

HOTDOG

Technician Guide

Configuration, tuning and maintenance of HotDog on the Lynx M20

Version 1.0 · 2026-11-01

For the technician who commissions, tunes and maintains the robot: shell, YAML and ROS 2 basics assumed.

Revision history

Version	Date	Changes
1.0	2026-11-01	First version

The first bring-up of a robot and a site, step by step, is in docs/commissioning.md; the operators' tasks in the HotDog User Manual. This guide explains what each setting does, how to change it for good, and how to keep the robot fit. The parameter reference at the end is generated from the packages' config files.

Contents

1 How configuration works 5
- 1.1 Where a value comes from 5
- 1.2 The env file 5
- 1.3 The site file 7
- 1.4 Try a value live, then keep it 7

2 The M20 8
- 2.1 The driver and its calibration 8
- 2.2 The robot profile and braking 8
- 2.3 Odometry only 9

3 LiDARs and the protective field 10
- 3.1 The Airy 10
- 3.2 What counts as an obstacle 10
- 3.3 The protective field 10

4 Patrols and the watchdog 11
- 4.1 Speed and stop points 11
- 4.2 The watchdog 11
- 4.3 The schedule 12

5 Thermal inspection 13
- 5.1 The pod 13
- 5.2 Targets in the inspection file 13
- 5.3 Limits 14

6 Partial discharge 15
- 6.1 The camera 15
- 6.2 One detection 15
- 6.3 Severities 15

7 Reports and alerts 16
- 7.1 Reports 16
- 7.2 Alerts 16

8 Network, time and the service 17
- 8.1 Addresses and ports 17
- 8.2 Service commands 17
- 8.3 Time 17

9 Maintenance 18
9.1 Schedule 18
9.2 Update the software 18
9.3 Back up and restore 19

10 Symptoms and fixes 20

11 Parameter reference 22
11.1 hotdog · robot.yaml 22
11.2 hotdog · m20_driver.yaml 23
11.3 hotdog · stack.yaml 24
11.4 hotdog_base · base.yaml 26
11.5 hotdog_inspection · inspection.yaml 27
11.6 hotdog_patrol · patrol.yaml 29
11.7 hotdog_reports · reports.yaml 30
11.8 hotdog_alerts · alerts.yaml 31

1 How configuration works

1.1 Where a value comes from

Every node reads its parameters in layers; a later layer wins, parameter by parameter:

Layer	File	Holds
package defaults	`src/<package>/config/*.yaml`	what every HotDog has
robot profile	`src/hotdog/config/robot.yaml` (`robot_params`)	the M20: footprint, height, limits, frames
the stack as tuned for HotDog	`src/hotdog/config/stack.yaml`	odometry only, the merged cloud, the hooks
this robot and site	`/etc/hotdog/site.yaml` (`PARAMS_FILE`)	what you tune: the only file you edit
launch arguments	`/etc/hotdog/hotdog.env`	addresses, the site, secrets, what runs

The repository files are never edited on a robot: an update would undo the change or conflict with it. Everything this robot and this substation need goes into `/etc/hotdog/site.yaml`, under the node it belongs to, and survives updates.

1.2 The env file

`/etc/hotdog/hotdog.env` is read by the service and by `scripts/env.sh` (the `hotdog` command). No comments after a value; quotes around values with spaces.

Variable	Meaning
`ROBOT_IP`	the M20's UDP server (`10.21.31.103`)
`DRIVER_PARAMS`	this robot's m20_driver calibration (`/etc/hotdog/m20_driver.yaml`)
`PARAMS_FILE`	this robot and site (`/etc/hotdog/site.yaml`)

Variable	Meaning
`ROBOT_PARAMS`	this robot's profile when it differs (`/etc/hotdog/robot.yaml`)
`SITE` , `SITES_DIR`	the site to load; where the sites live (`~/.hotdog/sites`)
`POD_IP`	the UniPod MT11
`HERTZINO_PASSWORD`	the acoustic camera's password
`TELEGRAM_BOT_TOKEN`	the alert bot
`HOTDOG_WEBHOOK_TOKEN`	bearer of the webhook
`INSPECTION` , `PATROL` , `REPORTS` , `ALERTS` , `POD`	`false` leaves that part out (default `true`)
`LIDAR`	`rslidar` (both Airy) or `external` (clouds published elsewhere)
`ROS_DOMAIN_ID` , `RMW_IMPLEMENTATION`	`0` , `rmw_fastrtps_cpp`

The file is mode 640: it holds secrets. After a change: `hotdog restart` .

