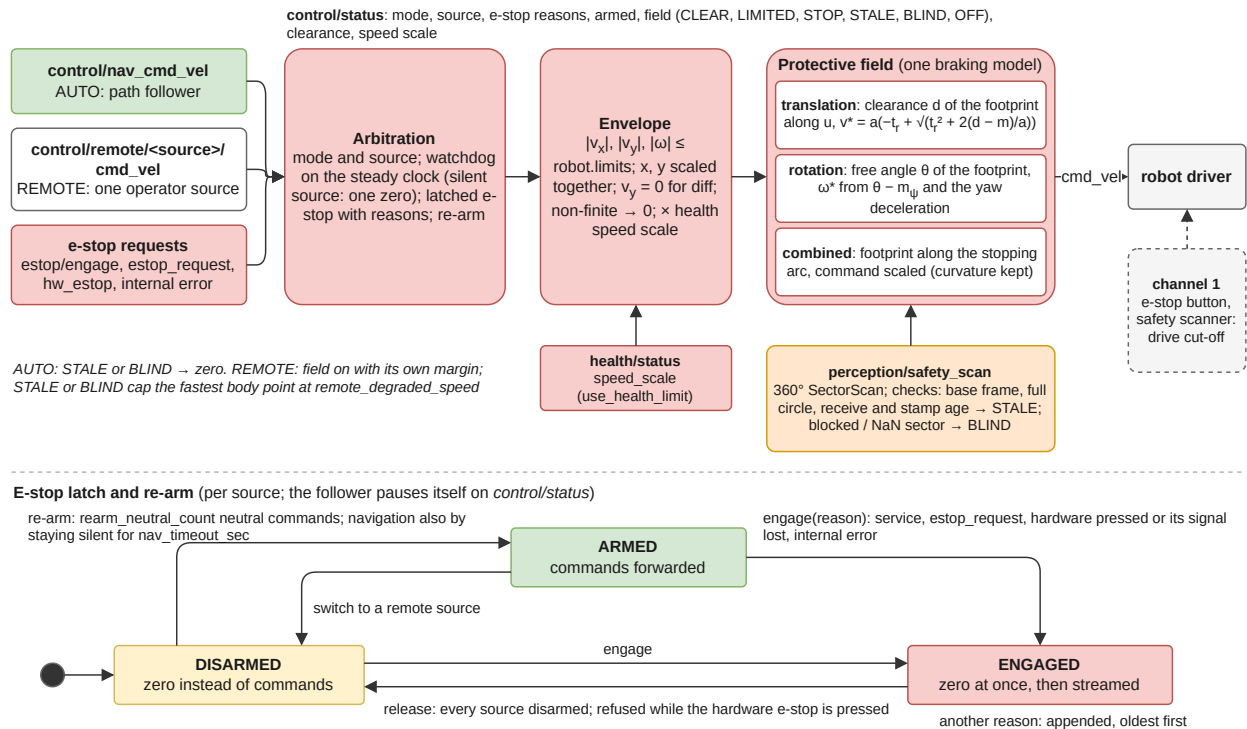


Figure 9.1: The safety layer and the e-stop latch with re-arm.

9.1 Two-channel safety architecture

The control mux is a software safety function in Python over DDS. It is *not* certified and must never be the only protection of a robot that drives among people (finding C2). Products for public spaces use two channels:

Channel	What	Stops the robot by
1 (hardware)	e-stop buttons and, where required, a safety-rated scanner or bumper wired to the drive's safe torque off or brake input, designed to ISO 13849-1 [16] (for example PL d) or IEC 61508 [17]; ISO 13482 [18] applies to personal care robots	cutting the drive, independently of any software
2 (<code>control_mux_node</code>)	envelope, protective field, watchdogs, latched software e-stop: smooth, early, configurable	publishing zero on <code>cmd_vel</code> and streaming zero while engaged

Wire channel 1's state to `control/hw_estop` (Bool, true = pressed): while it is pressed the software e-stop cannot be released, and after the button is released an operator still has to release the software e-stop explicitly. With `hw_estop_timeout_sec > 0` a silent publisher of `control/hw_estop` engages the e-stop (the signal is then required). The driver must stop on its own command timeout, which covers a crash of the mux until it respawns.

9.2 Pipeline

9.2.1 Arbitration

AUTO forwards `control/nav_cmd_vel`; REMOTE forwards `control/remote/<source>/cmd_vel` of the selected source only (`remote_sources`, default [`handheld`, `dashboard`]; review #8: several operators on one topic). `control/set_mode` with the current mode and source is a no-op (review #13); a change publishes one zero and resets the watchdog; unknown modes or sources are rejected. The **watchdog** gives the active source one zero when it is silent longer than `nav_timeout_sec` or `remote_timeout_sec`. All timeouts and the e-stop hold run on the steady clock: clock jumps and `use_sim_time` never disable them (review #4).

9.2.2 E-stop latch and re-arm

- **Engage.** `control/estop/engage` (Trigger), `control/estop_request` (String reason), `control/hw_estop`, a silent hardware signal, or an internal error. Reasons accumulate, oldest first, without duplicates (review #14). Zero is published at once and streamed at `estop_hold_hz`. Mode switches are allowed while engaged but forward nothing.
- **Release.** `control/estop/release` (Trigger) is refused while the hardware e-stop is pressed; it clears the reasons and disarms every source. There is no `SetBool`: in 0.1 a default request released the e-stop and anyone could release anyone's e-stop (review #3).
- **Re-arm.** At start, after a release and after switching to a remote source, a source is disarmed: it must send `rearm_neutral_count` consecutive neutral commands (every axis below `neutral_eps`) before anything it sends is forwarded, so a stick held during a release never moves the robot (review B3). Navigation also re-arms by staying silent for `nav_timeout_sec`.
- **Navigation pauses itself.** The mux never calls navigation services (review B3: the e-stop was resumed through `set_mode`); the follower reads `control/status:estop` is a sticky pause reason there (cleared by `navigation/resume`), remote a transient one.

9.2.3 Envelope

$|v_x| \leq \max_{vx}$, $|v_y| \leq \max_{vy}$ (0 for diff), $|\omega| \leq \max_{wz}$ from `robot.limits`; v_x and v_y are scaled together, keeping the direction. Any non-finite axis makes the command zero (review #5: NaN limits disabled the envelope). With `use_health_limit`, every axis is multiplied by `HealthStatus.speed_scale` (stale health: no change). Commands are clamped before the field.

9.2.4 Protective field

AUTO always; REMOTE when `field.in_remote` (default true since 0.2.0, with `field.remote_margin`). The model is speed and separation monitoring [19,20] with the braking model of Section 8.1.

Translation. For the commanded direction $\hat{u}$, the clearance d_i is the distance the footprint can move before touching a scan point (`translation_clearance`): points ahead of the leading edge count when they lie within the body corridor widened by `field.lateral_margin`; a point beside the body (behind the

leading edge, inside the corridor) or inside the footprint gives 0. The speed is limited to $v^*(d_t - m)$ with $m = \text{field.margin}$; below field.min_speed it stops. With $a = 0.6 \text{ m/s}^2$, $t_r = 0.25 \text{ s}$ and $m = 0.15 \text{ m}$:

free distance d_t	0.3 m	0.5 m	1.0 m	2.0 m
allowed speed v^*	0.30 m/s	0.52 m/s	0.87 m/s	1.35 m/s

Rotation (finding C1). In 0.1 only v_x, v_y were scaled: a pure rotation passed unchecked, although the corners of the footprint sweep the ring $r_{\text{in}} < r \leq r_c$ (0.28 to 0.49 m) and hit obstacles beside the robot. Now the free angle θ_f (`rotation_room`: the ring points are rotated in 1° steps against the turning direction until one enters the footprint; a point already inside gives 0) bounds the yaw rate: $|\omega| \leq \omega^*(\theta_f - m_\psi)$ with the yaw deceleration `max_yaw_accel` and $m_\psi = \text{field.yaw_margin}$; below `field.min_yaw_rate` it stops.

Combined motion. For (v_x, v_y, ω) the robot keeps the twist for t_r and then brakes both axes in proportion (the arc is kept) over $T_b = \max(v/a, |\omega|/\alpha)$. It covers the constant-twist arc

$$x(u) = \frac{v_x \sin \omega u - v_y (1 - \cos \omega u)}{\omega}, \quad y(u) = \frac{v_x (1 - \cos \omega u) + v_y \sin \omega u}{\omega}, \quad \psi(u) = \omega u \quad (9.1)$$

up to $u = t_r + T_b/2$, exactly $D(v)$ on a straight line, plus an extension of $\min(m/v, m_\psi/|\omega|)$. The footprint (widened by the lateral margin) is checked at poses along the arc; a point already beside the body may be passed but not approached closer than 2 cm. If the first contact is at u_c , scaling by s keeps the curvature and covers $st_r + s^2 T_b/2$, so the largest clear scale is

$$s = \frac{-t_r + \sqrt{t_r^2 + 2 T_b (u_c - u_{\text{ext}})}}{T_b}. \quad (9.2)$$

9.2.5 Scan validation and degraded sensing

`perception/safety_scan` is accepted only in `robot.frames.base` with full-circle coverage (review #7). It is STALE when invalid, not received for `field.scan_timeout_sec` (monotonic receive age) or when its stamp is older than `field.scan_max_age_sec` or more than `field.scan_clock_tolerance_sec` in the future. Any blocked or NaN sector makes it BLIND (review B1: an all-zero scan was read as clear).

- AUTO: STALE or BLIND $\rightarrow$ zero; no scan at all $\rightarrow$ zero.
- REMOTE: the fastest point of the body ($v + |\omega|r_c$) is capped at `field.remote_degraded_speed`; with a partly blind scan the field still applies to the points it sees, and a translation toward a blocked sector is refused.

`control/status` reports the field state (CLEAR, LIMITED, STOP, STALE, BLIND, OFF), the clearance and the applied speed scale.

9.2.6 Fail-safe behaviour

An exception in any callback engages the e-stop with the reason `internal_error` and publishes zero (review #6); on shutdown the node publishes zero three times. Safety parameters are read-only at runtime and validated at start (decelerations > 0 , timeouts > 0 , margins ≥ 0 , finite values): the node refuses to start on a bad value (review #5: `max_decel: 0` raised `ZeroDivisionError`).

9.3 Safety case

Hazards of the navigation function, their mitigations in the stack and the tests that verify them. Test names are `file::function` in the package's `test/` directory (u: unit, r: node test). The software mitigations

are channel 2; residual risks need channel 1 or a sensor the stack does not have.

	Hazard	Mitigation	Verification
H1	collision with an obstacle ahead while driving	protective field translation limit with one braking model; follower motion guard; speed profile	<code>u/test_field::test_wall_</code> <code>ahead_limits_with_the_</code> <code>shared_stopping_model,</code> <code>u/test_follower_safety::</code> <code>test_guard_checks_every_</code> <code>direction_of_the_</code> <code>footprint, sim E1 (Section 11.6)</code>
H2	body corner hits an obstacle while turning in place	rotation limit from the swept ring (C1)	<code>u/test_field::</code> <code>test_rotation_next_to_</code> <code>an_obstacle_is_limited,</code> <code>r/test_control_mux_node::</code> <code>test_field_limits_</code> <code>rotation_beside_an_</code> <code>obstacle</code>
H3	collision along a curved motion	stopping-arc check of combined motion	<code>u/test_field::</code> <code>test_combined_motion_</code> <code>checks_the_arc</code>
H4	blind LiDAR read as free space	BLOCKED sectors at the source, BLIND → zero in AUTO, creep in REMOTE (B1, F1)	<code>u/test_pipeline::</code> <code>test_blind_lidar_blocks_</code> <code>every_sector,</code> <code>u/test_scan::</code> <code>test_all_sectors_zero_</code> <code>is_blind_not_clear,</code> <code>u/test_field::test_blind_</code> <code>scan_stops_auto, sim E2</code> <code>(Section 11.6)</code>
H5	perception stops publishing or lags	receive and stamp age checks, STALE → zero; BLOCKED watchdog in perception; respawn	<code>u/test_scan::</code> <code>test_receive_age_on_</code> <code>monotonic_time,</code> <code>u/test_scan::test_header_</code> <code>age_old_and_future,</code> <code>r/test_lidar_sector_</code> <code>node::test_silent_input_</code> <code>publishes_blocked_scans</code>
H6	scan in a wrong frame or with partial coverage	frame and coverage validation (#7)	<code>u/test_scan::test_wrong_</code> <code>frame_is_rejected,</code> <code>u/test_scan::</code> <code>test_partial_coverage_</code> <code>is_rejected</code>
H7	navigation keeps the last moving command (stop, crash, silence)	watchdog zero; zero on INACTIVE, stop and shutdown (F2, #6)	<code>u/test_arbiter::</code> <code>test_watchdog_zeroes_</code> <code>once_per_silence,</code> <code>u/test_follower::</code> <code>test_stop_zeroes_once_</code> <code>and_start_resumes,</code> <code>r/test_control_mux_node::</code> <code>test_shutdown_hook_</code> <code>publishes_zeros</code>

