Nine Fives Programmable SP4T Switch

Reference Manual

Nine Fives

None

Table of contents

1. Introduction 3
1.1 Device Variants 3
1.2 About This Manual 3
2. Getting Started 4
2.1 Physical Connections 4
2.2 Available Ports 4
2.3 Accessing the Web UI 4
2.4 Connecting via SCPI 4
2.5 Using the REST API 5
2.6 Default Network Configuration 5
3. Touch UI 6
3.1 Touch UI - SP4T 6
4. Web UI 8
4.1 Web UI Overview 8
4.2 System Information 10
4.3 Network Configuration 12
4.4 Firmware Update 15
4.5 SP4T Control 19
5. REST API 22
5.1 REST API Overview 22
5.2 System Endpoints 23
5.3 Network Endpoints 25
5.4 Firmware Endpoints 28
5.5 SP4T Endpoints 32
6. SCPI API 34
6.1 SCPI Overview 34
6.2 Common Commands (IEEE 488.2) 37
6.3 System Commands 40
6.4 Network Commands 42
6.5 SP4T Switch Commands 48

1. Introduction

The Nine Fives Programmable SPDT Switch, Programmable SP4T Switch, and Programmable Attenuator are network-controlled RF instruments. Each device provides three communication interfaces for monitoring and control:

- **Web UI** (Port 80) — Browser-based graphical interface for configuration and control
- **REST API** (Port 80) — JSON API for integration with custom applications and automation scripts
- **SCPI over TCP** (Port 5025) — Industry-standard text protocol for test and measurement applications
- **SCPI over HiSLIP** (Port 4880) — High-Speed LAN Instrument Protocol (IVI-6.1) for VISA-compatible instrument communication

All interfaces provide equivalent control over the device. Choose the interface that best fits your application.

1.1 Device Variants

Nine Fives produces the following device variants:

VARIANT	DESCRIPTION
Programmable SPDT Switch	RF switch with two throws (RF0 / RF1)
Programmable SP4T Switch	RF switch with four throws (RF0–RF3)
Programmable Attenuator	Variable RF attenuator (0–95.25 dB in 0.25 dB steps)

See the SPDT, SP4T, or attenuator product specifications for RF performance, power limits, environmental ratings, and connector details.

All variants share common system, network, and firmware management capabilities. The variant-specific commands and endpoints are documented in their respective sections.

1.2 About This Manual

This manual covers all available commands and endpoints for your device variant. Each section includes complete syntax, parameters, examples, and error codes.

2. Getting Started

2.1 Physical Connections

The device has two network interfaces:

INTERFACE	CONNECTOR	DEFAULT IP	DESCRIPTION
Ethernet	RJ-45	Assigned by DHCP (fallback: 192.168.0.95)	Primary network connection
USB-C	USB-C port	192.168.42.95	Direct connection to a host computer

Connect to the device using either interface. The USB-C connection is useful for initial setup or when no Ethernet network is available.

2.2 Available Ports

PORT	PROTOCOL	DESCRIPTION
80	HTTP	Web UI and REST API
5025	TCP	SCPI raw socket
4880	TCP	SCPI over HiSLIP (IVI-6.1)

2.3 Accessing the Web UI

Open a web browser and navigate to the device's IP address:

```
http://192.168.0.95
```

Or, if connected via USB-C:

```
http://192.168.42.95
```

The web UI provides a graphical interface for all device functions including system monitoring, network configuration, firmware updates, and device control.

2.4 Connecting via SCPI

2.4.1 Raw TCP

Use any TCP client to connect to port 5025:

```
telnet <device-ip> 5025
```

Send a command to verify the connection:

```
*IDN?
```

The device responds with its identification string.

2.4.2 HiSLIP (VISA)

Use a VISA library (such as PyVISA) with the following resource string:

```
TCPIP::<device-ip>::hislip0::INSTR
```

Example using Python:

```
import pyvisa
rm = pyvisa.ResourceManager()
inst = rm.open_resource("TCPIP::192.168.0.95::hislip0::INSTR")
print(inst.query("*IDN?"))
```

2.5 Using the REST API

The REST API accepts JSON requests on port 80. Example using `curl`:

```
curl http://192.168.0.95/api/system/status
```

See the REST API section for complete endpoint documentation.

2.6 Default Network Configuration

SETTING	ETHERNET	USB-C
Mode	DHCP Client with Static Fallback	DHCP Host
IP Address	192.168.0.95 (fallback)	192.168.42.95
Subnet Mask	255.255.0.0	255.255.255.0
DHCP Timeout	30 seconds	—

In the default Ethernet mode (`DHCP Client with Static Fallback`), the device first attempts to obtain an IP address from a DHCP server. If no DHCP server responds within 30 seconds, it falls back to the configured static IP address.

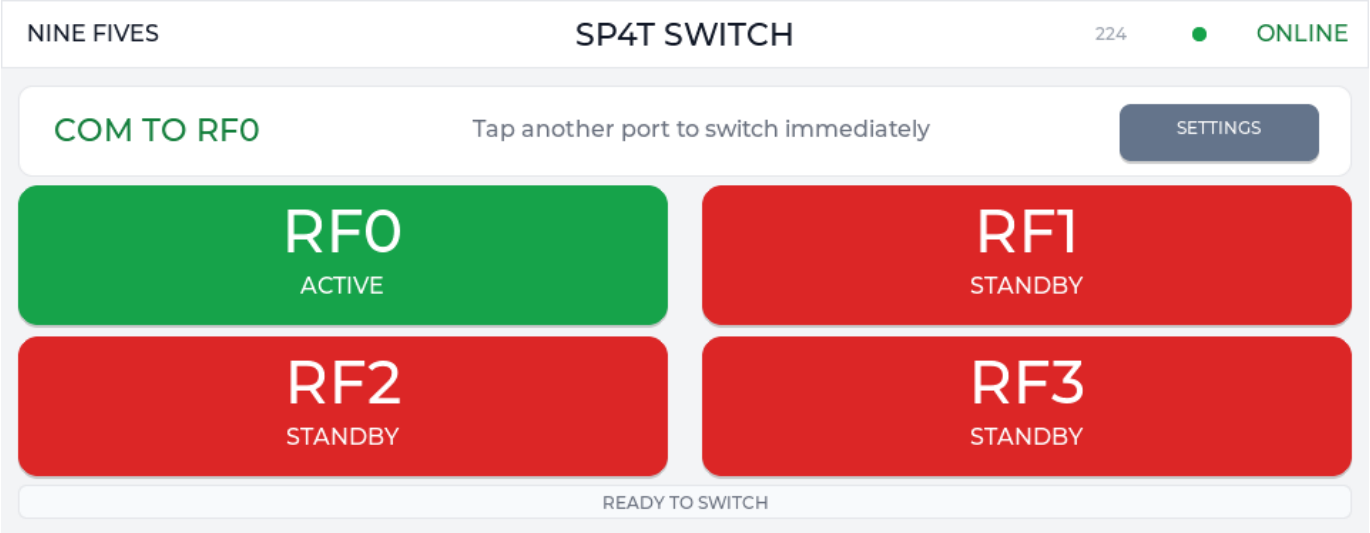
The USB-C interface runs a DHCP server by default, automatically assigning an IP address to the connected host computer.

3. Touch UI

3.1 Touch UI - SP4T

The SP4T touch UI routes a common port (`COM`) to one of four throws (`RF0`..`RF3`) from the front-panel touchscreen. It keeps the same chrome as the SPDT screen — top bar, route summary, and status footer — and replaces the two full-height port cards with a 2x2 grid so all four throws stay reachable on the 960x376 display.

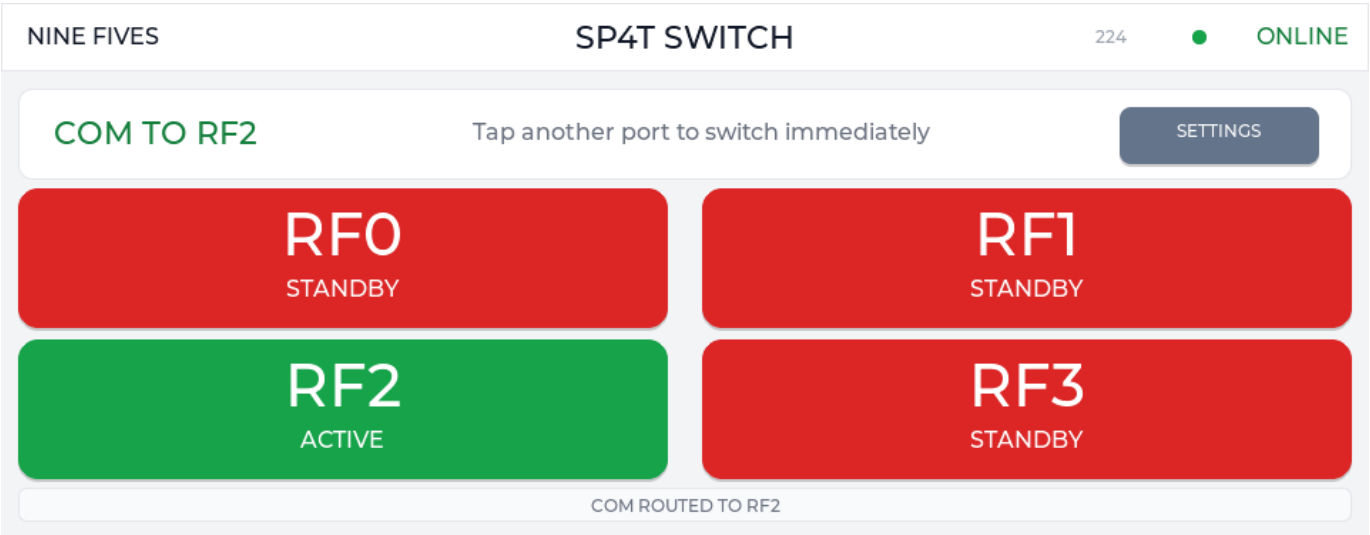
3.1.1 Main Screen



The active throw's card is green and reads **ACTIVE**; the other three are red and read **STANDBY**. The route summary at the top left reads **COM TO RF0**. Only one throw is closed at a time.

Each card is 456x112 px, well above the ~44 px minimum touch target, so the denser grid does not cost tap reliability.

3.1.2 Selecting a Throw



Tapping a standby card immediately requests a route change through the local runtime API — there is no confirm step. The request goes to `/api/sp4t` on SP4T hardware, so all four throws route; the card turns green optimistically while the request is in flight and reverts if the runtime rejects it.

3.1.3 Network Settings

The **Settings** control opens the shared network overview, which is identical to the SPDT variant. See Network Configuration and the SPDT touch UI page for those screens.

3.1.4 Controls

CONTROL	DESCRIPTION
RF0	Routes COM to RF0.
RF1	Routes COM to RF1.
RF2	Routes COM to RF2.
RF3	Routes COM to RF3.
Settings	Opens the shared network configuration screens.
Status bar	Shows the latest runtime status or switching message.

3.1.5 Hardware Type Gating

The touch UI bases its controls-enabled rule on the canonical hardware type (`hwtype`) shared with the runtime, preferring the `hardware_type` field of `/api/system/status`. `SPDT`, `SP4T`, and `ATTEN` render live controls; `unset` and `unknown` hardware disable the tiles and show a **HARDWARE TYPE UNSET/UNKNOWN** status instead of POSTing to a backend with no RF route.

