Devices

TobuFi Platform & Robotics Software

User guide

v0.4.1-r1247

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. Supported hardware

This document provides description for 8devices produced modules and development kits supported in this BSP layer.

1.1 TobuFi DVK

Supported hardware revisions: rev3, rev4, rev5.

Note: Rev3 includes EP92A6 HDMI bridge, which was removed in rev4 and later. Rev4 and later added an additional UART port (BLSP0 / UART block A0).

1.1.1 Peripherals

- 100M/1000M ETH port
- USB2 gadget port #1
- USB3 DRD OTG port #2
- 4x UART
- SD card
- NOR flash
- Coax audio in/out
- Optical audio out
- HDMI port
- MIPI display
- Power LED (fixed)
- Reset button (fixed)
- Boot configuration switches

1.1.2 Wi-Fi Radio #1

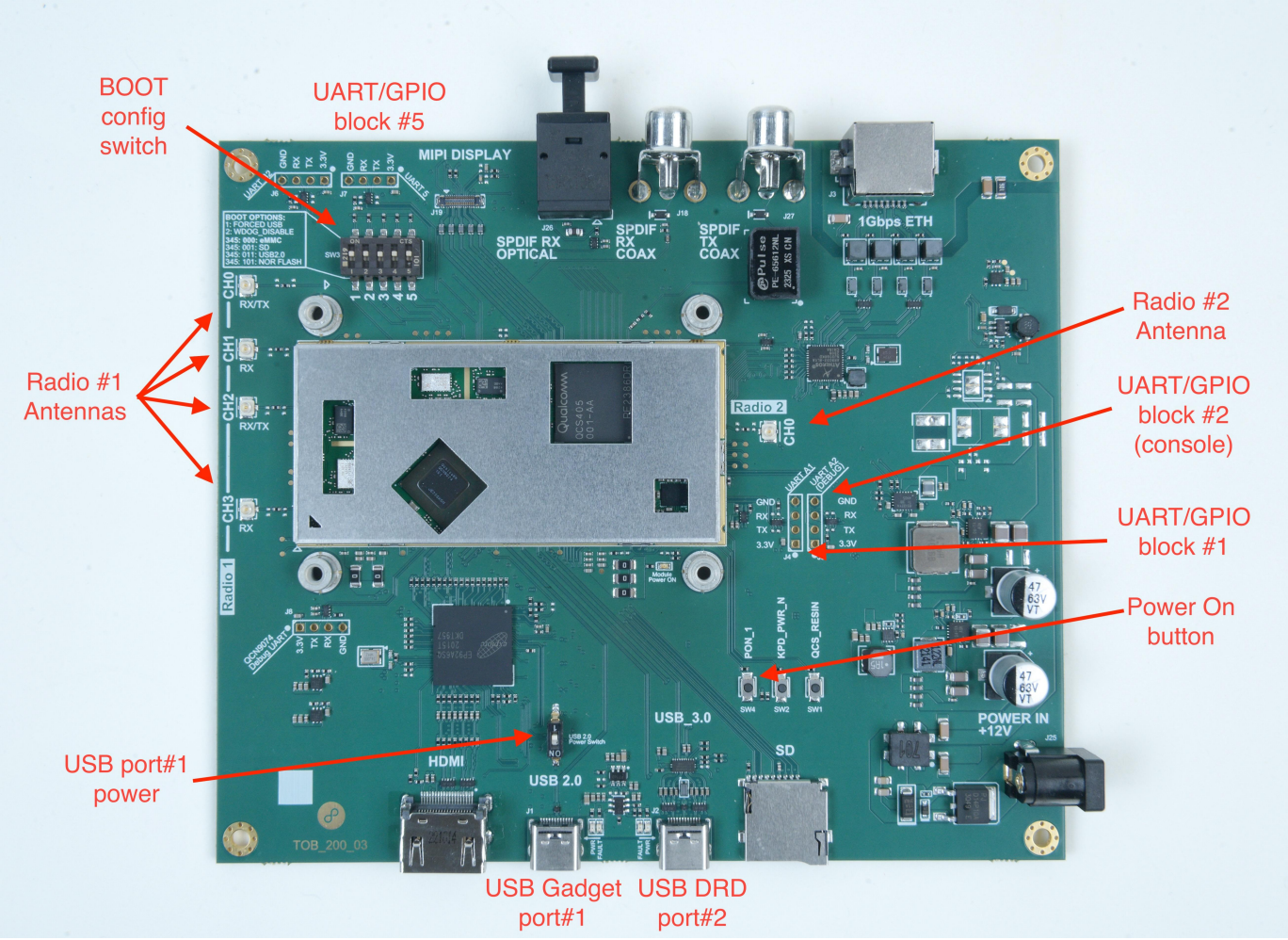
Parameter	Description
Operation band	2GHz + 5GHz
Operation mode	11ax 2x4
Antenna connectors	4x U.FL