	Hazard	Mitigation	Verification
H8	robot moves when an e-stop is released (held stick, resumed navigation)	re-arm with neutral commands; sticky estop pause in the follower (B3)	<code>u/test_arbiter::test_release_requires_rearm_and_never_forwards_the_held_command,</code> <code>u/test_follower_safety::test_estop_is_sticky_and_remote_is_transient</code>
H9	e-stop released by a default or foreign request	Trigger services only; release refused while the hardware e-stop is pressed (#3, C2)	<code>u/test_arbiter::test_hardware_estop_engages_and_blocks_release,</code> <code>r/test_control_mux_node::test_hardware_estop_blocks_release</code>
H10	watchdog disabled by a clock jump or simulated time	steady-clock timers (#4)	<code>r/test_control_mux_node::test_watchdog_and_hold_run_on_steady_time_with_sim_time</code>
H11	invalid safety parameter (zero deceleration, NaN limit)	validated typed configuration, read-only at runtime, refuse to start (#5, F11)	<code>r/test_control_mux_node::test_bad_values_refuse_to_start,</code> <code>r/test_control_mux_node::test_safety_parameters_are_read_only</code>
H12	software fault in the mux	e-stop on callback errors; respawn; driver command timeout (channel 1)	<code>r/test_control_mux_node::test_callback_error_engages_the_estop</code>
H13	floor seen as an obstacle or low obstacle hidden by body pitch	levelled height band, gravity-aligned odom (P1, O1)	<code>u/test_filtering::test_pitched_body_does_not_see_the_floor,</code> <code>r/test_lidar_sector_node::test_pitched_body_does_not_block_on_the_floor</code>
H14	low object near the robot below the lowest beam	obstacle memory in odom (B4); residual: objects appearing inside the zone → pitched mount or near-field sensor	<code>u/test_memory_blindness::test_memory_keeps_an_object_the_sensor_no_longer_sees; residual risk documented</code>
H15	negative obstacles (stairs down, edges)	not detected by the stack; residual: channel 1 sensor or site restriction	–
H16	planned path too narrow for the body when turning	radii from the footprint, oriented footprint check (N1)	<code>u/test_planning::test_door_narrower_than_the_body_is_refused,</code> <code>u/test_planning::test_oriented_footprint_against_a_wall</code>
H17	robot pose stale or wrong (TF latency, clock offset)	TF age checks with prediction and stop, TF_CLOCK; per-edge diagnosis in health	<code>u/test_follower_safety::test_tf_status_messages,</code> <code>u/test_follower_safety::test_slightly_old_pose_is_predicted_with_the_twist</code>
H18	wrong localization correction moves the robot off its path	chi-square gate on the robot pose; OFF_PATH stop (L1, #1)	<code>u/test_estimator::test_gate_rejects_outliers_and_accepts_consistent_fixes_far_from_origin, sim E3 (Section 11.6)</code>

	Hazard	Mitigation	Verification
H19	passing an obstacle at a stop point or while shifted	path shift limited by the measured offset; wait at stop points (F4, F6)	u/test_shift:: test_measured_offset_limits_the_speed, u/test_follower:: test_stop_point_with_an_obstacle_waits_instead_of_passing
H20	software safety mistaken for a certified function	two-channel architecture documented; product safety case required (C2)	review of the product design

9.4 Interfaces

Interface	Type	
cmd_vel	Twist	out: the robot's command
control/status	ControlStatus, latched	on change and at status_hz
control/nav_cmd_vel	Twist	in: navigation (AUTO)
control/remote/<source>/cmd_vel	Twist	in: one per remote_sources entry
perception/safety_scan	SectorScan	in: 360° scan in the base frame
control/estop_request	String	in: engage with a reason
control/hw_estop	Bool	in: hardware e-stop, true = pressed
health/status	HealthStatus	in, with use_health_limit
control/set_mode	SetMode	auto, or remote with a source
control/estop/engage, control/estop/release	Trigger	

9.5 Parameters

All parameters of the mux are read-only at runtime. Measure `robot.limits.max_decel`, `max_yaw_accel` and `reaction_time` on the robot before it drives on its own.

Parameter	Default	Unit	Range	Description
control_mux_node				
mode [†]	auto		auto, remote	mode at start
remote_sources [†]	[handheld, dashboard]			operator sources, each on control/re- mote/<source>/cmd_vel; the first is the default
nav_timeout_sec [†]	0.5	s	0.02..5	zero once when navigation is silent
remote_timeout_sec [†]	0.3	s	0.02..5	zero once when the operator is silent
estop_hold_hz [†]	10	Hz	1..100	zero commands while engaged

Parameter	Default	Unit	Range	Description
rearm_neutral_count [†]	3		1..100	consecutive neutral commands the source must send before it drives again
neutral_eps [†]	0.01	m/s, rad/s	0.0001..0.2	neutral: every axis below
hw_estop_timeout_sec [†]	0	s	0..10	engage when control/hw_estop is silent this long (0: the input is optional)
use_health_limit [†]	false			scale every command by health/status speed_scale
health_timeout_sec [†]	3	s	0.1..60	older health/status is ignored
status_hz [†]	2	Hz	0.2..50	control/status period (also on change)
control_mux_node				
field.in_remote [†]	true			apply the field in REMOTE too (remote_margin)
field.margin [†]	0.15	m	0..2	gap kept to an obstacle in AUTO
field.remote_margin [†]	0.1	m	0..2	gap kept to an obstacle in REMOTE
field.lateral_margin [†]	0.1	m	0..1	extra width of the swept path
field.yaw_margin [†]	0.1	rad	0..1	turn kept short of a contact
field.min_speed [†]	0.02	m/s	0..0.5	a slower allowed speed stops instead
field.min_yaw_rate [†]	0.05	rad/s	0..1	a slower allowed turn stops instead
field.remote_degraded_speed [†]	0.15	m/s	0..1	REMOTE with a stale or blind scan: fastest point of the body
field.scan_timeout_sec [†]	0.5	s	0.05..5	scan receive age (monotonic) limit
field.scan_max_age_sec [†]	0.5	s	0.05..5	scan header stamp age limit
field.scan_clock_tolerance_sec [†]	0.1	s	0..1	accepted negative stamp age (clocks)

Health and diagnostics

`dharmah_health` watches the stack and the computer. Its single node, `health_monitor_node`, runs every check each `period_sec` (1 s) and publishes a typed health/status (`HealthStatus`, latched: level OK / WARN / ERROR, issues with one line per problem, errors first, and `speed_scale` in $[0,1]$) and a diagnostics array with one `DiagnosticStatus` per checked item, named `health: <item>`, for `rqt_robot_monitor`. It reads the shared `robot.*` profile for the frames of the odometry chain.

10.1 Checks

Every check family is a `Check` class in `core/checks/` that returns one finding per item, OK included, so a diagnostic never disappears when it recovers.

Sensors (`sensor:<topic>`). Each entry of `sensors.topics` is `topic:min_hz[:max_age_sec]`. The topics are subscribed *raw*: the header stamp is read from the serialized CDR bytes (both byte orders), so a point cloud is never deserialized. The rate over `sensors.window_sec` is $\hat{f} = (N-1)/(t_N - t_1)$ (`dharmah_utils.math.RateMeter`); a topic is silent ($\hat{f} = 0$) when $t - t_N > \max(1, 3/\hat{f})$ and slow when $\hat{f} < \text{rate_tolerance} \cdot f_{\min}$ (0.8). The stamp age matters as much as the rate: a 10 Hz topic with stamps 2 s in the past passed the 0.1 checks and is now late. A stamp more than `max_future_sec` ahead of the clock is reported as a clock offset. A topic that is not published, or whose type is not installed, warns with the reason.

TF (`tf:<parent>-><child>`). The ages of every dynamic edge (`TfCache.edge_ages`) are kept over `tf.window_sec` and classified (`core/checks/tf.py`): a fresh edge is OK; a stamp more than `tf.max_future_sec` ahead is a clock offset (`cause: clock`); a late edge whose stamp stopped advancing is a stopped publisher (`stopped`); late ages that stay within `offset_jitter_sec` of each other are a steady source clock offset (`clock`), e.g. “camera_init→body 1.8 s old, steady (source clock offset?)”; late ages that vary are late delivery (`load`). Edges listed in `tf.edges` must exist; an edge that disappears is reported as gone. The robot’s `odom→base_link` chain gets its own finding.

Clock (`clock:ntp`, off by default). With `clock.enabled` the node runs `chronyc` tracking every `clock.period_sec` and warns when the reported offset exceeds `clock.max_offset_sec` (50 ms), when the clock is not synchronized or when the command fails or times out.

State (`state:*`). The typed statuses: odometry LOST is an error, INITIALIZING and DEGRADED warn; localization NONE and UNCERTAIN warn (review #9: 0.1 reported OK while the robot was not localized); control errors on an engaged e-stop (with its reasons) and on a STALE or BLIND protective field. A status that never arrived or is older than `state.timeout_sec` warns. The 360° safety scan is an error when it is older than `scan_timeout_sec` or blind (every sector BLOCKED or NaN) and a warning when some sectors are blocked; a scan that never arrived warns during `startup_grace_sec` and is an error afterwards. An empty topic parameter turns its check off.

Nodes (`node:<name>`). Every `nodes.expected` entry is a fully qualified name `/namespace/name`, compared with the running nodes including their namespaces (review #13). Missing nodes warn during the start-up grace and are errors afterwards; a node running twice warns.

System (`system:*`). Temperature, memory, free disk at `disk_path`, load per CPU, CPU busy share and Jetson GPU load, each against its limit (a limit of 0 turns the reading off; CPU and GPU are off by default). Thermal zones are read only when their type matches `system.thermal_zones` (CPU, GPU, junction, x86 package): some Jetson zones report bogus values (PMIC-Die at 100°C). Readings outside $-40..150^{\circ}\text{C}$ are ignored. A reading that cannot be obtained is a warning, never OK (review #17).

Products add checks without forking: a `Check` subclass registered as an entry point in the group `dharma_health.checks` and named in `extra_checks` (Section 14.2 shows the mechanism for a controller).

Actionable output (finding H1). The 0.1 monitor published JSON in a `std_msgs/String` that nothing acted on. The report (`core/report.py`) takes the worst level of all findings and the lowest speed scale any finding asks for: an error asks for 0, a warning for the lowest matching `degradation.rules` entry (`<pattern>=<scale>`, fnmatch on the item name) or `warn_scale` (1). The defaults halve the speed while the LiDAR or the estimator is slow or late, or a TF edge is late: `sensor:/livox/lidar=0.5`, `sensor:/Odometry=0.5`, `tf:*=0.5`. The control mux multiplies its commands by the scale when `use_health_limit` is set; a health status older than `health_timeout_sec` (3 s) is ignored (scale 1). Health is a diagnostic aid, not a safety function: the protective field and the e-stop never depend on it.

10.2 TF latency budget

The follower looks up `map→base_link` at the latest available time. `tf2` returns the transform at the *latest common time* of the chain, the newest stamp available on every edge, and the pose is stale when its age exceeds `safety.tf_timeout_sec` (0.5 s). With E the dynamic edges, τ_e their stamps and Δ_e the clock offsets of their sources,

$$\text{age}(t) = t_{\text{now}}^{\text{consumer}} - \min_{e \in E} \tau_e, \quad \tau_e = t_e^{\text{measured}} + \Delta_e, \quad (10.1)$$

$$\text{age} \approx \max_{e \in E} \left(\underbrace{L_e^{\text{acq}} + L_e^{\text{proc}} + L_e^{\text{tx}} + L_e^{\text{queue}}}_{\text{latency}} - \Delta_e \right) + \underbrace{\Delta_{\text{consumer}}}_{\text{clock offset}} + \underbrace{(t_{\text{now}} - t_{\text{received}})}_{\text{publish period}}. \quad (10.2)$$

In Dharma two edges are dynamic: `map→odom`, stamped with `now()` by localization at 20 Hz (almost always fresh), and `camera_init→body`, stamped by the estimator with the end time of the LiDAR scan from the LiDAR’s own clock. The LIO edge therefore dominates:

Contribution	Typical
acquisition (one Mid-360 frame)	≈ 100 ms
LIO processing	20–80 ms
DDS transport and queueing	a few ms, more under load
until the next control tick	up to one period
normal age	0.15–0.35 s (limit 0.5 s)
unsynchronized LiDAR clock	$+\Delta$, added directly

10.2.1 Causes, most likely first

Cause	Typical symptom	How to confirm
T1. LiDAR clock not synchronized with the host (no PTP, gPTP or GPS)	constant age independent of load; or extrapolation into the past or future errors; never in simulation (the simulator stamps with <code>now()</code> unless latency is modelled)	compare <code>header.stamp</code> of <code>/livox/lidar</code> or <code>/Odometry</code> with <code>date +%s.%N</code> ; a difference $\gg 0.2$ s or the year 1970 means T1
T2. Clocks of two computers differ (Jetson and PC, NUC, laptop)	only nodes on the other machine see stale poses	<code>chronyc tracking</code> on both; <code>ros2 run tf2_ros tf2_monitor map base_link</code> on each
T3. LIO latency grows (full CPU, thermal throttling)	age varies with load, often while mapping or with many Python nodes	<code>tf2_monitor</code> (average and maximum delay per edge), <code>ros2 topic delay /Odometry</code> , <code>tegrastats</code> , <code>health/status</code> (temperature, CPU)
T4. <code>/tf</code> queue in Python listeners	age grows gradually, then catches up; worst when <code>/tf</code> carries leg joints at hundreds of Hz (every Python node deserializes all of them under the GIL)	<code>ros2 topic hz /tf</code> , <code>ros2 topic bw /tf</code>
T5. An edge stops temporarily	at start (the gate holds the estimator), on estimator respawn, or (0.1) while <code>record_tags</code> blocked <code>map</code> → <code>odom</code> for 2 s (L2)	<code>odometry/status</code> <code>INITIALIZING / LOST</code> ; <code>localization log</code>
T6. Mixed <code>use_sim_time</code>	only in simulation or rosbag replay: some nodes on <code>/clock</code> , some on wall time	<code>ros2 param get <node> use_sim_time</code> for every node

10.2.2 Diagnosis in ten minutes

- Measure the sensor clock offset.** Print `header.stamp` of one `/livox/lidar` message and compare it with the host clock at reception. A robust estimate is the median of $\hat{\Delta} = t_{\text{received}} - \tau$ over 10 s; normally $\hat{\Delta} \approx 0.10\text{--}0.15$ s (acquisition and transport). Repeat for `/livox/imu`, `/Odometry` and the camera info.
- Find the old edge.** `ros2 run tf2_ros tf2_monitor map base_link` shows the average and maximum delay per edge. A large and stable delay of `camera_init`→`body` points to T1; a fluctuating one to T3 or T4.
- Check between machines.** `chronyc tracking`: the offset must be below 1 ms on a LAN.
- Check the load.** `tegrastats`, `ros2 topic hz /tf`, `ros2 topic bw /tf`.
- Repeat standing still and driving.** A constant offset means a clock problem, an offset that grows while moving means load.

