EZRA ROBOTICS

X30 Installation Guide

From nothing to the first patrol: the stack on the Jueying X30, the app on the Skydroid H12 and the NETRA command center

Version 1.0 • 2026-11-21

For the engineer who installs a robot and its command center: a Mac or a Linux laptop, a terminal and SSH. Every step is a command to copy; no ROS knowledge is needed to follow them.

About this document

For whom. For the engineer who installs a robot and its command center: a Mac or a Linux laptop, a terminal and SSH. Every step is a command to copy; no ROS knowledge is needed to follow them.

Operating the robot is in the X30 User Manual; what each setting does and how to tune it for a site in the X30 Tuning Guide. The robot's own hardware (battery, charging pile, power switch, the emergency stops) is described in Deep Robotics' Jueying X30 Pro User Manual, the cameras in the HIKMICRO and Hertzino manuals.

The X30 documents

Installation Guide	Getting the stack, the handheld app and NETRA onto a robot and a computer, from nothing to the first patrol.
User Manual	Operating the robot: safety, the Skydroid H12, routes, patrols, inspections, the recorder and NETRA.
Tuning Guide	What every setting does, the models behind them and how to tune a robot for a site; the full parameter reference.
Architecture	How the stack is built: hosts, nodes, topics, frames, the safety pipeline, the protocols to the handheld and to NETRA.
Site Acceptance Test	The handover tests, with pass criteria and sign-off.

All of them, and the same pages as a website with search: `https://darmawanrobotics.github.io/x30_ros/`.

Revisions

Version	Date	Changes
1.0	2026-11-21	First version: x30_ros 0.19, X30Controller 1.0, NETRA with the substation module

Contents

About this document . . . 2

1 What gets installed where . . . 5
- 1.1 The versions this guide installs . . . 6
- 1.2 The way through this guide . . . 6

2 Before you start . . . 7
- 2.1 What you need . . . 7
- 2.2 Addresses and passwords . . . 7
- 2.3 Time needed . . . 8

3 On the Mac, while it is online . . . 9
- 3.1 Tools . . . 9
- 3.2 The robot's sources in one archive . . . 9
- 3.3 The app . . . 9
- 3.4 NETRA . . . 9
- 3.5 The handheld's token . . . 11

4 The stack on the robot . . . 12
- 4.1 SSH keys . . . 12
- 4.2 Copy and unpack . . . 12
- 4.3 Dependencies . . . 12
- 4.4 Build . . . 13
- 4.5 Deep Robotics' autonomy off . . . 14
- 4.6 The motion host's reports . . . 14
- 4.7 The hosts and their clocks . . . 14

5 The services . . . 15
- 5.1 Install . . . 15
- 5.2 Settings . . . 15
- 5.3 Start, stop, watch . . . 16
- 5.4 What the watchdog does . . . 16
- 5.5 Run by hand . . . 17

6 The handheld . . . 18
- 6.1 Install the app . . . 18
- 6.2 First connection . . . 18
- 6.3 The H12's sticks . . . 18

7 NETRA on site . . . 20
- 7.1 Start NETRA . . . 20
- 7.2 Connect the robot . . . 20
- 7.3 What the robot sends and keeps . . . 20

8 A new robot . . . 22
- 8.1 Does the robot obey? . . . 22
- 8.2 The robot profile . . . 22

8.3 The first site and route . . . 23
9 Updates and backups . . . 24
9.1 A new version of the stack . . . 24
9.2 A new app . . . 24
9.3 A new NETRA . . . 24
9.4 What to keep . . . 25
10 When something does not work . . . 26
A Ports, files and folders . . . 27
A.1 Ports . . . 27
A.2 On the robot . . . 27

CHAPTER 1

What gets installed where

An X30 patrol system is four computers and a handheld. The robot carries three: the **motion host** walks the robot and is never changed; the **navigation host** on the robot's back runs the stack that drives it; the **perception host** inside the body runs the inspection cameras and records them. The operator's **Mac or laptop** runs NETRA, the command center, and the **Skydroid H12** runs the X30Controller app. Everything meets on the robot's own network: its Wi-Fi, or a cable to the Debug Ethernet port at the tail.

