Devices

Robosoft

Robonode quick start guide

v1.1.1-r1808

Notice

8devices reserves the right to change specifications without prior notice. While the information in this manual has been compiled with great care, it may not be deemed an assurance of product characteristics. 8devices shall be liable only to the degree specified in the terms of sale and delivery. The reproduction and distribution of the documentation and software supplied with this product and the use of its contents are subject to written authorization from 8devices.

Trademarks

8devices logo is a trademark of 8devices. All other registered and unregistered trademarks in this document are the sole property of their respective owners.

Table Of Contents

1. Robonode Board

1.1 Overview

Robonode is a development board for unmanned systems based on the TobuFi System-on-Module featuring Qualcomm QCS405 SoC. This document describes board-level hardware integration and peripherals.

1.2 Peripherals

- WiFi Radio #0 (HE/11ax)
- WiFi Radio #1 (VHT/11ac)
- Ethernet port (100M/1000M)
- USB Port #1 (USB 2.0 DRD/OTG)
- USB Port #2 (USB 3.0 host)
- I2C EEPROM (16KB)
- 4x BLSP UART/I2C
- 3x GPIO LEDs
- GPIO Push button
- GPIO Switch toggle
- Reboot button
- USB recovery button

1.3 Board Layout

1.3.1 Top View

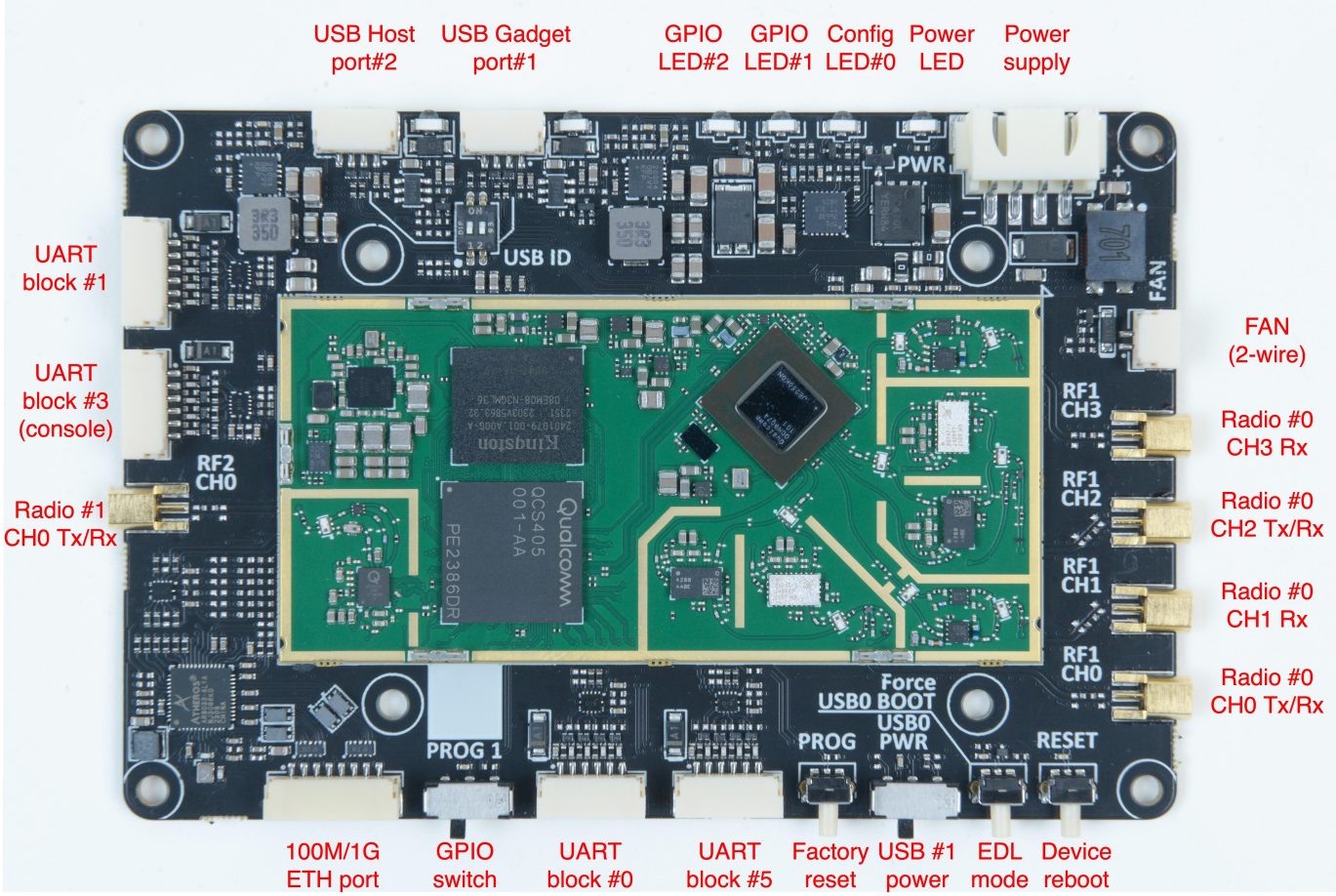