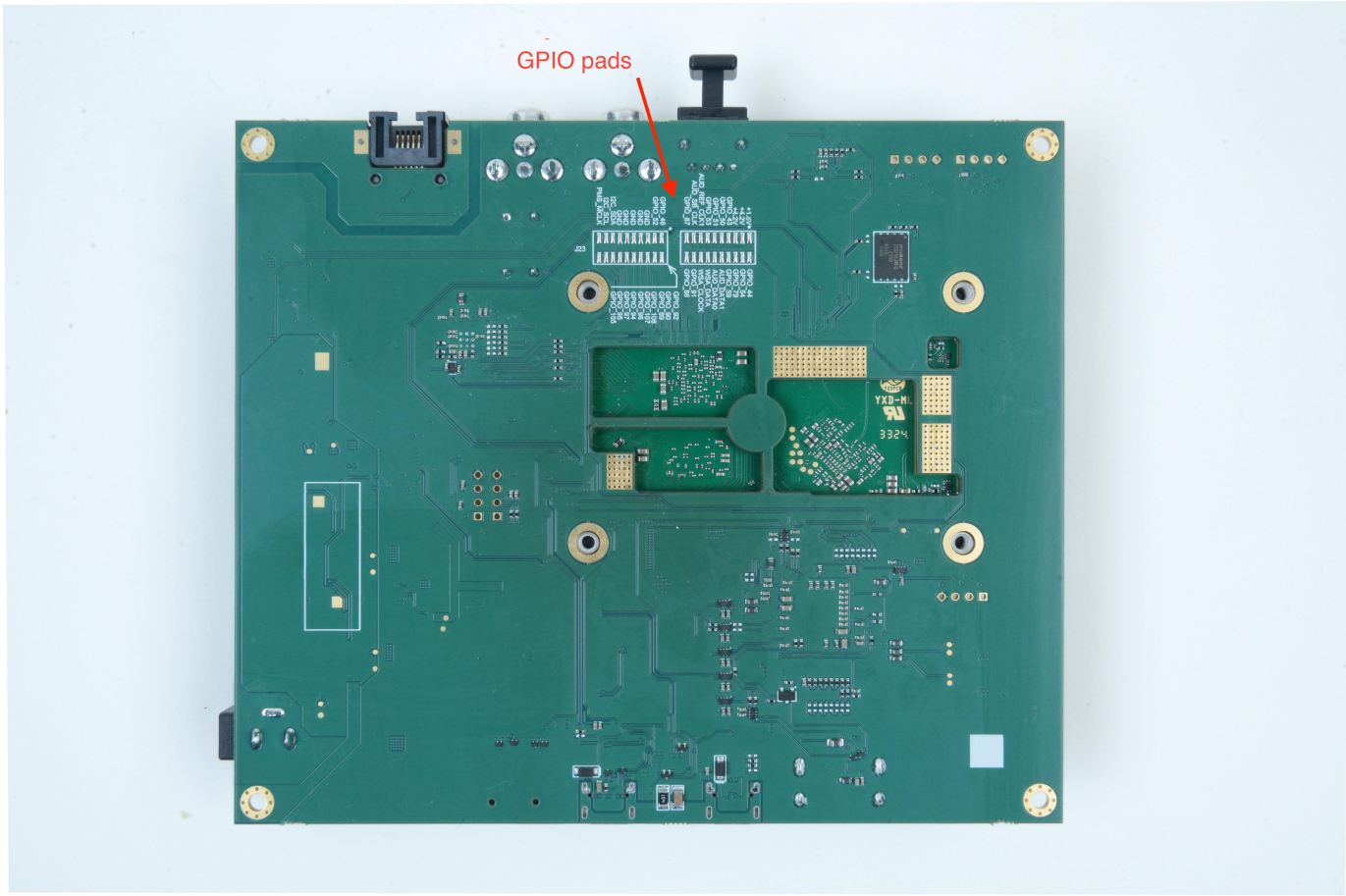
1.1.3 Wi-Fi Radio #2

Parameter	Description
Operation band	2GHz + 5GHz
Operation mode	11ac 1x1
Antenna connectors	1x U.FL