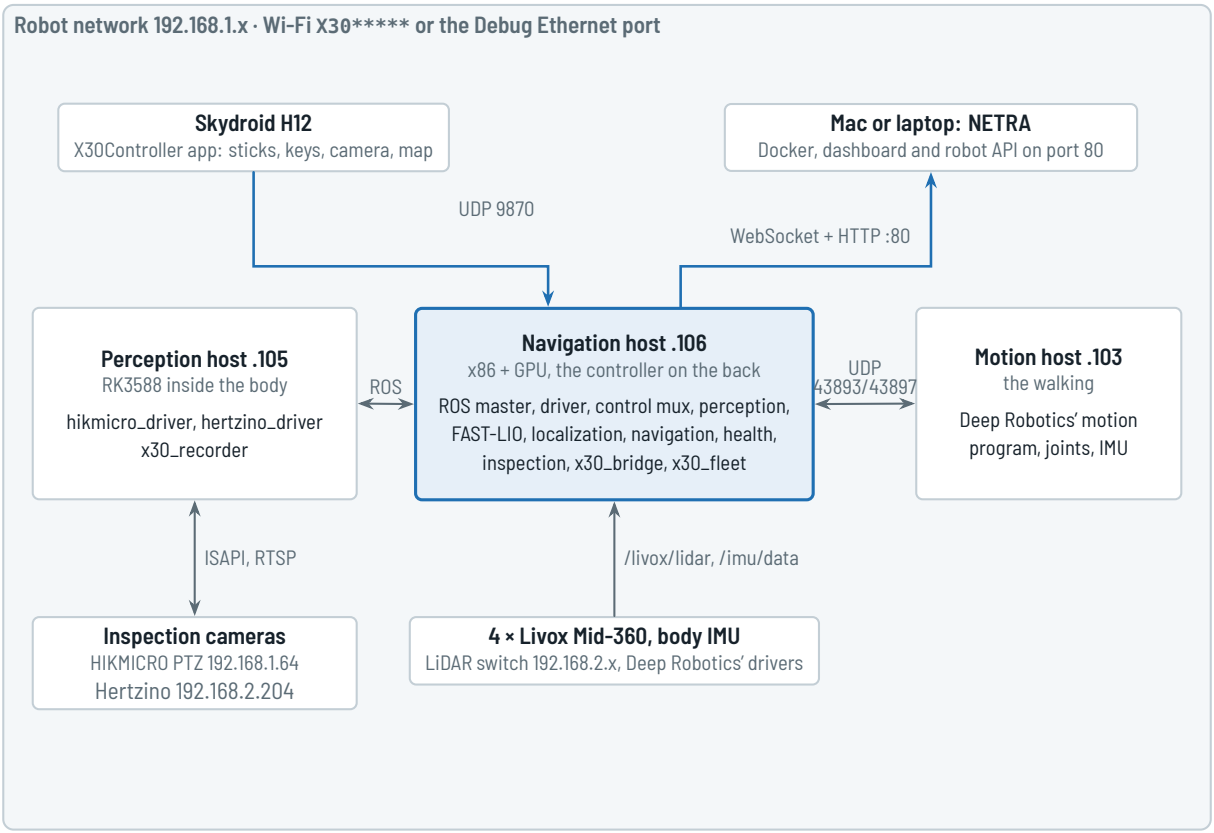


Figure 1.1: The computers of an X30 deployment and what talks to what.

Computer	Address	What you install
Navigation host	`192.168.1.106`	x30_ros (the whole workspace), the service `x30-navigation`; Deep Robotics' own autonomy turned off
Perception host	`192.168.1.105`	x30_ros (the camera drivers and the recorder only), the service `x30-perception`
Motion host	`192.168.1.103`	nothing: one line in its `network.toml`
Mac or laptop	the robot's network	NETRA (Docker), the tools to pack the sources and to install the app
Skydroid H12	the robot's Wi-Fi	the X30Controller app (APK)

1.1 The versions this guide installs

Part	Version	Where it comes from
x30_ros	0.19	`https://github.com/DarmawanRobotics/x30_ros`
X30Controller	1.0	`https://github.com/DarmawanRobotics/X30Controller/releases`
NETRA	with the sub-station module	`https://github.com/DarmawanWeb/Netra` (private)
Robot	Jueying X30 Pro	Deep Robotics; Ubuntu 20.04 and ROS Noetic on both hosts

1.2 The way through this guide

The robot has no internet. Everything that downloads is done first, on the Mac, while it is online; then the Mac joins the robot's Wi-Fi and carries it over.

1. **Before you start** (Chapter 2): the parts, the addresses and the passwords.
2. **On the Mac, online** (Chapter 3): pack the robot's sources, download the app, build NETRA and register the robot.
3. **On the robot** (Chapter 4): copy, build, turn Deep Robotics' autonomy off, introduce the hosts to each other.
4. **The services** (Chapter 5): the stack starts at boot under a watchdog; its settings.
5. **The handheld** (Chapter 6) and **NETRA** (Chapter 7).
6. **A new robot** (Chapter 8): the checks before it walks on its own, the first site and route.

> **Safety** The robot walks among energized equipment. Do the first bring-up in an open area, away from live parts, with one person at the handheld whose only job is to stop the robot. Keep the robot's hard emergency stop (at the tail) and the handheld's STOP within reach at every step.

CHAPTER 2

Before you start

2.1 What you need

Item	Notes
Jueying X30 Pro	charged, with its hard emergency stop working; the HIKMICRO PTZ camera and the Hertzino HZ-FA-371 mounted and powered
Skydroid H12	charged; or any Android phone or tablet (touch sticks)
Mac (or Linux laptop)	macOS on Apple Silicon recommended, 16 GB RAM, 30 GB free disk; Homebrew, Git with your GitHub SSH key, the `gh` CLI logged in
Network cable	optional: the robot's Debug Ethernet port, to keep the Mac online over Wi-Fi while it talks to the robot (Section 4.3)
USB-C cable	to install the app on the H12 with `adb`
Access	the GitHub repositories (x30_ros, X30Controller; NETRA is private); a NETRA license with the substation module (Chapter 7)