3.1.6 Reproducing These Renders

The screens above are captured from the LVGL X11 simulator against an x86 mock runtime:

```
# Run the simulator interactively
make run-touch-ui-sim-sp4t

# Recapture the screenshots on this page
python3 docs/screenshots/capture_touch_ui.py --set sp4t
```

The simulator target starts the runtime with the explicit SP4T variant:

```
NF_PART_VARIANT=SP4T ./server -variant SP4T
```

4. Web UI

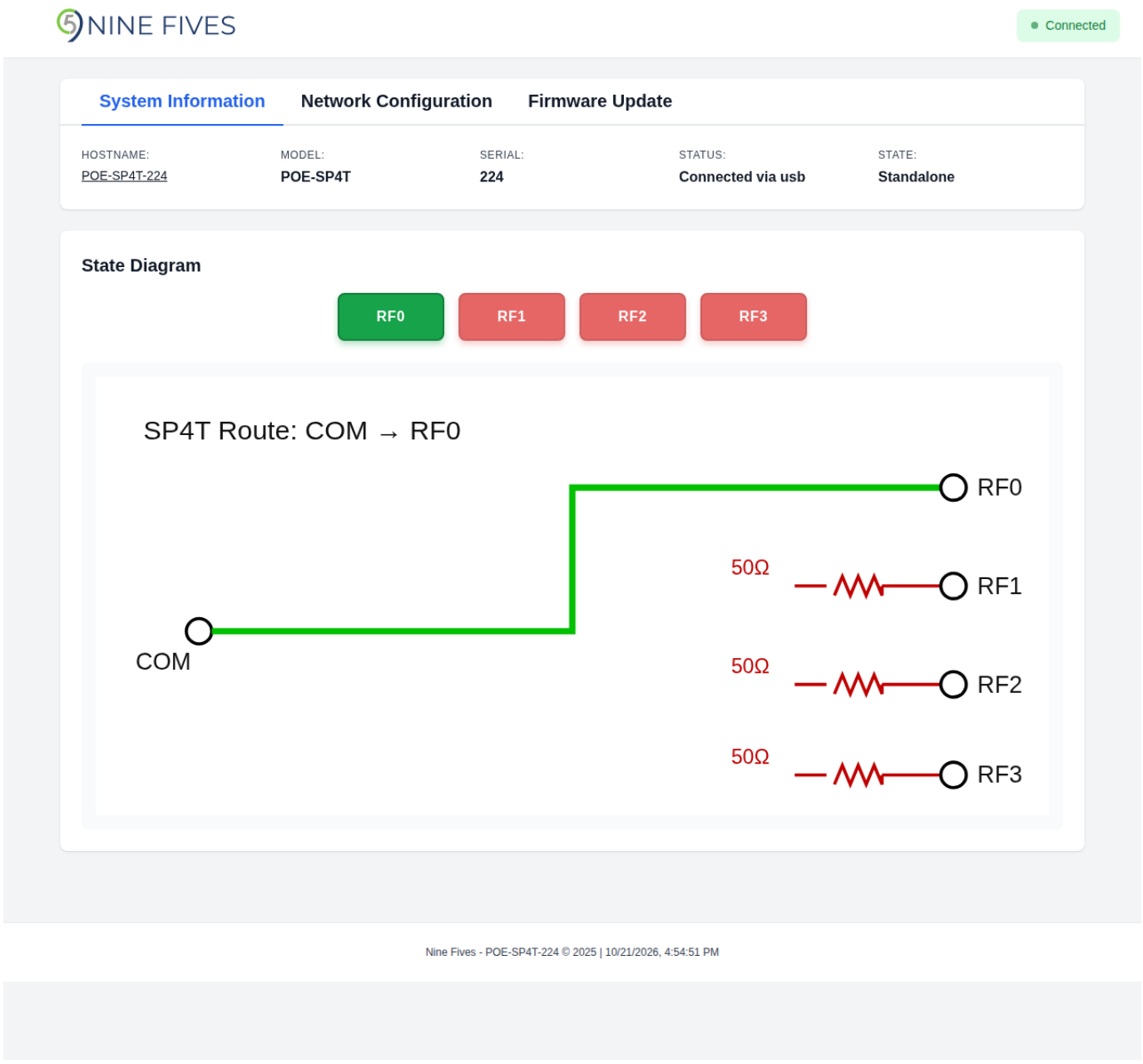
4.1 Web UI Overview

The Nine Fives web interface is accessible at `http://<device-ip>` on port 80. It provides a browser-based graphical interface for monitoring and controlling the SP4T device.

4.1.1 Layout

The interface consists of:

- **Header** — Nine Fives logo and connection status indicator
- **Tab bar** — Switches between System Information, Network Configuration, and Firmware Update views
- **SP4T control card** — Four-position RF route control
- **Notifications** — Success and error messages appear at the bottom of the page and auto-dismiss after a few seconds



4.1.2 Connection Status

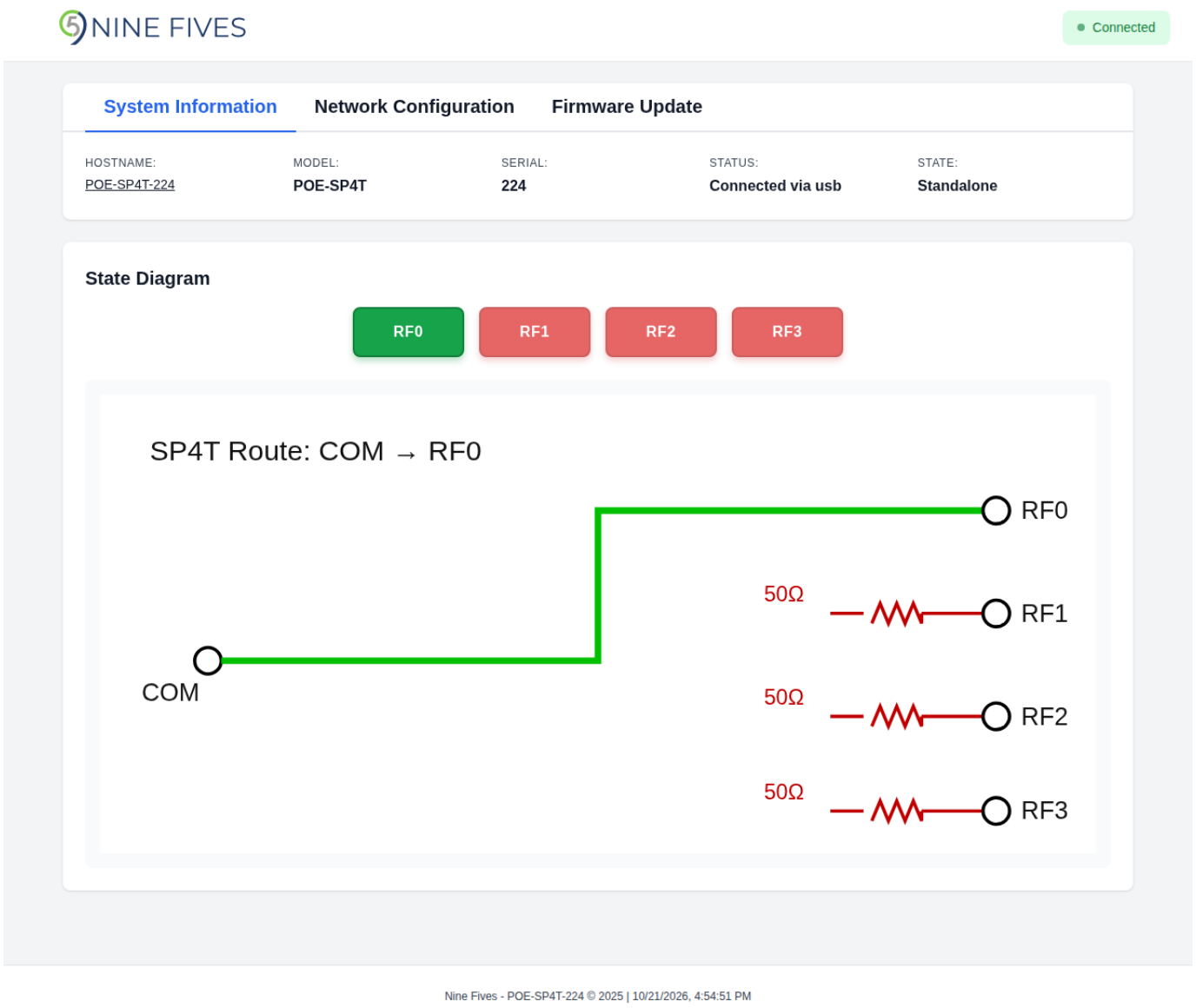
The header displays a connection status badge:

- **Connected** (green) — The browser is communicating with the device
- **Disconnected** (red) — Communication with the device has been lost

The interface polls the device every second to update the displayed state.

4.2 System Information

The **System Information** tab displays the current SP4T device status.



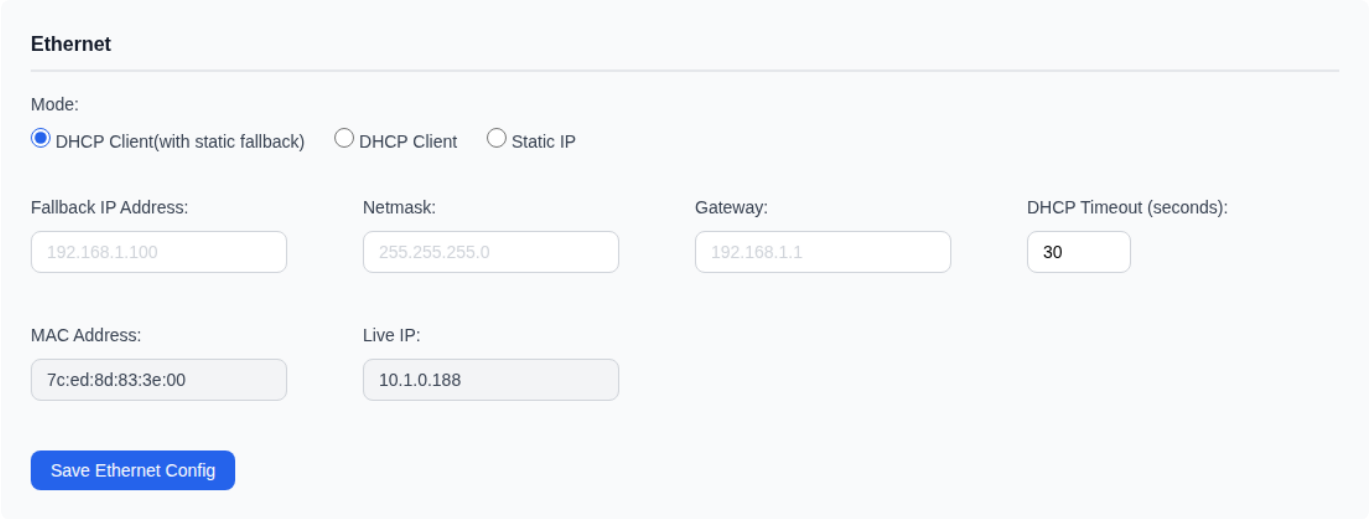
4.2.1 Displayed Fields

FIELD	DESCRIPTION
Hostname	The device hostname, displayed as a clickable link to `http://<hostname>.local`
Model	The device model identifier
Hardware Type	The detected hardware variant, `SP4T` for this manual
Serial	The device serial number
Status	Connection type — "Connected via eth", "Connected via usb", "Connected via dock", or "Disconnected"
State	"Docked" when the module is in its dock, otherwise "Standalone"

4.3 Network Configuration

The **Network Configuration** tab allows you to view and modify the network settings for both the Ethernet and USB-C interfaces.