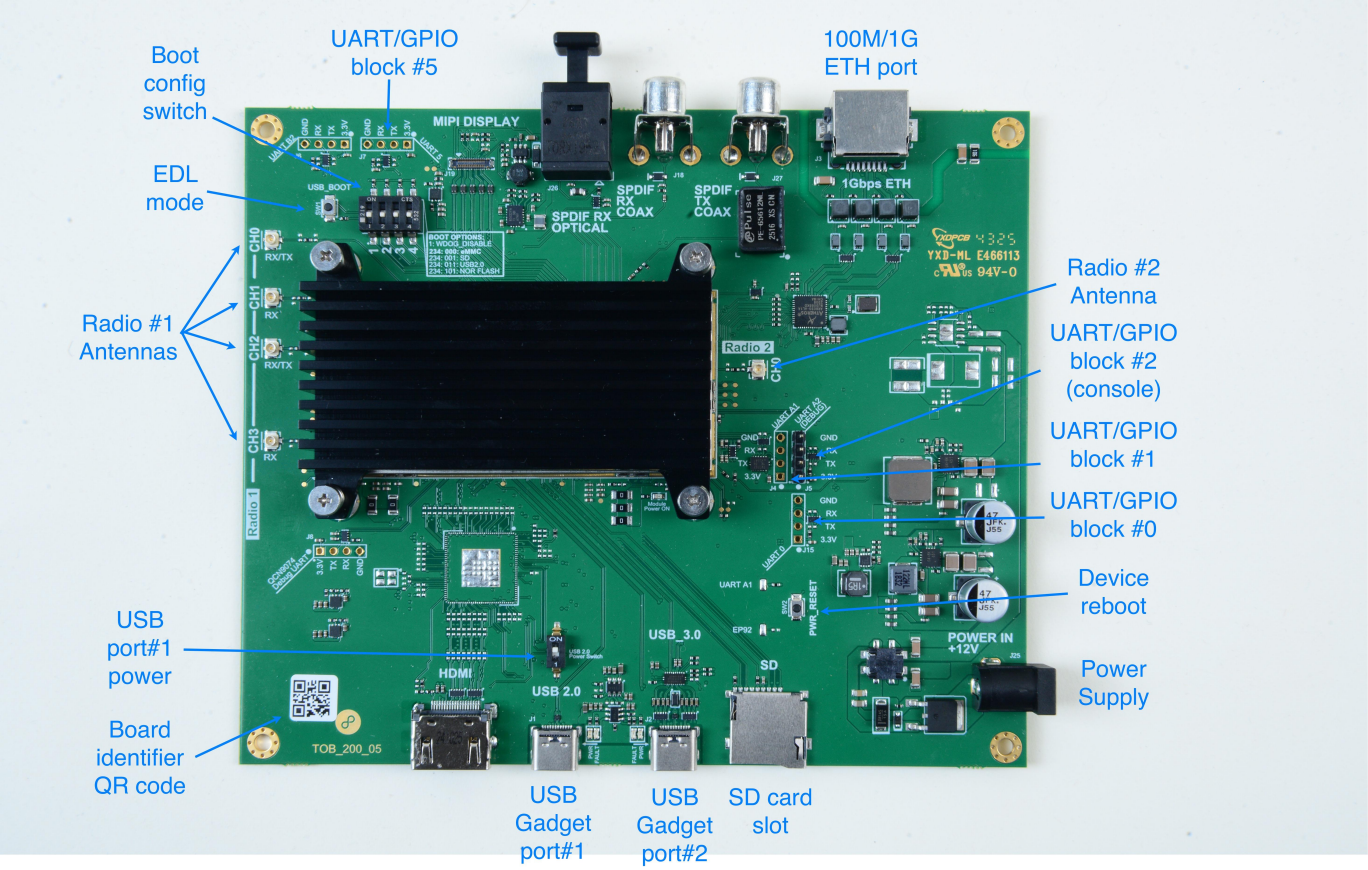
1.1.4 Board

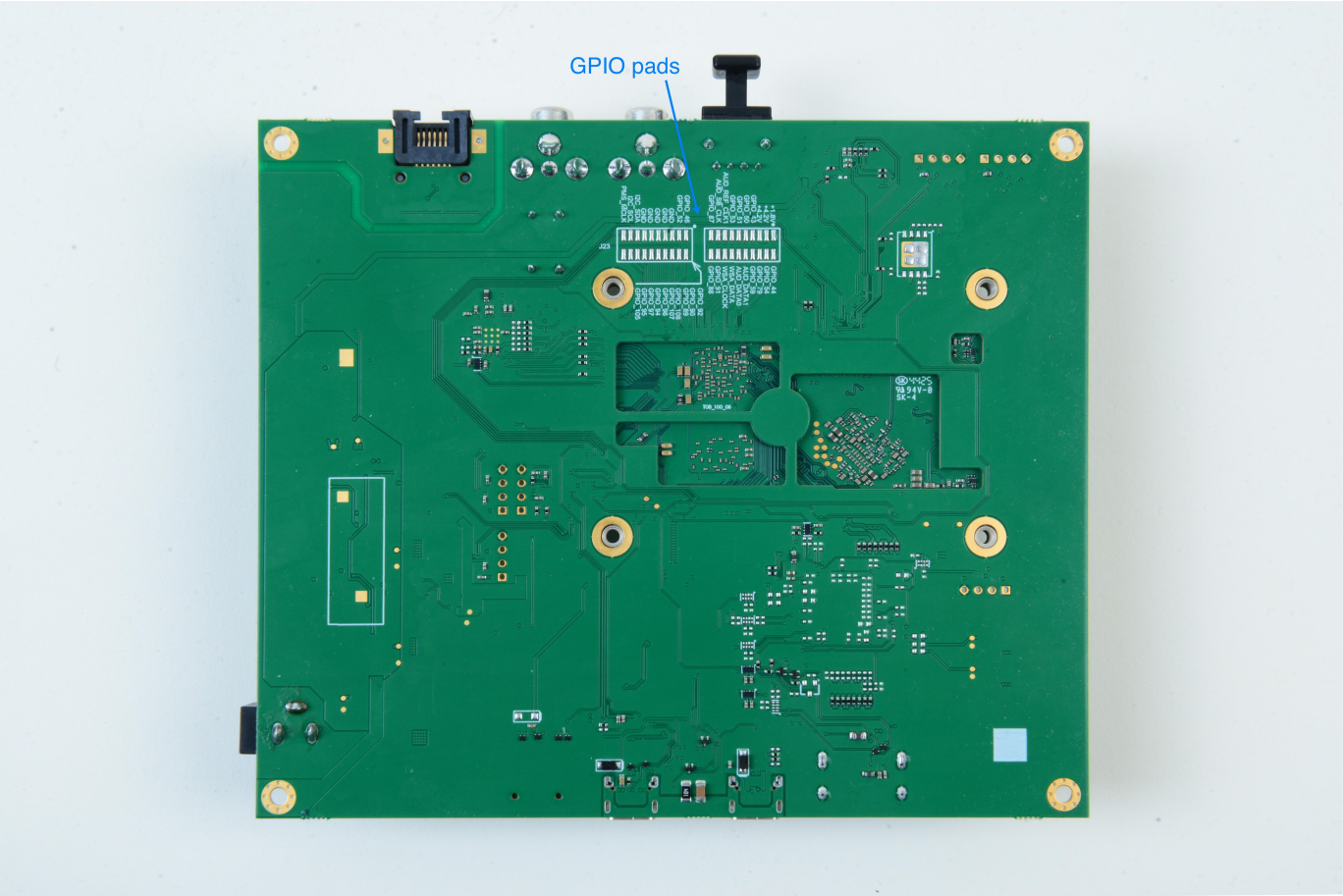
Rev3/Rev4





Rev5





USB DRD Port #2

USB dual-role (DRD) on-the-go (OTG) port supports automatic runtime switching between host and device modes. In host mode, it supports connecting USB flash drives, USB cameras and other standard USB peripherals like mice and keyboards. In device mode, it functions as a USB gadget port that can be configured with various functions in software. For details on configuring USB gadget functions, refer to USB Gadget setup.

USB Gadget Port #1

USB gadget port should be connected to a host PC using a standard USB cable for device access and configuration purposes. ADB debugging interface is available through this port.

USB PORT #1 POWER SWITCH

Slide switch controls USB power supply for USB gadget port. It should be enabled when USB port is configured as host controller. When USB port is working in gadget mode, switch should be disabled to save power.

Power ON and Reset

Rev3/Rev4: Device requires pressing the "Power ON" button after applying power. The PON button is used to start the device.

Rev5: Device starts automatically when power is applied (no button press required). The PON button is repurposed as a hard device reset.

Boot config (rev3/rev4)

Boot configuration DIP switches allow switching the boot devices order. The following combinations are supported:

Mode	SW #1	SW #2	SW #3	SW #4	SW #5
EDL recovery	ON	-	-	-	-
eMMC -> SD -> USB2	OFF	-	OFF	OFF	OFF
SD -> eMMC -> EDL	OFF	-	ON	OFF	OFF
eMMC -> EDL	OFF	-	OFF	ON	OFF
USB2	OFF	-	ON	ON	OFF
NAND -> EDL	OFF	-	OFF	OFF	ON
NOR -> EDL	OFF	-	ON	OFF	ON

Dashes (-) indicate that switch position is not relevant.

EDL mode entry (rev5)

Rev5 uses a dedicated EDL mode push button instead of DIP switch #1. The EDL entry procedure is the same as Robonode EDL recovery.