1.3 The site file

```
# /etc/hotdog/site.yaml: GI Cawang, HotDog 01
path_follower_node:
  ros__parameters:
    speed.cruise: 0.4
inspection_node:
  ros__parameters:
    thermal:
      ambient_c: 31.0
patrol_node:
  ros__parameters:
    schedule:
      enabled: true
      times: ["06:00", "14:00", "22:00"]
      route: "2026-11-01_09-30-00"
report_node:
  ros__parameters:
    title: GI Cawang 150 kV
    robot_name: HotDog 01
alert_node:
  ros__parameters:
    robot_name: HotDog 01 - GI Cawang
    language: id
    report_url: http://10.21.31.50:8088/
    telegram:
      chat_ids: ["-1001234567890"]
```

At start the bringup merges the layers, reports a section that matches no node (a typo) and refuses values out of range with the reason; check `hotdog logs` after every change.

1.4 Try a value live, then keep it

Most parameters are live:

```
source ~/dev/HotDog/scripts/env.sh
ros2 param describe /path_follower_node speed.cruise    # unit, range,
live or not
ros2 param set /path_follower_node speed.cruise 0.35
```

A live change is lost at the next restart: write the value you settled on into the site file. A parameter marked read-only needs the site file and `hotdog restart`.

2 The M20

2.1 The driver and its calibration

`m20_driver` talks to the M20's UDP server and turns `cmd_vel` into axis commands. Its own README has the verification on the real robot: the angle unit, standing, the emergency stop, the takeover by the remote, the speed calibration per gait, the signs of turning and strafing. Do it once per robot and keep the result in a copy of the bringup's driver file:

```
sudo install -D -m 644 ~/dev/HotDog/src/hotdog/config/m20_driver.yaml /
etc/hotdog/m20_driver.yaml
sudoedit /etc/hotdog/m20_driver.yaml      # angle_unit,
full_scale.<gait>.*, invert_*
```

Keep `odom_frame: camera_init`, `base_frame: body` and `publish_tf: true`: they make the M20's odometry HotDog's odometry. HotDog patrols in the agile flat gait (`stand_up_gait`).

2.2 The robot profile and braking

`src/hotdog/config/robot.yaml` describes the M20 for every node: the footprint (body, legs and wheels, **with the payload mounted**), `base_height`, the speed and acceleration limits and the braking. The protective field, the planner's clearance and the LiDAR self filter follow from it. A robot whose payload sticks out further needs its own profile: copy the file to `/etc/hotdog/robot.yaml`, change it, and name it in the env file (`ROBOT_PARAMS`).

`max_decel` and `reaction_time` decide how early the robot brakes for an obstacle. **Measure them** (the Dharma hub's Robot Profile guide): drive at full patrol speed on the site's ground, stop, measure the distance. Keep `max_decel` at or below the measured deceleration and `reaction_time` at or above the measured delay. Gravel and wet concrete brake worse than the workshop floor.

2.3 Odometry only

HotDog maps nothing. The M20's dead reckoning is the odometry and the odometry start is the map origin: the dock.

- **Drift.** The error grows with the distance walked; `localization/status` reports it as `sigma_xy` (`hotdog status`: drift). Check on the first patrols of a route that the robot stops where the stop points were taught; split routes that end up too far off.
- **Re-anchoring.** Every patrol starts with `localization/assume_origin` at the dock (`patrol_node` `start.reanchor`). A robot moved by hand must be put back on the dock and re-anchored (`hotdog anchor`).
- **A driver restart resets the odometry.** The odometry monitor reports it, localization drops to NONE, the follower pauses and the watchdog stops the patrol after `abort.hold_timeout_sec`.
- **Tags** along a long route correct the drift: `fiducials:=true` with a calibrated camera and AprilTags recorded on the site (dharma_localization). Without them, keep routes short.

3 LiDARs and the protective field

3.1 The Airy

`rslidar_sdk` runs both RoboSense Airy from `src/hotdog_base/config/rslidar.yaml`: the front one on UDP 6699/7788, the rear one on 6698/7789, stamped on arrival (no PTP needed). `cloud_merger_node` merges them into `/lidar/points` in `base_link`: each input transformed by the URDF plus `input_rpy_deg`, the robot's own body removed (footprint, `self_height` for the pod and rails), leveled with the M20's IMU, cut to `range_max` and `height_max`. A stale input stops the merged cloud (`require_all`): perception then reports BLOCKED and the robot stops instead of driving half blind.