Figure 1: Robonode board top view components

1.3.2 Bottom View

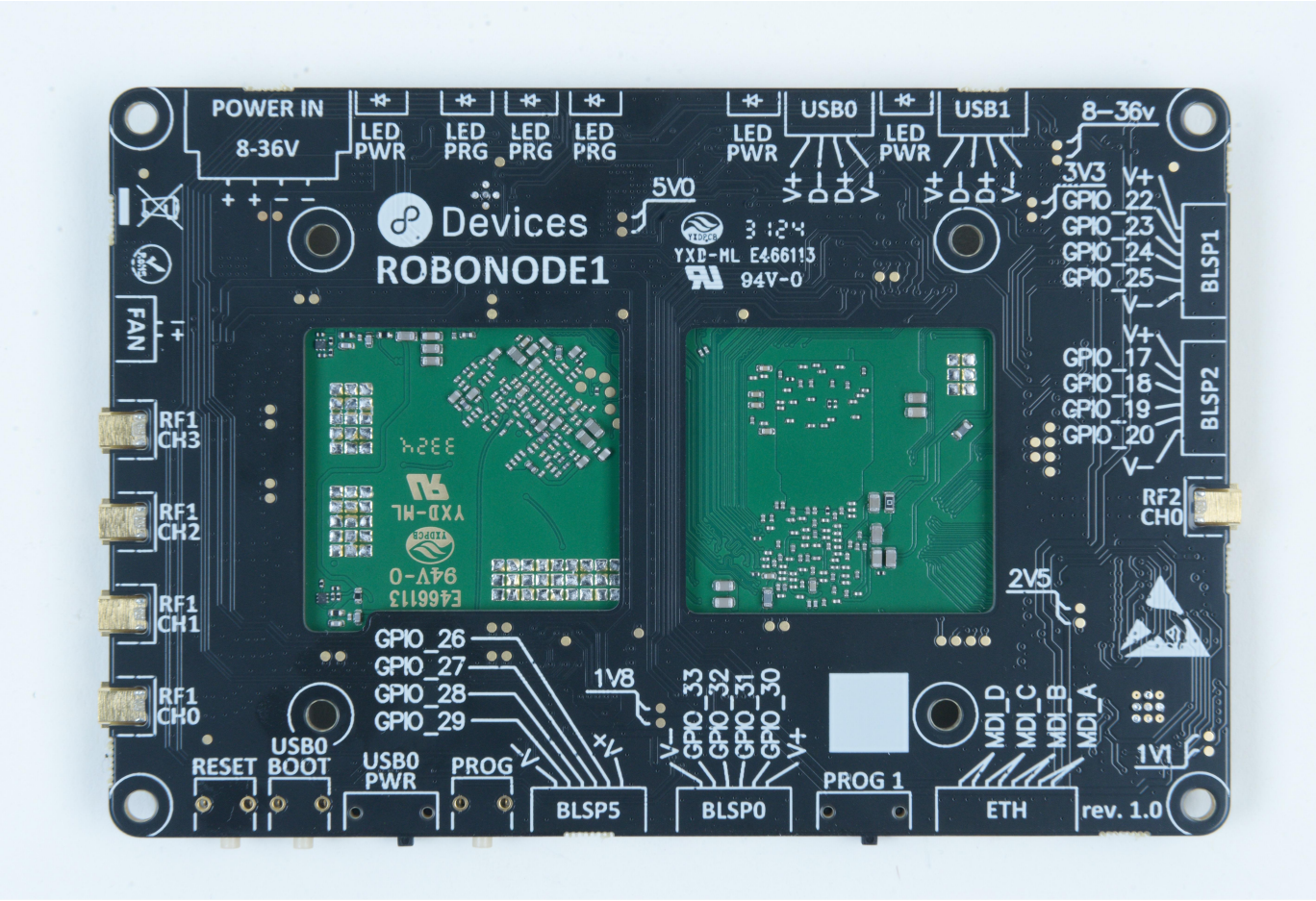

Figure 2: Robonode board bottom view components

1.3.3 Connector Identification

Top Side:

- **Ethernet Port**: RJ45 connector, labeled "ETH"
- **USB Port #1**: USB-C connector (left side)
- **USB Port #2**: USB-C connector (right side)
- **WiFi Antennas**: 4x MMCX (Radio #0), 1x MMCX (Radio #1)
- **Power Connector**: DC barrel jack (8-36V)

Bottom Side:

- **BLSP0 Header**: 4-pin header for UART/I2C (GPIOs 30-33)
- **BLSP1 Header**: 4-pin header for UART/I2C (GPIOs 22-25)
- **BLSP2 Header**: 4-pin header for UART/I2C (GPIOs 17-20), includes serial console
- **BLSP5 Header**: 4-pin header for UART/I2C (GPIOs 26-29)
- **LED0, LED1, LED2**: User-defined GPIO LEDs
- **Push Button**: User-defined GPIO button
- **Slide Switch**: User-defined GPIO switch
- **Reboot Button**: System reset button
- **Recovery Button**: EDL mode entry button

1.4 Board Interfaces

1.4.1 I2C EEPROM

Chip: AT24C128C **Capacity:** 16KB **Interface:** BLSP1 I2C (GPIO 24-25)

Non-volatile storage for board identification and production storage. Accessible through standard Linux I2C interface.

1.4.2 WiFi Radio #0 (Pine)

Parameter	Description
Radio Chip	PCIe/QCN9074
WiFi Standard	IEEE 802.11ax (WiFi 6/6E)
Channel Width	20/40/80/160 MHz