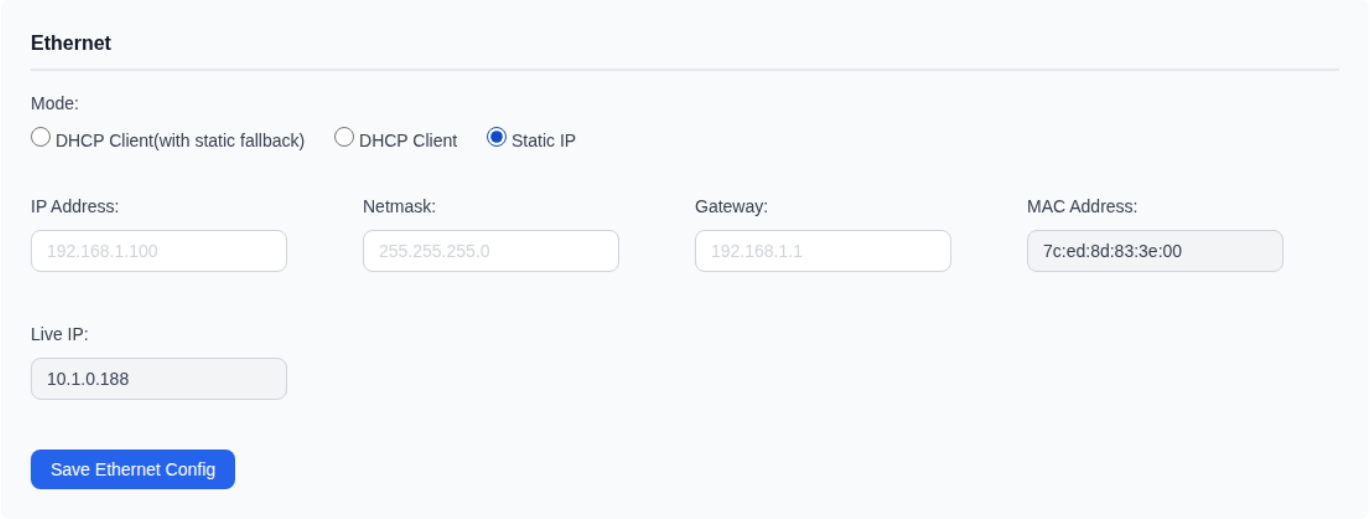
4.3.1 Ethernet Settings



Mode Selection

Select the Ethernet operating mode using the radio buttons:

MODE	DESCRIPTION
DHCP Client (with static fallback)	Attempts DHCP; falls back to the configured static IP address after a timeout. This is the factory default.
DHCP Client	Obtains an IP address from a DHCP server on the network. The IP address and netmask fields are disabled in this mode.
Static IP	Uses the configured IP address, netmask, and gateway.



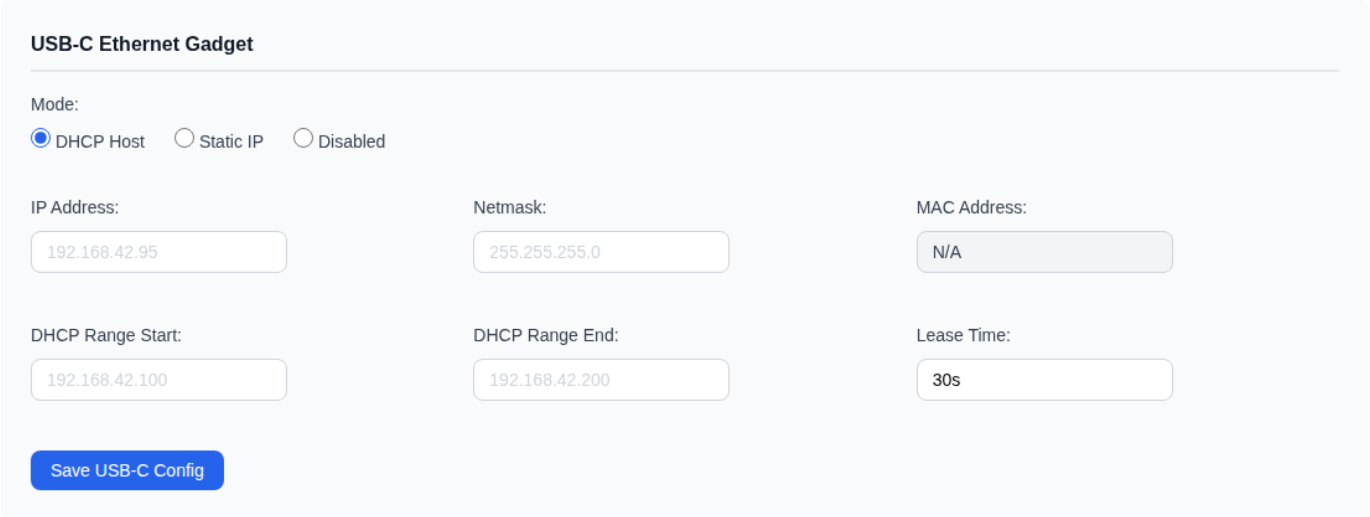
Configuration Fields

FIELD	DESCRIPTION	EDITABLE
IP Address / Fallback IP Address	The configured IP address. Label changes based on mode.	Yes (except in DHCP Client mode)
Netmask	The subnet mask	Yes (except in DHCP Client mode)
Gateway	The default gateway	Yes (except in DHCP Client mode)
DHCP Timeout	Seconds to wait for DHCP before falling back to static. Only shown in DHCP Client (with static fallback) mode.	Yes
MAC Address	The Ethernet interface MAC address	No (read-only)
Live IP	The currently active IP address on the interface	No (read-only)

Click **Save Ethernet Config** to apply changes. Settings take effect immediately.

> **WARNING**
>
> Changing network settings while connected via Ethernet may result in loss of connectivity. Ensure you have an alternative way to reach the device (e.g., via USB-C) before making changes.

4.3.2 USB-C Ethernet Gadget Settings



Mode Selection

MODE	DESCRIPTION
DHCP Host	Runs a DHCP server, automatically assigning an IP address to the connected computer. This is the factory default.
Static IP	Uses the configured IP address and netmask.
Disabled	Brings down the USB-C network interface. All fields are hidden.

Configuration Fields

FIELD	DESCRIPTION	EDITABLE
IP Address	The device IP address on the USB-C interface	Yes
Netmask	The subnet mask	Yes
MAC Address	The USB-C interface MAC address	No (read-only)
Live IP	The currently active IP address on the interface	No (read-only)

When **DHCP Host** mode is selected, additional fields appear:

FIELD	DESCRIPTION
DHCP Range Start	First IP address in the DHCP pool
DHCP Range End	Last IP address in the DHCP pool
Lease Time	DHCP lease duration (e.g., `30s`, `1h`, `24h`)

Click **Save USB-C Config** to apply changes.

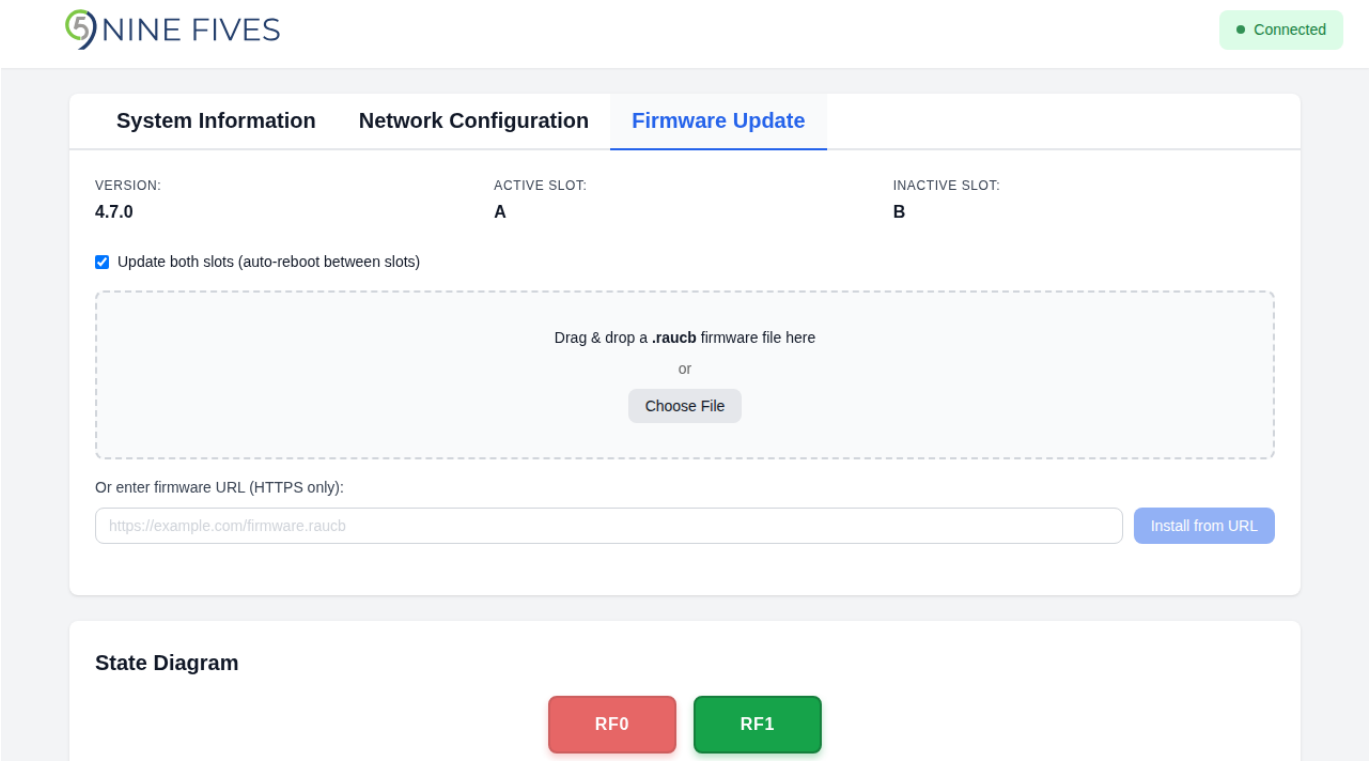
4.3.3 Factory Reset

Click **Reset Networking Settings to Factory Defaults** at the bottom of the network configuration panel to restore all network settings to their factory defaults. A confirmation dialog appears before the reset is applied.

> **WARNING**
>
> The device will briefly be unreachable while factory default settings are applied. The default configuration uses DHCP Client with Static Fallback mode on Ethernet (fallback IP: 192.168.0.95) and DHCP Host mode on USB-C (IP: 192.168.42.95).

4.4 Firmware Update

The **Firmware Update** tab displays the current firmware version and provides tools for updating the device firmware.



4.4.1 Firmware Information

FIELD	DESCRIPTION
Version	The currently running firmware version
Active Slot	The firmware slot currently booted (A or B)
Inactive Slot	The alternate (not booted) firmware slot (A or B)

The device uses an A/B slot scheme. Firmware is always installed to the inactive slot, so the running firmware is never modified while it is in use. A both-slots update replaces the previous firmware in both slots, so no copy of it remains to fall back to.

4.4.2 Update Methods

File Upload

Drag and drop a `.raucb` firmware bundle onto the upload area, or click **Choose File** to select one from your computer.

URL Install

Enter an HTTPS URL pointing to a `.raucb` firmware bundle and click **Install from URL**. The device downloads and installs the firmware directly. The server must present a valid TLS certificate.

Either way, the browser asks you to confirm before the install starts.

4.4.3 Update Modes

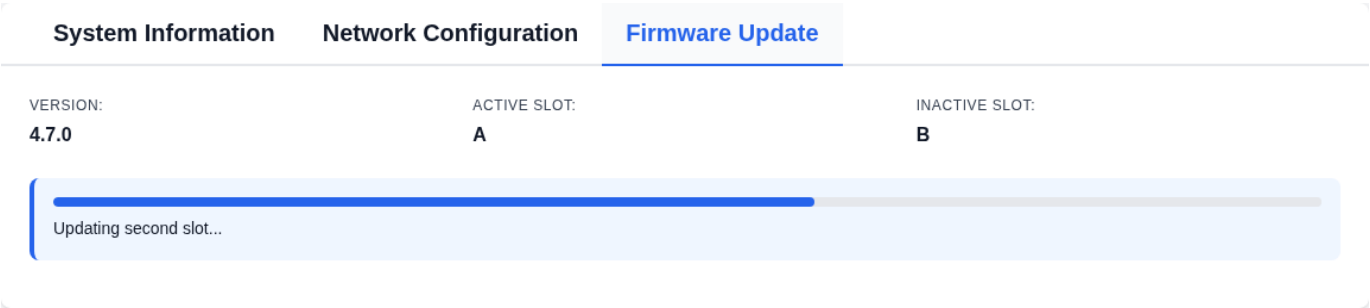
The **Update both slots (auto-reboot between slots)** checkbox selects the mode. It is checked each time the page loads.