10.2.3 Fixes

In the system (T1, T2).

- PTP for the Mid-360: run `ptp41` as master on the Jetson NIC connected to the LiDAR, with hard-

ware timestamping when `ethtool -T` supports it, and `phc2sys` when the NIC clock must follow the system clock. The LiDAR stamps then share the host time base and FAST-LIO, perception and localization are consistent.

- Without PTP: restamp in the driver, $\tau' = t_{\text{received}} - \hat{L}$ with the nominal latency $\hat{L}$ (scan duration plus transport), keeping the per-point time offsets for deskewing. Better than a free-running device clock, but its jitter enters the stamps. FAST-LIO's `time_sync_en` only aligns the LiDAR with the IMU, not with the host.
- Several computers: `chrony` with the Jetson as local server (or both on the same NTP server), `makestep` at boot, and the `chrony` check in health.
- Camera: make the driver use the host clock (RealSense: global time enabled), so the tag age limit and the odometry lookup at the tag stamp do not fail.

In the stack (T3–T5), all in 0.2.0:

- per-edge ages in `TfCache` and in health, so the message names the edge and the likely cause;
- stamp-age checks in health, not only rates;
- latency compensation in the follower: for ages between `safety.tf_compensate_sec` and the timeout the pose is predicted to the current time with the odometry twist, ${}^{odom}\mathbf{T}_{base}(t_{\text{now}}) \approx {}^{odom}\mathbf{T}_{base}(\tau) \exp(\hat{\xi}(t_{\text{now}} - \tau))$, and the robot stops only above the timeout;
- distinct states `TF_STARTING` (the gate holds), `TF_STALE` (age) and `TF_CLOCK` (a pose from the future), so an operator knows what to fix;
- no blocking service callbacks (L2), so `map→odom` never pauses;
- a simulator that stamps with a latency, so regressions of T1 and T3 show in CI (Chapter 11).

Reducing the `/tf` load (joint states at most 30 Hz or on a separate topic; heavy Python nodes reading `localization/pose` instead of the whole `/tf`) and real-time priority with core isolation for the estimator help further.

10.3 Parameters

Parameters marked [†] are read-only at runtime (`live=False`); every other one can be changed with `ros2 param set` and is validated before it applies.

Parameter	Default	Unit	Range	Description
health_monitor_node				
<code>period_sec</code> [†]	1	s	0.1..10	health/status and diagnostics period
<code>startup_grace_sec</code>	30	s	0..600	missing nodes and inputs only warn this long
<code>diagnostics_prefix</code>	health			"<prefix>: <check>" in <code>rqt_robot_monitor</code>
<code>extra_checks</code> [†]	[]			registered checks to add (entry points <code>dharmah.health.checks</code>)
<code>sensors.topics</code> [†]	<code>[/livox/ lidar:5.0, /Odometry: 8.0]</code>			"topic:min_hz[:max_age_sec]" (max_age_sec 0: no stamp check)

Parameter	Default	Unit	Range	Description
sensors.rate_tolerance	0.8		0..1	a rate below this share of min_hz is an issue
sensors.window_sec [†]	5	s	0.5..60	rate window
sensors.max_age_sec	0.5	s	0..60	stamp age limit of entries without one; 0: off
sensors.max_future_sec	0.1	s	0..60	stamps further ahead of the clock: clock offset
sensors.discover_period_sec [†]	2	s	0.1..60	look for unsubscribed topics
system.thermal_zones	[cpu*, gpu*, tj*, x86_pkg_temp]			thermal zone types read (fnmatch, any case); others are ignored, some are bogus
system.max_temp_c	85	degC	0..150	hottest zone
system.max_memory_percent	92	%	0..100	memory in use
system.min_disk_free_gb	5	GB	0..1e+06	free at disk_path
system.disk_path	~			file system to watch (~ and \$VAR expanded)
system.max_load_per_cpu	2		0..100	1 min load average / CPUs
system.max_cpu_percent	0	%	0..100	CPU busy share
system.max_gpu_percent	0	%	0..100	Jetson GPU load
nodes.expected	[]			"/namespace/name" of every node that must run
state.odometry_topic [†]	odometry/status			OdometryStatus
state.localization_topic [†]	localization/status			LocalizationStatus
state.control_topic [†]	control/status			ControlStatus
state.safety_scan_topic [†]	perception/safety_scan			SectorScan (360)
state.timeout_sec	3	s	0.1..600	a status older than this is stale
state.scan_timeout_sec	1	s	0.05..60	a safety scan older than this is stale
tf.enabled [†]	true			listen to TF
tf.edges	[]			"parent->child" dynamic edges that must exist (every dynamic edge is watched)
tf.max_age_sec	0.5	s	0.01..60	newest transform age
tf.max_future_sec	0.1	s	0..60	stamps further ahead of the clock: clock offset
tf.window_sec	5	s	1..60	age history per edge
tf.offset_jitter_sec	0.2	s	0..10	late ages steadier than this: a clock offset
clock.enabled [†]	false			run the command
clock.command [†]	chronyc tracking			prints "System time : x seconds"
clock.period_sec	10	s	1..3600	run it this often

Parameter	Default	Unit	Range	Description
<code>clock.timeout_sec</code>	1	s	0.1..10	command timeout
<code>clock.max_offset_sec</code>	0.05	s	0..60	offset limit
<code>degradation.rules</code>	[sensor: /livox/lidar= 0.5, sensor: /Odometry=0. 5, tf:*=0.5]			"<issue name pattern>=<scale>" for warnings (fnmatch); the lowest match wins
<code>degradation.warn_scale</code>	1		0..1	scale of warnings without a rule

Chapter 11

Simulation

`dharma_sim` is a software-in-the-loop robot: one node, `sim_robot_node`, replaces the robot driver, the Livox Mid-360, its IMU and the LiDAR-inertial estimator, so the whole stack runs without hardware, in CI and on a laptop. It is kinematic (no Gazebo, no GPU). A product runs its own bringup against it with `odometry method:=external` and drives it like the real robot; its end-to-end tests run the same way.

11.1 Why the simulator models errors

The 0.1 simulator had perfect odometry and stamped everything with `now()` (finding S1). The code paths that handle drift (localization corrections, L1), map smearing (M1), a tilted start (O1) and stale TF (T1, T3) were therefore never exercised. Version 0.2.0 models them, with a seed so every run is reproducible.

11.2 Models

All models are pure Python and numpy in `dharma_sim/core` and unit-tested without ROS; `simulation.py` runs them on one clock.

Body (`body.py`). Omnidirectional or differential from `robot.kinematics` (a diff robot never strafes), with the acceleration limits of the profile. It integrates the *measured* time step as an exact arc, clamped to `max_dt` (0.1 used a fixed step per wall-timer tick, so the simulated robot slowed down under load, review #14). A command older than `cmd_timeout_sec` brakes the body to a stop, like the driver. A contact of the footprint with the world stops the body and latches `sim/collision` (an obstacle dropped onto the robot counts as a contact).

World (`world.py`). Walls (polylines with a height) and circles (pillars, people, boxes) with a height, from a YAML world in `config/worlds/`; dynamic circles from the scenario or from `sim/obstacles`. Ray casting is correct for origins inside a circle (0.1 used $t = -b - \sqrt{\text{disc}}$ there and was blind, review #10), for tangent rays and for circles behind the origin.

LiDAR (`lidar.py`). A Mid-360: `lidar.rings` elevations over the vertical field of view from -7° to 52° at the mount height (an obstacle lower than the lowest beam is not seen, as on the robot), `azimuth_rays` rays per ring, 0.1–40 m, floor returns of the downward rings, Gaussian range noise and random point dropout from the seeded RNG. The cloud is dense enough for perception (well above `blind.min_points` in the height band of the mall world).

Odometry drift (`odometry.py`). Every true step ΔT of the body is measured as

$$\xi = ((1+s)\Delta x, (1+s)\Delta y, \Delta\psi + b\Delta d) + n, \quad n \sim \mathcal{N}(0, \Sigma\Delta s), \quad {}^{odom}\mathbf{T}_{base} \leftarrow {}^{odom}\mathbf{T}_{base} \oplus \xi, \quad (11.1)$$

with the scale error s , the yaw bias b per metre, the distance Δd and the travel $\Delta s = \Delta d + w_\psi|\Delta\psi|$ (`rotation_weight` w_ψ , so turning in place drifts too). The variance grows linearly with the distance (a random walk, standard deviation $\propto \sqrt{d}$) and the biases accumulate with it. An odometry restart starts a new origin at the current pose, like a restarted estimator.

Latency (`latency.py`). Sensor messages carry $\tau = t - L$ and show the state of that time (an interpolated pose history), so a late stamp is late data, not only a late label; commands act L after they arrive and become zero once stale. A clock jump back clears the command queue.

IMU (`imu.py`). Specific force in g and yaw rate on `/livox/imu`, so the bridge levels at once. The IMU sits at $(0.011, 0.02329, -0.04412)$ m in `livox_frame` (Livox manual); FAST-LIO's `extrinsic_T` is the LiDAR in the IMU frame, its negative. 0.1 used FAST-LIO's T_{LL} with the wrong sign (review #11).

The body is planar: no gait, pitch or roll, no ceiling, no negative obstacles. Experiments that depend on body motion (the pitched floor of H13) or on the real estimator therefore need recorded data or the robot.

11.3 Robot-side contract

On the robot	In the simulator
driver	subscribes <code>/<ns>/cmd_vel</code> ; publishes <code>/<ns>/odom</code> (twist, <code>robot.frames.odom→base</code>) and <code>/<ns>/battery</code>
Mid-360 and FAST-LIO2	<code>/cloud_registered_body</code> (PointCloud2 in <code>body</code>), <code>/Odometry</code> and TF <code>camera_init→body</code> , <code>/livox/imu</code>
URDF	static TF <code>base_link→livox_frame→livox_imu_frame</code>
a tag in view	publishes <code>/initialpose</code> at the true pose while localization/status is NONE or not yet published (<code>initial_pose_hz, 0: off</code>)
inspection at a stop point	<code>sim/inspect</code> (<code>std_srvs/Trigger</code>) answers <code>clear</code>
people and boxes	<code>sim/obstacles</code> (PoseArray in the map frame) replaces the dynamic circles
the truth	<code>sim/ground_truth</code> (PoseStamped in <code>robot.frames.map</code>), <code>sim/collision</code> (Bool, latched)

With the localization bridge (`odom→camera_init`, `body→base_link`) this gives `odom→base_link`. Perception publishes BLOCKED scans unless it gets that chain; the mux forwards nothing without a fresh, stamped, 360° safety scan in `base_link`. The simulator publishes no `/livox/lidar` (the raw driver topic): the hub's `robot_bringup/config/sim.yaml` points health at `/cloud_registered_body` instead.

11.4 Scenarios and site fixture

Scenarios are YAML files in `config/scenarios/` (times in seconds since start), loaded by the node (`scenario_file`, a path or a name) and by tests. A scenario may set the seed (it replaces the seed parameter, so the same file gives the same run), override the latency and odometry noise parameters, add circles that appear, move with (v_x, v_y) and leave, silence sensors (`lidar`, `odometry`, `imu` or `all`) for a duration, and restart the odometry at given times. Independent generators are derived from the seed for the LiDAR, the odometry and the IMU (`numpy.random.SeedSequence`), so changing one model does not change the others' noise.

Scenario	Seed	Content
drift	1	$\sigma_{x,y} = 0.01 \text{ m}/\sqrt{\text{m}}$, $\sigma_\psi = 0.3^\circ/\sqrt{\text{m}}$, 1 % scale error, $0.1^\circ/\text{m}$ yaw bias; localization corrections (L1)
person_crossing	2	a person crosses the south corridor (8–30 s), a 0.5 m box stays on the route from 20 s: avoidance, protective field, replanning
sensor_faults	3	latency 0.15 s LiDAR, 0.05 s odometry and commands; LiDAR (1.5 s), odometry (0.8 s) and full (1 s) dropouts; an odometry restart at 40 s

The site fixture (`site_fixture.py`) rasterizes a world into a loadable site: `map.pgm` and `map.yaml` (`map_server` trinary, with size and CRC), `tags.json` with the world's tag plates and `routes/*.csv` (`dharma_utils.io`). The launch argument `site:=mall_loop` draws it into `sites_dir` (default `/tmp/dharma_sim/sites`) when the site has no map yet, including the taught loop config/`routes/mall_loop.csv` with one stop point; `dharma_mapping`'s site store reads it unchanged.

11.5 Launch

```
ros2 launch dharma_sim sim.launch.py robot_params:=robot.yaml site:=
mall_loop
ros2 launch dharma_sim sim.launch.py scenario:=person_crossing rviz:=true
ros2 topic pub --once /sim/obstacles geometry_msgs/msg/PoseArray \
  "{poses: [{position: {x: 2.0, y: -3.0}}]}"
```

Arguments: `params_file`, `robot_params`, `use_sim_time`, `log_level` (the launch contract of Section 3.3), `world`, `scenario`, `robot_ns` (empty: from the parameter files), `site`, `sites_dir` and `rviz` (the `dharma_rviz` layout with the driver topics remapped).