2.2 Addresses and passwords

What	Value	Notes
Robot Wi-Fi	`X30*****`	the serial number of the robot; password `12345678` (Deep Robotics' default)
SSH to every host	user `ysc`	password `'` (one single quote, Deep Robotics' default)
Navigation host	`192.168.1.106`	the stack, its ROS master
Perception host	`192.168.1.105`	the cameras, the recorder
Motion host	`192.168.1.103`	commands to UDP 43893, reports from it to UDP 43897
HIKMICRO PTZ camera	`192.168.1.64`	its admin password: `HIKMICRO_PASSWORD`
Hertzino acoustic camera	`192.168.2.204`	its password: `HERTZINO_PASSWORD`
H12 gateway	UDP `9870` on .106	the app's token: `X30_TELEOP_TOKEN`
NETRA	port `80` on the Mac	the robot's token: `X30_NETRA_TOKEN`

Secrets The camera passwords and the two tokens live only in the private settings files of the hosts (`/etc/x30/*.env`, readable by root and `ysc`). They are never launch arguments or ROS parameters: `roslaunch` prints every parameter, and anyone on the robot's network can read them from the master.

2.3 Time needed

About an hour on the Mac (most of it downloads: NETRA's images and its AI model), half an hour on the robot for the build, half an hour for the services, the app and NETRA, and an hour for the checks of a new robot.

CHAPTER 3

On the Mac, while it is online

Everything that needs the internet happens here, before the Mac joins the robot's Wi-Fi (it has no internet). At the end of this chapter the Mac holds the robot's sources in one archive, the app's APK and a NETRA that runs offline.

3.1 Tools

Docker Desktop, adb and GitHub's gh (skip what you have already):

```
brew install --cask docker android-platform-tools
brew install gh
gh auth login
```

Start Docker Desktop once and give it at least 4 GB of memory (Settings › Resources).

3.2 The robot's sources in one archive

`tools/pack_sources.sh` clones x30_ros, every repository of `x30.repos` with their submodules (FAST-LIO with its ikd-Tree, the Livox driver, the HIKMICRO and Hertzino drivers) and the Livox SDK, and packs them into one archive of about 5 MB, without Git history:

```
mkdir -p ~/x30 && cd ~/x30
git clone git@github.com:DarmawanRobotics/x30_ros.git
cd x30_ros
bash tools/pack_sources.sh ~/x30/x30_sources.tar.gz
```

A GitHub repository that HTTPS refuses, a private driver, comes over SSH with your GitHub key; nothing in your Git configuration changes. `src/SOURCES.txt` in the archive lists the commit of every repository.

3.3 The app

```
cd ~/x30
gh release download -R DarmawanRobotics/X30Controller \
  -p 'x30-controller-*.apk'
```

Every release carries Skydroid's SDK, so it reads the H12's own sticks; on any other Android device the app drives with touch sticks.

3.4 NETRA

NETRA builds its images and downloads its AI model once; afterwards it starts without internet.

```
cd ~/x30
git clone --recursive git@github.com:DarmawanWeb/Netra.git
cd Netra
scripts/local.sh setup
```

```
scripts/local.sh up
```

`setup` checks the Mac, writes `.env` with generated passwords and fetches Ollama and its model; `up` builds and starts everything (a few minutes the first time). Open `http://localhost` and log in as `admin` with the password `setup` printed (`NETRA_ADMIN_PASSWORD` in `.env`).

3.4.1 License

Without a license NETRA runs its core with one robot and drops the inspection uploads of the robot. The X30 needs the **substation** module. NETRA accepts only a license signed by a key built into it: `be/internal/license/keys/*.pub`. The image carries the license tool, so no Go is needed:

```
netra-license() {
  docker run --rm --entrypoint /netra-license -u "$(id -u)" \
    -v ~/netra-keys:/keys -v "$PWD/license":/out \
    ghcr.io/darmawanweb/netra:main "$@"
}
```

Once, where licenses are issued (skip it when `~/netra-keys/ezra.key` exists), the key pair:

```
mkdir -p ~/netra-keys
netra-license keygen -out /keys/ezra
```

NETRA trusts the public half once it is built in (commit `ezra.pub` to netra-fleet-be, so that every build does):

```
cp ~/netra-keys/ezra.pub be/internal/license/keys/
scripts/local.sh up
```

1. Read the installation ID in **Settings › License** (e.g. `NETRA-7KQ3-M9XD-2WPA`).
2. Issue the license (or in the License Manager):

```
netra-license issue -key /keys/ezra.key \
  -client "Gardu X30" -edition substation -tier pro \
  -robots 1 -days 365 \
  -installation NETRA-7KQ3-M9XD-2WPA \
  -out /out/netra.lic
docker compose restart api
```

The API reads `license/netra.lic` at its start.

> **The private key** `ezra.key` signs any license. Keep it offline with a backup, never in a repository, an image or a server.

The substation edition draws robots on a GPS basemap. The X30 has no GPS: it appears there through the site's reference points (**Site map**). For a drawing of the substation instead, issue the license with `-map floorplan`.

3.4.2 The robot

```
scripts/local.sh robot x30-01 "X30 01"
```

It prints `ROBOT_ID`, `CC_URL` and `CC_TOKEN`: keep the token, it becomes `X30_NETRA_TOKEN` on the robot (Chapter 7). A robot ID is 2 to 32 lower-case letters, digits, `-` and `_`.

3.5 The handheld's token

The H12 gateway accepts only a handheld that knows its shared secret. Make one and keep it with NETRA's:

```
openssl rand -hex 16
```

> **Tip** Before you leave: open `http://localhost` once with Wi-Fi off. Everything NETRA needs is then on the Mac.