MODE	DESCRIPTION
Both slots (default)	The device installs the firmware to the inactive slot, reboots automatically into it, then installs it to the second slot.
Single slot	Uncheck **Update both slots** before uploading *or* installing from a URL. The device installs the firmware to the inactive slot only. After installation, click **Reboot to Activate New Firmware** to boot into the new firmware. The previous firmware stays in the other slot.

4.4.4 Both-Slots Updates

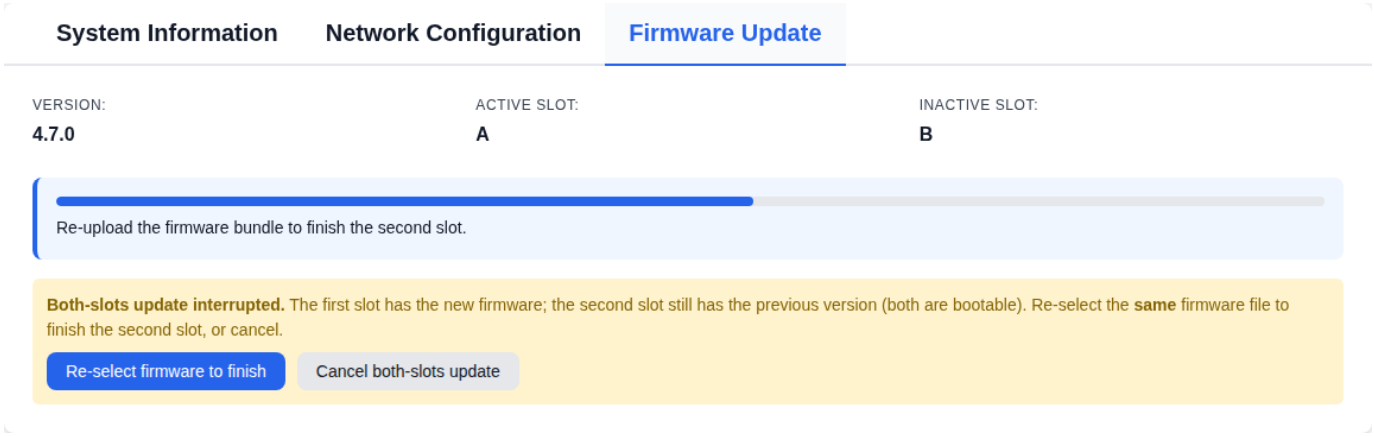
The firmware bundle is too large to stage on the device's persistent storage, so it must be re-supplied for the second slot after the mid-cycle reboot. How that happens depends on how you started the update:

- **File upload:** keep the browser tab open. The page holds the bundle in memory and re-uploads it automatically once the device comes back up to finish the second slot.
- **URL install:** the device re-downloads the bundle from the URL on its own, so there is no need to keep the tab open.



If a file-upload both-slots update is interrupted (for example the tab is closed or reloaded before the second slot finishes), the device is left running the new firmware on the first slot with the previous firmware still on the second slot. Both remain bootable. When you next open the firmware page, a **Both-slots update interrupted** banner offers two choices:

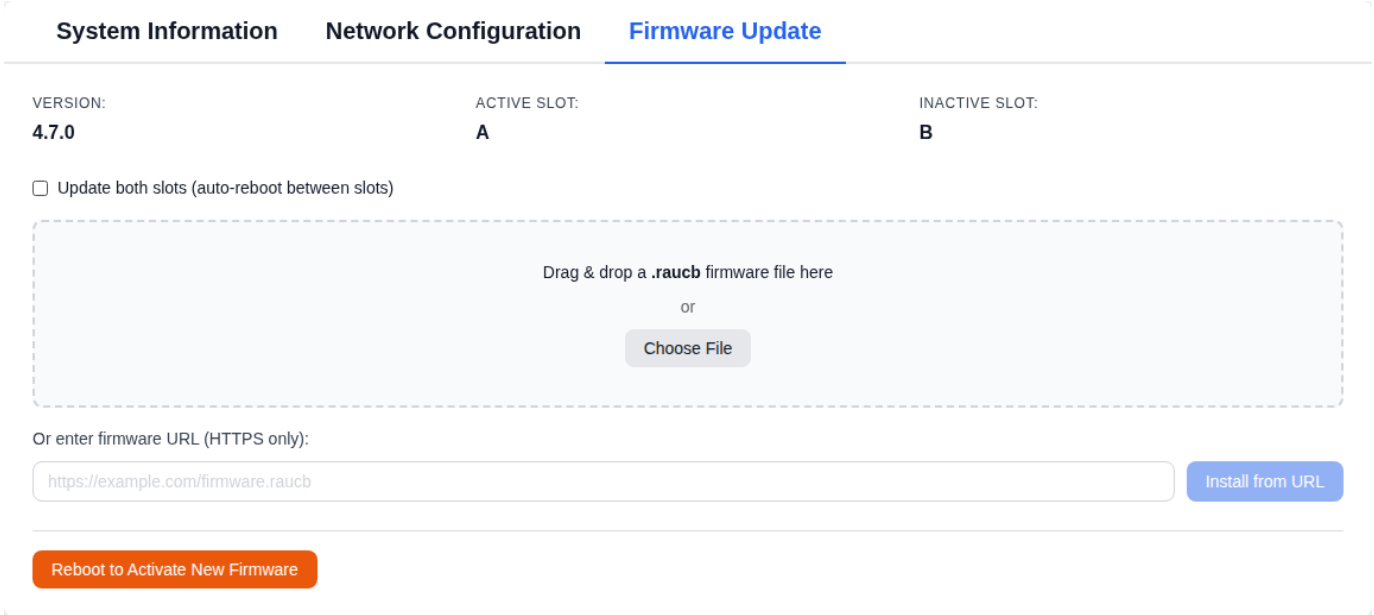
- **Re-select firmware to finish:** choose the *same* `.raucb` file to install it to the second slot. A file with a different version is rejected and the banner stays.
- **Cancel both-slots update:** leave the second slot on the previous firmware.



4.4.5 Update Progress

During an update, the checkbox, upload area and URL field are replaced by a progress bar and status message. The page polls for status every 2 seconds, and keeps polling across the automatic reboot during a both-slots update.

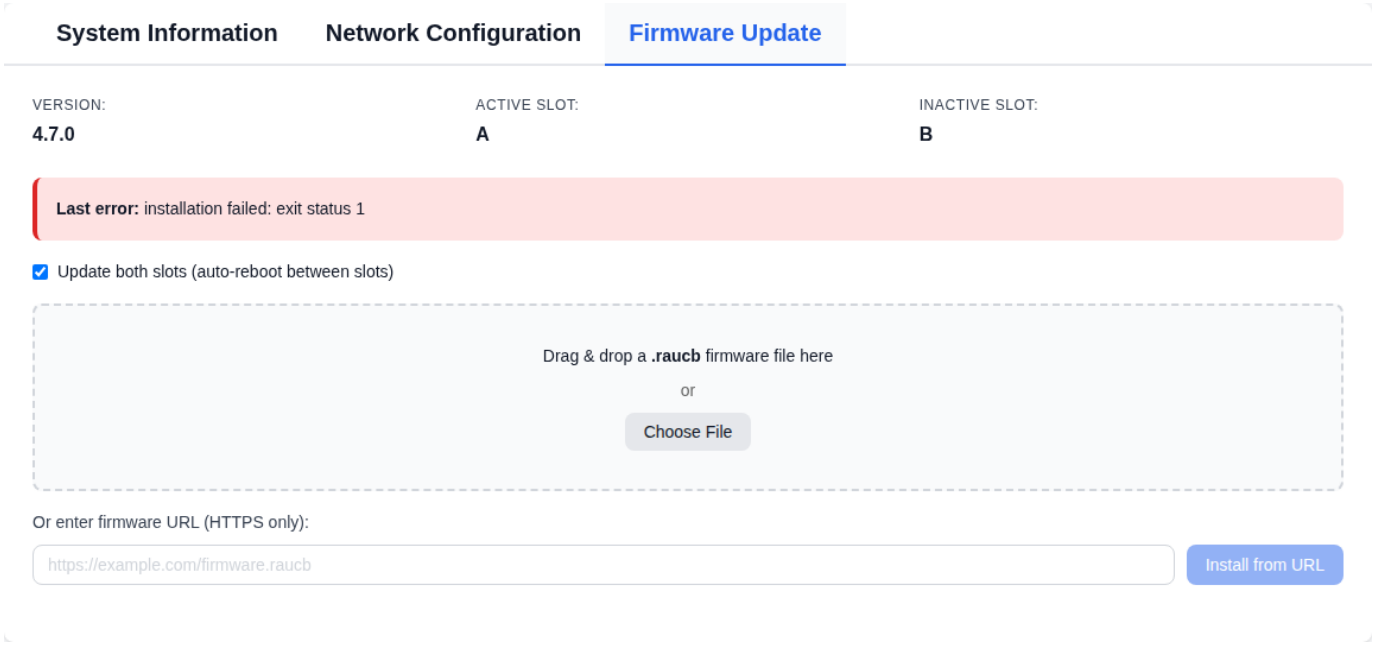
After a successful single-slot update, a **Reboot to Activate New Firmware** button appears. Clicking it (and confirming) reboots the device. The page reloads 30 seconds later.



After a successful both-slots update, the message **Both firmware slots updated successfully.** is displayed. No reboot is needed.

4.4.6 Errors

If an update fails, a **Last error** box below the firmware information shows the reason, and the upload area returns so you can try again. A request the device rejects (for example a non-HTTPS URL) is reported in a notification that disappears after a few seconds.


System Information
Network Configuration
Firmware Update
VERSION:
4.7.0
ACTIVE SLOT:
A
INACTIVE SLOT:
B
Last error: installation failed: exit status 1
Update both slots (auto-reboot between slots)
Drag & drop a .raucb firmware file here
or
Choose File
Or enter firmware URL (HTTPS only):
https://example.com/firmware.raucb
Install from URL