Operation Band	2GHz + 5GHz or 2GHz + 6GHz
Operation Mode	2x4 (2T4R)
Antenna Connectors	4x MMCX
Max Conducted 2GHz Tx Power (aggregate)	32 dBm
Max Conducted 5GHz Tx power (aggregate)	29 dBm
Max Conducted 6GHz Tx Power (aggregate)	29 dBm
2GHz Frequency Range	2360–3150 MHz
5GHz Frequency Range	4550–6630 MHz
6GHz Frequency Range	5325–7495 MHz

High-performance WiFi radio supporting WiFi 6E (802.11ax) with MIMO capability. Suitable for high-throughput wireless applications, mesh networking, and dual-band simultaneous operation.

1.4.3 WiFi Radio #1

Parameter	Description
Radio Chip	SNOC/WCN3980
WiFi Standard	IEEE 802.11ac (WiFi 5)
Channel Width	20/40/80 MHz
Operation Band	2GHz + 5GHz
Operation Mode	1x1 (1T1R)
Antenna Connectors	1x MMCX
Max 2GHz Tx Power	16 dBm
Max 5GHz Tx power	16 dBm
2.4 GHz Frequency Range	2412-2484 MHz
5 GHz Frequency Range	5180-5825 MHz

Integrated WiFi radio supporting dual-band operation. Suitable for management interface, station mode connectivity, or additional access point deployment.

1.4.4 Ethernet Port

Parameter	Description
Trancever Chip	RGMII/AR8033A
Speed	10/100/1000 Mbps
Duplex	Half/Full
MDI/MDI-X	Auto-crossover

Gigabit Ethernet port with RJ45 connector. Supports network connectivity for wired applications, device management, and high-bandwidth data transfer.