CHAPTER 4

The stack on the robot

From here on the Mac is on the robot's Wi-Fi (`X30*****`, password `12345678`), or on a cable to the Debug Ethernet port at the tail.

4.1 SSH keys

Every host takes user `ysc` with password `'` (one single quote). Install your key once, so the commands below never ask again:

```
ssh-copy-id ysc@192.168.1.106
ssh-copy-id ysc@192.168.1.105
```

> **Go by the address, not the name** Deep Robotics names the navigation host (.106) `ysc-perception` and the perception host (.105) `ysc`. `hostname -I` tells which one a shell is on. The tools of this chapter take the role from that address and refuse what belongs to the other host.

4.2 Copy and unpack

```
cd ~/x30
scp x30_sources.tar.gz ysc@192.168.1.106:~/
scp x30_sources.tar.gz ysc@192.168.1.105:~/
```

On each host (`ssh ysc@192.168.1.106`, then the same on .105), into a clean workspace: the one before stays as `src.prev`, so a package x30_ros no longer has is never built again. One command a line, each copied whole:

```
mkdir -p ~/x30_ws && cd ~/x30_ws
rm -rf src.prev build devel
[ -d src ] && mv src src.prev
tar -xzf ~/x30_sources.tar.gz
```

The workspace is `~/x30_ws` with `src/` (x30_ros and its sources) and `Livox-SDK/`.

4.3 Dependencies

`check_deps.sh` reads only what is installed (`dpkg`, the workspace, `/usr/local/lib`) and prints one `apt install` line for what is missing:

```
bash ~/x30_ws/src/x30_ros/tools/check_deps.sh
```

On the perception host it checks what that host builds, the cameras and the recorder, by its address (`--perception` for it on any other computer).

The robot borrows the Mac's internet through SSH for the missing packages. The Mac must then be on the robot *and* online: a cable to the Debug Ethernet port and Wi-Fi to the internet, or the robot's Wi-Fi and a phone on USB tethering. On the Mac, kept open (a SOCKS proxy back to it):

```
ssh -R 1080 ysc@192.168.1.106
```

In that session, on the robot:

```
PX=socks5h://127.0.0.1:1080
P="-o Acquire::http::Proxy=$PX"
P="$P -o Acquire::https::Proxy=$PX"
sudo apt $P update
sudo apt $P install -y <the packages check_deps.sh printed>
```

Where SSH forwarding is refused, the robot names the files it needs and the Mac fetches them. On the robot:

```
sudo apt-get install --print-uris -qq <packages> |
  cut -d"'" -f2 > ~/urls.txt
```

On the Mac, online:

```
scp ysc@192.168.1.106:~/urls.txt .
mkdir -p debs && cd debs
xargs -n1 curl -fsSLO < ../urls.txt
cd .. && scp -r debs ysc@192.168.1.106:~/
```

Back on the robot:

```
sudo dpkg -i ~/debs/*.deb
```

> **No upgrades on the robot** Never `apt upgrade` there: Deep Robotics' software is built against the packages it has. A robot that gets internet also starts Ubuntu's own upgrades, which hold apt's lock (`Could not get lock`, `unattended-upgr`): let them finish or take the internet away, and turn them off once with the two lines below. `EXPKEYSIG` from `apt update` is the expired key of ROS's repository: a warning, the stack's packages are installed already.

```
sudo systemctl disable --now apt-daily.timer
sudo systemctl disable --now apt-daily-upgrade.timer
```

4.4 Build

Navigation host (.106), once: the Livox SDK, which FAST-LIO's Livox driver links.

```
cd ~/x30_ws/Livox-SDK/build
cmake .. && make -j4
sudo make install
```

Then **on each host**:

```
cd ~/x30_ws/src/x30_ros/tools
bash x30_host.sh build
```

`x30_host.sh build` is `catkin_make` as the robot needs it: ROS sourced, Release, no tests, at most one compile job per 2 GB of memory, and on the perception host only the cameras, the recorder and what launches them. Its arguments go to `catkin_make` (`--pkg x30_navigation`).

> **Note** A `pip install --user` that put a setuptools in `~/.local` stops catkin (`module 'importlib_metadata' has no attribute 'EntryPoints'`): the build leaves `~/.local` out (`PYTHONNOUSERSITE=1`) without touching it. Deep Robotics' catkin on the perception host includes the workspace's `src/config.cmake` (`include could not find load file: config.cmake`): the build makes an empty one there.

4.5 Deep Robotics' autonomy off

The navigation host also runs Deep Robotics' localization, navigation, mapping, URDF publisher and its bridge to the robot. Next to the stack they only take CPU and get in its way (`/initialpose`, `/map`, TF, the UDP port of the motion host's reports). Turn them off, now and at boot, on the navigation host only (the perception host and the motion host refuse it: what Deep Robotics runs there stays):

```
cd ~/x30_ws/src/x30_ros/tools
bash dr_services.sh off
bash dr_services.sh status
```

`status` then says `udp 43897 ... free`.

The sensor drivers (`/livox/lidar`, `/imu/data`), the ROS master and the robot server stay. The motion host and the perception host are never touched. `dr_services.sh on` puts everything back as it was.