4.5 SP4T Control

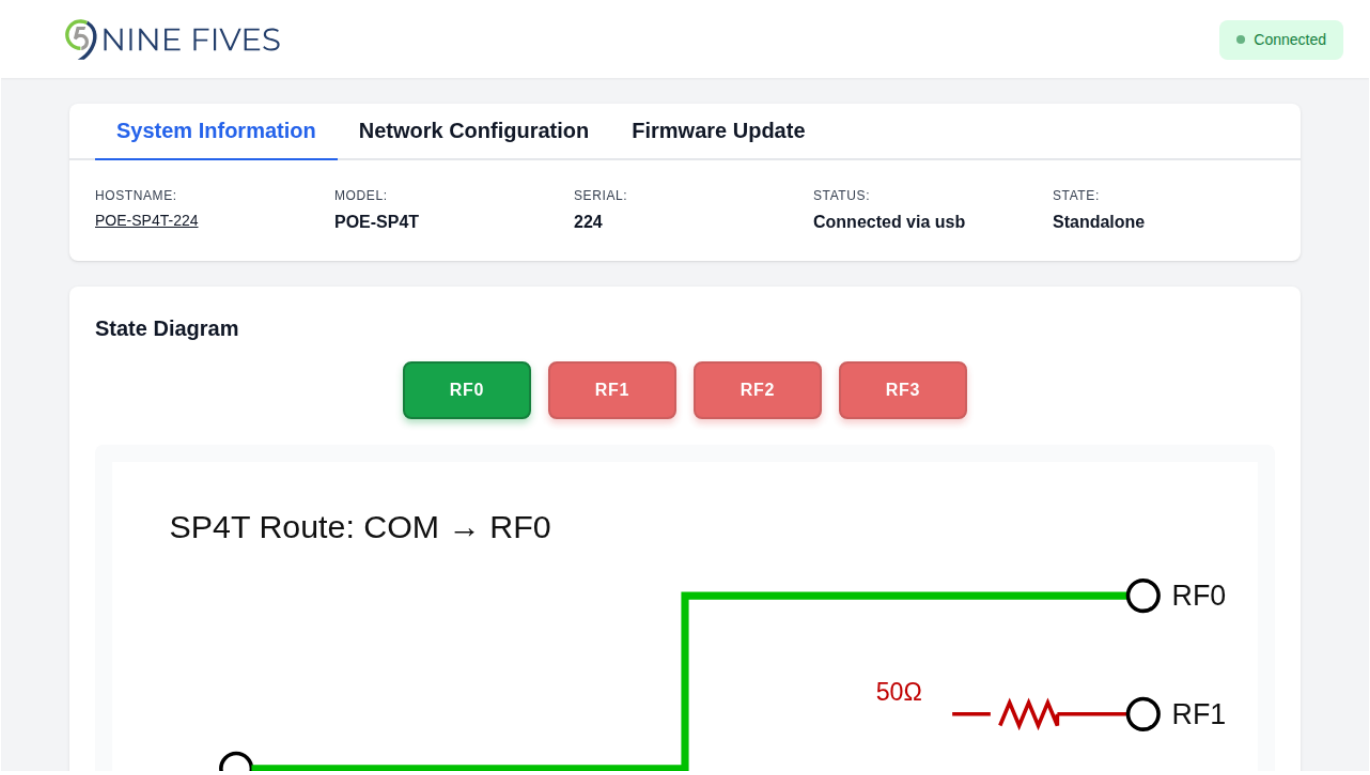
The SP4T web UI shows a four-position control card mirroring the on-device touch UI's 2x2 grid, backed by `/api/sp4t`.

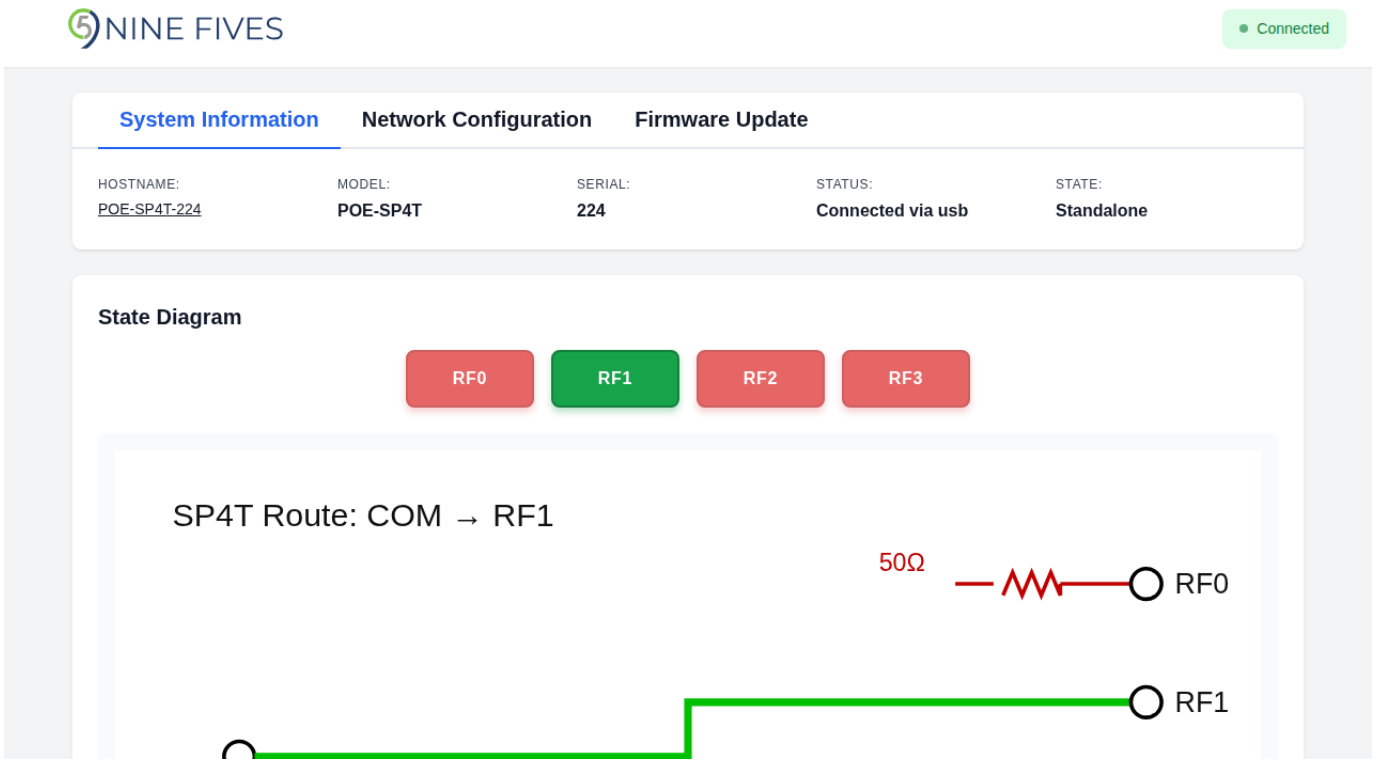
4.5.1 Control Card

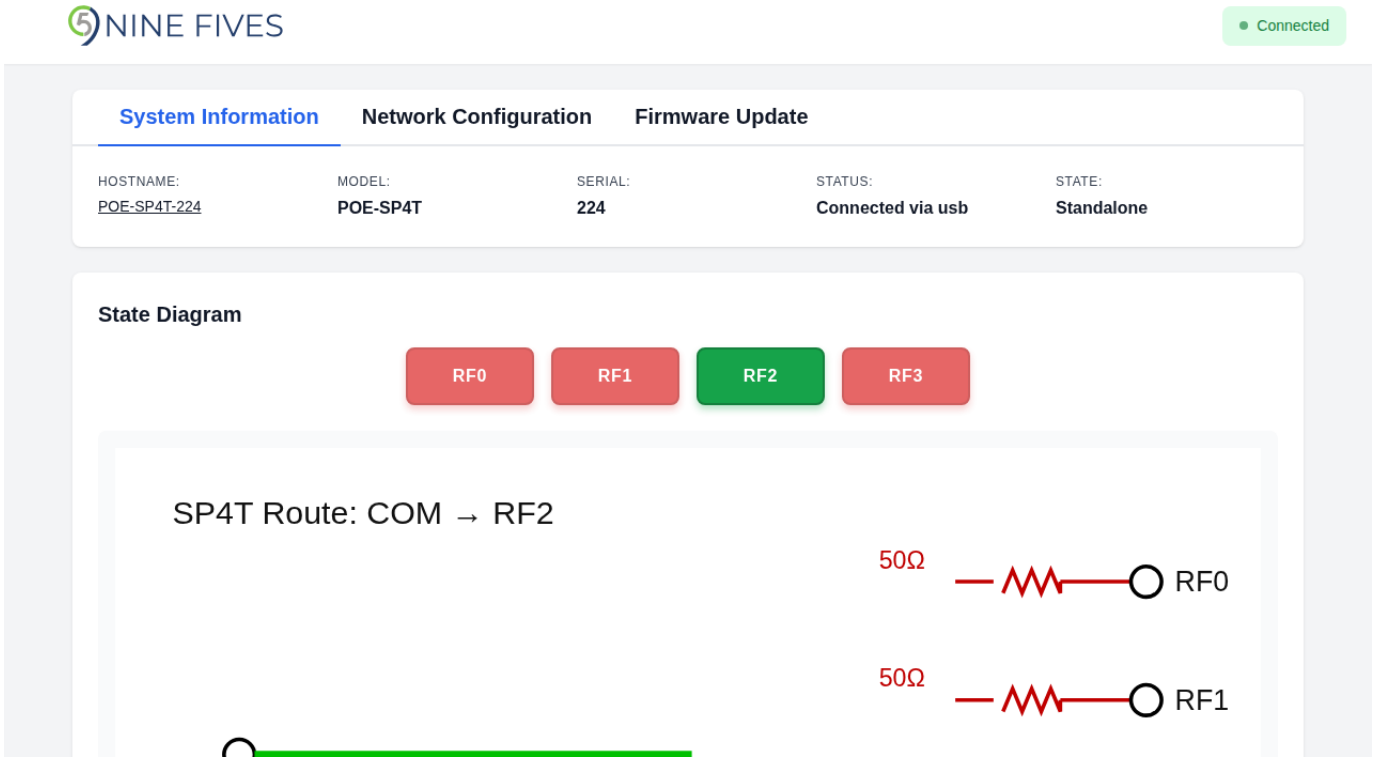
The card renders **RF0–RF3** buttons above the state diagram. The active route's button is green and the diagram shows the corresponding routed path; the other three throws are drawn 50 Ω terminated.

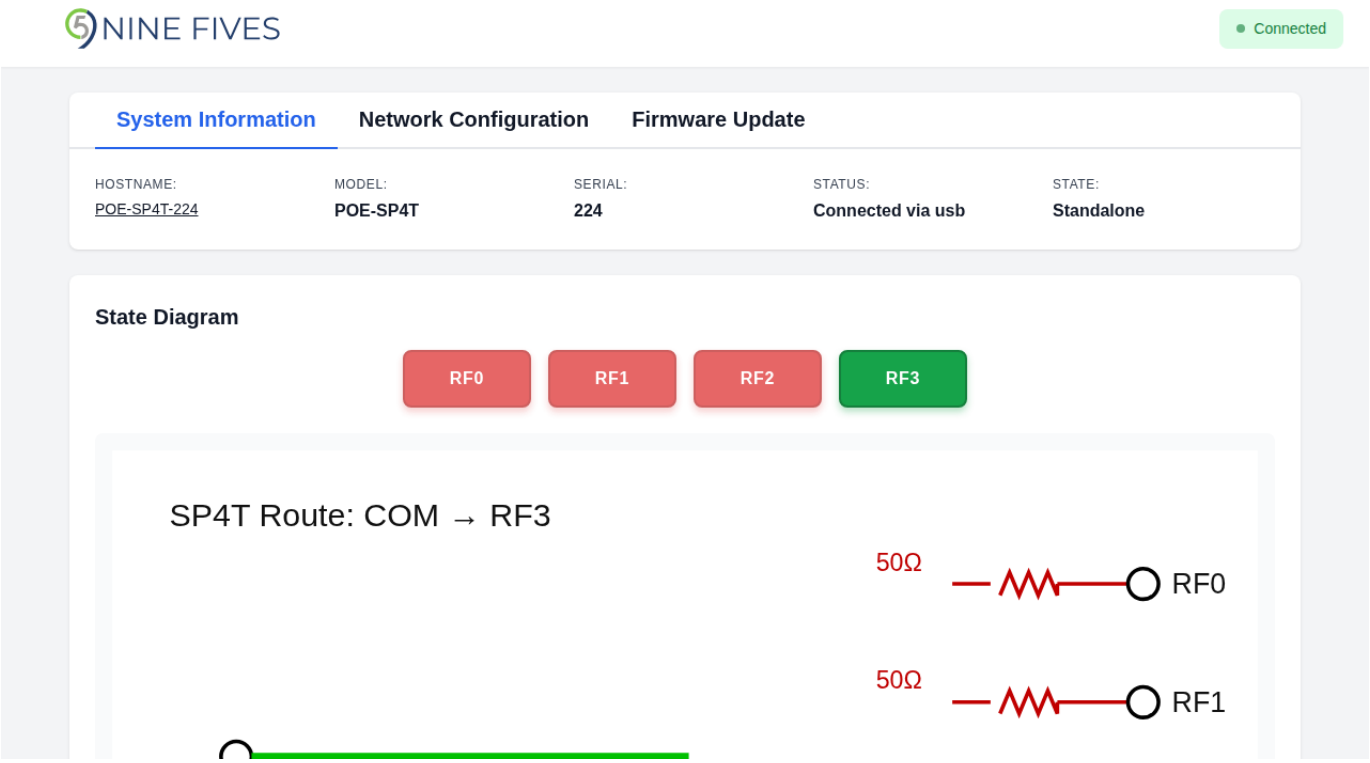
The frontend selects the card from the canonical `hardware type` field of `/api/system/status` rather than substring-matching model strings: `SP4T` renders the four-position card, `SPDT` the two-position card, and `ATTEN` the attenuator controls.