11.6 Evaluation protocol

The experiments of the paper are closed-loop simulations built from the pure `core/` modules of the packages: `dharma_sim`'s world, body, LiDAR, odometry drift, latency and scenario models drive the perception, control, localization and navigation cores at the node rates, without ROS, deterministic per seed. E1 measures the stopping distance against $D(v)$, E2 a blind LiDAR in AUTO and REMOTE, E3 the localization error with and without the EKF and scan matching under seeded odometry drift, E4 the path shift clearance and completion time with standing and crossing people. One end-to-end run of the real nodes (`ros2 launch robot_bringup sim.launch.py`) checks the composition. The scripts, the raw results and the seeds of every condition are in the `experiments/` folder of the Dharma hub repository; the measured results are reported in the paper's evaluation (Section VI of `docs/paper`) and not duplicated here.

11.7 Parameters

Parameters marked [†] are read-only at runtime (`live=False`); every other one can be changed with `ros2 param set` and is validated before it applies.

Parameter	Default	Unit	Range	Description
sim_robot_node				
<code>world_file</code> [†]	" "			world YAML path or name in <code>config/worlds</code>
<code>scenario_file</code> [†]	" "			scenario YAML path or name; empty: none
<code>robot_ns</code> [†]	<code>robot</code>			driver namespace: <code>/<ns>/cmd_vel</code> , <code>odom</code>
<code>seed</code> [†]	0		0..	RNG seed (a scenario may set it)
<code>rate_hz</code> [†]	100	Hz	5..1000	physics
<code>odom_rate_hz</code> [†]	20	Hz	1..500	/Odometry
<code>imu_rate_hz</code> [†]	50	Hz	0..500	0: off

Parameter	Default	Unit	Range	Description
imu_topic [†]	/livox/imu			IMU (sensor_msgs/Imu, in g)
max_dt [†]	0.1	s	0.001..1	longest physics step (stalls)
cmd_timeout_sec [†]	0.5	s	0.05..10	older command: the body stops
obstacle_radius	0.25	m	0.01..5	sim/obstacles size
obstacle_height	1.7	m	0.01..5	sim/obstacles height
battery_start [†]	0.9		0..1	charge at start
battery_drain_per_hour [†]	0.2		0..10	charge lost per hour
initial_pose_hz [†]	1	Hz	0..10	/initialpose at the true pose while localization is NONE; 0: off
lio_odom_frame [†]	camera_init			FAST-LIO world frame
lio_body_frame [†]	body			FAST-LIO body frame (the IMU)
lidar.rate_hz [†]	10	Hz	0.5..50	scans per second
lidar.frame [†]	livox_frame			URDF frame of the LiDAR
lidar.imu_frame [†]	livox_imu_frame			URDF frame of the built-in IMU
lidar.mount_xyz [†]	[0.13011, -0.02329, 0.17598]	m		livox_frame in base_link
lidar.imu_xyz [†]	[0.011, 0.02329, -0.04412]	m		IMU in livox_frame (Livox Mid-360 manual; FAST-LIO extrinsic_T is its negative)
lidar.azimuth_rays [†]	360		8..3600	rays per ring over 360 deg
lidar.rings [†]	16		1..128	elevations over the vertical fov
lidar.fov_min_deg [†]	-7	deg	-90..90	lowest
lidar.fov_max_deg [†]	52	deg	-90..90	highest
lidar.range_min [†]	0.1	m	0..10	closer: no return
lidar.range_max [†]	40	m	0.5..500	no return
lidar.noise_m [†]	0.01	m	0..1	range noise, 1 sigma
lidar.point_dropout [†]	0		0..1	fraction of returns lost at random
lidar.floor_returns [†]	true			rays pointing down hit the floor
odometry.sigma_x [†]	0.005	m/sqrt(m)	0..1	along-track noise
odometry.sigma_y [†]	0.005	m/sqrt(m)	0..1	cross-track noise (slip)
odometry.sigma_yaw_deg [†]	0.1	deg/sqrt(m)	0..10	heading noise
odometry.scale_error [†]	0		-0.5..0.5	measured / true distance - 1
odometry.yaw_bias_deg_per_m [†]	0	deg/m	-10..10	heading bias
odometry.rotation_weight [†]	0.2	m/rad	0..10	turning counts as this travel
latency.lidar_sec [†]	0.05	s	0..1	cloud age
latency.odometry_sec [†]	0.01	s	0..1	pose age
latency.command_sec [†]	0.02	s	0..1	cmd_vel delay

Operator panels

`dharma_rviz` provides five RViz panels (C++, Qt) to operate and tune a robot, and the layout `rviz/robot.rviz` with one of each next to the map, the planner path and costmap and the markers of navigation, routes, avoidance, perception and tags. The panels use relative names; the robot's own topics are remapped to its namespace:

```
rviz2 -d $(ros2 pkg prefix dharma_rviz)/share/dharma_rviz/rviz/robot.rviz \
  --ros-args -r battery:=/robot/battery -r cmd_vel:=/robot/cmd_vel \
  -r motor_temperature:=/robot/motor_temperature
```

12.1 Panels

Panel	Calls	Shows
Mission (ControlPanel)	localization/start, route/start, route/stop, route/mark_stop_point, navigation/load_path, navigation/start, stop, pause, resume, planner/go_to, planner/cancel	localization/status, navigation/status (state, progress, pause reasons), planner/status, route/saved, the e-stop
Safety (SafetyPanel)	control/estop/engage and control/estop_request, control/estop/release, control/set_mode	control/status: mode, remote source, e-stop reasons, armed, protective field, clearance, speed scale
Status (StatusPanel)	–	battery, motor_temperature, diagnostics (worst of all), health/status, odometry/status, cmd_vel
Tuning (TuningPanel)	the parameter services of any node	its parameters, ranges and read-only flags from the node's descriptors
Tags (TagPanel)	localization/record_tags, localization/start	localization/status

12.2 Behaviour

- **Never blocking.** Each panel instance has its own node (`dharma_rviz_<panel>_<pid>_<n>`, review #10), spun from the Qt event loop. Service calls are asynchronous with a client per request and fail with a message after a timeout (5 s, the e-stop 2 s) instead of freezing RViz or leaving buttons disabled (#15, #16).
- **E-stop first** (#11). The E-STOP button is never disabled and sends both the Trigger service and the `control/estop_request` topic on every press, so it works even when one path is down. Start, Resume and Go of the Mission panel are disabled while `control/status` reports the e-stop (navigation refuses them anyway); Stop is always available. Release is a separate button; the mux still requires the re-arm (Section 9.3).

- **Stale values** (#16). Values that stop arriving for 3 s are greyed out with their age; the diagnostics line shows the worst status among those received within that time.
- **Tuning** (#9, #15). Edits wait for *Apply* (*Apply on change* is off by default); spin boxes take the mouse wheel only when focused; ranges come from `describe_parameters` and read-only parameters (`live=False`) are disabled; choices listed in a description ((a | b)) become a combo box; string arrays are edited one item per line; `controller` and `policy` select which implementation section is shown; a second Load replaces the first and its late answers are dropped. *Save YAML* writes every shown value at full precision.
- **Goals**. RViz's *2D Goal Pose* tool only fills in the Go To pose of the Mission panel; Go sends it to `planner/go_to`.

The tests are gtests: the pure helpers (`src/common`), the panels in-process with an offscreen Qt and a fake server node, and every panel loaded through `pluginlib` (Appendix A).

Chapter 13

Interfaces reference

All Dharma packages talk through the messages and services of `dharma_interfaces` 0.2.0 and standard ROS types; products add their own in their own package. The tables below are generated from `msg/` and `srv/`.

13.1 Conventions

- SI units, radians, REP-103 and REP-105 frames; units are written next to every field.
- States are `uint8` constants, never free strings; message fields are for humans only. Status topics are latched.
- No `std_srvs/SetBool` for safety actions: a default request must never release or stop something, so these are separate `std_srvs/Trigger` services.
- `SectorScan`: `ranges <= BLOCKED (0)` or `NaN` is blocked or unknown and is an obstacle at the robot; `>= range_max` is free.
- `Waypoint.yaw` is in radians (route CSV files store degrees for humans); `dwel_sec > 0` marks a stop point.
- Paths in requests (`LoadFile`, `LoadPath`) must resolve inside the sites directory.
- Removed in 0.2.0: `Poi`, `PoiArray`, `Tour`, `TourArray`, `Visitors`, `MissionStatus`, `SavePoi`, `SaveTour`, `DeleteById`; `AvoidanceCommand.steering_bias` became `heading_offset` (rad) and `yaw_rate_bias` (rad/s); `SectorScan.num_sectors` was dropped.

13.2 Topics and services by package

Name	Type	Server / publisher
<code>odometry/status</code>	<code>OdometryStatus</code>	odometry monitor
<code>odometry/restarted</code>	<code>std_msgs/Empty</code>	odometry monitor
<code>localization/status</code>	<code>LocalizationStatus</code>	localization
<code>localization/pose</code>	<code>PoseWithCovarianceStamped</code>	localization
<code>localization/tag_markers</code>	<code>MarkerArray</code>	localization
<code>localization/start, reset, assume_origin, record_tags</code>	<code>Trigger</code>	localization
<code>localization/load_tags</code>	<code>LoadFile</code>	localization
<code>perception/sector_scan, perception/safety_scan</code>	<code>SectorScan</code>	lidar_sector_node
<code>perception/obstacles</code>	<code>MarkerArray</code>	lidar_sector_node
<code>/map, mapping/grid</code>	<code>OccupancyGrid</code>	site manager, grid mapper
<code>mapping/status</code>	<code>MappingStatus</code>	grid mapper
<code>site/status</code>	<code>SiteStatus</code>	site manager
<code>site/list, site/load</code>	<code>ListSites, LoadSite</code>	site manager

Name	Type	Server / publisher
mapping/start	StartMapping	site manager
mapping/stop, mapping/save	Trigger	site manager
route/start, route/stop	Trigger	route recorder
route/mark_stop_point	MarkStopPoint	route recorder
route/saved	std_msgs/String	route recorder
planner/go_to, planner/plan, planner/cancel	GoTo, PlanPath, Trigger	planner
planner/status, planner/path, planner/costmap	PlannerStatus, Path, Occupancy-Grid	planner
navigation/set_path	SetPath	follower
navigation/start, stop, pause, resume	Trigger	follower
navigation/status, navigation/stop_point	NavigationStatus, StopPointEvent	follower
navigation/load_path	LoadPath	path loader
navigation/waypoints, navigation/path	WaypointPath, Path	path loader
navigation/avoidance	AvoidanceCommand	avoidance node
control/nav_cmd_vel	Twist	follower
cmd_vel	Twist	control mux (only)
control/status	ControlStatus	control mux
control/set_mode	SetMode	control mux
control/estop/engage, control/estop/release	Trigger	control mux
control/estop_request, control/hw_estop	String, Bool	to the control mux
health/status, diagnostics	HealthStatus, DiagnosticArray	health monitor

13.3 Messages

AvoidanceCommand (msg)

fields

header	std_msgs/Header	
heading_offset	float32	rad, rotate the commanded direction (+left)
yaw_rate_bias	float32	rad/s, added to the commanded yaw rate (+left)
lateral_bias	float32	m/s, sideways push (+left), omni robots only
velocity_scale	float32	0..1 multiplier on the commanded speed
emergency	bool	obstacle inside the emergency cone: stop

ControlStatus (msg)*constants*

MODE_AUTO = 0	uint8	
MODE_REMOTE = 1	uint8	
FIELD_OFF = 0	uint8	not applied in this mode
FIELD_CLEAR = 1	uint8	
FIELD_LIMITED = 2	uint8	slowed down for an obstacle
FIELD_STOP = 3	uint8	obstacle within the stopping distance
FIELD_STALE = 4	uint8	no fresh safety scan
FIELD_BLIND = 5	uint8	the scan reports blocked or unknown sectors

fields

header	std_msgs/Header	
mode	uint8	
remote_source	string	the operator source that drives in MODE_REMOTE
estop	bool	latched e-stop (software or hardware)
estop_reasons	string[]	oldest first
armed	bool	false after a release until the source sent a neutral command
field	uint8	
clearance	float32	m, free distance along the commanded motion (inf: clear)
speed_scale	float32	0..1, applied to the commanded speed

HealthStatus (msg)*constants*

OK = 0	uint8
WARN = 1	uint8
ERROR = 2	uint8

fields

header	std_msgs/Header	
level	uint8	
issues	string[]	one line per problem
speed_scale	float32	0..1 speed limit requested from the control mux (1: none)

LocalizationStatus (msg)*constants*

NONE = 0	uint8	no map -> odom yet
LOCALIZED = 1	uint8	
UNCERTAIN = 2	uint8	localized, but the uncertainty grew past the limits

fields

header	std_msgs/Header	
state	uint8	
source	string	last correction: tag:<frame>, pose_estimate, origin, scan_match
sigma_xy	float32	m, 1-sigma position uncertainty (largest axis)
sigma_yaw	float32	rad, 1-sigma heading uncertainty
since_correction	float32	s since the last accepted correction
message	string	

MappingStatus (msg)*fields*

header	std_msgs/Header	
active	bool	scans are being integrated
site	string	
scans	uint32	integrated in this session
rejected	uint32	skipped: jumps, corrections in flight, poor matches
resolution	float32	m per cell
width	uint32	cells
height	uint32	cells
known_area_m2	float32	free + occupied
message	string	