1.2 Robonode

1.2.1 Peripherals

- 100M/1000M ETH port
- USB Gadget port #1
- USB Host port #2

- 4x UART
- 3x LEDs
- Push button
- Slide button
- USB boot (EDL mode) button

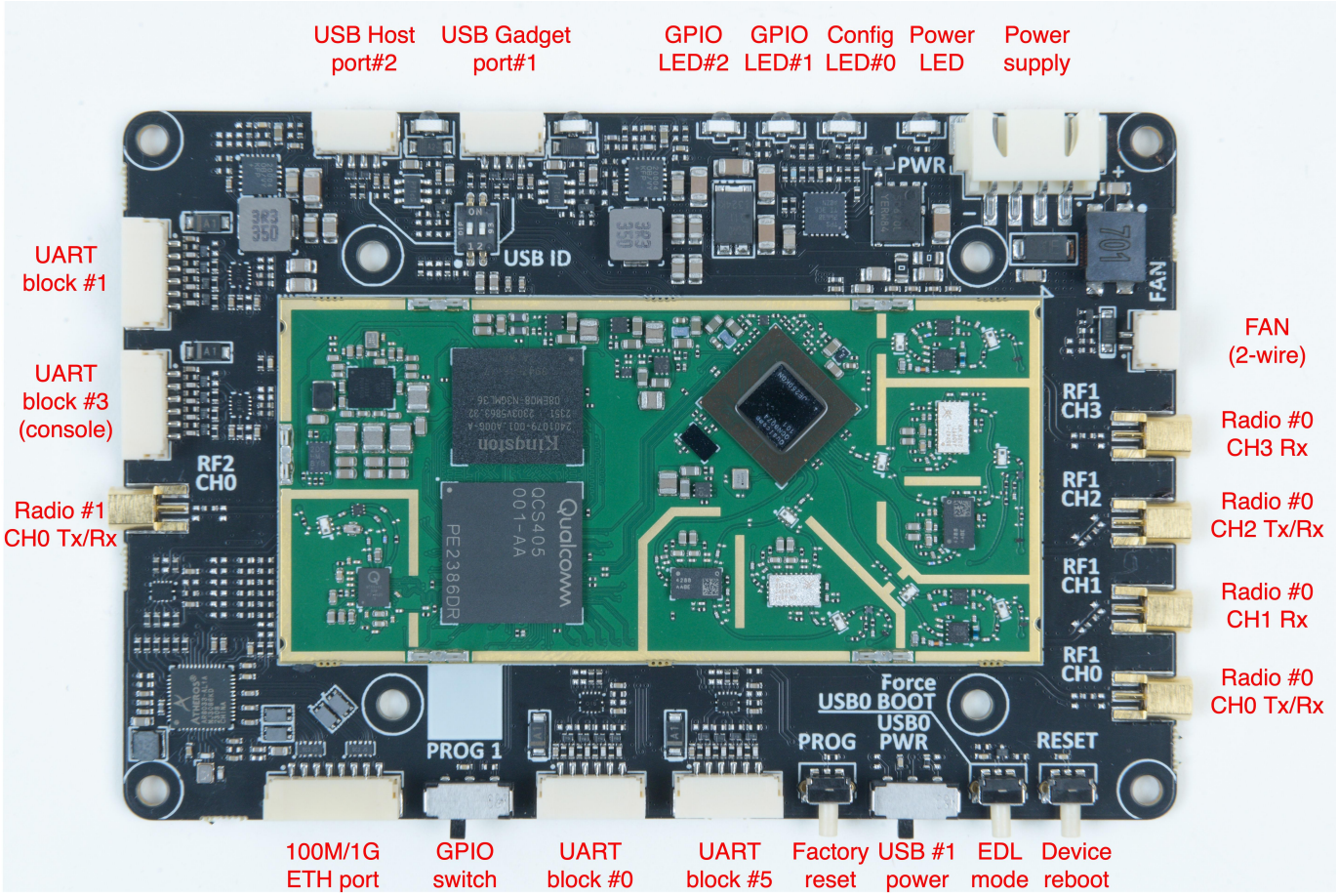
1.2.2 Wi-Fi Radio #0

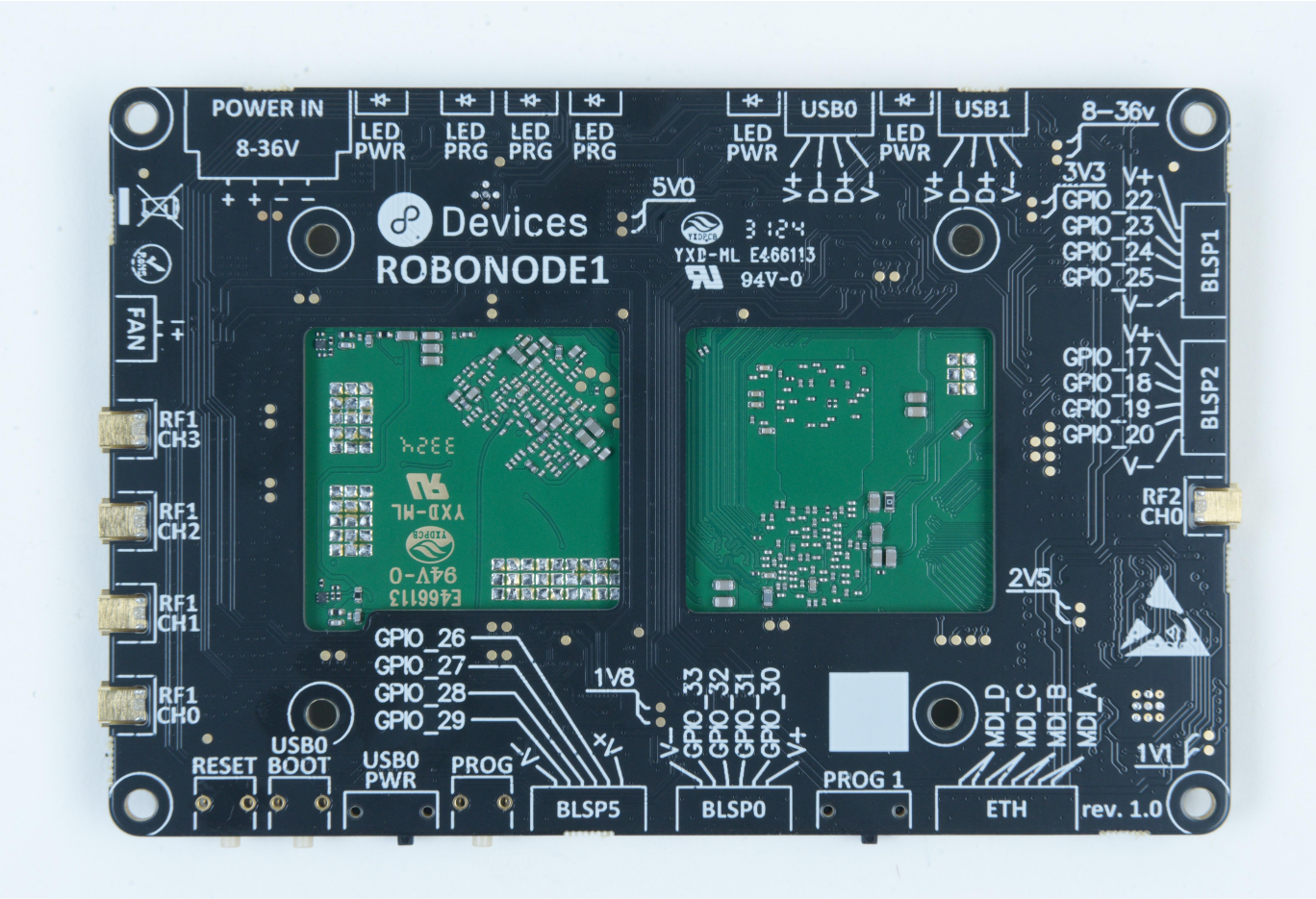
Parameter	Description
Operation band	2GHz + 5GHz or 2GHz + 6GHz
Operation mode	11ax 2x4
Antenna connectors	4x MMCX
Max Conducted 2GHz Tx Power (aggregate)	32 dBm
Max Conducted 5GHz Tx power (aggregate)	29 dBm
Max Conducted 6GHz Tx Power (aggregate)	29 dBm
2GHz frequency range	2360–3150 MHz
5GHz frequency range	4550–6630 MHz
6GHz frequency range	5325–7495 MHz

1.2.3 Wi-Fi Radio #1

Parameter	Description
Operation band	2GHz + 5GHz
Operation mode	11ac 1x1
Antenna connectors	1x MMCX
Max 2GHz Tx Power	16 dBm
Max 5GHz Tx power	16 dBm

1.2.4 Board





USB Host Port #2

USB host controller port supports connecting USB flash drives, USB cameras and other standard USB peripherals like mice and keyboards.

USB Gadget Port #1

USB gadget port should be connected to a host PC using a standard USB cable for device access and configuration purposes. ADB debugging interface is available through this port.

USB PORT #1 POWER SWITCH

Slide switch controls USB power supply for USB gadget port. It should be enabled when USB port is configured as host controller. When USB port is working in gadget mode, switch should be disabled to save power.

Configuration LED #0

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.

GPIO LED #1

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

GPIO LED #2

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

GPIO Switch

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

Factory Reset

Device factory reset button, pressing and holding the button for at least 10 seconds resets device configuration to factory defaults. This is also indicated by the configuration LED slowly blinking while the button is being held.

Device Reboot / EDL Mode

Pressing and releasing "Device reboot" button restarts the device.

To enter EDL (Emergency Download) mode for software recovery or initial device flashing, hold the "EDL mode" button pressed when power cycling or rebooting the device.

2. Robonode set-up

Refer to Robonode board section for available connectors and other peripherals location on PCB.

2.1 Power ON

Device requires power supply ranging from `8V` to `36V`.

- Connect power cords to power supply.
- Device automatically starts up.

2.2 Serial console

Connect 3.3V TTL serial console cable to serial console TTL connect. Follow the steps in console access guide to open serial connection.

2.3 EDL recovery

Software EDL recovery procedure is defined in system maintenance software recovery manual section.

To enter EDL recovery refer to robonode device reboot / EDL mode section for button location on PCB and proceed with following steps:

- Connect USB gadget port#1 to PC.
- Press and hold "EDL mode" button.
- Press and release "Device reboot" button.
- Release "EDL mode" button.
- Device appears on host as `05c6:9008` "Qualcomm, Inc. Gobi Wireless Modem (QDL mode)".

3. TobuFi set-up

Refer to TobuFi DVK board section for available connectors and other peripherals location on PCB.

3.1 Power ON

Device requires **12V 1.5A** power supply.

- Plug-in power jack into device.
- Press "Power ON" button to start device.