4.5.2 State Diagrams




NINE FIVES
Connected
System Information
Network Configuration
Firmware Update
HOSTNAME:
POE-SP4T-224
MODEL:
POE-SP4T
SERIAL:
224
STATUS:
Connected via usb
STATE:
Standalone
State Diagram
RF0
RF1
RF2
RF3
SP4T Route: COM → RF1
50Ω
RF0
RF1


NINE FIVES
Connected
System Information
Network Configuration
Firmware Update
HOSTNAME:
POE-SP4T-224
MODEL:
POE-SP4T
SERIAL:
224
STATUS:
Connected via usb
STATE:
Standalone
State Diagram
RF0
RF1
RF2
RF3
SP4T Route: COM → RF2
50Ω
RF0
50Ω
RF1



Four SVG diagrams (`poe_sp4t.rf0.svg` … `poe_sp4t.rf3.svg`) show the route for each throw:

STATE	DIAGRAM
RF0	COM → RF0
RF1	COM → RF1
RF2	COM → RF2
RF3	COM → RF3

4.5.3 Hardware Type Unset / Unknown

When `hardware_type` is `unset` or `unknown`, the runtime starts no RF manager and the web UI renders a disabled-hardware card instead of any control card. Network configuration remains available from the Network Configuration tab.

> **PROVISIONING**
>
> SP4T boards must have `get_nf_part_variant.sh` return `SP4T`. A board left as `SWITCH` runs as SPDT, never requests EXT_GPIO1, and leaves the route indeterminate in whichever half the undriven line selects — the runtime startup log is the only signal.

See SPDT Control for the two-position variant.

5. REST API

5.1 REST API Overview

The device provides a JSON REST API on port 80 for programmatic control and monitoring.

5.1.1 Base URL

```
http://<device-ip>
```

All endpoints accept and return JSON (`Content-Type: application/json`) unless otherwise noted.

5.1.2 Error Handling

HTTP STATUS	DESCRIPTION
200	Success
202	Accepted (asynchronous operation started)
400	Bad Request — invalid input or parameters
405	Method Not Allowed — wrong HTTP method
409	Conflict — operation already in progress
500	Internal Server Error
503	Service Unavailable

Error responses are returned as plain text with a descriptive message.

5.1.3 Examples

The examples in this section use `curl`. Replace `<device-ip>` with your device's IP address.

```
# Query system status
curl http://<device-ip>/api/system/status

# POST with JSON body
curl -X POST -H "Content-Type: application/json" \
  -d '{"state": 1}' \
  http://<device-ip>/api/spdt
```

5.2 System Endpoints

5.2.1 GET /api/system/status

Returns complete system status information.

Response:

```
{
  "device": "Nine Fives Programmable SPDT Switch",
  "hostname": "nf-device",
  "model": "NF-SW-01",
  "serial": "ABC123",
  "status": "Connected via eth",
  "temperature": 45.2,
  "connectedIface": "eth0"
}
```

FIELD	TYPE	DESCRIPTION
`device`	string	Device name
`hostname`	string	System hostname
`model`	string	Device model identifier
`serial`	string	Serial number
`status`	string	Connection status: `"Connected via eth"`, `"Connected via usb"`, or `"Disconnected"`
`temperature`	number	Device temperature in °C
`connectedIface`	string	Network interface the client is connected through (`eth0` or `usb0`)

Sub-Endpoints

Individual fields are available at dedicated endpoints:

ENDPOINT	RETURNS
`GET /api/system/status/device`	`{"device": "..."}`
`GET /api/system/status/hostname`	`{"hostname": "..."}`
`GET /api/system/status/model`	`{"model": "..."}`
`GET /api/system/status/serial`	`{"serial": "..."}`
`GET /api/system/status/status`	`{"status": "..."}`
`GET /api/system/status/temperature`	`{"temperature": 45.2}`
`GET /api/system/status/connectedIface`	`{"connectedIface": "eth0"}`

5.2.2 POST /api/system/reboot

Reboots the device. The device will be unreachable for approximately 30 seconds.

Request: No body required.

Response:

```
{
  "status": "ok",
```

```
  "message": "Rebooting in 3 seconds..."
}
```

5.3 Network Endpoints

5.3.1 GET /api/network/config

Returns network configuration for both the Ethernet and USB-C interfaces, including configured settings and live interface information.

Response:


```
{
  "ethernet": {
    "mode": "dhcp_client_fallback",
    "ip_address": "192.168.0.95",
    "netmask": "255.255.0.0",
    "gateway": "",
    "dhcp_fallback_timeout": 30,
    "dhcp_server": {
      "range_start": "192.168.0.100",
      "range_end": "192.168.0.199",
      "lease_time": "30s"
    },
    "live_ip": "192.168.0.95",
    "mac": "00:11:22:33:44:55"
  },
  "usb_gadget": {
    "mode": "dhcp_host",
    "ip_address": "192.168.42.95",
    "netmask": "255.255.255.0",
    "enabled": true,
    "dhcp_server": {
      "range_start": "192.168.42.100",
      "range_end": "192.168.42.200",
      "lease_time": "30s"
    },
    "live_ip": "192.168.42.95",
    "mac": "00:11:22:33:44:56"
  }
}
```


Ethernet Modes

MODE	DESCRIPTION
`dhcp client fallback`	Try DHCP; fall back to static IP after timeout (factory default)
`static`	Static IP address
`dhcp client`	Obtain IP from a network DHCP server

USB-C Modes

MODE	DESCRIPTION
`dhcp host`	Run DHCP server for the connected host computer (factory default)
`static`	Static IP address
`disabled`	Interface is brought down

Read-Only Fields

The `live ip` and `mac` fields are read-only and reflect the current state of the interface.

5.3.2 POST /api/network/config

Updates network configuration and applies it immediately. You can update Ethernet, USB-C, or both interfaces in a single request. Only include the fields you want to change.

> **WARNING**
>
> Changing network settings while connected may result in loss of connectivity. Settings take effect immediately.

Example — set Ethernet to static IP:

```
curl -X POST -H "Content-Type: application/json" -d '{
  "ethernet": {
    "mode": "static",
    "ip_address": "192.168.1.100",
    "netmask": "255.255.255.0",
    "gateway": "192.168.1.1"
  }
}' http://<device-ip>/api/network/config
```

Example — set Ethernet to DHCP client mode:

```
curl -X POST -H "Content-Type: application/json" -d '{
  "ethernet": {
    "mode": "dhcp_client"
  }
}' http://<device-ip>/api/network/config
```

Example — set Ethernet to DHCP client with static fallback:

```
curl -X POST -H "Content-Type: application/json" -d '{
  "ethernet": {
    "mode": "dhcp_client_fallback",
    "ip_address": "192.168.0.95",
    "netmask": "255.255.0.0",
    "dhcp_fallback_timeout": 30
  }
}' http://<device-ip>/api/network/config
```

Example — disable USB-C networking:

```
curl -X POST -H "Content-Type: application/json" -d '{
  "usb_gadget": {
    "mode": "disabled",
    "enabled": false
  }
}' http://<device-ip>/api/network/config
```

Response:

```
{
  "status": "ok"
}
```

Errors:

- `400` — Invalid request body, invalid or unknown mode, or invalid IP address
- `500` — Failed to save or apply configuration

5.3.3 POST /api/network/reset

Restores factory default network configuration and applies it immediately.

Request: No body required.

Response:

```
{
  "status": "ok"
}
```

> **WARNING**
>
> The device will briefly be unreachable while factory default settings are applied. The default configuration is DHCP Client with Static Fallback on Ethernet (fallback IP: 192.168.0.95) and DHCP Host on USB-C (IP: 192.168.42.95).

5.4 Firmware Endpoints

5.4.1 GET /api/firmware/status

Returns current firmware version, slot information, and update progress.

Response (idle):

```
{
  "version": "1.2.0-abc1234",
  "activeSlot": "A",
  "inactiveSlot": "B",
  "inactiveVersion": "",
  "updateInProgress": false
}
```

Response (both-slots update, second slot installing):

```
{
  "version": "1.3.0-def5678",
  "activeSlot": "B",
  "inactiveSlot": "A",
  "inactiveVersion": "",
  "updateInProgress": true,
  "updateProgress": 60,
  "updateMessage": "Updating second slot...",
  "updateMode": "full",
  "updatePhase": "installing_second"
}
```

FIELD	TYPE	DESCRIPTION
`version`	string	Currently running firmware version
`activeSlot`	string	Active firmware slot (`A` or `B`)
`inactiveSlot`	string	Inactive firmware slot
`inactiveVersion`	string	Reserved; currently always empty
`updateInProgress`	boolean	Whether an update is currently running
`updateProgress`	number	Progress percentage (0–100)
`updateMessage`	string	Human-readable status message
`updateMode`	string	`"full"` for a both-slots update; omitted for single-slot
`updatePhase`	string	Current phase of a both-slots update (see below)
`lastError`	string	Error message from the last failed update

`updateProgress`, `updateMessage`, `updateMode`, `updatePhase` and `lastError` are omitted when they are zero or empty. After an update finishes, `updateProgress` stays at `100` and `updateMessage` keeps the result until the device restarts:

- Single slot: `"Installation complete. Reboot to activate new firmware."` Reboot with `POST /api/system/reboot` to activate it.
- Both slots: `"Both slots updated successfully."` with `updateMode: "full"`. No reboot is needed.

Full Update Phases

PHASE	DESCRIPTION
`installing_first`	Installing firmware to the inactive slot
`rebooting`	First slot done, device is about to reboot
`awaiting_second_upload`	After the reboot, an upload-sourced update is waiting for the bundle at `/api/firmware/update/resume` (`updateInProgress` is `true`, progress 55)
`installing_second`	After reboot, installing firmware to the second slot

`updatePhase` is omitted once a both-slots update completes. If the update fails, it keeps the phase in which the failure happened.

Errors:

- `503` — Firmware manager not available

5.4.2 POST /api/firmware/update

Upload a `.raucb` firmware bundle for installation. Uses `multipart/form-data` encoding.

PARAMETER	DESCRIPTION
Form field: `firmware`	The `.raucb` file (max 256 MiB)
Query: `mode`	Empty (default) for single slot, `full` for both slots

Example — single slot:

```
curl -X POST -F "firmware=@firmware-1.2.0.raucb" \
  http://<device-ip>/api/firmware/update
```

Example — both slots:

```
curl -X POST -F "firmware=@firmware-1.2.0.raucb" \
  "http://<device-ip>/api/firmware/update?mode=full"
```

Response (202 Accepted):

```
{
  "status": "accepted",
  "message": "Installing firmware from firmware-1.2.0.raucb"
}
```

With `mode=full`, the message is `"Installing firmware from firmware-1.2.0.raucb (both slots, will reboot automatically)"`.

Installation runs asynchronously. Poll `GET /api/firmware/status` for progress.

In `mode=full`, the device reboots automatically after the first slot. The bundle is **not** staged on the device, and only a small marker is kept across the reboot. After the reboot the status reports `updatePhase: "awaiting_second_upload"`, and the caller must re-upload the same bundle to `POST /api/firmware/update/resume` to finish the second slot. (URL-sourced full updates skip this; see below.)

Errors:

- `400` — Missing file, file larger than 256 MiB, invalid form data, or a file name not ending in `.raucb`
- `409` — Update already in progress
- `500` — The upload could not be saved on the device
- `503` — Firmware manager not available

5.4.3 POST /api/firmware/update/url

Download and install a firmware bundle from a URL.