NavigationStatus (msg)*constants*

INACTIVE = 0	uint8	
JOINING = 1	uint8	walking to the path
FOLLOWING = 2	uint8	
APPROACHING = 3	uint8	final x, y, yaw approach to a stop point or the goal
DWELLING = 4	uint8	waiting at a stop point
PAUSED = 5	uint8	see pause_reasons
BLOCKED = 6	uint8	
OFF_PATH = 7	uint8	
GOAL_REACHED = 8	uint8	
FAILED = 9	uint8	

fields

header	std_msgs/Header	
state	uint8	
pause_reasons	string[]	operator, estop, remote, localization, battery
index	uint32	current waypoint
count	uint32	waypoints in the path
remaining	float32	m along the path
cross_track	float32	m, signed (+left)
message	string	

OdometryStatus (msg)*constants*

INITIALIZING = 0	uint8	
OK = 1	uint8	
DEGRADED = 2	uint8	still usable, e.g. speed disagrees with the robot odometry
LOST = 3	uint8	stale or jumping, do not drive on it

fields

header	std_msgs/Header	
state	uint8	
speed	float32	m/s, estimator
tilt	float32	rad, roll / pitch of body in odom while standing still
message	string	

PlannerStatus (msg)*constants*

IDLE = 0	uint8
PLANNING = 1	uint8
FOLLOWING = 2	uint8
REPLANNING = 3	uint8
ARRIVED = 4	uint8
FAILED = 5	uint8
CANCELLED = 6	uint8

fields

header	std_msgs/Header	
state	uint8	
goal_x	float64	m, map frame
goal_y	float64	
goal_yaw	float64	rad
path_length	float32	m
remaining	float32	m
replans	uint32	
message	string	

SectorScan (msg)*constants*

BLOCKED = 0.0	float32
---------------	---------

fields

header	std_msgs/Header	
angle_min	float32	rad
angle_increment	float32	rad
range_max	float32	m
ranges	float32[]	m

SiteStatus (msg)*fields*

header	std_msgs/Header	
site	string	empty: none loaded
sites	string[]	every site on the robot
has_map	bool	
mapping	bool	a mapping session is running
tags	uint32	recorded fiducials
routes	uint32	taught routes
message	string	

StopPointEvent (msg)*constants*

ARRIVED = 0	uint8
INSPECTED = 1	uint8
DEPARTED = 2	uint8
SKIPPED = 3	uint8

fields

header	std_msgs/Header	
event	uint8	
index	uint32	waypoint index of the stop point
x	float64	m, map frame
y	float64	
yaw	float64	rad
dwel_sec	float32	
message	string	

Waypoint (msg)*fields*

x	float64	m
y	float64	m
yaw	float64	rad
dwel_sec	float32	> 0: stop point, wait this long

WaypointPath (msg)*fields*

header	std_msgs/Header	frame_id: map
waypoints	Waypoint []	

13.4 Services**GoTo (srv)***request*

x	float64	m
y	float64	m
yaw	float64	rad

response

success	bool
message	string

ListSites (srv)

request

(empty)

response

sites string[]

active string

LoadFile (srv)

request

path string

create bool

response

success bool

message string

LoadPath (srv)

request

path string

response

success bool

message string

LoadSite (srv)

request

site string

response

success bool

message string

MarkStopPoint (srv)

request

dwell_sec float32

response

success bool

message string

PlanPath (srv)

<i>request</i>		
x	float64	m, map frame
y	float64	m
yaw	float64	rad
<i>response</i>		
success	bool	
message	string	
waypoints	Waypoint[]	
length	float32	m

SetMode (srv)

<i>request</i>		
mode	string	
source	string	remote source, e.g. handheld; empty: the first configured one
<i>response</i>		
success	bool	
message	string	

SetPath (srv)

<i>request</i>		
waypoints	Waypoint[]	
process	bool	resample and smooth it like a taught route
source	string	who sent it, for the log
<i>response</i>		
success	bool	
message	string	

StartMapping (srv)

<i>request</i>		
site	string	1-48 of a-z, 0-9, _ and -
reset	bool	
<i>response</i>		
success	bool	
message	string	

Development

14.1 Package layout

```

<pkg>/
  <pkg>/
    __init__.py
    <name>_node.py      thin ROS nodes: parameters, pubs/subs, services,
timers, conversions
    core/              pure Python + numpy: algorithms, models, state
machines, configs
      config.py        configuration dataclasses (dharma_utils.config.
param)
      <topic>/         sub-packages for families (controllers/, avoidance
/, follower/ ...)
      utils/           ROS-side helpers of this package: msg <-> core
conversions, markers
config/<pkg>.yaml      defaults (only what differs from the dataclass
defaults, plus docs)
launch/<pkg>.launch.py
test/unit/test_*.py   pytest on core/ without ROS
test/ros/test_*.py    rclpy in-process node tests
test/test_style.py    dharma_utils.testing.check_package
package.xml setup.py setup.cfg resource/<pkg> README.md CHANGELOG.md

```

- core/ never imports rclpy, ROS messages, tf2 or launch; nodes convert messages to core types (numpy arrays, Pose2D, dataclasses) at the edge.
- Modules have at most 300 lines (tests 400), one class per algorithm. Families are OOP: an abstract base class in base.py, one implementation per file, a dharma_utils.plugins.Registry in __init__.py, open to entry points so a product adds an implementation without forking.
- Nodes subclass dharma_utils.ros.DharmaNode and start with run(MyNode): declare_config, declare_robot, self.log, self.tf (with per-edge ages), self.every() (exception-guarded timers), shutdown hooks.
- setup.py uses find_packages(exclude=['test', 'test.*']) and extras_require={'test': ['pytest']}: without a pytest test dependency colcon runs no tests and Jazzy CI fails with exit code 5.

14.2 Adding a plugin

Algorithm families are open to other packages through Python entry points (dharma_utils.plugins.Registry): a product adds an implementation without forking.

Family	Base class	Entry point group	Selected by
path controllers	PathController	dharma.controllers	controller of path_follower_node
avoidance policies	AvoidancePolicy	dharma.avoidance	policy of obstacle_avoidance_node
health checks	Check	dharma_health.checks	extra_checks of health_monitor_node

The node looks the name up in the registry, which loads every entry point of its group on first use and checks that the class derives from the base class; an unknown name is refused with the list of available ones. `path_follower_node` declares the `Config` dataclass of every registered controller as parameters `<name>.*` at start, so a plugin controller's parameters have descriptors, ranges and live validation like the built-in ones. In 0.2.0 a plugin avoidance policy runs with its `Config` defaults (the node declares parameters only for the built-in policies), and a plugin health check is built without arguments.

14.2.1 Example: a controller package

A complete package `acme_controllers` with one controller, `carrot_p`, that turns toward a point ahead on the route in proportion to its bearing:

```
acme_controllers/
  acme_controllers/__init__.py      (empty)
  acme_controllers/carrot_p.py
  test/test_carrot_p.py
  resource/acme_controllers         (empty, ament index marker)
  package.xml  setup.py  setup.cfg
```

`acme_controllers/carrot_p.py`: a frozen configuration dataclass and the controller. The base class applies the avoidance biases and the path shift offset is in the request, so compute only tracks the (offset) route.

```
"""Carrot P: turn toward a point ahead on the route, proportionally to its
   bearing."""

from dataclasses import dataclass
import math

from dharma_navigation.core.controllers import Command, PathController,
    TrackRequest
from dharma_navigation.core.controllers.base import carrot, to_body
from dharma_utils.config import param

@dataclass(frozen=True)
class CarrotPConfig:
    """Parameters, declared as carrot_p.* on path_follower_node."""

    lookahead: float = param(0.8, min=0.1, max=5.0, unit='m', doc='carrot
    ahead on the route')
    gain: float = param(1.5, min=0.0, max=10.0, unit='1/s', doc='yaw rate
    per rad of bearing')

class CarrotPController(PathController):
```

```

"""Drive at the requested speed, slower when the carrot is off to the
side."""

Config = CarrotPConfig

def compute(self, request: TrackRequest) -> Command:
    """Return the forward speed and a yaw rate proportional to the
    carrot bearing."""
    x, y = to_body(request.pose, carrot(request, self.config.lookahead))
    bearing = math.atan2(y, x)
    speed = request.speed * max(0.0, math.cos(bearing))
    return Command(speed, 0.0, self.clamp_wz(self.config.gain * bearing))

```

setup.py registers the class under the group (no decorator: the entry point is the registration):

```

from setuptools import find_packages, setup

setup(
    name='acme_controllers',
    version='0.1.0',
    packages=find_packages(exclude=['test', 'test.*']),
    data_files=[
        ('share/ament_index/resource_index/packages', ['resource/
acme_controllers']),
        ('share/acme_controllers', ['package.xml']),
    ],
    install_requires=['setuptools'],
    extras_require={'test': ['pytest']},
    zip_safe=True,
    maintainer='ACME Robotics',
    maintainer_email='dev@example.com',
    description='A path controller plugin for Dharma.',
    license='Apache-2.0',
    entry_points={
        'dharma.controllers': ['carrot_p = acme_controllers.carrot_p:
CarrotPController'],
    },
)

```

package.xml declares `<exec_depend>dharma_navigation</exec_depend>`, `<test_depend>python3-pytest</test_depend>` and `<build_type>ament_python</build_type>`; setup.cfg has the usual `script_dir / install_scripts` lines. test/test_carrot_p.py checks the registration and the behaviour on a kinematic robot:

```

import math

from acme_controllers.carrot_p import CarrotPController
from dharma_navigation.core.controllers import CONTROLLERS, ControllerLimits, TrackRequest
from dharma_navigation.core.path.route import Route
from dharma_utils.io import RoutePoint
from dharma_utils.math import Pose2D

def test_registered_through_the_entry_point():
    assert CONTROLLERS.get('carrot_p') is CarrotPController

```

```
def test_converges_onto_a_straight_route():
    route = Route.from_points([RoutePoint(0.1 * i, 0.0) for i in range(101)])
    controller = CarrotPController(limits=ControllerLimits(max_wz=1.0))
    pose, dt = Pose2D(0.0, 0.5, 0.0), 0.05
    for _ in range(200):
        s = max(0.0, min(pose.x, route.length))
        cmd = controller.update(TrackRequest(route, s, pose, speed=0.5, dt=dt))
        yaw = pose.yaw + cmd.wz * dt
        x = pose.x + cmd.vx * math.cos(yaw) * dt
        pose = Pose2D(x, pose.y + cmd.vy * math.sin(yaw) * dt, yaw)
    assert abs(pose.y) < 0.05 and pose.x > 3.0
```

Build it next to the stack and select it in the product's parameter file:

```
colcon build --packages-select acme_controllers
colcon test --packages-select acme_controllers && colcon test-result --verbose
```

```
path_follower_node:
  ros__parameters:
    controller: carrot_p
    carrot_p: {lookahead: 0.8, gain: 1.5}
```

`ros2 param describe /path_follower_node carrot_p.lookahead` then shows the description, unit and range, and `ros2 param set /path_follower_node carrot_p.gain -1.0` is refused with the range. The controller can also be switched at runtime (controller is live); the follower rebuilds it with the robot's limits. An avoidance policy works the same way: subclass `AvoidancePolicy`, implement `compute(angles, ranges, scan)` returning an `AvoidanceOut` (blocked and unknown sectors arrive as range 0) and register it under `dharma.avoidance`.

14.3 Code style

- English, clean, readable; one-line docstrings on public classes and functions; comments only where the code cannot say it. No commented-out code, no `print`, no bare `except`.
- Type hints on public functions; frozen dataclasses for data and configuration; Enum for states; no mutable default arguments.
- Units in names or docstrings (`_sec`, `_m`, `_deg`); radians internally.
- Python 3.8 and numpy 1.17 compatible (Foxy): `from __future__ import annotations` first in every module (ruff's FA rule with `target-version="py38"`), so hints keep `tuple[str, ...]` and `float|None`; no 3.9+ runtime features (`zip(strict=)`, `itertools.pairwise`, `functools.cache`, runtime `list[int]` or `X|Y`, `str.removeprefix`, `match`) and no numpy 2 APIs; `dharma_utils.math.default_rng` instead of `np.random.default_rng`.
- Distribution differences only through `dharma_utils.ros`: `compat` (rcply init and shutdown, parameter descriptors, executors, the TF buffer) and `launch` (`node(respawn=, ros_arguments=)`, `group(forwarding=)`, `node_overrides`); a package never branches on the distribution. `check_package` reports what is left.
- Short functions: ruff's pylint rules (branches, statements, complexity 12) are on; split instead of adding `noqa`.

14.4 Lint and format

The same pre-commit hooks run locally and in CI: whitespace and end-of-file fixes, YAML and XML checks, large-file guard, ruff check -fix and ruff format (single quotes, line length 99), clang-format for C++, prettier for YAML, Markdown and JSON, shellcheck.

```
pip install pre-commit && pre-commit install
pre-commit run --all-files
```

14.5 Tests

test/unit pytest on core/, fast and without ROS: `PYTHONPATH=$PWD python -m pytest -q test/unit`.

test/ros rclpy in-process node tests with DharmaNode, Spinner and Collector (dharma_utils.testing); unique node names; `rclpy.init()` and `try_shutdown()` per test. Module-level helpers must not be named `setup` or `teardown`: the pytest versions of the Foxy, Humble and Jazzy apt packages run them as nose-style fixtures before `rclpy.init()`. Foxy's pytest is 4.6.

test/test_style.py `check_package`: module sizes, pure core/, no shadowed node members.