This procedure is needed every time device is power cycled. However it is not required when performing normal linux reboot.

Note: Rev5 devices start automatically when power is applied. No button press is required.

3.2 Serial console

Connect 3.3V TTL serial console cable to serial console TTL header. Follow the steps in console access guide to open serial connection.

3.3 EDL recovery

Software EDL recovery procedure is defined in system maintenance software recovery manual section.

3.3.1 Enable EDL recovery mode (rev3/rev4)

- Set boot config switch #1 to ON.
- Power cycle the device.
- Press and release "Power ON" button.
- Device appears on host as `05c6:9008` "Qualcomm, Inc. Gobi Wireless Modem (QDL mode)".

3.3.2 Disable EDL recovery mode (rev3/rev4)

- Set boot config switch #1 to OFF.
- Power cycle the device.
- Press and release "Power ON" button to start normal device boot.

3.3.3 EDL recovery mode (rev5)

Rev5 uses a dedicated push button for EDL mode entry. Follow the same procedure as Robonode EDL recovery.

4. TobuFi bring-up

4.1 Ethernet

Ethernet MAC and PHY control done through **emac-dwc-eqos** driver. Driver is started automatically on boot and does not require any additional special initialisation.

4.1.1 MAC address

Ethernet port MAC address is pre-programmed during board production and is stored in `/persist/emac_config.ini` file. When MAC config file is missing a static default MAC is used.

Configuration example:

```
# cat /persist/emac_config.ini
00:55:7b:b5:7d:f9
```

4.1.2 Sources

Driver sources are cloned into `src/data-kernel/drivers/emac-dwc-eqos` directory during initial build environment init. It's cloned from open open source codelinaro repository.

4.2 Wi-Fi Radio #1 (Pine)

Radio #1 supports operation in 2.4GHz and 5GHz band. It is managed by **qca-wifi** proprietary Qualcomm driver through PCIe bus. Radio requires target firmware, board data file and optionally regulatory database file. Driver supports both wext ioctl wireless extensions and nl80211 interface.

Resources required for radio operation:

- Radio sub-processor M3 core image `/lib/firmware/qcn9000/m3.bin`
- Radio sub-processor target firmware `/lib/firmware/qcn9000/amss.bin`
- Radio board data files `/lib/firmware/qcn9000/bdwlan.bin`
- Radio regulatory database `/lib/firmware/qcn9000/regdb.bin`

Radio MAC address and calibration data are pre-programmed during production into radio's internal EEPROM storage.

4.2.1 Operation band

Despite radio supports operating 2.4GHz and 5GHz bands it requires full radio reconfiguration to switch between bands. Radio band switch is done by supplying radio appropriate board data file. This simply could be done linking appropriate file into `/lib/firmware/qcn9000/bdwlan.bin`.

Applicable board data files:

- `/lib/firmware/qcn9000/bdwlan.pn2` -- 2.4GHz operation board data file
- `/lib/firmware/qcn9000/bdwlan.pn5` -- 5GHz operation board data file

Procedure to switching band:

```
rmmod wifi_3_0
rm /lib/firmware/qcn9000/bdwlan.bin
ln -s /lib/firmware/qcn9000/bdwlan.pn2 c/lib/firmware/qcn9000/bdwlan.bin
modprobe wifi_3_0
```

4.2.2 Driver load

Driver load is deferred to application services startup. Overall driver load sequence consists of following steps:

- Loaded **qca-wifi** driver modules
- Driver initiates CNSS subsystem
- Target firmware, board data and regdb are loaded
- Host driver initialises network devices

Main driver module is called **wifi_3_0**. To initialise the radio it is sufficient to load this module:

```
modprobe wifi_3_0
```

4.3 Wi-Fi Radio #2 (Cherokee)

Radio #2 supports working in 2.4GHz and 5GHz band. It is managed by **qcacld** Qualcomm open source driver. Radio requires target firmware image and board data configuration files, which gets loaded by controlling driver during initialization. Driver has no wext ioctl wireless extensions and is managed through nl80211 interface.

Resources required for radio operation:

- Target firmware image `/lib/firmware/qdsp6sw.mbn`
- Board data files `/lib/firmware/bdwlan*`
- Driver default configuration `/lib/firmware/wlan/qca_cld/WCNSS_qcom_cfg.ini`

4.3.1 Driver load

Driver load is deferred to application services startup. Overall driver load sequence consists of following steps:

- Started **cnss_daemon** service
- Loaded **wlan** driver module
- CNSS service starts radio initialisation
- Target firmware and calibration data is loaded
- Host driver initialises network devices

Main driver module is called **wlan**.

4.3.2 MAC address

The wireless radio MAC address is pre-programmed during board production and stored in the /persist/wlan_mac.bin file. When MAC config file is missing a static default MAC is used.

Configuration example:

```
# cat /persist/wlan_mac.bin
Intf0MacAddress=00037f129eab
END
```

4.3.3 Tx/Rx calibration data

Wireless radio Tx/Rx calibration data is pre-programmed during board production and is stored in `/persist/bdwlan.bin`.

4.3.4 XTAL calibration data

Wireless radio XTAL calibration data is pre-programmed during board production and is stored in `/persist/xo_cal_data.bin`.

4.4 Production EEPROM

Production EEPROM is production-time pre-programmed memory block storing essential board unit metadata. Textual representation of EEPROM memory can be found in `/persist/eeprom.cfg`.

NOTE: When `eeprom.cfg` file is missing means production information is not written. In this case it is recommended to backup `/persist` directory contents so it would be possible to recover on accidental erase.

EEPROM configuration example:

```
# cat /persist/eeprom.cfg
PRODUCT_ID=TobuFi-DVK
PCB_REVISION=01
PCB_NAME=TOB_200
PCB_SN=1234567890
PCB_PROD_DATE=2025-01-01
PCB_PROD_LOCATION=Somewhere
SERIAL_NO=0987654321
MAC_ADDR_eth0=00:03:7F:12:90:8F
MAC_ADDR_wlan0=00:03:7F:12:90:90
MAC_ADDR_wlan1=00:03:7F:12:90:91
```

5. System access

- Console access (UART)
- SSH connection (Network)
- ADB access (USB)
- GUI access (Network)
- USB ethernet (USB/Network)

5.1 Console access

Connect serial console cable to board's UART TTL header. Refer to hardware reference manual for serial console TTL header location for your specific board.

Default serial console baud rate is 115200.

Run desired terminal emulator:

- picocom
- microcom
- minicom
- etc.

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

```
microcom -s 115200 -p /dev/ttyUSB0
```

Upon successful connection user name prompt will be displayed. Default user name is `root`, no password is required.

5.2 SSH connection

SSH connection requires connectivity over IP networking. Connecting to the device may be done either over:

- Wi-Fi
- Wired ethernet via UTP cable
- Virtual RNDIS/modem interface via USB cable

Target device and host systems IP address subnets must match. For the configuration details to set up target device network please refer to:

- System network config
- System network setup

5.2.1 Shell access

Establish SSH connection using desired SSH client, e.g.:

```
ssh root@192.168.2.1
```

5.2.2 Upload and download files

File operations are supported either via legacy scp or sftp protocols thereby any scp/sftp protocol utility should work (scp, sftp, winscp, filezilla, etc.).

```
scp <local files...> root@192.168.2.1:<remote path>
```

```
scp root@192.168.2.1:<remote file> <local path>
```

5.3 ADB access

ADB connection uses a USB bus configured in device mode providing ADB function and companion `adbd` service to handle host requests. In contrast, `fastboot` provides similar functionality through USB but is controlled by lower level firmware (i.e. bootloader).