1.4.5 USB Port #1

Parameter	Description
Standard	USB 2.0
Speed	480 Mbps (HS)
Connector	USB-C
Mode	DRD (Host/Gadget/OTG)
Power	Switch-controlled
Identifier	usb0

Dual-role port capable of host, device and OTG modes with manual power switch control. In device mode, provides access to ADB debugging interface for development and configuration. In host mode, supports standard USB peripherals.

USB Port #1 Power Switch

Slide switch controls USB mode and power supply. When toggled on USB port is working in host controller mode, and port is powered locally, when toggled off USB port is working in device mode and it expects to be powered from the host device.

Enable USB power when desirable to use port to connect additional periphery, i.e. USB camera. Keep USB power disabled to keep port in device mode, i.e. for ADB access, USB storage or USB ethernet adapter mode.

1.4.6 USB Port #2

Parameter	Description
Standard	USB 2.0
Speed	480 Mbps (HS)
Connector	USB-C
Mode	Host only
Power	Always powered
Identifier	usb1

Host-only port for connecting USB flash drives, USB cameras, and other standard USB peripherals like mice and keyboards.

1.4.7 LED #0

GPIO: 47

Application controlled LED dedicated for system configuration state feedback. LED is fully controlled by application software therefore may be repurposed on demand, changing application source code and rebuilding software.

1.4.8 LED #1

GPIO: 48

Software controlled LED. Currently unused, dedicated for future application needs.

1.4.9 LED #2

GPIO: 52

Software controlled LED, currently unused, dedicated for future application needs.

For LED control via sysfs and device tree configuration, see GPIO Configuration.

1.4.10 GPIO Push Button

GPIO: 58

Software controlled push button. Pressing and holding the button for at least 10 seconds triggers factory reset, resetting device configuration to factory defaults. This is also indicated by the configuration LED slowly blinking while the button is being held.

1.4.11 GPIO Switch Toggle

GPIO: 51

Software controlled slide switch, currently unused, dedicated for future application needs.

For button and switch programming via Linux input subsystem, see GPIO Configuration.

1.4.12 Reboot Button

Pressing and releasing the button restarts the device.

1.4.13 USB Recovery Button

Hold the button pressed when power cycling or rebooting to enter EDL (Emergency Download) mode for software recovery or initial device flashing.

1.5 Related Documentation

- QCS405 SoC - QCS405 SoC (System-on-Chip) details
- TobuFi SoM - TobuFi SoM (System-on-Module) details
- Robonode GPIO - Board GPIO interfaces and control
- Robonode BLSP - Board UART and I2C interfaces
- Robonode Initialization - Board initialization sequence

2. Robonode Quick Start

2.1 Recipe 1: Power On & First Connection

Goal: Power on the board and establish first access.

2.1.1 Power

- Supply: **8–36V DC, ~24W peak**.
- Board **auto-starts** on power.

See Robonode Board for connector pinout.

2.1.2 Serial Console (UART)

Connect a USB-to-serial (3.3V TTL) adapter to the BLSP2 (console) connector on the side of the board — see Board Layout.

On the host:

```
picocom -b 115200 /dev/ttyUSB0
```

Settings: 115200 baud, 8N1. Login: `root`, no password.

2.1.3 SSH

```
ssh root@192.168.2.1
```

2.1.4 ADB (USB)

Connect a USB-C cable from the host to **USB Port #1** (USB 2.0) on the board top side — see Board Layout.

On the host:

```
adb devices
adb shell
```

2.1.5 Default Network

- Device IP: `192.168.2.1`

- Web GUI: `http://192.168.2.1`

2.2 Recipe 2: Display Board and Software Info

Goal: Check board identity and installed software version.