Every fixed defect has a regression test (Appendix A).

```
colcon build --packages-select <pkg>
colcon test --packages-select <pkg> --event-handlers console_direct+
colcon test-result --verbose
```

14.6 Continuous integration

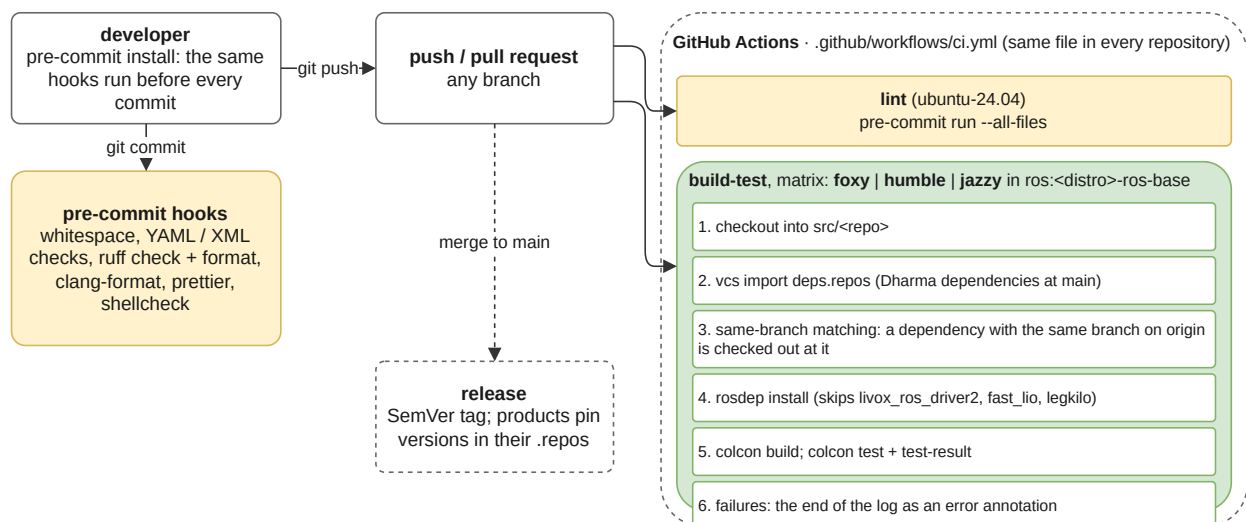


Figure 14.1: CI of every Dharma repository: lint, then build and test on Foxy, Humble and Jazzy with the Dharma dependencies on the same branch when it exists.

Every repository carries the same `.github/workflows/ci.yml` (Fig. 14.1):

1. **lint** on Ubuntu 24.04: `pre-commit run --all-files`; the end of the log is reported as an error annotation on failure.

2. **build-test**, matrix `foxy`, `humble` and `jazzy` in `ros:<distro>-ros-base` (Python 3.8 with numpy 1.17, Python 3.10 with numpy 1.21, and Python 3.12 with numpy 1.26); the frozen Foxy image first gets the current ROS apt key:
 - (a) check out the repository into `src/<repo>`;
 - (b) `vc`s import the Dharma dependencies from `deps.repos` (at main);
 - (c) **same-branch matching**: every dependency that has the branch under test on its origin is checked out at that branch, so a change spanning several repositories is built and tested together before it is merged;
 - (d) `rosdep install` (skipping `livox_ros_driver2`, `fast_lio`, `legkilo`, which are built from source on robots; `loguru` from pip on Foxy);
 - (e) `colcon build`, then `colcon test` of the repository's packages and `colcon test-result`.

Discovery is limited to localhost in CI (`ROS_LOCALHOST_ONLY`, `ROS_AUTOMATIC_DISCOVERY_RANGE`).

14.7 Releases

- SemVer; before 1.0 a breaking change bumps the minor version and every package of the stack moves with it (0.1 → 0.2). `dharma_utils` is a dependency of every package: its API is deprecated one minor release before anything is removed.
- `CHANGELOG.md` in `Keep` a Changelog format (Added, Changed, Fixed, Removed); `package.xml` carries the version.
- Conventional commits, one short line each (`fix(mux): rotation inside the field`).
- Products pin released versions in their `.repos` file (`vcstool`).

14.8 The template repository

The Dharma repository is the hub of the stack and a GitHub template a robot product starts from. It holds:

dharma.repos, **external.repos** every Dharma repository at main; the Livox driver (`livox_ros_driver2`), FAST-LIO and, commented out, the Leg-KILO fork.

docker/, **compose.yaml**, **.devcontainer/** a development image for Foxy, Humble and Jazzy (`ARG ROS_DISTRO, ros:<distro>-ros-base`) with an XFCE desktop over VNC and noVNC; one compose service per distribution (profiles `foxy`, `humble` and `jazzy`), separate build, install and log volumes per distribution (builds of two distributions never mix), ports bound to localhost (Foxy 5903 / 6082, Humble 5901 / 6080, Jazzy 5902 / 6081).

scripts/dev.sh `[-distro foxy|humble|jazzy] up | shell | build | test | vnc | down | clean | import` (default distribution from `DHARMA_DISTRO`, else Humble).

scripts/setup.sh check or install `<components>` on a robot computer or a development machine: `ros`, `tools`, `repos`, `livox`, `rosdep`, `build` and `service` (a systemd unit running the bringup at boot); Ubuntu 20.04 with Foxy, 22.04 with Humble or 24.04 with Jazzy.

scripts/start_tmux.sh a tmux session with one window per subsystem from a YAML layout in `scripts/tmux/` (robot or sim).

scripts/sync_tooling.sh, tooling/ the canonical tooling files and their copy into every check-out (maintainers).

src/robot_bringup the placeholder product package: `config/robot.yaml` (the robot profile), `config/stack.yaml` (per-node overrides), `config/sim.yaml` (on top of it for the simulator), `launch/robot.launch.py` (an optional robot driver and the stack) and `launch/sim.launch.py` (dharma_sim and the stack, odometry external), with the composition in `robot_bringup/stack.py` and pure helpers in `bringup.py`, all unit-tested.

.github/, CONTRIBUTING.md, docs/ CI, the pull request template, the conventions and this documentation.

Bringup. `ros2 launch robot_bringup robot.launch.py` validates every argument and the robot profile (unknown names and out-of-range values stop the launch with the reason), merges `stack.yaml` (and `sim.yaml`) with `params_file` into one parameter file, adds every started node to the health monitor's `nodes.expected`, reports sections that match no node, and includes each package launch in its own scope with `robot_params`, the merged file and only the arguments that are set. It remaps the `cmd_vel` to `/<robot_ns>/cmd_vel`. **Arguments:** `robot_params`, `params_file`, `use_sim_time`, `log_level`, `robot_ns` (ROBOT_NS), `robot_driver` (`<package>[/<file>.launch.py]`), **started with** `namespace:=<robot_ns>`, `odometry` (`fast_lio`, `leg_kilo` or `external`), `site`, `sites_dir`, `start_mapping`, `fiducials`, `camera_image`, `camera_info`, `health`, and for the simulator world, `scenario`, `rviz`. A robot driver package provides `launch/bringup.launch.py` accepting `namespace`; it publishes `/<ns>/odom`, `/<ns>/battery`, the LiDAR and IMU and the URDF TF, and subscribes `/<ns>/cmd_vel`.

A new product. “Use this template” on GitHub, clone, then

```
scripts/init_project.py --name MyRobot --package my_robot_bringup --robot-ns
  my_robot --dry-run
scripts/init_project.py --name MyRobot --package my_robot_bringup --robot-ns
  my_robot
```

renames `robot_bringup` everywhere, sets the robot namespace and the container names, removes what only the hub needs (`tooling/`, the tooling sync, the script itself, the manual, the paper, the diagram sources and the wiki unless `--keep-docs`), resets the version to 0.1.0 and rewrites `README.md` and `CHANGELOG.md`. The wiki page *New product from the template* walks through the remaining steps (robot profile, driver, limits, first map).

14.9 Building this documentation

```
make -C docs                # diagrams (when draw.io is installed), manual,
  paper
make -C docs diagrams      # diagrams/*.drawio -> diagrams/pdf (cropped),
  diagrams/svg
make -C docs manual        # build/manual/dharma_manual.pdf
make -C docs paper         # build/paper/dharma.pdf
make -C docs reference SRC=~ /ws/src # regenerate manual/generated from the
  sources
```

Diagrams are draw.io files (Helvetica; colours per layer: estimation orange #ffe6cc / #d79b00, world model blue #dae8fc / #6c8ebf, planning and execution green #d5e8d4 / #82b366, safety red #f8cecc / #b85450, utilities yellow #ffff2c / #d6b656, hardware grey #f5f5f5 / #666666). Their PDF and SVG exports are committed, so LaTeX builds without draw.io.

Appendix A

Traceability: findings, fixes and tests

Every finding of the two reviews of the 0.1 stack, the change that fixes it in 0.2.0 and the regression tests that cover it. IDs with letters (O1, L1, ...) come from the first review (the mathematical analysis); #*n* IDs come from the second review, numbered per repository. Priorities: P0 safety, P1 robustness, P2 maintainability (first review). Tests are `file::function` in the package's `test/` directory (u: `test/unit`, r: `test/ros`); the C++ panels of `dharmarviz` have gtests, written `file::Suite.Test` (g: `test/`). A cell that says “no dedicated test” marks a fix that is verified only by review or indirectly.

A.1 dharmarodometry

ID	Finding	Fix	Tests
O1 (P1)	the start gate let the estimator start up to 8° tilted, tilting odom and map	<code>max_tilt_deg 3.0</code> (<code>core/config.py</code>); <code>Leg-KILO init_type: 2</code> (<code>config/leg_kilo.yaml</code>); tilt of odom while still in <code>OdometryStatus</code> (<code>core/monitor.py</code>); levelling in localization	<code>u/test_still::</code> <code>test_default_start_tilt_limit_is_three_degrees,</code> <code>u/test_still::test_gate_holds_a_tilted_start,</code> <code>u/test_monitor::</code> <code>test_level_body_in_a_tilted_odom_is_degraded,</code> <code>u/test_monitor::</code> <code>test_tilted_body_in_a_level_odom_is_ok</code>
O1	monitor thresholds per robot speed, not a statistical test	residual χ^2 with sensor variances and CUSUM (<code>core/residual.py</code>)	<code>u/test_residual::</code> <code>test_lasting_mismatch_alarms_after_a_while_and_recovers,</code> <code>u/test_residual::</code> <code>test_threshold_scales_with_the_variances,</code> <code>u/test_residual::test_agreeing_speeds_never_alarm_with_sensor_noise</code>
#7	false restart when turning in place near the start pose	a restart needs a message gap or a translation jump to the origin, never yaw alone (<code>core/restart.py</code>)	<code>u/test_restart::</code> <code>test_turn_in_place_near_the_start_is_no_restart,</code> <code>u/test_restart::test_yaw_snap_alone_is_no_restart,</code> <code>r/test_monitor_node::</code> <code>test_turn_in_place_at_the_start_is_no_restart</code>
#8	<code>extrinsic_est_en: true</code> for the factory-calibrated Mid-360 IMU	<code>false</code> (<code>config/fast_lio.yaml</code>)	<code>r/test_launch::test_fast_lio_does_not_estimate_the_factory_extrinsic</code>
#17	a mistyped <code>site_config</code> silently ignored	a set but missing file fails the launch	<code>r/test_launch::</code> <code>test_missing_site_config_fails_the_launch</code>

ID	Finding	Fix	Tests
#3, #15	route recorder in odometry (records while not localized, non-atomic CSV)	moved to dharma_mapping and fixed there	see dharma_mapping, recorder
H1	status as a string	OdometryStatus (typed)	r/test_monitor_node:: test_status_is_typed_ and_restart_needs_gap_ and_origin
–	layer inversion: odometry launched localization	include and exec_depend removed	r/test_launch:: test_external_starts_ only_the_monitor
–	empty launch arguments override parameter files	only non-empty arguments override	r/test_launch:: test_empty_arguments_ never_override, r/test_launch::test_ parameter_files_in_order

A.2 dharma_localization

ID	Finding	Fix	Tests
L1 (P1)	one tag detection hard-replaced map→odom: no covariance, no continuous correction, ad-hoc 0.5 m / 15° rejection	SE(2) EKF over map→odom with a chi-square gate (core/estimator.py), sources (core/sources/), optional scan matching, localization/pose with covariance	u/test_estimator:: test_predict_grows_with_ distance_not_time, u/test_estimator:: test_gate_grows_with_ uncertainty, u/test_localizer:: test_tag_in_view_ localizes_then_corrects_ through_the_gate, u/test_scan_match::test_ match_recovers_the_pose_ and_reports_a_covariance
L2 (P2)	record_tags slept 2 s in a service callback and blocked the map→odom timer	timer-driven recording session, own callback group (core/tags.py, utils/ tag_services.py)	r/test_localization_ node::test_record_tags_ keeps_tf_flowng_and_ relocalizes
O1 (P1)	odom not level	levelling in the bridge (core/leveling.py, core/bridge.py)	u/test_leveling_bridge:: test_level_rotation_ removes_the_start_tilt_ of_the_base, u/test_leveling_bridge:: test_level_bridge_keeps_ lio_motion_planar_on_a_ tilted_start, r/test_ localization_node:: test_bridge_levels_odom_ and_reset_clears