To setup the USB ADB connection:

- Connect the device's USB gadget port to the PC using a USB cable capable of data transfer. Refer to hardware section for USB gadget port description and location on board.
- Use ADB tools on your host computer to access device as described in section below.

5.3.1 Required Software Installation

Before using ADB, you'll need to install some software on your computer. Follow the instructions for your operating system:

Linux (Ubuntu/Debian)

Open a terminal and run:

```
sudo apt install android-tools-adb android-tools-fastboot
```

MacOS

Using Homebrew package manager, open a terminal and run:

```
brew install android-platform-tools
```

Windows

- Install Python:
  - Download and install Python 3.8 from https://www.python.org/downloads/windows/
  - Ensure "Add Python to PATH" is checked during installation
- Install USB Driver (choose one):
  - Install Google USB Driver from https://developer.android.com/studio/run/win-usb
  - Or install Universal ADB Driver from https://adb.clockworkmod.com/
  - Or install your device manufacturer's specific USB driver from their website
- Install ADB Tools:
  - Go to https://developer.android.com/tools/releases/platform-tools
  - Download and install Android SDK Platform Tools

Once ADB/fastboot tools installed you can start using the tools in Windows CLI shell/prompt.

5.3.2 Listing devices

```
adb devices
```

5.3.3 Shell access

```
adb [-s <serial>] shell
```

ADB shell is accessed using command above. When there is only single device connected to host PC, serial number may be omitted. When host PC detects more than one ADB device connected, serial number is required. Available devices serial numbers can be retrieved by listing the devices

5.3.4 Upload and download files

```
adb [-s <serial>] push <local files...> <remote path>
```

```
adb [-s <serial>] pull <remote files...> <local path>
```

5.4 GUI access

The device hosts a web-based management interface that can be accessed through a standard web browser over the network. The web UI provides a graphical interface for device configuration, monitoring, and management without requiring command-line access.

To access the device's web management interface:

- Ensure your computer is connected to the same network as the device, either through Wi-Fi, Ethernet, or USB RNDIS connection.
- Open a web browser on your computer and navigate to the device's IP address:

```
http://192.168.2.1
```

5.5 USB ethernet

USB ethernet uses RNDIS (Remote Network Driver Interface Specification) protocol to provide network connectivity over a USB cable by creating a virtual network interface. This allows the device to appear as a network adapter to the host computer, enabling IP-based communication.

To setup the USB ethernet connection:

- Connect the device's USB gadget port to the PC using a USB cable capable of data transfer. Refer to hardware section for USB gadget port description and location on board.
- On your host computer, the device should appear as a new network interface. The interface name varies by operating system:
- Linux: typically `usb0` or `enp...`
- macOS: typically `en<number>`
- Windows: appears as "Remote NDIS based Internet Sharing Device"

Once USB RNDIS connectivity is established, you can use it for:

- SSH connections
- GUI access

6. System maintenance

6.1 Software version

Software release version is stored in file and can be retrieved via linux CLI shell.

```
# cat /etc/version
v0.1.5-r523
```

Software release build details are located in `/lib/release` directory files.

```
# cat /etc/release/version
v0.1.5-r523
```

```
# cat /lib/release/hash-sdk
405b8dc
```

```
# cat /lib/release/build
BUILD_MACHINE=tobufi-robonode
BUILD_DISTRO=tobufi-drone
BUILD_IMAGE=8dev-drone-image
BUILD_VERSION=v0.1.5-r523
BUILD_HOST=runner-wwd4mxxm-project-833-concurrent-0
```

6.2 Software update

There are two different software image packages types used for different software update methods.

- Software update image used for on-system OTA update.
- Recovery flash image used for fastboot and EDL flashing over USB.

6.2.1 OTA update

Update procedure requires standard system update image. Firmware update package may be uploaded to device via network (e.g. ethernet connection or Wi-Fi), or using USB ADB interface. Software update package can be placed in any directory on the device, however it's recommended to place it in `/tmp` directory.

- Upload firmware update package to device
- Run `swupgrade <firmware.zip>`

6.2.2 Fastboot flashing

Update procedure requires fastboot/EDL recovery flash update package. For required software installation instructions, refer to ADB Required Software Installation.

- Connect USB gadget port#1 to PC.
- Enter fastboot mode (optional).
- Execute fastboot flashing script (`fastboot_flash.py`) from software flash image package.
- Device reboots automatically after successful update.

MULTIPLE FASTBOOT DEVICES

Multiple devices might be connected to the PC at the same time. When multiple devices are found in fastboot state, the `fastboot_flash.py` utility must be explicitly specified which device to upgrade. Devices are identified by serial number which should be passed to the utility.

```
./fastboot_flash.py <serial>
```

Refer to check fastboot devices section to get the list of available fastboot devices.

Fastboot mode (advanced)

Fastboot is special device mode allowing low-level system controls in bootloader. Enter to **fastboot** mode may be triggered either from device shell or through ADB.

ENTER FASTBOOT

To enter **fastboot** requires device reboot running either commands

On-device linux command:

```
reboot bootloader
```

On-host ADB command:

```
adb reboot bootloader
```

EXIT FASTBOOT

To leave **fastboot** mode reboot the device or run on-host fastboot command

```
fastboot reboot
```

CHECK FASTBOOT DEVICES

Listing devices displays detected devices in fastboot mode and their serial numbers. Serial number is required to specify using `-s <serial>` fastboot option when more than one device is detected.

```
fastboot devices
```

6.3 Software recovery

Update procedure requires fastboot/EDL recovery flash image package.

The procedure requires the device to be forced into EDL recovery mode. Refer to desired board initial setup section how to force device into EDL recovery mode:

- Robonode
- TobuFi DVK

NOTE: By default, performing EDL recovery fully erases device storage including production data inside `/persist` directory. The `-p` flag can be used to preserve data partitions during flashing (see examples below). When not using the `-p` flag, it is recommended to backup data in `/persist` directory before starting EDL recovery procedure. Refer to desired board bring-up section for preservable production data:

- TobuFi DVK
- Connect USB gadget port#1 to PC.
- Enter EDL recovery mode.
- Execute EDL flashing script (`edl_flash.py`) from software flash image package.
- Power cycle the device to boot new firmware.

Data preservation options

The `edl_flash.py` utility supports preserving data partitions during recovery flash using the `-p` flag:

```
./edl_flash.py -p
```

Preserve `/persist` partition (production data).

```
./edl_flash.py -pp
```

Preserve both `/persist` and `userdata` partitions.

```
./edl_flash.py -p <serial>
```

Preserve `/persist` partition with serial number for multi-device setup.

6.3.1 Multiple EDL devices

Multiple devices might be connected to the PC at the same time. When multiple devices are in EDL state when attempting software recovery, the `edl_flash.py` utility must be explicitly specified which device to upgrade. Devices are identified by serial number which should be passed to the utility.

```
./edl_flash.py <serial>
```

Listing available EDL devices:

```
./edl_flash.py -l
```

6.4 Configuration management

6.4.1 Apply configuration

System config settings need to be applied when they are modified or replaced. This is done by reloading configuration using command:

```
reload-config
```

6.4.2 Reset configuration to factory defaults

Resetting configuration erases all user system modifications and restores system to original factory state. After reset system is automatically rebooted.

```
reset-config
```

6.5 Diagnostics and Troubleshooting

The software cointains diagnostic tools for system troubleshooting and issue analysis. The `trouble` utility automatically collects comprehensive system information for troubleshooting purposes. It gathers configuration files, logs, network information, and hardware diagnostics.