Check after mounting or replacing a LiDAR: in RViz the rear cloud lies behind the robot model and the ground stays flat at `-base_height` while the body tilts; else fix `input_rpy_deg` in the site file.

3.2 What counts as an obstacle

`lidar_sector_node` turns the merged cloud into 360° sectors between `filter.height_min` (0.15 m: gravel, kerbs, trench covers stay out) and `filter.height_max` (1.6 m) above the ground. Lower objects, pits and drops are not seen: the route must not need it.

3.3 The protective field

`control_mux_node` is the only publisher of `/m20/cmd_vel`. For every command it checks the 360° scan along the direction of motion with the footprint and the braking model: it slows the robot down (field LIMITED) and stops it (field STOP) so that it can always stop before the nearest obstacle. A stale or blind scan stops it too. The health monitor's speed limit and the latched e-stop apply to every source.

Test it after every change of the profile or the LiDARs, in an open area, at patrol speed: a person walks slowly towards the robot from the front, the back and each side; the robot must stop before them every time.

4 Patrols and the watchdog

4.1 Speed and stop points

Parameter (node)	Default	Effect
`speed.cruise` (`path_follower_node`)	0.5	m/s along the route; 0.3 for the first patrols of a route
`hooks.inspection_timeout_sec` (`path_follower_node`)	180	s the robot waits at most for the inspection of a stop
`max_duration_sec` (`inspection_node`)	150	s for all targets of a point; keep below the hook timeout
`match_radius_m` (`inspection_node`)	1.5	m, a stop this close to an inspection point inspects it

A stop point waits its dwell (taught, default 5 s) plus the inspection. Each target takes the gimbal settle time (2.5 s), the thermal samples and, with `acoustic`, the detection time (5 s).

4.2 The watchdog

`patrol_node` `abort.*`, every second while a patrol runs:

Parameter	Default	When exceeded
`min_battery`	0.2	back to the dock; keep above the follower's `critical_battery` (0.1)
`max_duration_min`	120	back to the dock; set it to about twice the route's duration
`hold_timeout_sec`	180	blocked: back to the dock; off the route or lost: stop
`nav_timeout_sec`	5	the follower is silent: stop
`return_to_dock`	true	false: every watchdog action stops the robot instead
`near_m` , `dock_clearance_m`	2.0, 1.0	where the way back starts and how far before the dock it ends

An operator pause, the e-stop or REMOTE is never overruled. The way back is the taught route, reversed from the nearest waypoint of its first pass, without stop points, into `<site>/patrol/return.csv`; then `charge_start` docks the M20.

4.3 The schedule

`schedule.enabled`, `schedule.times` (local time of the robot computer, `HH:MM`), `schedule.route` (empty: the route patrolled last) and `schedule.window_min` (a time missed by more than this is skipped). Keep them in the site file; operators switch the schedule on and off with `hotdog schedule`. The robot computer's time zone must be the substation's (`timedatectl`).

5 Thermal inspection

5.1 The pod

`unipod_driver` runs the UniPod MT11 in the `/unipod` namespace (`POD_IP`). `inspection_node` points it with `set_angle` and `set_zoom` , waits `pod.settle_sec` , refreshes the flat-field correction once per point (`thermal.refresh_shutter`) and asks the temperatures of boxes in thermal pixels. With the camera's AI super-resolution on, the thermal frame is 1280 × 1024: set `thermal.width` and `thermal.height` to match.

The MT11 gives temperatures per point, box and frame, not a radiometric image: the reports show the camera's colour picture as evidence, and the values come from the box measurements.

5.2 Targets in the inspection file

The operators teach targets with `hotdog teach target` ; refine them in `<sites_dir>/<site>/inspection.yaml` (read at every inspection, no restart):

```
points:
  - name: bay1_disconnector
    x: 12.40
    y: -3.10
    targets:
      - name: bay1_ct_terminals
        pan_deg: -20.0
        tilt_deg: 5.0
        zoom: 2.0
        thermal:
          regions: # boxes in thermal pixels, compared with each other
            - { name: r, box: [80, 120, 200, 260] }
            - { name: s, box: [260, 120, 380, 260] }
            - { name: t, box: [440, 120, 560, 260] }
          max_temp_c: 95.0 # this part's own critical limit
        acoustic: { enabled: true, duration_sec: 8.0 }
```

Regions are better than strips when the phases are not evenly spread across the picture or other equipment shares the frame. Find the boxes on a thermal snapshot of the target (its report folder) at its zoom.