Request:

```
{
  "url": "https://example.com/firmware-1.2.0.raucb"
}
```

Only HTTPS URLs are accepted, and the server must present a valid TLS certificate. The bundle must be no larger than 256 MiB.

PARAMETER	DESCRIPTION
Query: `mode`	Empty (default) for single slot, `full` for both slots

With `mode=full`, the device re-downloads the bundle from the URL after the reboot and finishes the second slot on its own, so no further client action is required. If the re-download fails, it retries up to 5 times, 10 seconds apart.

Response (202 Accepted):

```
{
  "status": "accepted",
  "message": "Downloading and installing firmware from URL"
}
```

With `mode=full`, the message is `"Downloading and installing firmware from URL (both slots, will reboot automatically)"`.

Errors:

- `400` — Invalid request body, missing URL, non-HTTPS URL, or update already in progress
- `503` — Firmware manager not available

Download failures after the request is accepted are reported in `lastError` of `GET /api/firmware/status`.

5.4.4 POST /api/firmware/update/resume

Finish the second slot of an interrupted **upload-sourced** full update by re-uploading the bundle. Uses `multipart/form-data` with a `firmware` field, like `/api/firmware/update`. Valid only while `updatePhase` is `awaiting_second_upload`. The bundle version must match the version installed to the first slot.

```
curl -X POST -F "firmware=@firmware-1.2.0.raucb" \
  http://<device-ip>/api/firmware/update/resume
```

Response (202 Accepted):

```
{
  "status": "accepted",
  "message": "Finishing second slot..."
}
```

Errors:

- `400` — Missing file, file larger than 256 MiB, invalid form data, or a file name not ending in `.raucb`
- `409` — Not awaiting a second-slot upload, or bundle version mismatch. On a mismatch the device stays in `awaiting second upload` with the reason in `lastError`, so the correct bundle can be uploaded.
- `500` — The upload could not be saved on the device
- `503` — Firmware manager not available

5.4.5 POST /api/firmware/update/cancel

Abandon an interrupted full update, clearing the pending second-slot marker and the update status. The device stays on the new firmware; the second slot keeps its previous version. Both slots remain bootable. Use it only while `updatePhase` is `awaiting second upload`; it does not stop an install that is running.

```
curl -X POST http://<device-ip>/api/firmware/update/cancel
```

Response (200 OK):

```
{
  "status": "ok",
  "message": "Full update cancelled. Both slots remain bootable."
}
```

Errors:

- `503` — Firmware manager not available

5.5 SP4T Endpoints

These endpoints are available on SP4T switch hardware. States are zero-based and there is no "all open" state — exactly one throw is routed at all times.

5.5.1 GET /api/sp4t

Returns the currently routed throw.

Response:

```
{
  "state": 0
}
```

STATE	DESCRIPTION
`0`	COM routed to RF0
`1`	COM routed to RF1
`2`	COM routed to RF2
`3`	COM routed to RF3

5.5.2 POST /api/sp4t

Sets the routed throw.

Request:

```
{
  "state": 2
}
```

Response:

```
{
  "state": 2
}
```

Errors:

- `400` — Invalid state value (must be `0`–`3`)
- `500` — Failed to set SP4T state

Example:

```
# Route COM to RF2
curl -X POST -H "Content-Type: application/json" \
  -d '{"state": 2}' \
  http://<device-ip>/api/sp4t

# Query the current throw
curl http://<device-ip>/api/sp4t
```

5.5.3 Deprecated /api/switch Alias

On SP4T hardware, `/api/switch` is a deprecated alias for `/api/sp4t` and carries the same 0–3 state range. Existing bench scripts and saved automation that predate the rename keep working; new automation should use `/api/sp4t`.

For the two-throw variant see SPDT Endpoints.

6. SCPI API

6.1 SCPI Overview

The device implements a SCPI (Standard Commands for Programmable Instruments) command interface over two transport protocols.

6.1.1 Transport Protocols

Raw TCP (Port 5025)

PROPERTY	VALUE
Protocol	TCP
Port	5025
Format	Text-based, newline-terminated

Example connections:

```
telnet <device-ip> 5025
nc <device-ip> 5025
```

HiSLIP (Port 4880)

HiSLIP (High-Speed LAN Instrument Protocol) is defined by IVI Foundation specification IVI-6.1. It provides improved performance over raw TCP and supports VISA resource strings.

PROPERTY	VALUE
Protocol	HiSLIP (IVI-6.1)
Port	4880
Vendor ID	NF

VISA resource string:

```
TCPIP::<device-ip>::hislip0::INSTR
```

Example (Python with PyVISA):

```
import pyvisa
rm = pyvisa.ResourceManager()
inst = rm.open_resource("TCPIP::192.168.0.95::hislip0::INSTR")
print(inst.query("*IDN?"))
```

All commands documented in this section are available on both transport protocols.

6.1.2 Command Syntax

Case Insensitivity

Commands are case-insensitive:

```
*IDN?
*idn?
*Idn?
```

Short Forms

SCPI supports abbreviated commands. Uppercase letters in the documentation indicate the minimum required characters:

FULL FORM	SHORT FORM
`:SYSTem:NETwork:ETHernet:MODE?`	`:SYST:NET:ETH:MODE?`
`:SYSTem:FIRMware:VERSion?`	`:SYST:FIRM:VERS?`
`SWITch:MAIN:STATe?`	`SWIT:MAIN:STAT?`

Multiple Commands

Multiple commands can be sent on one line, separated by semicolons:

```
*IDN?; :ATT?
```

6.1.3 Error Codes

Errors are queued and read with `:SYSTem:ERRor?` (see Common Commands).

CODE	DESCRIPTION (AS RETURNED BY `:SYSTEM:ERROR?`)	RAISED WHEN
0	No error	Queue is empty
-100	Command error	Unparseable command line
-103	Invalid separator	Parameters not separated by commas
-104	Data type error	Wrong parameter type, e.g. `:SETATT abc`
-108	Parameter not allowed	More parameters than the command takes
-109	Missing parameter	Required parameter omitted
-113	Undefined header	Unknown command
-200	Execution error	Hardware or configuration operation failed
-221	Settings conflict	Network settings rejected as a combination
-222	Data out of range	Value outside the legal range, e.g. attenuation or switch state
-224	Illegal parameter value	Value not in the allowed set, e.g. bad mode name or IPv4 address
-350	Queue overflow	Error queue was full; newest errors were dropped
-363	Input buffer overrun	Command line exceeded the input buffer

Descriptions may be followed by `;<detail>` with more context. Class ranges: -100 to -199 command errors, -200 to -299 execution errors, -300 to -399 device-specific errors, -400 to -499 query errors.

6.2 Common Commands (IEEE 488.2)

These commands are available on all device variants.

6.2.1 *IDN?

Query instrument identification.

Response:

```
Nine Fives,<controller_type>,<serial>,<version>
```

Where `<controller_type>` is `Switch Controller` or `Attenuator Controller`.

Example:

```
*IDN?
Nine Fives,Switch_Controller,0001,1.0.0
```

6.2.2 *RST

Reset instrument to default state.

- **Programmable SPDT Switch:** Sets switch to RF1 (GPIO low — preserving the pre-rename physical effect)
- **Programmable SP4T Switch:** Sets switch to RF0 (both select lines low — the power-on route)
- **Programmable Attenuator:** Sets attenuation to 0 dB

> **RESET STATE IS ASYMMETRIC BETWEEN SWITCH VARIANTS**
>
> SPDT resets to **RF1** while SP4T resets to **RF0**. The SPDT reset preserves the physical GPIO-low behaviour the hardware has always had; on SP4T both select lines low is the RF0 route, so that is the reset state.

Example:

```
*RST
```

6.2.3 *CLS

Clear the error queue and the Standard Event Status Register. Enable registers (`*ESE`, `*SRE`) are not changed.

Example:

```
*CLS
```

6.2.4 *ESR?

Query and clear the Standard Event Status Register. Response is an integer 0–255.

BIT	VALUE	MEANING
0	1	Operation complete (`*OPC`)
2	4	Query error
3	8	Device-specific error
4	16	Execution error
5	32	Command error

6.2.5 *ESE / *ESE?

Set / query the Standard Event Status Enable register (0–255). `*CLS` does not change it.

6.2.6 *SRE / *SRE?

Set / query the Service Request Enable register (0–255; bit 6 always reads 0). `*CLS` does not change it.

6.2.7 *STB?

Query the Status Byte. Bit 2 (value 4) is set while the error queue is not empty, bit 5 (32) summarises `*ESR? & *ESE`, and bit 6 (64) is set when any enabled summary bit is set.

6.2.8 *OPC / *OPC? / *WAI

All commands finish executing before the next one starts, so `*OPC` sets the Operation Complete bit of `*ESR?` immediately, `*OPC?` returns `1`, and `*WAI` returns immediately.

6.2.9 *TST?

Self-test query. No self-test is implemented; always returns `0` (pass).

6.2.10 :SYSTem:ERRor[:NEXT]?

Read and remove the oldest entry in the error queue. Response is `<code>,"<description>[;<detail>]"`, or `0,"No error"` when the queue is empty.

Example:

```
:SETATT 999
:SYST:ERR?
-222,"Data out of range;attenuation out of range: 999.00 dB is not valid for 1 attenuators (0 to 31.75 dB in 0.25 dB steps)"
:SYST:ERR?
0,"No error"
```

The queue holds 16 entries and is kept per connection (per session on HiSLIP). When it is full the oldest entries are kept, the last entry is replaced with `-350,"Queue overflow"`, and newer errors are discarded. The queue is emptied by `*CLS` and by reading it.

COMMAND	DESCRIPTION
`:SYSTem:ERRor:ALL?`	Read and remove all entries as `code,"description",...` (or `0,"No error"`)
`:SYSTem:ERRor:COUNt?`	Number of unread entries
`:SYSTem:ERRor:CODE[:NEXT]?`	Like `:SYSTem:ERRor?` but the code only
`:SYSTem:ERRor:CODE:ALL?`	All codes only, comma separated (`0` when empty)

6.2.11 :SYSTem:VERSion?

Returns the SCPI version this instrument complies with: `1999.0`.

6.2.12 :STATus subsystem

COMMAND	DESCRIPTION
`:STATus:OPERation[:EVENt]?`	Operation event register (always 0; no conditions are reported)
`:STATus:OPERation:CONDition?`	Operation condition register (always 0)
`:STATus:OPERation:ENABle <n>` / `?`	Operation enable register (0–65535)
`:STATus:QUEStionable[:EVENt]?`	Questionable event register (always 0)
`:STATus:QUEStionable:CONDition?`	Questionable condition register (always 0)
`:STATus:QUEStionable:ENABle <n>` / `?`	Questionable enable register (0–65535)
`:STATus:PRESet`	Clear the OPERation and QUEStionable enable registers

6.3 System Commands

These commands are available on all device variants.

6.3.1 :SYSTem:FIRMware:VERSion?

Query the current firmware version.

Short form: `:SYST:FIRM:VERS?`

Response: Quoted string

Example:

```
:SYST:FIRM:VERS?
"1.2.0-abc1234"
```

6.3.2 :SYSTem:FIRMware:STATus?

Query firmware update status.

Short form: `:SYST:FIRM:STAT?`

Response: Status string in one of the following formats:

RESPONSE	DESCRIPTION
`"IDLE"`	No update in progress
`"UPDATING,<progress>,<message>"`	Update in progress with percentage and status
`"COMPLETE"`	Last update completed. After a single-slot update, reboot (`:SYSTem:REBoot`) to activate it.
`"ERROR,<message>"`	Last update failed

`COMPLETE` is reported until the device restarts, including after a both-slots update started from the web UI or REST API. A both-slots update interrupted after its reboot reports `"UPDATING,55,Re-upload the firmware bundle to finish the second slot."`. It can only be finished or cancelled from the web UI or REST API.