6.5.1 Collected Information

The troubleshoot utility captures:

- System information
- logs and statistics
- Configuration files
- Network diagnostrics
- Network packet dump

6.5.2 Example Usage

Generate troubleshoot archive with default naming Default output location: `/tmp/<hostname>-<MAC>.tar.gz`

```
trouble
```

Generate troubleshoot archive to specific file

```
trouble /tmp/system-diag.tar.gz
```

6.5.3 Common Troubleshooting Steps

- **Collect diagnostic data**: Run `trouble` to gather comprehensive system information
- **Check service status**: Verify critical services are running properly
- **Review logs**: Check system and service logs for error messages
- **Verify configuration**: Ensure system configuration is applied correctly
- **Test connectivity**: Verify network interfaces and connectivity
- **Monitor resources**: Check system resources (CPU, memory, disk usage)

6.6 Interfaces Monitoring

The `watchirq` service continuously monitors network interface performance and interrupt statistics, providing real-time throughput analysis for system monitoring and troubleshooting.

The service runs automatically in the background and generates JSON formatted statistics files for programmatic access and integration with monitoring tools.

6.6.1 Output Files

The service statistics into files at `/var/stats/`:

- `/var/stats/watchirq.json`: Current performance metrics snapshot for all interfaces in JSON format
- `/var/stats/watchirq/<interface>.log`: Chronological tabular data per interface with timestamp, packet rates, throughput (Mbps), and CPU usage

```
# View current interface statistics
cat /var/stats/watchirq.json

# Monitor interface performance files
tail -f /var/stats/watchirq/<inteface>.log
```

6.6.2 Collected Metrics

- **Throughput**: TX/RX data rates in Mbps and packets per second
- **Error Statistics**: Packet errors and dropped packets per interface
- **System Performance**: CPU idle percentage correlation
- **Real-time Updates**: Statistics updated every second

6.7 WiFi Spectrum Analysis

The `spectral` service runs in the background to continuously monitor WiFi spectrum usage and channel interference. This provides insights into RF environment and helps optimize channel selection for WiFi broadcast operations.

6.7.1 Output Files

The service generates spectrum analysis data in `/var/stats/` (volatile memory - lost after reboot):

- **`/var/stats/spectral.json`**: Current spectrum analysis snapshot in JSON format
- **`/var/stats/spectral.csv`**: Chronological spectral measurements continuously updated (rotates every 24h)

```
# View current spectrum analysis
cat /var/stats/spectral.json

# Monitor spectrum data files
tail -f /var/stats/spectral.csv
```

6.7.2 Manual Spectrum Analysis

The `spectral` utility allows on-demand spectrum analysis but interrupts the background scan service while running.

Single Spectrum Scan

```
spectral
```

Continuous Monitoring

```
spectral -c
```

Frequency Range Scan

```
# Scan specific frequency range (2.4GHz band)
spectral -a -b 2400:2500
```

Save to File

```
# Save results to custom CSV file
spectral -f /tmp/spectrum-scan.csv
```

7. System setup

7.1 Apply configuration

System config data needs to be applied when it's modified or replaced. Applying configuration makes config settings effective.

```
reload-config
```

7.2 Network control

Network controls are available through `network` utility. Executing the command applies the change in system and updates system configuration. Multiple options may be specified in single command to apply them in one shot.

7.2.1 IP address

```
network ipaddr <A.D.D.R>
```

7.2.2 IP mode

```
network ipmode <mode>
```

Mode	Description
static	Static IP configuration
dhcp-client	DHCP client, preserves IP address as fallback
dhcp-server	DHCP server, IP address must be configured

7.3 Wi-Fi Radio0 control

Wi-Fi radio0 controls are available through `radio0` utility. Executing command applies the change in system and updates system configuration. Multiple options may be specified in single command to apply them in one shot.

7.3.1 Getting parameters

Active radio0 parameters can be retrieved using the `get` command. Caller may retrieve either single, multiple or all parmeters.

```
radio0 get [param1 [param2] ...]
```

Available parameters: `mode`, `ctry`, `chwidth`, `frequency`, `txpower`, `cca`, `protocol`, `gi`, `nss`, `mcs`, `nawds`.

7.3.2 Operation mode

Controls both `DEVICE_MODE` and `WIFI0_VAP_MODE` parameters in configuration.

To set up a specific radio mode:

```
radio0 setup <mode>
```

To tear down the current radio mode:

```
radio0 teardown
```

Mode	Description
ap	BSS access-point mode
sta	BSS station mode
nawds	Non-associated WDS mode (requires NAWDS peer configuration)
wfb	Wi-Fi broadcast mode (see Wi-Fi broadcast section)

When changing modes, first tear down the current mode before setting up the new one:

```
radio0 teardown
radio0 setup <new-mode>
```

7.3.3 Country code

Country code should be a two-letter alpha2 country code from regdb supported countries list or special `CT` for unrestricted operation.

```
radio0 ccode <country-code>
```

7.3.4 Channel width

Channel width support is limited for WFB mode. When WFB mode is enabled only 5, 10 and 20 MHz channel widths are supported.

```
radio0 chwidth <5|10|20|40|80|160>
```

7.3.5 Frequency

Radio frequency parameters accept Wi-Fi channel number or frequency value in MHz scale and with 1 MHz precision. Frequency value should be compatible with radio supported bands. Radio band change is performed automatically.

```
radio0 frequency <frequency|channel>
```

7.3.6 Tx Power

Radio Tx power parameter value should be in dBm or `0` (auto) maximal possible value.

```
radio0 txpower <dbm>
```

7.3.7 CCA Threshold

Controls the Clear Channel Assessment (CCA) threshold for the radio transmitter. This value is in dBm, allowed valus range from -94 to -11 or 0 to set up vendor default.

```
radio0 cca <dBm>
```

7.3.8 NAWDS peer(s)

NAWDS supports more than one peer connection. Therefore to manage peers list couple control options are available.