ID	Finding	Fix	Tests
T1/T3, #14	tf2 interpolated a ramp across a correction	old value at $t - \epsilon$, new at t (core/timeline.py)	u/test_leveling_bridge:: test_timeline_steps_within_epsilon_never_ramps, u/test_localizer:: test_correction_samples_step_without_ramp
#1	auto-correct gate compared map→odom instead of map→base	innovation on the robot pose	u/test_estimator::test_gate_rejects_outliers_and_accepts_consistent_fixes_far_from_origin, u/test_estimator::test_heading_uncertainty_far_from_origin_moves_the_robot_sideways
#2	record_tags allowed while not localized, overwriting the site anchors	refuse unless LOCALIZED and not circular	u/test_tags:: test_recordable_refuses_when_not_localized_or_circular, u/test_localizer:: test_record_refused_when_not_localized_and_when_circular
#3	a corrupt tags file crashed record_tags	tolerant reads, success: false	u/test_tags:: test_merge_refuses_to_overwrite_a_corrupt_file, u/test_localizer:: test_record_session_writes_the_tag_and_survives_a_corrupt_file
#10	an empty launch argument overrode the params file	non-empty overrides only	u/test_launch:: test_empty_arguments_do_not_override
#11	~ in the tags file not expanded	dharma_utils.io.expand	r/test_localization_node::test_tags_file_with_tilde_is_expanded
#12	load_tags wrote anywhere	path must resolve inside sites_dir	r/test_localization_node:: test_load_tags_path_guard_and_corrupt_file
#15	tag tests without the real optical rotation	tests with $(-\pi/2, 0, -\pi/2)$ and non-zero yaws, hand-computed poses	u/test_tag_geometry:: test_optical_rotation_is_right_down_forward, u/test_tag_geometry:: test_tag_off_axis_and_rotated_robot_yawed_90_degrees

A.3 dharma_perception

ID	Finding	Fix	Tests
P1 (P1)	height band in base_link, which pitches with the gait; base height 0.35 here and 0.30 elsewhere	levelled points, band above the floor from robot.base_height, optional ground segmentation (core/filtering.py)	u/test_filtering:: test_pitched_body_does_not_see_the_floor, u/test_filtering:: test_band_is_relative_to_the_floor_through_base_height, r/test_lidar_sector_node:: test_pitched_body_does_not_block_on_the_floor
P1	minimum of one point per sector sensitive to outliers	<i>k</i> -th nearest point (core/sectors.py)	u/test_sectors::test_kth_nearest_ignores_a_single_spurious_return
P2 (P1)	visitor tracker associated in a rotating frame	removed (product feature)	–
B4	low objects near the robot drop out of view; sectors without returns read free	obstacle memory in odom (core/memory.py); limits documented	u/test_memory_blindness:: test_memory_keeps_an_object_the_sensor_no_longer_sees, u/test_pipeline:: test_memory_keeps_a_low_box_in_the_safety_scan
–	blind sensor not reported	every sector BLOCKED when blind, TF missing, errors or silence (core/blindness.py, node watchdog)	u/test_pipeline:: test_blind_lidar_blocks_every_sector, r/test_lidar_sector_node::test_blind_livox_blocks_every_sector, r/test_lidar_sector_node::test_silent_input_publishes_blocked_scans
#4	one malformed cloud killed the producer	validated parsing, per-message try/except, respawn	u/test_livox:: test_fuzz_never_raises_anything_else, r/test_lidar_sector_node::test_malformed_livox_bytes_keep_the_node_alive, r/test_lidar_sector_node:: test_pointcloud_input_and_malformed_clouds
#12	Livox tag bits ignored	noise filter per field and confidence	u/test_livox:: test_noise_mask_selects_fields_and_levels, r/test_lidar_sector_node::test_livox_noise_tags_are_dropped
#16	the Livox path had no real test	raw-bytes node test, packer checked against the documented layout	r/test_lidar_sector_node::test_livox_raw_bytes_give_front_and_safety_scans, u/test_livox:: test_packer_matches_the_documented_layout

A.4 dharma_mapping

ID	Finding	Fix	Tests
M1 (P1)	scans integrated at the TF pose without any check	jump and correction gates, optional refinement (core/integrator.py); pose graph future work	u/test_integrator:: test_jump_is_rejected_ and_never_grows_the_grid, u/test_integrator::test_ refinement_keeps_walls_ sharp_under_pose_error, u/test_integrator:: test_refinement_refuses_ an_ambiguous_corridor
O1	band assumed a level map frame without saying so	band in the levelled map frame, documented	u/test_integrator:: test_band_drops_floor_ ceiling_and_the_robot
I1, §5.4	site manager mixed map storage with POIs and tours	POIs and tours removed	-
#4	mapping/save wrote another site's grid; a timed-out stop counted as stopped	the session remembers its site (core/session.py); a timeout is a failure	u/test_session:: test_session_refuses_to_ save_another_sites_grid, r/test_site_manager_ node::test_timed_out_ stop_is_a_failure
#5	no .bak, map pair not atomic, reset of the same site skipped the archive	atomic pair with size and CRC, .bak pair (core/map_io.py); reset always archives	u/test_map_io::test_ previous_pair_is_kept_ as_a_loadable_backup, u/test_map_io::test_torn_ pair_is_detected, u/test_map_io:: test_failed_write_ leaves_the_old_pair, u/test_session::test_ reset_always_archives_ existing_content
#7	corrupt site files crashed the site manager at every boot	tolerant loads, notes in site/status (core/site_store.py)	u/test_site_store:: test_corrupt_files_ never_raise, r/test_site_ manager_node:: test_corrupt_site_files_ do_not_crash_the_boot
#8	unbounded grid growth on odometry jumps (300 m: 109 MB)	size cap and jump gate (core/grid.py)	u/test_grid::test_growth_ is_capped_on_a_pose_jump, u/test_grid::test_cap_ splits_the_room_when_ both_sides_grow
#9	an origin localization at boot trusted as the frame of an existing site	only tag or pose-estimate localization continues a site	u/test_session:: test_origin_at_boot_is_ not_the_frame_of_an_ existing_site, r/test_site_manager_ node::test_origin_at_ boot_does_not_continue_ an_existing_site

ID	Finding	Fix	Tests
#10	launch defaults overrode the params file	non-empty overrides only	<code>r/test_launch::</code> <code>test_empty_arguments_</code> <code>add_no_overrides,</code> <code>r/test_launch::</code> <code>test_missing_params_</code> <code>file_fails_the_launch</code>
#11	~ not expanded (recorder output, tags path)	<code>dharma_utils.io.</code> <code>expand</code>	<code>u/test_routes::test_</code> <code>output_path_is_expanded_</code> <code>and_empty_refuses</code>
#12	<code>grid_mapper/save</code> load accepted any path	site names only	<code>r/test_grid_mapper_node::</code> <code>test_scans_become_a_</code> <code>grid_saved_into_its_own_</code> <code>site,</code> <code>r/test_grid_mapper_node::</code> <code>test_continue_needs_the_</code> <code>site_loaded</code>
#13	archives collided within a second, nested routes, were made before the preconditions	<code>unique_dir</code> , preconditions first	<code>u/test_site_store::</code> <code>test_archive_twice_in_</code> <code>one_second_never_nests,</code> <code>r/test_site_manager_</code> <code>node::</code> <code>test_reset_archives_</code> <code>into_unique_directories,</code> <code>r/test_site_manager_</code> <code>node::</code> <code>test_reset_archives_</code> <code>only_when_it_can_start</code>
#14	scans integrated during a map→odom correction	correction gate	<code>u/test_integrator::</code> <code>test_scans_during_a_</code> <code>correction_are_skipped</code>
rec.	recorder recorded while not localized (TF exists ≠ localized)	only while LOCALIZED (<code>core/routes.py</code>)	<code>u/test_routes::</code> <code>test_refuses_when_not_</code> <code>localized, r/test_route_</code> <code>recorder_node::</code> <code>test_refuses_while_not_</code> <code>localized_even_with_tf</code>
rec.	non-atomic, all-in-memory CSV	incremental atomic writes	<code>u/test_routes::</code> <code>test_route_goes_to_the_</code> <code>site_and_is_written_</code> <code>incrementally</code>
rec.	an <code>OSError</code> in the service killed the node	<code>success: false</code>	<code>r/test_route_recorder_</code> <code>node::test_file_errors_</code> <code>are_answers_not_crashes</code>

A.5 dharma_navigation

ID	Finding	Fix	Tests
N1 (P0)	planner radius 0.35 m below the half diagonal 0.49 m	lethal = inscribed radius + margin, oriented footprint check up to the circumscribed radius (core/planning/costmap.py, footprint.py)	u/test_planning:: test_radii_come_from_the_footprint, u/test_planning:: test_oriented_footprint_against_a_wall, u/test_planning:: test_door_narrower_than_the_body_is_refused
N2 (P2)	MPPI without obstacle cost, warm start or seed; PID without anti-windup; SpeedRegulator integrating $ \omega $	obstacle cost, warm start, seeded RNG (core/controllers/mppi.py); clamped integral (pid.py); signed error, reset on new paths	u/test_controllers:: test_mppi_is_seeded_and_warm_started, u/test_controllers:: test_mppi_steers_around_scan_obstacles, u/test_controllers:: test_pid_integral_is_clamped_and_recovers; regulator: no dedicated test
N3 (P1)	two stopping models (path shift without reaction time)	dharm_utils.safety for path shift, profile and guard	u/test_path:: test_profile_brakes_into_stops_with_the_stopping_model, u/test_shift::test_no_gap_slows_to_a_stop_with_the_stopping_model
N4 (P0)	steering_bias a rate in the message, an angle in one controller	heading_offset (rad) and yaw_rate_bias (rad/s) in every controller	u/test_controllers::test_biases_have_one_meaning_in_every_controller
N5 (P2)	hard-coded 'map' / 'base_link'	robot.frames	no dedicated test: the node tests run with the default frames; no frame literal is left in the package sources
–	speed profile without a forward pass	forward pass (core/path/profile.py)	u/test_path:: test_profile_forward_pass_limits_acceleration
§4	stale TF stopped the robot without distinction	ages from self.tf, prediction between tf_compensate_sec and the timeout, TF_STARTING / TF_STALE / TF_CLOCK (core/follower/supervisor.py)	u/test_follower_safety:: test_tf_status_messages, u/test_follower_safety:: test_slightly_old_pose_is_predicted_with_the_twist
–	path_follower_node 1429 lines	pure follower in core/follower/, node 277 lines	test_style:: test_package_conventions

ID	Finding	Fix	Tests
F1	blind scan (0 / NaN sectors) read as clear in path shift and planner	blocked = stop	u/test_shift:: test_blind_scan_stops, r/test_planner_node:: test_blocked_replans_ and_blind_scan_fails, u/test_avoidance:: test_blind_sectors_ ahead_stop_the_robot, r/test_obstacle_ avoidance_node:: test_blind_scan_is_an_ emergency
F2	a stop could leave a moving command last; no zero on shutdown	zero on entering INACTIVE, shutdown hook, state lock	u/test_follower:: test_stop_zeroes_once_ and_start_resumes, r/test_ path_follower_node:: test_follow_pause_estop_ resume_and_shutdown
F3	start raced the control tick	one mutually exclusive group for state changes and the control loop, asynchronous hook calls	r/test_path_follower_ node:: test_follow_pause_estop_ resume_and_shutdown
F4	path-shift speed limit from the commanded offset only	also the measured offset	u/test_shift:: test_measured_offset_ limits_the_speed
F5	progress jumped to a parallel leg while shifted	windowed, offset-compensated projection, capped advance (core/path/progress.py)	u/test_path:: test_shifted_progress_ stays_on_its_leg, u/test_follower:: test_overlapping_legs_ keep_progress_and_stops
F6	stop points skipped while shifted or triggered early from another leg	detection by progress and distance minus offset; wait at a stop with an obstacle	u/test_follower:: test_overlapping_legs_ keep_progress_and_stops, u/test_follower:: test_stop_point_with_an_ obstacle_waits_instead_ of_passing
F7	joining and final approach checked by the front cone only; timers ran while paused	360° motion guard in every phase; timers frozen	u/test_follower_safety:: test_joining_is_guarded_ by_the_scan, u/test_follower:: test_dwell_timer_is_ frozen_while_paused
F8	snapped goals never arrived	arrival against the plan goal	u/test_session:: test_arrival_is_tested_ against_the_snapped_goal, r/test_planner_node:: test_go_to_arrives_on_ the_plan_goal_and_stops
F9	planner failed but the follower kept going	stop on every failure after set_path	u/test_session:: test_follower_failure_ stops_and_fails, u/test_session:: test_blocked_replans_ then_gives_up_and_stops

ID	Finding	Fix	Tests
F10	cancel blocked behind planning	own callback group, cancel flag	<code>r/test_planner_node::test_cancel_is_not_blocked_by_a_slow_send</code>
F11	numeric parameters unchecked (NaN, zero temperature, negative accelerations)	typed configuration with ranges, NaN $\rightarrow$ 0	<code>u/test_follower_safety::test_config_ranges_and_cross_checks,</code> <code>u/test_follower_safety::test_shaper_envelope_slew_and_nan,</code> <code>r/test_path_follower_node::test_invalid_parameters_are_rejected</code>
F12	route processing dropped stop points closer than 3 cm	anchors kept	<code>u/test_path::test_close_stop_points_are_all_kept</code>
F13	avoidance smoothing slowed braking	asymmetric smoothing	<code>u/test_avoidance::test_smoothing_never_slows_braking,</code> <code>u/test_avoidance::test_policy_brakes_in_one_scan_with_heavy_smoothing</code>
F14, B2	default <code>cmd_vel</code> bypassed the mux	<code>control/nav_cmd_vel</code> (fixed name, <code>utils/outputs.py</code>)	<code>r/test_path_follower_node::test_follow_pause_estop_resume_and_shutdown</code>
F15	path shift kept after a mode change; costmap cache not invalidated; planner pose without TF age check	reset on mode change, rebuild on parameter change, age check	<code>u/test_follower::test_mode_change_drops_the_path_shift,</code> <code>r/test_planner_node::test_go_to_arrives_on_the_plan_goal_and_stops</code> (costmap dropped on a parameter change); planner pose age (<code>tf.fresh</code> with <code>tf_timeout_sec</code>): no dedicated test
F16	“paused until relocalized” never un-paused	pause reasons (<code>core/follower/pause.py</code>)	<code>u/test_follower_safety::test_relocalization_unpauses,</code> <code>u/test_follower_safety::test_estop_is_sticky_and_remote_is_transient</code>
F17	one-point path accepted; pending inspections survived a restart; route markers keyed on the count; line of sight through diagonal pinches	two points required; inspection rounds; markers by route; corner cells counted	<code>u/test_path::test_route_needs_two_points_and_length,</code> <code>u/test_follower::test_late_inspection_answers_are_dropped_after_a_restart,</code> <code>u/test_planning::test_line_of_sight_blocks_diagonal_pinches</code>