5.3 Limits

`thermal.absolute`, `thermal.ambient`, `thermal.similar` each hold `attention`, `warning` and `critical` in °C. The defaults follow the usual infrared inspection practice (NETA MTS):

Criterion	Value judged	attention / warning / critical
absolute	the hottest temperature	80 / 90 / 105 °C
ambient	the hottest temperature over `thermal.ambient_c`	11 / 21 / 41 °C
similar	the hottest region over the median of the others	2 / 4 / 16 °C

Use the substation owner's limits where they exist, and the equipment makers' limits for parts with their own (`max_temp_c`). Readings depend on the load: note in the site's records that limits were set for the usual load, and read rises over ambient with the load of the day.

`thermal.ambient_c` is live; operators set it every shift (`hotdog ambient`).

6 Partial discharge

6.1 The camera

The HZ-FA-371 is driven over its HTTP API (manual V5.1) by `inspection_node`; its ROS driver is ROS 1 only. Set up on the camera's web page once: **partial discharge mode** (gas leak mode is refused), the analysis band and the imaging threshold the manual gives for HV equipment. `acoustic.host` (factory `192.168.2.204`), `acoustic.port`, `acoustic.user`; the password only in the env file (`HERTZINO_PASSWORD`).

6.2 One detection

For each acoustic target, after the robot stopped: a detection already running is stopped, the newest record noted, a detection of `duration_sec` started, the record appearing a whole detection later taken (an older one never), the detection stopped, the sound map and PRPD photos saved next to the report.

6.3 Severities

Type	Default	Parameter
noise	normal	
corona	attention	`acoustic.corona`
surface	warning	`acoustic.surface`
floating	critical	`acoustic.floating`

`acoustic.intensity_warning` and `intensity_critical` raise the severity by the sound map's intensity once the site's levels are known (0: off): record a few rounds first, then set them above the usual levels of healthy equipment.

7 Reports and alerts

7.1 Reports

`report_node` renders a page per patrol, a page per day and a front page, and serves `~/.hotdog/reports` read-only on `server.port` (8088). `title` and `robot_name` name the site and the robot on every page. Retention: days older than `retention.days` (365), then the oldest days while less than `retention.min_free_gb` (5 GB) is free; never today. `reports/rebuild` renders every page again (after changing the title).

Keep what must be archived elsewhere: copy the day folders off the robot (with their images), e.g. weekly with `rsync -a robot@<ip>:.hotdog/reports/ <archive>/`.

7.2 Alerts

`alert_node` sends to the Telegram chats in `telegram.chat_ids` (the bot token in `TELEGRAM_BOT_TOKEN`) and to `webhook.url`. Set up a bot: @BotFather, add it to the crew's group, send a message there, read the group id (negative) from `https://api.telegram.org/bot<token>/getUpdates`. `language: id` writes the alerts in Bahasa Indonesia; `report_url` adds the link to the patrol page; `findings.min_severity` (warning) and `findings.repeat_hours` (12) decide what is sent. Alerts are spooled in `~/.hotdog/alerts` and sent every `retry_sec` until delivered, for `max_age_hours`.

The robot computer needs outbound HTTPS to `api.telegram.org`; a proxy goes into the env file (`HTTPS_PROXY=...`).

8 Network, time and the service

8.1 Addresses and ports

Device / service	Address (default)	Port
Lynx M20	`10.21.31.103`	UDP 30000
RoboSense Airy front / rear	send to the payload computer	UDP 6699/7788, 6698/7789
UniPod MT11	`10.21.31.26` (factory `192.168.144.25`)	TCP 37260, RTSP 8554
Hertzinno HZ-FA-371	`192.168.2.204` (factory)	HTTP 80
report server	the payload computer	TCP 8088
SSH	the payload computer	TCP 22

The payload computer needs an address in each subnet (netplan, docs/commissioning.md). ROS 2 runs with `rmw_fastrtps_cpp` on domain 0.