4.6 The motion host's reports

The motion host sends its 200 Hz reports only to the addresses in its `network.toml` (Deep Robotics' Application Manual, 4.3.2). Add the navigation host with port 43897 and restart the motion program; otherwise the robot takes commands but reports nothing, and the driver's state says `no reports from 192.168.1.103 yet (network.toml on the motion host?)`.

4.7 The hosts and their clocks

Deep Robotics' drivers on the navigation host register under the name `ysc-perception`. On the perception host, make that name resolve:

```
echo '192.168.1.106 ysc-perception' |
  sudo tee -a /etc/hosts
```

Without internet the two clocks drift apart; nothing in the stack depends on it, but logs and reports of the two hosts then disagree. Let the navigation host be the perception host's clock with `chrony` (`allow 192.168.1.0/24` and `local stratum 10` on .106, `server 192.168.1.106 iburst` on .105), or once after each boot on .105:

```
sudo date -s "$(ssh ysc@192.168.1.106 date -Ins)"
```

CHAPTER 5

The services

Each host runs its part of the stack as a systemd service under a watchdog: it starts at boot, restarts when something it depends on goes away, and is never given up on.

5.1 Install

The same on each host:

```
cd ~/x30_ws/src/x30_ros/tools
sudo bash x30_host.sh install
```

The role comes from the host's address: `navigation` on .106, `perception` on .105. A role named for the other host is refused, and a service an older x30_host.sh put on the wrong host is removed with its settings. This writes `/etc/systemd/system/x30-<role>.service` and its settings file `/etc/x30/<role>.env` (mode 600: root and `ysc` only), and enables the service. Run it with the robot up: the navigation host's service is ordered after Deep Robotics' services and the one running the ROS master, as it finds them. At the end it prints the next commands: the settings to edit, then the service's start and its log.

> **An existing settings file is kept** `install` never overwrites `/etc/x30/<role>.env`. On a robot installed before version 0.19, add the line `X30_NETRA_TOKEN=` to `/etc/x30/navigation.env` yourself.

5.2 Settings

`/etc/x30/navigation.env` (shell words: a quoted value stays one argument; NETRA's arguments join `X30_LAUNCH_ARGS` in Chapter 7):

```
X30_LAUNCH_ARGS="site:=gardu controller:=neupan"
# only for controller:=neupan
X30_PYTHON=/home/ysc/x30_ws/neupan/bin/python
X30_TELEOP_TOKEN=<the handheld's token>
X30_NETRA_TOKEN=<CC_TOKEN from NETRA>
# each node's JSON lines log
# X30_LOG_DIR=/home/ysc/x30/logs
# ROS's own logs, pruned at every start (MB)
# X30_ROS_LOG_MB=2048
```

`/etc/x30/perception.env`:

```
# e.g. hikmicro_host:=192.168.1.64 hertzino_host:=192.168.2.204
X30_LAUNCH_ARGS=""
HIKMICRO_PASSWORD=<thermal camera admin password>
HERTZINO_PASSWORD=<acoustic camera password>
```

> **A comment goes on a line of its own** systemd reads no comment after a value: whatever follows a value on its line becomes part of it, the comment's words in a token, a password or the launch arguments.

5.2.1 Launch arguments of the navigation host

Argument	Default	What it does
`site`	last run's	the site at start (routes, tags, inspection points), created when missing
`sites_dir`	`~/x30/map`	where the sites are
`controller`	`holonomic`	the path controller; `neupan` needs its venv and `X30_PYTHON`
`health`, `inspection`	`true`	the health monitor; the inspection at the stop points
`bridge`	`true`	the H12 gateway on UDP 9870
`netra_url`, `netra_robot_id`	empty	NETRA's address and this robot's ID; the uplink runs when `netra_url` is set
`reports_dir`	`~/x30/reports`	the inspection reports
`lidar_frame`, `lidar_pose`	`livox_frame`, 0s	the frame of `/livox/lidar` and its mount (Section 8.2)
`odometry_method`	`fast_lio`	`external` when another estimator publishes `/Odometry`
`blackbox`, `blackbox_dir`	`true`, `~/x30/blackbox`	the rolling rosbag of the last 3 h

5.3 Start, stop, watch

The navigation host first: the perception host uses its master. On .106:

```
sudo systemctl restart x30-navigation
```

Then on .105:

```
sudo systemctl restart x30-perception
```

The watchdog's status line, the log and the check, here on .106 (`x30-perception` on .105):

```
systemctl status x30-navigation
journalctl -u x30-navigation -f
bash ~/x30_ws/src/x30_ros/tools/x30_host.sh check
```

`x30_host.sh check` tells this host's role and address, whether the other host answers, the ROS master and every registered node from here, the two clocks, and on the navigation host that Deep Robotics' autonomy is off, on the perception host the cameras' ports.

5.4 What the watchdog does

The service's main process starts the host's launch file, tells systemd it is ready once the master and the critical nodes answer (on the navigation host: the driver, the control mux, the LiDAR scans and the path follower), then checks them every 5 s. The service restarts when:

- the master restarted (its nodes are lost with it; the perception host follows this way);
- the master or a critical node stays away longer than 45 s;
- the start takes longer than 4 min, or roslaunch exits;
- the watchdog itself hangs (systemd's `WatchdogSec=60`).