2.2.1 Steps

```
boardinfo
swinfo
```

2.2.2 Example `boardinfo` output

```
Board name: Robonode rev2.0
Radio ID: Premium
Radio features: 2-6GHz 2x4
Serial Number: 601821d6
```

2.2.3 Example `swinfo` output

```
Board name: Robonode rev2.0
Radio ID: Premium
Radio features: 2-5GHz 2x4
Serial Number: 601821d6
SW partition: A
SW Version: 1.1.0
Image: 8dev-robo-image
Distro: 8dev-drone
Machine: robonode
Build hashes:
  meta-8dev: 48009c1
  meta-8dev-premium: 53040f4
```

3. Quick Start Recipes

Cookbook-style recipes for common tasks. Each recipe has a **Goal**, **Steps**, and **Validate** section.

3.1 Recipe 3: Update Software (OTA)

Goal: Update the device firmware using an SWU update image.

The unified Robosoft image is applicable for both TobuFi-DVK and Robonode boards.

3.1.1 Steps

- Transfer the image to the device:

```
scp update-image.swu root@192.168.2.1:/tmp/
```

- Apply the update and reboot:

```
update /tmp/update-image.swu
```

The device reboots automatically after a successful update.

3.1.2 Validate

```
swinfo
```

Check that `SW Version` matches the expected version.

!!! note The device uses A/B partitions. A failed update triggers automatic rollback after 7 unsuccessful boots.

3.2 Recipe 4: Flash Software via USB (Fastboot)

Goal: Flash boot and rootfs images via USB.

3.2.1 Steps

- Connect the USB port #1 (one with ADB function) to the host PC.

- Flash the images:

```
python3 fastboot_flash.py --boot boot-image.img --system rootfs-image.img
```

For multiple devices, list them first:

```
fastboot devices
python3 fastboot_flash.py <serial> --boot boot-image.img --system rootfs-image.img
```

3.2.2 Validate

After reboot:

```
swinfo
```

3.3 Recipe 5: Upgrade from Legacy v0.x to v1.x

Goal: Upgrade from legacy v0.x firmware to v1.x.

3.3.1 Steps

- Extract the v1.x flash update package.
- Connect the USB gadget port to the host.
- Run from the extracted package directory:

```
python3 fastboot_flash.py
```

3.3.2 Validate

```
swinfo
```

Confirm `SW Version` shows a v1.x version.

3.4 Recipe 6: Downgrade to Legacy v0.x

Goal: Restore device to legacy v0.x firmware.

3.4.1 Steps

- Extract the legacy v0.x flash update package.
- Connect the USB gadget port to the host.
- Run from the extracted package directory:

```
python3 fastboot_flash.py -a .
```

3.4.2 Validate

Device runs v0.x after reboot.

3.5 Recipe 7: Collect Diagnostic Data

Goal: Collect a diagnostic archive for troubleshooting or support.

3.5.1 Steps

On the device:

```
trouble
```

Creates `/tmp/<hostname>-<MAC>-<timestamp>.zip`.

Transfer to your host:

```
scp root@192.168.2.1:/tmp/\*.zip .
```

The archive includes: system logs, kernel messages, network config, service status, process list, and config files.

3.6 Recipe 8: Remote WiFi Monitoring with Wireshark

Goal: Capture WiFi packets on the device and view them live in Wireshark on the host.

3.6.1 Step 1 — Enable monitor interface

Add to `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    mon:
      enabled: true
```

Apply:

```
config-validate && esconf reload
```

3.6.2 Option A — Piped tcpdump

```
ssh root@192.168.2.1 "tcpdump -i wlan0.mon -U -w -" | wireshark -k -i -
```

3.6.3 Option B — Wireshark sshdump (GUI)

In Wireshark: **Capture > Options > Manage Interfaces > SSH remote capture (sshdump)**:

Setting	Value
Remote SSH server address	`192.168.2.1`
Remote SSH server port	`22`
Remote username	`root`
Remote interface	`wlan0.mon`
Remote capture command	`tcpdump`

Click **Start**.

3.7 Recipe 9: Configuration Basics

Goal: Understand the standard 3-step workflow used in all configuration recipes.

3.7.1 Steps

```
vi /etc/roboconf/config.yaml    # 1. Edit
config-validate                  # 2. Validate
esconf reload                    # 3. Apply
```

Config file: `/etc/roboconf/config.yaml`. Must contain `version: "1.1"`.

See Configuration Parameters for the full reference.

3.8 Recipe 10: Change IP Address

Goal: Change the device IP address.

3.8.1 Steps

Edit `/etc/roboconf/config.yaml`:

```
network:
  zones:
    mgmt:
      mode: "static"
      address: 192.168.10.1
```

Apply: `config-validate && esconf reload`

3.8.2 Validate

```
ip addr show
```

3.9 Recipe 11: Set Up a BSS Link (AP + Station)

Goal: Establish a WiFi link between two devices — one as AP, the other as Station.

3.9.1 AP device

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    mode: "BSS"
    ap:
      ssid: "MyNetwork"
      passphrase: "mypassword"
      enabled: true
      zone: "data"
    sta:
      enabled: false
```

3.9.2 Station device

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    mode: "BSS"
    ap:
      enabled: false
    sta:
      ssid: "MyNetwork"
      passphrase: "mypassword"
      enabled: true
      zone: "data"
```

Apply on both devices: `config-validate && esconf reload`

3.9.3 Validate

- AP: `iw dev wlan0 station dump` — shows connected stations.
- Station: `iw dev wlan0 link` — shows connected SSID and signal.
- Both: `ping <other-device-ip>`

!!! note Radio parameters (`channel`, `width`, `txpower`, etc.) retain defaults. See Configuration Parameters.

3.10 Recipe 12: Set Up a NAW Link

Goal: Establish a Non-Associated WiFi link between two devices.

3.10.1 Steps — On both devices (identical config)

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    mode: "NAW"
    channel: 36
    naw:
      link_id: "MyNAWLink"
      passphrase: "nawpassword"
      enabled: true
      zone: "data"
```

Apply on both devices: `config-validate && esconf reload`

!!! note NAW requires a fixed channel or frequency. Pine Radio (`radio0`) only. Both sides must match `channel`/`frequency`, `link_id`, and `passphrase`.

3.10.2 Validate

- `iw dev wlan0 link` — shows link state.
- `ping <other-device-ip>`

3.11 Recipe 13: Set Up a WFB Link

Goal: Establish a unidirectional WiFi Broadcast link using IP socket-based forwarding.

Both devices must use the same `channel`, `width`, and `stream_id`.

3.11.1 TX device

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    mode: "WFB"
    channel: 36
    width: 20
wfb:
  streams:
    tx:
      enabled: true
      mode: "tx"
      listen_port: 5600
```

```
      stream_id: 1
    rx:
      enabled: false
```

3.11.2 RX device

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    mode: "WFB"
    channel: 36
    width: 20
wfb:
  streams:
    tx:
      enabled: false
    rx:
      enabled: true
      mode: "rx"
      forward_addr: 127.0.0.1
      forward_port: 5600
      stream_id: 1
```

Apply on both devices: `config-validate && esconf reload`

3.11.3 Validate

- `iw dev` — look for `wlan0.mon` type monitor.
- `systemctl status wifibroadcast@tx` / `systemctl status wifibroadcast@rx`
- Test with `wfb_test` — see Recipe 14.

3.12 Recipe 14: Validate WFB Link

Goal: Verify WFB link end-to-end using `wfb_test`.

3.12.1 Steps

- On the **RX device**:

```
wfb_test -r 5600 -n 10
```

- On the **TX device**:

```
wfb_test -t 5600 -n 10 -d 100
```

TX sends 10 packets, 100 ms apart. RX reports received/lost.

3.12.2 Validate

RX shows `Received data` for each packet. `Lost data` indicates packet loss.

3.13 Recipe 15: Optimize WFB Transmission

Goal: Maximize WFB throughput by disabling CCA and enabling TX burst. WFB unidirectional mode only.