8.2 Service commands

```
sudo systemctl status hotdog          # running? since when?
hotdog logs                           # journalctl -fu hotdog
hotdog restart                        # sudo systemctl restart hotdog
scripts/setup.sh                      # every component: ok or missing
```

The service restarts HotDog 5 s after it stops for any reason and deletes ROS logs older than `LOG_DAYS` (7) at each start.

8.3 Time

Every stamp comes from the robot computer's clock: `scripts/setup.sh install timesync` (chrony) keeps it right and `health/status` warns when it drifts. The schedule runs on its local time: set the time zone with `sudo timedatectl set-timezone Asia/Jakarta` (or the site's).

9 Maintenance

9.1 Schedule

When	Task
every shift	`hotdog status`; ambient temperature; reports and alerts of the previous shift read
weekly	clean the Airy windows, the pod's lenses and thermal window, the acoustic camera's front with a soft, dry cloth; check the payload rails' fasteners; dock contacts clean; copy the reports off the robot
monthly	`scripts/setup.sh`; free disk (`hotdog status`: disk); a patrol watched end to end; the protective field test (section 4.3); stop point positions on the longest route
quarterly	thermal check against a reference (a calibrated contact thermometer on a warm part, or a blackbody), within the pod maker's accuracy; acoustic camera against a known source; the M20's battery health and maintenance per its manual
yearly	the Site Acceptance Test again; the limits reviewed with the substation owner

Record every maintenance with its date and result.

9.2 Update the software

```
cd ~/dev/HotDog
git pull
scripts/setup.sh install repos rosdep build
hotdog restart
scripts/setup.sh && hotdog status
```

Update with the robot on its dock and the schedule off; switch it on again after a patrol was watched end to end. Read the CHANGELOG first: a new parameter may need a value in the site file.

9.3 Back up and restore

What	Where	Why
the sites	`~/.hotdog/sites/`	routes and inspection points: hours of teaching
the robot's settings	`/etc/hotdog/` (env, site, driver)	everything tuned on this robot
the reports	`~/.hotdog/reports/`	the inspection record

```
sudo tar czf hotdog-backup-$(date +%Y%m%d).tgz /etc/hotdog ~/.hotdog/
sites
```

Restore onto a new payload computer: install (docs/commissioning.md), unpack the archive, `hotdog restart`. A site taught on one M20 works on another only if its dock is the same.

10 Symptoms and fixes

Symptom	Cause and fix
the bringup stops at start	`hotdog logs` : an unknown section, a value out of range, a missing file named in the env file
health WARN: a sensor below its rate	the LiDAR (`ros2 topic hz /lidar/front/points`), the M20 odometry (`/m20/odom`), the network cable
health ERROR: tf, odometry stale	the M20 driver lost the robot: `ping $ROBOT_IP` , the M20 switched on, `/m20/state`
field BLIND or STALE, robot does not move	a LiDAR is down or its packets don't arrive: `require_all` stops the cloud; check both
the robot stops for nothing	dust or rain on the Airy windows; `filter.height_min` too low on uneven ground; the self filter too small after a payload change
it drifts off the route in turns	`speed.cruise` lower; the M20's speed calibration (driver README)
it stops short of the stop points far from the dock	odometry drift: split the route, or tags
thermal values far from the reference	lens dirty or wet; emissivity of the target (the pod's settings); the target's region misses the part
`similar` findings on equipment with one hot phase by design	use `regions` covering comparable parts only, or `thermal: { enabled: true, strips: 0 }`
acoustic: every target `noise` , known discharge missed	the camera's band and threshold; `duration_sec` longer; distance and direction of the target
acoustic: "gas leak mode"	the camera's web page: partial discharge mode
alerts waiting, never sent	`hotdog logs` (alert not sent: HTTP 401: wrong token; 400: wrong chat id; network: no route or proxy)
reports server unreachable	the substation firewall (TCP 8088), `server.host` , another program on the port (`hotdog logs`)

Symptom	Cause and fix
disk full	`retention.days` lower, `retention.min_free_gb` ; ROS logs are pruned at each start

11 Parameter reference

Every parameter of HotDog's own config files, with its default and what it does, read from the files themselves. Dharma's parameters are in each `dharma_*` repository's README; `ros2 param describe /<node> <name>` gives the unit, the range and whether it is live.