Meanwhile the robot stands: the control mux stops a command that is not renewed, and the motion host stops taking commands without the driver's heartbeat. On the navigation host the watchdog also turns off, every 30 s, whatever of Deep Robotics' autonomy one of its services started again late, and names it in the journal.

5.5 Run by hand

For a test without the services (stop them first), on each host this shell's ROS network:

```
eval "$(bash ~/x30_ws/src/x30_ros/tools/x30_host.sh env)"
```

Then on .106:

```
roslaunch x30_bringup navigation_host.launch site:=gardu
```

And on .105:

```
roslaunch --wait x30_bringup perception_host.launch
```

CHAPTER 6

The handheld

6.1 Install the app

With a cable (USB debugging on: **Settings › About**, tap **Build number** seven times, then **Developer options › USB debugging**):

```
adb devices
adb install -r ~/x30/x30-controller-*.apk
```

`adb devices` lists the H12 before the install.

Without a cable: on a network with internet, open `https://github.com/DarmawanRobotics/X30Controller/releases/latest` in the H12's browser, download `x30-controller-<version>.apk`, open it and allow **Install unknown apps** for the browser.

> **Note** Until the repository has a release key (README, "Releases"), every APK is signed by CI's debug key: when a new version refuses to install over the old one, uninstall the old app first.

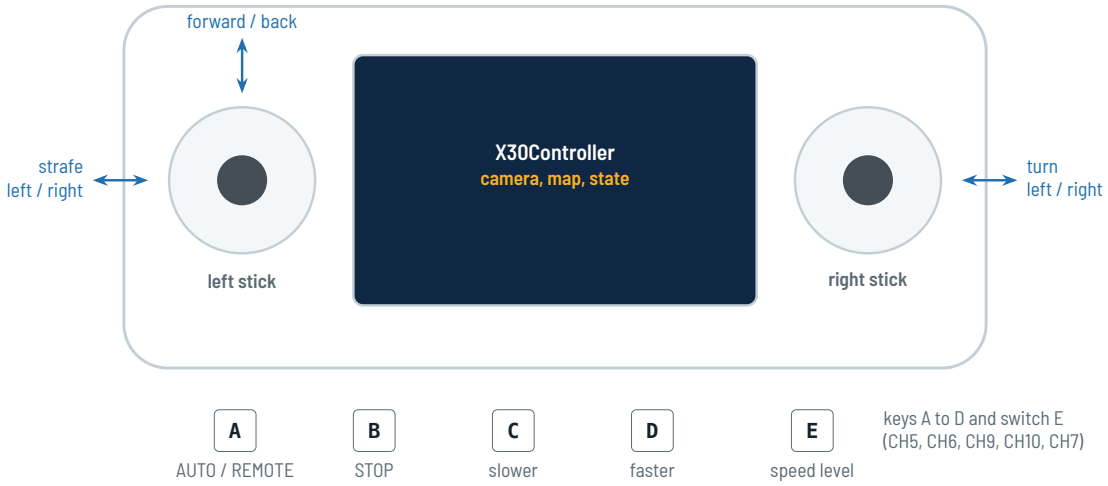
6.2 First connection

1. Join the H12 to the robot's Wi-Fi (`X30*****`, `12345678`).
2. Open X30Controller. On the start screen tap the version number three times.
3. Robot: `192.168.1.106`, port `9870`; token: the value of `X30_TELEOP_TOKEN` in `/etc/x30/navigation.env`.
4. The drive screen shows the robot's battery, mode, localization and the camera.

6.3 The H12's sticks

The H12 sends its sticks over its own radio, not as an Android gamepad; the app reads them through Skydroid's SDK. In **Settings (gear) › Skydroid H12 sticks** the link and every channel show live:

- pick the channels of forward, strafe and turn (default 2, 1, 3: the left stick drives and strafes, the right stick turns) and tick **reverse** for one that goes the wrong way;
- a released stick must read about 1500. A stick without a centring spring (a throttle stick) must never drive: map forward to a stick that centres by itself, such as CH4.



On any other Android phone or tablet the app drives with touch sticks on the screen.

CHAPTER 7

NETRA on site

At the site the Mac joins the robot's Wi-Fi and runs NETRA without internet; the robot connects to it directly.

7.1 Start NETRA

```
cd ~/x30/Netra
scripts/local.sh start
scripts/local.sh awake
scripts/local.sh lan
```

`start` runs the images built before (no internet needed), `awake` keeps the Mac awake for schedules and the robot, `lan` prints the Mac's address on each network, e.g. `http://192.168.1.50`. Reserve that address for the Mac in the robot's router, or check it after each join: the robot uses it.

7.2 Connect the robot

On the navigation host, check that NETRA answers, then give the robot its address, its ID and its token:

```
curl http://192.168.1.50/healthz
sudo nano /etc/x30/navigation.env
```

NETRA answers `"status":"ok"`. In the settings:

```
X30_LAUNCH_ARGS="site:=gardu netra_url:=http://192.168.1.50 netra_robot_id:=x30-01"
X30_NETRA_TOKEN=<CC_TOKEN>
```

```
sudo systemctl restart x30-navigation
journalctl -u x30-navigation -f | grep -i netra
```

`command center` in the log: connected.