3.13.1 Steps

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    cca_threshold: 0    # Disable CCA (0 = off)
    tx_burst: true       # Enable WMM Tx burst
```

Apply: `config-validate && esconf reload`

- **CCA disable**: radio transmits without checking channel. Use only in WFB mode — breaks BSS/NAW.
- **TX burst**: burst transmission on monitor interface. Only effective in WFB mode.

3.13.2 Validate

Run `iperf` throughput test across the WFB link:

On RX device:

```
iperf -u -s -p 5600
```

On TX device:

```
iperf -u -c 127.0.0.1 -p 5600 -b 50M -t 10
```

Compare throughput before and after.

3.14 Recipe 16: Change Band

Goal: Switch Pine Radio between 2 GHz and 5 GHz band.

The radio band is determined automatically from the configured channel or frequency. To switch band, set a channel or frequency in the target band.

3.14.1 Switch to 2.4 GHz

```
wifi:
  radio0:
    channel: 6
    frequency: 0
```

3.14.2 Switch to 5 GHz

```
wifi:
  radio0:
    channel: 36
    frequency: 0
```

Apply: `config-validate && esconf reload`

!!! note Band switching requires a driver reload which happens automatically during `esconf reload`. All WiFi connections are briefly interrupted.

3.14.3 Validate

```
wifistats
```

Check that the operating frequency is in the target band.

3.15 Recipe 17: Change Channel

Goal: Switch WiFi to a specific standard channel.

3.15.1 Steps

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    channel: 36
    frequency: 0          # 0 = use channel
```

Apply: `config-validate && esconf reload`

3.15.2 Validate

```
iw dev wlan0 info | grep channel
```

3.16 Recipe 18: Set Custom Frequency

Goal: Tune Pine Radio to a non-standard frequency.

3.16.1 Steps

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    channel: 0            # 0 = use frequency
    frequency: 4987        # MHz
```

Apply: `config-validate && esconf reload`

3.16.2 Validate

```
wifistats
```

Shows operating frequency.

!!! note Custom frequencies outside standard bands are not reported by the `iw` command; use `wifistats`.

3.17 Recipe 19: Set Narrow Channel Width

Goal: Use narrow bandwidth (5 or 10 MHz) for long-range or interference avoidance.

3.17.1 Steps

Edit `/etc/roboconf/config.yaml`:

```
wifi:
  radio0:
    width: 10            # 5, 10, 20, 40, 80, 160 MHz
```

Apply: `config-validate && esconf reload`

!!! note 5 and 10 MHz widths are available on Pine Radio (`radio0`) only. Narrow channel widths are not reported by the `iw` command; use `wifistats`.

3.17.2 Validate

```
wifistats
```

3.18 Recipe 20: UART Access via Network

Goal: Access a board serial port over TCP from a remote host.

The default config enables TCP access on ports 3001–3003 for UART0–UART2.

3.18.1 Steps

Config (customise if needed) in `/etc/roboconf/config.yaml`:

```
uart:
  ttyMSM1:
    access_protocol: "tcp"
    access_port: 3001
    serial_baud: 115200
    serial_frame: "N81"
```

Apply: `config-validate && esconf reload`

3.18.2 Validate

From host:

```
nc <device-ip> 3001
```

Interactive serial access to the UART.

3.19 Recipe 21: Stream UART as UDP

Goal: Stream serial port data as UDP to a remote destination.

3.19.1 Steps

Edit `/etc/roboconf/config.yaml`:

```
uart:
  ttyMSM1:
    access_protocol: "udp"
    access_port: 3001
    serial_baud: 115200
    serial_frame: "N81"
    udp_stream: true
    udp_stream_addr: 192.168.1.100
    udp_stream_port: 4000
```

Apply: `config-validate && esconf reload`

3.19.2 Validate

On receiving host:

```
nc -ul 4000
```

Serial data appears as it arrives.

3.20 Recipe 22: Enable USB Ethernet (RNDIS)

Goal: Access device over USB cable as a network connection.

3.20.1 Steps

Edit `/etc/roboconf/config.yaml`:

```
usb:
  usb0:
    eth-rndis: true
```

Apply: `config-validate && esconf reload`

3.20.2 Validate

Host sees a new network interface. Device is reachable at `192.168.2.1`.

3.21 Recipe 23: Disable ADB

Goal: Disable ADB for production deployment.

3.21.1 Steps

Edit `/etc/roboconf/config.yaml`:

```
usb:
  usb0:
    adb: false
```

Apply: `config-validate && esconf reload`

3.21.2 Validate

```
adb devices
```

The device no longer appears in the list.