11.1 hotdog · robot.yaml

robot profile: footprint, limits, base height, frames · `src/hotdog/config/robot.yaml`

```
/**
```

Parameter	Default	Meaning
`robot.name`	`lynx_m20`	name of the robot profile
`robot.kinematics`	`omni`	the M20 walks and rolls in x, y and yaw
`robot.base_height`	`0.42`	m, base_link (body centre) above the ground when standing
`robot.footprint.front`	`0.43`	
`robot.footprint.rear`	`0.43`	
`robot.footprint.half_width`	`0.25`	
`robot.limits.max_vx`	`0.8`	m/s, patrol on gravel and cable trenches
`robot.limits.max_vy`	`0.3`	m/s
`robot.limits.max_wz`	`0.8`	rad/s
`robot.limits.max_accel`	`0.6`	m/s^2
`robot.limits.max_decel`	`0.8`	m/s^2, braking: measure it (Dharma hub, Robot Profile)
`robot.limits.max_yaw_accel`	`1.5`	rad/s^2
`robot.limits.reaction_time`	`0.35`	s, cmd_vel -> UDP axis command at 20 Hz -> RL controller
`robot.frames.map`	`map`	TF frame name
`robot.frames.odom`	`odom`	TF frame name

Parameter	Default	Meaning
`robot.frames.base`	`base_link`	TF frame name

11.2 hotdog · m20_driver.yaml

the M20 driver as HotDog runs it · `src/hotdog/config/m20_driver.yaml`

```
/**
```

Parameter	Default	Meaning
`robot_ip`	`10.21.31.103`	the M20's UDP server on the wired payload network
`odom_frame`	`camera_init`	The robot's own odometry is HotDog's odometry: the driver publishes it as the LiDAR- inertial estimators of Dharma do (camera_init -> body on TF), dharma_localization bridges it into odom -> base_link (body is imu_link, the identity of base_link in the URDF).
`base_frame`	`body`	TF frame name
`publish_tf`	`true`	
`stand_up_gait`	`flat_agile`	Patrol in the agile flat gait: the protocol document recommends it for autonomy.
`max_vx`	`1.0`	m/s, above robot.limits (0.8): the control mux limits first
`max_vy`	`0.5`	
`max_wz`	`1.0`	
`angle_unit`	`auto`	set rad or deg once the driver logged which one this robot reports

11.3 hotdog · stack.yaml

the Dharma stack as tuned for HotDog: odometry only · `src/hotdog/config/stack.yaml`

odometry_monitor_node

Parameter	Default	Meaning
`odom_topic`	`/m20/odom`	the driver's odometry is the estimator
`imu_topic`	`""`	planar odometry: no tilt check against the IMU
`leg_odom_topic`	`""`	nothing independent to compare it with
`stale_sec`	`0.5`	the motion report comes at 10 Hz

localization_node

Parameter	Default	Meaning
`imu_frame`	`imu_link`	the driver's body frame in the URDF (identity of base_link)
`assume_start_origin`	`true`	the odometry start is the map origin: start at the dock
`leveling.enabled`	`false`	the odometry is planar already
`estimator.sigma_xy_per_sqrt_m`	`0.05`	No corrections without tags: the state stays LOCALIZED and sigma only reports the drift. Re-anchor at the dock before every patrol (localization/assume_origin).
`estimator.max_sigma_xy`	`100.0`	
`estimator.max_sigma_yaw_deg`	`180.0`	
`tag.auto_localize`	`false`	
`tag.auto_correct`	`false`	

lidar_sector_node

Parameter	Default	Meaning
`input.type`	`pointcloud2`	

Parameter	Default	Meaning
`input.pointcloud_topic`	`/lidar/points`	hotdog_base cloud_merger_node, front and rear Airy
`input.frame`	`base_link`	
`input.timeout_sec`	`0.5`	
`front.fov_deg`	`360.0`	the rear Airy sees behind: avoidance while backing up too
`front.range_max`	`8.0`	
`filter.level_source`	`none`	cloud_merger_node levels the cloud with the IMU
`filter.height_min`	`0.15`	m above the ground: gravel, kerbs and cable trench covers stay out
`filter.height_max`	`1.60`	
`blind.min_points`	`200`	