Within seconds the robot is online on the dashboard: battery, mode, the navigation state, the recorder and the inspection in **Live control**. `site:=` must be there: NETRA's routes and their inspection plans go into the active site (`~/x30/map/<site>/netra/`).

7.3 What the robot sends and keeps

- **Telemetry** every second; patrol and stop events, alerts and the answers to commands, all over one WebSocket that comes back by itself.
- **Uploads**: every taught route and every inspection of NETRA's points wait in a spool on the robot (`~/x30/spool`) until NETRA has them, across restarts and while NETRA is away.
- **Clock**: the robot has no internet clock; NETRA measures the offset and puts every measurement on its own clock.

Install the license first Without the substation module NETRA accepts an inspection upload but drops it, and the robot does not send it again. The reports stay on the robot (`~/x30/reports`), but NETRA will not have them.

CHAPTER 8

A new robot

Before the stack walks a robot on its own, check that the robot does what the stack tells it, and measure what the stack assumes about it. In an open area, one person at the handheld whose only job is to stop the robot.

8.1 Does the robot obey?

Stop the service (`sudo systemctl stop x30-navigation`) and start only the driver:

```
eval "$(bash ~/x30_ws/src/x30_ros/tools/x30_host.sh env navigation)"
roslaunch x30_bringup bringup.launch
```

	Do	Expect
1	`rostopic hz /x30/odom`	about 50 Hz; `/x30/state` says `connected: True`. If not: `network.toml` (Section 4.6)
2	`rostopic echo -n 1 /x30/state`	`rcs_layout` and `motion_layout` read `natural`
3	`rosservice call /x30/stand_up`	ends in `torque_standing`; `/x30/start_stepping` in `stepping`
4	`rostopic pub -r 20 /x30/cmd_vel geometry_msgs/Twist '{linear: {x: 0.2}}'`	the robot walks forward, `control_mode` turns `non_manual`; stop the publisher: it stops within 0.5 s
5	the same with `angular: {z: 0.3}`	it turns **left** (counter-clockwise seen from above); else `invert_wz: true`
6	the same with `linear: {y: 0.1}`	it steps **left**; else `invert_vy: true`
7	walking, switch to manual on Deep Robotics' handheld	the robot follows that handheld; `velocity` reads `takeover (manual mode)` until it is handed back
8	`rostopic echo -n 1 /x30/battery`	voltage and current as Deep Robotics' app shows them
9	`roslaunch x30_bringup bringup.launch rviz:=true`	the legs in RViz follow the real ones

A robot that disagrees with step 5 or 6 gets its own copy of `x30_driver/config/driver_params.yaml` with the inversion, passed as `params_file:=`. The Tuning Guide explains every driver setting.

8.2 The robot profile

`x30_bringup/config/robot.yaml` is the X30 for every node: its footprint, its height, its limits and its frames. Three things are particular to each robot and are measured, not taken from the drawing:

1. **`base_height`**: `base_link` (the body centre) above the floor while it stands. The scans measure their height band from the floor `base_height` below it; 0.48 m is the model's.

2. **The LiDAR mount**: `rostopic echo -n 1 --noarr /livox/lidar/header` gives the frame the robot's driver publishes in. Set `lidar_frame` and `lidar_pose` so that a wall 1 m ahead shows at 1 m in `rostopic echo -n 1 /perception/safety_scan`.
3. **Braking**: `limits/max_decel`, `max_yaw_accel` and `reaction_time`, worst case (floor, payload, low battery). The protective field is sized from them; the Tuning Guide shows how to measure them.

8.3 The first site and route

With the services running again (`sudo systemctl start x30-navigation`):

1. Put the robot on its start spot (the dock), facing the same way every time: the start of the odometry is the site's origin, and one degree off there is 1.7 cm per metre on every route.
2. `rostopic echo -n 1 /localization/status`: `LOCALIZED`, source `origin`.
3. Teach a short route with the H12 (User Manual): REMOTE, **Record**, walk, **Point** at a stop, **Finish**.
4. Walk back to the start, pick the route in **Routes**, AUTO, **Start**. The robot repeats it and stops at the stop point.
5. In NETRA the route appears under **Routes**, and the robot's patrol state on the dashboard.

Then the acceptance tests (X30 Site Acceptance Test) before the robot patrols for real.

CHAPTER 9

Updates and backups

9.1 A new version of the stack

On the Mac, online:

```
cd ~/x30/x30_ros
git pull
bash tools/pack_sources.sh ~/x30/x30_sources.tar.gz
```

At the robot, to both hosts:

```
cd ~/x30
scp x30_sources.tar.gz ysc@192.168.1.106:~/
scp x30_sources.tar.gz ysc@192.168.1.105:~/
```

On each host a clean workspace and its build (Section 4.4):

```
cd ~/x30_ws
rm -rf src.prev build devel
mv src src.prev
tar -xzf ~/x30_sources.tar.gz
bash src/x30_ros/tools/check_deps.sh
bash src/x30_ros/tools/x30_host.sh build
```

The build ends with the host's restart: `sudo systemctl restart x30-navigation` on .106, `sudo systemctl restart x30-perception` on .105; the services stay installed. A new version may need a new package (`check_deps.sh` says so); the `CHANGELOG.md` of x30_ros says what changed and whether a settings file needs a new line.