Example:

```
:SYST:FIRM:STAT?
"IDLE"
```

6.3.3 :SYSTem:FIRMware:UPDate \<url>

Download a firmware bundle from a URL and install it to the inactive slot. Only HTTPS URLs are accepted. The update runs asynchronously — poll `:SYSTem:FIRMware:STATus?` for progress, then reboot with `:SYSTem:REBoot` to activate the new firmware. To update both slots, use the web UI or the REST API.

Short form: `:SYST:FIRM:UPD`

Example:

```
:SYST:FIRM:UPD "https://example.com/firmware-1.2.0.raucb"
```

Errors:

- `-109` — Missing URL parameter
- `-200` — Update failed to start (already in progress, empty or non-HTTPS URL, etc.)

6.3.4 :SYSTem:REBoot

Reboot the device. The device will be unreachable for approximately 30 seconds.

Short form: `:SYST:REB`

Example:

```
:SYST:REB
```

6.4 Network Commands

These commands configure the Ethernet and USB-C network interfaces. Write commands take effect immediately.

6.4.1 Ethernet Interface

:SYSTem:NETwork:ETHernet:MODE

Query or set the Ethernet operating mode.

Short form: `:SYST:NET:ETH:MODE`

MODE	DESCRIPTION
`static`	Static IP address
`dhcp client`	Obtain IP from a DHCP server
`dhcp client fallback`	Try DHCP; fall back to static after timeout

Examples:

```
:SYST:NET:ETH:MODE?
dhcp_client_fallback

:SYST:NET:ETH:MODE static
```

:SYSTem:NETwork:ETHernet:IPADdress

Query or set the configured Ethernet IP address.

Short form: `:SYST:NET:ETH:IPAD`

Examples:

```
:SYST:NET:ETH:IPAD?
192.168.0.95

:SYST:NET:ETH:IPAD 192.168.1.100
```

:SYSTem:NETwork:ETHernet:SMASk

Query or set the Ethernet subnet mask.

Short form: `:SYST:NET:ETH:SMAS`

Examples:

```
:SYST:NET:ETH:SMAS?
255.255.0.0

:SYST:NET:ETH:SMAS 255.255.255.0
```

:SYSTem:NETwork:ETHernet:GATEway

Query or set the Ethernet default gateway.

Short form: `:SYST:NET:ETH:GATE`

Examples:

```
:SYST:NET:ETH:GATE?
192.168.1.1

:SYST:NET:ETH:GATE 10.0.0.1
```

:SYSTem:NETwork:ETHernet:DHCPTimeout

Query or set the DHCP fallback timeout in seconds. Used in `dhcp client fallback` mode.

Short form: `:SYST:NET:ETH:DHCPT`

Examples:

```
:SYST:NET:ETH:DHCPT?
30

:SYST:NET:ETH:DHCPT 60
```

:SYSTem:NETwork:ETHernet:DHCP:STARt

Query or set the DHCP server pool start address.

Short form: `:SYST:NET:ETH:DHCP:STAR`

Examples:

```
:SYST:NET:ETH:DHCP:STAR?
192.168.0.100

:SYST:NET:ETH:DHCP:STAR 192.168.0.50
```

:SYSTem:NETwork:ETHernet:DHCP:END

Query or set the DHCP server pool end address.

Short form: `:SYST:NET:ETH:DHCP:END`

Examples:

```
:SYST:NET:ETH:DHCP:END?
192.168.0.199

:SYST:NET:ETH:DHCP:END 192.168.0.250
```

:SYSTem:NETwork:ETHernet:DHCP:LEASetime

Query or set the DHCP lease time.

Short form: `:SYST:NET:ETH:DHCP:LEAS`

Examples:

```
:SYST:NET:ETH:DHCP:LEAS?
30s

:SYST:NET:ETH:DHCP:LEAS 1h
```

:SYSTem:NETwork:ETHernet:LIVeip?

Query the live IP address currently active on the Ethernet interface. Read-only.

Short form: `:SYST:NET:ETH:LIV?`

Example:

```
:SYST:NET:ETH:LIV?
192.168.0.95
```

:SYSTem:NETwork:ETHernet:MAC?

Query the Ethernet MAC address. Read-only.

Short form: `:SYST:NET:ETH:MAC?`

Example:

```
:SYST:NET:ETH:MAC?
00:11:22:33:44:55
```

6.4.2 USB-C Gadget Interface

:SYSTem:NETwork:USB:MODE

Query or set the USB-C networking mode.

Short form: `:SYST:NET:USB:MODE`

MODE	DESCRIPTION
`static`	Static IP address
`dhcp host`	Run DHCP server for connected host
`disabled`	Interface is brought down

Examples:

```
:SYST:NET:USB:MODE?
dhcp_host

:SYST:NET:USB:MODE static
```

:SYSTem:NETwork:USB:IPADdress

Query or set the configured USB-C IP address.

Short form: `:SYST:NET:USB:IPAD`

Examples:

```
:SYST:NET:USB:IPAD?
192.168.42.95

:SYST:NET:USB:IPAD 192.168.42.10
```

:SYSTem:NETwork:USB:SMASk

Query or set the USB-C subnet mask.

Short form: `:SYST:NET:USB:SMAS`

Examples:

```
:SYST:NET:USB:SMAS?
255.255.255.0

:SYST:NET:USB:SMAS 255.255.255.0
```

:SYSTem:NETwork:USB:ENABled

Query or set the USB-C interface enabled state.

Short form: `:SYST:NET:USB:ENAB`

Examples:

```
:SYST:NET:USB:ENAB?
1

:SYST:NET:USB:ENAB 0
```

:SYSTem:NETwork:USB:DHCP:STARt

Query or set the USB-C DHCP server pool start address.

Short form: `:SYST:NET:USB:DHCP:STAR`

Examples:

```
:SYST:NET:USB:DHCP:STAR?
192.168.42.100

:SYST:NET:USB:DHCP:STAR 192.168.42.50
```

:SYSTem:NETwork:USB:DHCP:END

Query or set the USB-C DHCP server pool end address.

Short form: `:SYST:NET:USB:DHCP:END`

Examples:

```
:SYST:NET:USB:DHCP:END?
192.168.42.200

:SYST:NET:USB:DHCP:END 192.168.42.250
```

:SYSTem:NETwork:USB:DHCP:LEASetime

Query or set the USB-C DHCP lease time.

Short form: `:SYST:NET:USB:DHCP:LEAS`

Examples:

```
:SYST:NET:USB:DHCP:LEAS?
30s

:SYST:NET:USB:DHCP:LEAS 24h
```

:SYSTem:NETwork:USB:LIVeip?

Query the live IP address currently active on the USB-C interface. Read-only.

Short form: `:SYST:NET:USB:LIV?`

Example:

```
:SYST:NET:USB:LIV?
192.168.42.95
```

:SYSTem:NETwork:USB:MAC?

Query the USB-C MAC address. Read-only.

Short form: `:SYST:NET:USB:MAC?`

Example:

```
:SYST:NET:USB:MAC?
00:11:22:33:44:56
```

6.4.3 Network Reset

:SYSTem:NETwork:RESet

Restore factory default network configuration and apply immediately.

Short form: `:SYST:NET:RES`

Example:

```
:SYST:NET:RES
```

6.4.4 Quick Reference

Ethernet

COMMAND	SHORT FORM	R/W	DESCRIPTION
`:SYSTem:NETwork:ETHernet:MODE`	`:SYST:NET:ETH:MODE`	R/W	Ethernet operating mode
`:SYSTem:NETwork:ETHernet:IPADdress`	`:SYST:NET:ETH:IPAD`	R/W	Configured IP address
`:SYSTem:NETwork:ETHernet:SMASk`	`:SYST:NET:ETH:SMAS`	R/W	Subnet mask
`:SYSTem:NETwork:ETHernet:GATEway`	`:SYST:NET:ETH:GATE`	R/W	Default gateway
`:SYSTem:NETwork:ETHernet:DHCPTimeout`	`:SYST:NET:ETH:DHCPT`	R/W	DHCP fallback timeout (seconds)
`:SYSTem:NETwork:ETHernet:DHCP:STARt`	`:SYST:NET:ETH:DHCP:STAR`	R/W	DHCP pool start
`:SYSTem:NETwork:ETHernet:DHCP:END`	`:SYST:NET:ETH:DHCP:END`	R/W	DHCP pool end
`:SYSTem:NETwork:ETHernet:DHCP:LEASetime`	`:SYST:NET:ETH:DHCP:LEAS`	R/W	DHCP lease time
`:SYSTem:NETwork:ETHernet:LIVeip?`	`:SYST:NET:ETH:LIV?`	R	Live IP address
`:SYSTem:NETwork:ETHernet:MAC?`	`:SYST:NET:ETH:MAC?`	R	MAC address

USB-C

COMMAND	SHORT FORM	R/W	DESCRIPTION
`:SYSTem:NETwork:USB:MODE`	`:SYST:NET:USB:MODE`	R/W	USB-C operating mode
`:SYSTem:NETwork:USB:IPADdress`	`:SYST:NET:USB:IPAD`	R/W	Configured IP address
`:SYSTem:NETwork:USB:SMASk`	`:SYST:NET:USB:SMAS`	R/W	Subnet mask
`:SYSTem:NETwork:USB:ENABled`	`:SYST:NET:USB:ENAB`	R/W	Enabled state (0/1)
`:SYSTem:NETwork:USB:DHCP:STARt`	`:SYST:NET:USB:DHCP:STAR`	R/W	DHCP pool start
`:SYSTem:NETwork:USB:DHCP:END`	`:SYST:NET:USB:DHCP:END`	R/W	DHCP pool end
`:SYSTem:NETwork:USB:DHCP:LEASetime`	`:SYST:NET:USB:DHCP:LEAS`	R/W	DHCP lease time
`:SYSTem:NETwork:USB:LIVeip?`	`:SYST:NET:USB:LIV?`	R	Live IP address
`:SYSTem:NETwork:USB:MAC?`	`:SYST:NET:USB:MAC?`	R	MAC address

6.5 SP4T Switch Commands

The SP4T routes a common port (`COM`) to one of four throws (`RF0`..`RF3`). Only one throw is closed at a time; states are zero-based and there is no "all open" state.

6.5.1 SP4T:MAIN:STATE?

Query the currently routed throw.

Response: `0` to `3`

VALUE	DESCRIPTION
`0`	`COM` ↔ `RF0`
`1`	`COM` ↔ `RF1`
`2`	`COM` ↔ `RF2`
`3`	`COM` ↔ `RF3`

Example:

```
SP4T:MAIN:STATE?
2
```

6.5.2 SP4T:MAIN:STATE \<value>

Set the routed throw.

Parameters:

VALUE	DESCRIPTION
`0`	Route `COM` to `RF0`
`1`	Route `COM` to `RF1`
`2`	Route `COM` to `RF2`
`3`	Route `COM` to `RF3`

Example:

```
SP4T:MAIN:STATE 2
```

Errors:

- `-222` — Data out of range; expected 0 to 3
- `-104` / `-109` / `-108` — Non-numeric, missing, or extra parameter
- `-200` — Switch write failed

6.5.3 Short Forms and Aliases

Both mnemonics are all-caps, so `SP4T` has no abbreviated form:

COMMAND	SHORT FORM
`SP4T:MAIN:STATE?`	—
`SP4T:MAIN:STATE`	—
`SP4T:MAIN?`	—
`SP4T:MAIN`	—

The legacy `SWITch:MAIN[:STATe]` alias is also registered on SP4T hardware and keeps its documented `SWIT:MAIN:STAT` short form. It carries the SP4T valid range (0–3).

6.5.4 *RST Behaviour

`*RST` returns the switch to its power-on/reset state, **RF0** (both select lines low). Note the asymmetry with the SPDT variant, where `*RST` restores RF1 — see common-commands.md.

6.5.5 Common (inherited)

`*IDN?`, `*RST`, `*CLS` per common-commands.md.

`*IDN?` response:

```
Nine Fives,POE-SP4T,<serial>,<version>
```