`path_follower_node`

Parameter	Default	Meaning
`controller`	`holonomic`	
`avoidance_mode`	`path_shift`	
`speed.cruise`	`0.5`	m/s
`inspection.reverse_distance_m`	`0.0`	
`hooks.inspection`	`[/inspection/inspect]`	inspection_node at every stop point; a missing service never holds the patrol. The stop waits dwell + inspection_timeout_sec at most: above inspection_node max_duration_sec.
`hooks.inspection_timeout_sec`	`180.0`	

control_mux_node

Parameter	Default	Meaning
use_health_limit	true	scale commands by health/status speed_scale
remote_sources	[handheld, dashboard]	control/remote/<source>/cmd_vel

health_monitor_node

Parameter	Default	Meaning
nodes.expected	[""]	"topic:min_hz[:max_age_sec]" The bringup adds every node it starts, and the M20 driver's when it starts it.
degradation.rules	["sensor:/lidar/points=0.5", "sensor:/m20/odom=0.5", "tf:*=0.5"]	
clock.enabled	true	the offset of this computer's clock to its time source (scripts/setup.sh install timesync)

11.4 hotdog_base · base.yaml

both Airy as one leveled cloud · src/hotdog_base/config/base.yaml

cloud_merger_node

Parameter	Default	Meaning
inputs	[/lidar/front/points, /lidar/rear/points]	rslidar_sdk publishes one cloud per Airy (config/rslidar.yaml); the first input triggers the merged cloud, which perception (lidar_sector_node) and health read.

Parameter	Default	Meaning
`input_rpy_deg`	`[0.0, 0.0, 0.0, 0.0, 0.0, 180.0]`	roll, pitch, yaw (deg) of each input on top of its URDF frame. The M20 model gives the rear LiDAR no rotation, but the rear Airy looks backwards: check both against their data (RViz, the robot in front of a wall) before the robot moves on its own.
`output_topic`	`/lidar/points`	
`require_all`	`true`	a stale side stops the output: perception goes BLOCKED, not half blind
`max_age_sec`	`0.3`	
`range_max`	`30.0`	m from base_link
`height_max`	`2.5`	m above the floor
`self_height`	`0.6`	m above base_link: the pod and the payload rails
`level_source`	`imu`	the M20's odometry is planar: level the cloud here, not from TF
`imu_topic`	`/m20/imu`	

11.5 hotdog_inspection · inspection.yaml

overheat analysis and partial discharge · `src/hotdog_inspection/config/inspection.yaml`

`inspection_node`

Parameter	Default	Meaning
`enabled`	`true`	live: false lets the stop points pass without inspection
`sites_dir`	`~/.hotdog/sites`	<site>/inspection.yaml: points (map frame) and their targets
`reports_dir`	`~/.hotdog/reports`	<day>/<time>_<point>/report.json + images, <day>/findings.csv
`match_radius_m`	`1.5`	a stop this close to a point inspects it (taught at the same spot)

Parameter	Default	Meaning
`max_duration_sec`	`150.0`	per point: keep below the follower's hooks.inspection_timeout_sec
`pod.namespace`	`/unipod`	unipod_driver
`pod.settle_sec`	`2.5`	gimbal and zoom (auto focus) before measuring
`thermal.ambient_c`	`30.0`	live: the air temperature at the site (ros2 param set ... thermal.ambient_c 33.0)
`thermal.width`	`640`	MT11 thermal pixels (1280 x 1024 with AI super-resolution on)
`thermal.height`	`512`	
`thermal.samples`	`3`	per region, the median is used
`thermal.refresh_shutter`	`true`	flat-field correction once per point
`thermal.absolute`	`{attention: 80.0, warning: 90.0, critical: 105.0}`	degC
`thermal.ambient`	`{attention: 11.0, warning: 21.0, critical: 41.0}`	degC over ambient_c
`thermal.similar`	`{attention: 2.0, warning: 4.0, critical: 16.0}`	degC over the other phases
`acoustic.enabled`	`true`	live
`acoustic.host`	`192.168.2.204`	factory address; this computer needs an address in 192.168.2.x
`acoustic.port`	`80`	
`acoustic.user`	`admin`	
`acoustic.duration_sec`	`5.0`	detection time when a target sets none

Parameter	Default	Meaning
`acoustic.corona`	`attention`	common on outdoor HV conductors, watch the trend
`acoustic.surface`	`warning`	tracking on insulators and terminations
`acoustic.floating`	`critical`	a floating (loose) part: arcing
`acoustic.intensity_warning`	`0.0`	a discharge this intense is at least a warning; 0: off
`acoustic.intensity_critical`	`0.0`	and critical; set both once the site's levels are known