9.2 A new app

```
cd ~/x30
gh release download -R DarmawanRobotics/X30Controller \
  -p 'x30-controller-*.apk' --clobber
adb install -r x30-controller-*.apk
```

9.3 A new NETRA

Online:

```
cd ~/x30/Netra
git pull
git submodule update --init --recursive
scripts/local.sh up
```

up rebuilds what changed.

9.4 What to keep

Where	What
`~/x30/map/<site>/` (.106)	the site: routes, tags, inspection points. Small, precious: copy it after every change
`~/x30/reports/` (.106)	the inspection reports; days go when the disk is short
`~/x30/recordings/` (.105)	the camera recordings; the oldest go first
`/etc/x30/*.env`	the settings and secrets of each host
NETRA	`scripts/local.sh export`: database, photos and plans in one file

```
rsync -a ysc@192.168.1.106:x30/map/ ~/x30/backup/map/
rsync -a ysc@192.168.1.106:x30/reports/ ~/x30/backup/reports/
```

CHAPTER 10

When something does not work

What you see	What to do
`no reports from 192.168.1.103 yet`	the navigation host is not in the motion host's `network.toml`, or another process holds UDP 43897: `dr_services.sh status`
`catkin_make: importlib_metadata has no attribute 'EntryPoints'`	build with `x30_host.sh build`: it leaves `~/.local` out (`PYTHONNOUSERSITE=1`)
The first build fails: `fast_lio/Pose6D.h: No such file or directory`	FAST-LIO compiled before its messages: build again (they are there the second time); an archive of `pack_sources.sh` from 0.19 on makes it wait
`catkin_make: include could not find load file: config.cmake`	Deep Robotics' catkin on the perception host includes `src/config.cmake`: `x30_host.sh build` from 0.19 on makes an empty one (by hand: `touch ~/x30_ws/src/config.cmake`)
The service fails at once: `ROS_DISTRO: unbound variable`	an x30_ros before 0.19 on the robot: pack the current one, unpack it clean and build (Section 4.4)
`this is the perception host (192.168.1.105), this is the navigation host`	a command for the other host: `hostname -I` tells which one this shell is on (Deep Robotics names .106 `ysc-perception`); left without a role, `x30_host.sh` takes this host's
`apt: Could not get lock, unattended-upgr`	Ubuntu upgrades itself while the robot has internet: let it finish or take the internet away, never `apt upgrade`, and turn the timers off (Section 4.3)
`apt: Unable to locate package`	the robot has no package lists: use the SSH proxy, not the file list
The service restarts again and again	`journalctl -u x30-navigation`: the watchdog names the node that stayed away; `x30_host.sh check`
Nodes of the other host unreachable	`x30_host.sh check` lists them; a name that does not resolve goes into `/etc/hosts`
The H12 does not connect	H12 on the robot's Wi-Fi; host `192.168.1.106:9870`; the token as in `X30_TELEOP_TOKEN`; the service running
The H12's sticks do nothing	an APK from a release (it carries Skydroid's SDK); **Settings › Skydroid H12 sticks** shows the link; mode REMOTE (key A)
`local.sh up: "/fe/package-lock.json": not found`	`be/` and `fe/` of NETRA are empty (cloned without `--recursive`): `git submodule update --init --recursive` in `Netra`, then `scripts/local.sh up` again
`adb devices` lists nothing	USB debugging on the H12 (**Settings › About** › tap **Build number** 7 times › **Developer options**); a data cable, not a charging one; allow the computer on the H12's screen. Or copy the APK to the H12 and open it in the file manager
The robot offline in NETRA	from .106: `curl http://<Mac>/healthz`; another network or the macOS firewall (allow Docker); `no NETRA token` in the journal: the token is empty; `401` in `scripts/local.sh logs api`: the token is wrong
Inspections never reach NETRA	the license's substation module; NETRA's inspection points on the stops of the route patrolled; `~/x30/spool` holds what is still waiting
Recorder "No status" in NETRA	the perception host's service: `systemctl status x30-perception` on .105

APPENDIX A

Ports, files and folders

A.1 Ports

Where	Port	What
Motion host .103	UDP 43893	commands from the driver
Navigation host .106	UDP 43897	the motion host's reports
Navigation host .106	TCP 11311	the ROS master (both hosts)
Navigation host .106	UDP 9870	the H12 gateway
Perception host .105	UDP 43899, 3333, 60000	elevation map, charging, its odometry
HIKMICRO .1.64	TCP 80, 554	ISAPI, RTSP
Hertzino .2.204	TCP 80, 443	its API
Mac	TCP 80	NETRA: dashboard and the robot API

A.2 On the robot

Path	Holds
`~/x30_ws/`	the workspace: `src/`, `Livox-SDK/`, the build
`/etc/x30/navigation.env`, `perception.env`	the services' settings and secrets
`~/x30/map/<site>/`	`tags.json`, `inspection.yaml`, `routes/*.csv`, `netra/` (NETRA's routes and plan)
`~/x30/reports/<date>/`	one folder per inspected point, `findings.csv` per day
`~/x30/recordings/<date>/` (.105)	one folder per recorder session
`~/x30/spool/`	uploads waiting for NETRA
`~/x30/blackbox/`	the last 3 h as rosbag splits
`~/x30/logs/<node>.jsonl`	each node's log, rotated
`~/.x30/trip.json`	the trip meter