ID	Finding	Fix	Tests
F18	missing tests (transitions, zero on stop / e-stop / stale / pause / shutdown, path shift lag, blind scans, overlapping legs)	added	u/test_follower:: test_transition_table, u/test_follower_safety:: test_every_hold_ publishes_zero_every_ tick and the tests above

A.6 dharma_control

ID	Finding	Fix	Tests
C1 (P0)	rotation not limited by the field	rotation room and combined-motion arc (core/field.py, core/arc.py)	u/test_field:: test_rotation_next_to_ an_obstacle_is_limited, u/test_field::test_ rotation_limit_follows_ the_yaw_stopping_model, u/test_field:: test_combined_motion_ checks_the_arc, r/test_control_mux_node:: test_field_limits_ rotation_beside_an_ obstacle
C2 (P0)	software-only safety	control/hw_estop, two-channel architecture documented (README, Section 9.1)	u/test_arbiter::test_ hardware_estop_engages_ and_blocks_release, u/test_arbiter::test_ silent_hardware_estop_ engages_when_required
N3	own stopping model	dharma_utils.safety only	u/test_field::test_wall_ ahead_limits_with_the_ shared_stopping_model
–	field off in REMOTE by default	field.in_remote true with its own margin	r/test_control_mux_node:: test_field_on_in_remote_ by_default, u/test_field::test_ remote_uses_its_margin_ and_can_be_switched_off
B1	blind LiDAR (all sectors 0) read as clear	blocked sectors are obstacles, BLIND → zero	u/test_scan:: test_all_sectors_zero_ is_blind_not_clear, u/test_field::test_blind_ scan_stops_auto, r/test_control_mux_node:: test_blind_scan_stops_ auto_and_creeps_remote

ID	Finding	Fix	Tests
B3	e-stop resumed through set_mode; fire-and-forget pause calls; release forwarded the held stick command	no navigation calls; re-arm with neutral commands (core/arbiter.py)	u/test_arbiter::test_release_requires_rearm_and_never_forwards_the_held_command, u/test_arbiter::test_switch_to_remote_ignores_a_stick_held_during_the_switch
#3	SetBool default request released the e-stop; anyone released anyone's	Trigger engage and release	r/test_control_mux_node::test_estop_service_holds_zero_and_release_needs_rearm
#4	watchdog and hold on the ROS clock	steady clock	r/test_control_mux_node::test_watchdog_and_hold_run_on_steady_time_with_sim_time
#5	parameters ignored at runtime; max_decel: 0 crashed; NaN limits disabled the envelope	read-only validated parameters; non-finite → zero	u/test_arbiter::test_config_validation, r/test_control_mux_node::test_bad_values_refuse_to_start, u/test_envelope::test_non_finite_on_any_axis_is_zero
#6	no zero on shutdown or crash	shutdown hook, e-stop on callback errors	r/test_control_mux_node::test_shutdown_hook_publishes_zeros, r/test_control_mux_node::test_callback_error_engages_the_estop
#7	scan freshness by receive time only; NaN sectors free; frame and coverage unchecked	scan validation (core/scan.py)	u/test_scan::test_header_age_old_and_future, u/test_scan::test_wrong_frame_is_rejected, u/test_scan::test_partial_coverage_is_rejected, u/test_scan::test_blocked_or_unknown_sectors_are_blind
#8	several remote operators on one topic	one topic per source	u/test_arbiter::test_only_the_active_source_is_forwarded, r/test_control_mux_node::test_remote_sources_and_set_mode
#13	re-sending the same mode injected a zero	no-op	u/test_arbiter::test_set_mode_same_mode_and_source_is_a_no_op
#14	a second e-stop reason overwrote the first without republish	reasons accumulate, status republished	u/test_arbiter::test_estop_reasons_are_kept_oldest_first_without_duplicates, r/test_control_mux_node::test_estop_request_topic_appends_reasons

A.7 dharma_health

ID	Finding	Fix	Tests
H1 (P2)	status as JSON in std_msgs/String, not actionable	HealthStatus + DiagnosticArray, speed_scale for the mux	u/test_report:: test_lidar_rate_low_or_ tf_late_halves_the_speed, u/test_report:: test_error_stops_and_ comes_first, u/test_report:: test_lowest_matching_ rule_wins_and_rules_ never_raise_an_error, r/test_health_node:: test_core_states_match_ the_messages
–	rates only: a 10 Hz topic with 2 s old stamps passed	header stamp age per topic, negative ages; raw subscription, stamp read from the CDR bytes	u/test_sensors:: test_10hz_topic_with_2s_ old_stamps_warns, u/test_sensors::test_ stamps_from_the_future_ are_a_clock_offset, u/test_sensors::test_ slow_and_silent_topics, u/test_sensors::test_cdr_ stamp_both_byte_orders, r/test_health_node:: test_stamp_from_a_real_ serialized_message
T1–T5	TF staleness without diagnosis	per-edge ages classified as clock offset or load; optional chrony check	u/test_tf:: test_steady_late_edge_ is_a_clock_offset, u/test_tf::test_growing_ ages_are_load, u/test_tf:: test_stopped_publisher, u/test_tf::test_future_ stamps_and_few_samples, u/test_tf::test_clock_ check_runs_the_command_ once_per_period, r/test_health_node:: test_tf_edge_with_a_ steady_offset_is_a_ clock_problem
#9	health OK while not localized	NONE is a warning	u/test_state::test_not_ localized_is_a_warning, r/test_health_node:: test_old_stamps_not_ localized_then_ok_then_ lost

ID	Finding	Fix	Tests
–	safety scan and control field not watched	state:safety_scan and state:control (stale or blind is an error)	u/test_state:: test_safety_scan, u/test_state:: test_control_states, r/test_health_node:: test_blind_safety_scan_ is_an_error
#13	missing-node check ignored namespaces	fully qualified names	u/test_state::test_nodes_ with_namespaces, r/test_health_node:: test_expected_node_in_ another_namespace_is_ missing
#17	unreadable thermal zone or disk reported OK	“unreadable” warning, thermal zone whitelist	u/test_system::test_ unreadable_values_warn, u/test_system::test_ whitelisted_zones_only, u/test_system::test_no_ whitelisted_zone_warns

A.8 dharma_sim

ID	Finding	Fix	Tests
S1 (P2)	perfect odometry, stamps with now()	drift model, latency, LiDAR dropout, seeded scenarios	u/test_odometry:: test_random_walk_grows_ with_the_square_root_of_ distance, u/test_odometry:: test_scale_error_and_ yaw_bias_accumulate, u/test_odometry:: test_same_seed_same_run, u/test_latency:: test_sensor_stamp_is_ now_minus_latency, u/test_scenario:: test_seeded_generators_ are_reproducible_and_ independent, u/test_scenario::test_ dropouts_and_restarts, r/test_sim_robot_node:: test_stamps_carry_the_ latency, r/test_sim_robot_node:: test_scenario_dropout_ silences_the_lidar

ID	Finding	Fix	Tests
#10	ray cast blind from inside a circle; no collisions; LiDAR ignored the vertical field of view and the minimum range	fixed ray cast, collision latch, Mid-360 model	u/test_world:: test_ray_from_inside_a_circle_sees_its_far_side, u/test_world:: test_collisions_with_walls_and_circles, u/test_body:: test_collision_stops_and_latches, u/test_lidar::test_low_obstacle_hidden_below_the_lowest_beam, u/test_lidar:: test_min_and_max_range, r/test_sim_robot_node:: test_collision_latches_and_stops
#11	imu_in_lidar sign (FAST-LIO's T_{IL})	Livox manual values, derived consistently	u/test_odometry:: test_imu_offset_follows_the_livox_manual, u/test_odometry:: test_lio_pose_closes_the_bridge_chain
#14	fixed Δt per wall-timer tick (time dilation under load)	measured Δt , clamped to max_dt	u/test_body::test_measured_dt_not_ticks, u/test_body::test_stall_is_clamped_to_max_dt

A.9 dharma_rviz

ID	Finding	Fix	Tests
#9	Tuning: live apply by default, wheel scrolling edits gains, no bounds	apply-on-change off, ranges from descriptors, wheel only with focus	g/test_tuning:: TuningTest.EditorsFollowTheDescriptors, g/test_tuning::TuningTest. WheelIgnoredAndEditsWaitForApply
#10	duplicate panels in robot.rviz; fixed node names per panel	duplicates removed, unique node names	g/test_panels:: PanelTest.PanelsGetUniqueNodeNames, g/test_common:: Names.UniqueNodeNamesPerInstance, g/test_plugins::Plugins.EveryPanelLoads; robot.rviz: no dedicated test
#11	no e-stop, mode or field state in RViz; Start resumed during an e-stop	Safety panel; Start disabled while ControlStatus. estop	g/test_panels::PanelTest. MissionStartDisabledDuringEstop, g/test_panels:: PanelTest.SafetyEstopAlwaysEnabled, g/test_common::Format. NavigationButtonsFollowStateAndEstop

ID	Finding	Fix	Tests
#15	<code>wait_for_service</code> blocked the Qt thread; load race; 4 decimals; string arrays split on commas	asynchronous readiness, client per request, full precision	<code>g/test_panels::PanelTest.ServiceCallTimesOut,</code> <code>g/test_panels::PanelTest.ServiceCallReportsUnavailableAndSuccess,</code> <code>g/test_tuning::TuningTest.SecondLoadWins,</code> <code>g/test_tuning::TuningYaml.ExportKeepsPrecision,</code> <code>g/test_common::Yaml.ListLines</code>
#16	Tag panel buttons stuck; only the first diagnostic shown; values never stale	timeouts, worst status, age indicator	<code>g/test_panels::PanelTest.TagButtonsComeBackAfterTheReply,</code> <code>g/test_common::Diagnostics.WorstOfAllFreshStatuses,</code> <code>g/test_common::Staleness.GoesStaleAfterTimeout</code>

A.10 dharma_interfaces and cross-stack findings

ID	Finding	Fix	Tests
I1 (P2)	core and application (tour guide) types mixed	POI, tour, visitor and mission types removed; products own them	–
N4	<code>AvoidanceCommand.steering_bias</code> unit ambiguous	<code>heading_offset</code> (rad), <code>yaw_rate_bias</code> (rad/s)	see <code>dharma_navigation</code> N4
H1	status as strings everywhere	typed status messages with enum constants	package tests of every publisher
P2 (§5.3)	duplication: 15 <code>main()</code> boilerplates, 11 latched QoS definitions, 5 yaw implementations, 7 identical <code>test_node_members.py</code>	<code>dharma_utils</code> (run, QoS, math, check_package)	<code>dharma_utils</code> unit tests; <code>test_style</code> in every package
§5.2	robot description in about six files	<code>RobotProfile</code> shared through <code>robot.*</code>	<code>u/test_config::test_robot_profile_derived_values</code> (<code>dharma_utils</code>)

Bibliography

- [1] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, “FAST-LIO2: Fast direct LiDAR-inertial odometry,” *IEEE Transactions on Robotics*, 2022.
- [2] G. Ou, D. Li, and H. Li, “Leg-KILO: Robust kinematic-inertial-lidar odometry for dynamic legged robots,” *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8194–8201, 2024.
- [3] E. S. Page, “Continuous inspection schemes,” *Biometrika*, vol. 41, no. 1-2, pp. 100–115, 1954.
- [4] E. Olson, “AprilTag: A robust and flexible visual fiducial system,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011.
- [5] J. Wang and E. Olson, “AprilTag 2: Efficient and robust fiducial detection,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2016.
- [6] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York: John Wiley & Sons, 2001.
- [7] E. B. Olson, “Real-time correlative scan matching,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 4387–4393.
- [8] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [9] R. C. Coulter, “Implementation of the pure pursuit path tracking algorithm,” The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, Jan. 1992.
- [10] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron *et al.*, “Stanley: The robot that won the DARPA Grand Challenge,” *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [11] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, “Information theoretic MPC for model-based reinforcement learning,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.
- [12] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a Frenet frame,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010, pp. 987–993.
- [13] J. Borenstein and Y. Koren, “The vector field histogram – fast obstacle avoidance for mobile robots,” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.
- [14] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
- [15] V. Sezer and M. Gokasan, “A novel obstacle avoidance algorithm: “Follow the Gap Method”,” *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.
- [16] ISO, “ISO 13849-1:2023, safety of machinery – safety-related parts of control systems – part 1: General principles for design,” International Organization for Standardization, 2023.
- [17] IEC, “IEC 61508-1:2010, functional safety of electrical/electronic/programmable electronic safety-related systems – part 1: General requirements,” International Electrotechnical Commission, 2010.
- [18] ISO, “ISO 13482:2014, robots and robotic devices – safety requirements for personal care robots,” International Organization for Standardization, 2014.

- [19] —, “ISO/TS 15066:2016, robots and robotic devices – collaborative robots,” International Organization for Standardization, 2016.
- [20] J. A. Marvel and R. J. Norcross, “Implementing speed and separation monitoring in collaborative robot workcells,” *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 144–155, 2016.