11.6 hotdog_patrol · patrol.yaml

patrols, the watchdog, the schedule · `src/hotdog_patrol/config/patrol.yaml`

`patrol_node`

Parameter	Default	Meaning
`robot_ns`	`m20`	
`sites_dir`	`~/.hotdog/sites`	routes: <site>/routes/<route>.csv
`reports_dir`	`~/.hotdog/reports`	<day>/patrol_<time>.json
`battery_topic`	`/{robot_ns}/battery`	
`start.min_battery`	`0.3`	fraction of a full battery
`start.reanchor`	`true`	localization/assume_origin: a patrol starts at the dock (the map origin)
`start.stand_up_service`	`/{robot_ns}/stand_up`	m20_driver: stand, then RL control
`start.max_health`	`1`	start with health WARN, never with ERROR
`finish.dock_service`	`/{robot_ns}/charge_start`	m20_driver: the M20 goes back to its dock

Parameter	Default	Meaning
`abort.return_to_dock`	`true`	false: a watchdog return stops the robot where it is instead
`abort.min_battery`	`0.2`	back to the dock below this; keep above the follower's critical_battery
`abort.max_duration_min`	`120.0`	a patrol running longer goes back (0: off)
`abort.hold_timeout_sec`	`180.0`	blocked this long: back; off the route or not localized: stop
`abort.nav_timeout_sec`	`5.0`	navigation/status silent this long: the patrol fails
`abort.near_m`	`2.0`	the way back starts at the nearest point of the first pass this near
`abort.dock_clearance_m`	`1.0`	the way back ends this far from the route's start (the dock)
`schedule.enabled`	`false`	live: ros2 param set /patrol_node schedule.enabled true
`schedule.times`	`["06:00", "14:00", "22:00"]`	local time of this computer
`schedule.route`	`""`	the route of the scheduled patrols; empty: the route loaded last
`schedule.window_min`	`10.0`	a time missed by more than this (robot busy or off) is skipped

11.7 hotdog_reports · reports.yaml

report pages, retention, the report server · `src/hotdog_reports/config/reports.yaml`

`report_node`

Parameter	Default	Meaning
`reports_dir`	`~/.hotdog/reports`	inspection_node and patrol_node write here

Parameter	Default	Meaning
`title`	`HotDog substation patrol`	heading of every page
`robot_name`	`HotDog`	e.g. "HotDog 01 - GI Cawang"
`trend_days`	`30`	each finding is compared with the last round within this many days
`retention.days`	`365`	delete days older than this (0: keep everything)
`retention.min_free_gb`	`5.0`	then delete the oldest days while less is free (today never)
`retention.period_min`	`60.0`	min between two retention checks
`server.enabled`	`true`	http://<robot>:8088/ on the substation network, read-only
`server.host`	`0.0.0.0`	127.0.0.1: only this computer (open it through an SSH tunnel)
`server.port`	`8088`	TCP port of the report server

11.8 hotdog_alerts · alerts.yaml

Telegram and webhook alerts · `src/hotdog_alerts/config/alerts.yaml`

`alert_node`

Parameter	Default	Meaning
`enabled`	`true`	live
`language`	`en`	en | id (Bahasa Indonesia)
`robot_name`	`HotDog`	e.g. "HotDog 01 - GI Cawang"
`report_url`	`""`	e.g. http://10.21.31.50:8088/ : a link to the patrol page in every patrol alert
`telegram.chat_ids`	`[""]`	e.g. ["-1001234567890"] (a group) and ["123456789"] (a person)
`telegram.photos`	`true`	the thermal image or the sound map with each finding

Parameter	Default	Meaning
`webhook.url`	`""`	e.g. https://scada.example/api/hotdog : one JSON POST per alert
`findings.min_severity`	`warning`	attention | warning | critical
`findings.repeat_hours`	`12.0`	the same finding again only after this, unless it got worse
`patrol_done`	`true`	every finished patrol (failed, stopped and returned ones always)
`health_repeat_min`	`30.0`	min, health ERROR announced again only after this
`retry_sec`	`15.0`	the spool is sent every 15 s while the uplink is down
`max_age_hours`	`48.0`	older unsent alerts are dropped
`max_queued`	`500`	alerts kept waiting at most; the oldest are dropped