Reset single NAWDS peer:

```
radio0 nawds-set <MAC>
```

Add additional NAWDS peer:

```
radio0 nawds-add <MAC>
```

To find the peer MAC address, check the VAP interface MAC on the remote device:

```
ip link show wifi0
```

7.3.9 Protocol mode

Controls modulation protocol used by the radio.

```
radio0 protocol <ht|vht|he|auto>
```

Value	Description
ht	High Throughput (802.11n) protocol and modulations
vht	Very High Throughput (802.11ac) protocol and modulations, 5GHz only
he	High Efficiency (802.11ax) protocol and modulations
auto	Automatic selection, in BSS/NAWDS mode unrestricted mixed mode; in WFB mode defaults to HT

7.3.10 Guard interval

Controls guard interval duration.

```
radio0 gi <400|800|1600|3200>
```

Value	Description
400	0.4µs guard interval (short GI)
800	0.8µs guard interval (normal GI)
1600	1.6µs guard interval (HE only)
3200	3.2µs guard interval (HE only)

7.3.11 NSS count

NSS (Number of Spatial Streams) controls how many spatial data streams are used for transmission. This feature is only supported with VHT (802.11ac) and HE (802.11ax) protocols.

```
radio0 nss <NSS|auto>
```

Value	Description
1-2	Number of spatial streams
auto	Automatic selection, in BSS/NAWDS mode defaults maximum available; in WFB mode defaults to 1

7.3.12 MCS index

MCS (Modulation and Coding Scheme) determines the data rate. When set to a specific value, the system will use that fixed rate. The auto option behavior depends on the operating mode.

```
radio0 mcs <MCS|auto>
```

Protocol	Range	Description
HT (11n)	0-7, 0-15	Single/Dual stream MCS indexes
VHT (11ac)	0-9	Per stream MCS indexes
HE (11ax)	0-11	Per stream MCS indexes
auto	-	Automatic rate adaption in BSS/NAWDS mode; in WFB mode defaults to MCS 1

7.4 Wi-Fi Radio1 control

Wi-Fi radio1 controls are available through `radio1` utility. Executing command applies the change in system and updates system configuration. Multiple options may be specified in single command to apply them in one shot.

7.4.1 Getting parameters

Active radio1 parameters can be retrieved using the `get` command. Caller may retrieve either single, multiple or all parmeters.

```
radio1 get [param1 [param2] ...]
```

Available parameters: `mode`, `ctry`, `chwidth`, `frequency`, `txpower`.

7.4.2 Operation mode

Controls `WIFI1_VAP_MODE` parameter in configuration.

To set up a specific radio mode:

```
radio1 setup <mode>
```

To tear down the current radio mode:

```
radio1 teardown
```

Mode	Description
ap	BSS access-point mode
sta	BSS station mode

When changing modes, first tear down the current mode before setting up the new one:

```
radio1 teardown
radio1 setup <new-mode>
```

7.4.3 Country code

Country code should be a two-letter alpha2 country code. You can get the list of supported country codes from the Linux Wireless Regulatory Database source: https://git.kernel.org/pub/scm/linux/kernel/git/sforshee/wireless-regdb.git/tree/db.txt

```
radio1 ccode <country-code>
```

7.4.4 Channel width

```
radio1 chwidth <20|40|80>
```

7.4.5 Frequency

Radio frequency parameters accept Wi-Fi channel number or channel frequency value. Radio band is switched automatically.

```
radio1 frequency <frequency|channel>
```

7.4.6 Tx Power

Radio Tx power parameter value should be in dBm or `0` (auto) maximal possible value.

```
radio1 txpower <dbm>
```

8. System config

Single config file defines system boot-time and run-time configuration. It is located at `/etc/robonode/system.cfg`. Config file is a plain text containing ini like key-value pairs.

8.1 Configuration Version

The configuration file includes a version field that defines the schema version and supported keys:

```
VERSION="0.3"
```

This field is automatically managed by the system and should not be modified manually.

8.2 Device mode

The device mode configures how Wi-Fi connectivity operates. Set it in the config file:

```
DEVICE_MODE=wfb
```

The following modes are supported:

Mode	Description
wfb	Wi-Fi Broadcast mode - Enables high-performance, ACK-less communication between devices. Limited to 5/10/20 MHz channel widths.
nawds	Non-Associated WDS mode - Direct device-to-device communication with ACKs enabled. Requires peer MAC addresses to be configured via `WIFI0_NAWDS_MAC`.
bss	Basic Service Set mode - Standard Wi-Fi AP/STA operation for traditional access point and client connections.

8.3 Network settings

Configured bridged networking IP settings.

```
NET_IP_MODE=dhcp-client
NET_IP_ADDR=192.168.2.1
```

IP Mode	Description
static	Static IP configuration defined in `NET_IP_ADDR`
dhcp-client	DHCP client, `NET_IP_ADDR` is used as fallback IP
dhcp-server	DHCP server, uses `NET_IP_ADDR` to derive server params

8.4 Wi-Fi radio0 (Pine) settings

```
WIFI0_CTRY=CT
WIFI0_FREQ=2437
WIFI0_CHW=20
WIFI0_TXP=20
WIFI0_CCA=0
WIFI0_PROTO=auto
WIFI0_GI=800
WIFI0_MCS=auto
WIFI0_NSS=auto
```

Parameter	Value options	Description
WIFI0_CTRY	alpha2 country code	Radio country code. May be any country from regdb, or special CT country code for unrestricted operation
WIFI0_FREQ	radio frequency of supported bands	Radio frequency in MHz
WIFI0_CHW	5, 10, 20, 40, 80, 160	Radio operating channel width
WIFI0_TXP	0, 1, ...	Radio Tx power in dBm. Zero means auto-power setting to max value
WIFI0_CCA	-94 - -11, 0, empty	Radio CCA threshold in dBm for transmitter sensitivity. Empty value or 0 stands for vendor default threshold.
WIFI0_PROTO	auto, ht, vht, he	Wi-Fi frame type. Auto value defaults to HT mode in WFB, and does not imply any restriction in BSS/NAWDS mode
WIFI0_GI	auto, 400, 800, 1600, 3200	Transmission guard interval. For HT and VHT proto: 400, 800, for HE proto: 800, 1600, 3200. Value auto defaults to GI 800
WIFI0_MCS	auto, 0-15, 0-9, 0-11	Fixed transmission MCS. For HT mode 0-15, VHT mode 0-9, HE mode 0-11. Value auto defaults to auto-rate for BSS/NAWDS mode, and MCS 3 for WFB mode
WIFI0_NSS	auto, 1-2	Transmission NSS, for VHT and HE proto. Value auto defaults to NSS 1

NOTE: for WFB mode only 5/10/20 MHz channel widths are supported.

8.5 Wi-Fi radio1 settings

```
WIFI1_CTRY=LT
WIFI1_FREQ=5180
WIFI1_CHW=20
WIFI1_TXP=0
```

Parameter	Value options	Description
WIFI1_CTRY	alpha2 country code	Radio country code. May be any country from regdb
WIFI1_FREQ	Valid 2GHz or 5GHz channel frequency	Radio channel frequency in MHz
WIFI1_CHW	20, 40, 80	Radio operating channel width
WIFI1_TXP	0, 1, ...	Radio Tx power in dBm. Zero means auto-power setting to max value

8.6 Wi-Fi Broadcast settings

```
WFB_FEC_K=1
WFB_FEC_N=2
WFB_FEC_DELAY=0
WFB_FEC_TIMEOUT=0
WFB_SECURITY=on
WFB_STREAMS=bridge
```

Parameter	Value options	Description
WFB_FEC_K	1, 2, ...	Number of blocks required to fully recover the data
WFB_FEC_N	2, 3, ...	Total number of produced blocks data and redundancy
WFB_FEC_DELAY	0, 1, ...	Wi-Fi broadcast packets transmission delay
WFB_FEC_TIMEOUT	0, 1, ...	Wi-Fi broadcast link timeout
WFB_SECURITY	on, off	Wi-Fi broadcast security mode: enabled or disabled encryption
WFB_STREAMS	bridge, dial, drone, gs, ...	Arbitrary names of Wi-Fi broadcast streams to enable
WFB_MODE	tx, rx, tun, tap	Wi-Fi broadcast operation mode: tx (IP socket Tx mode), rx (IP socket Rx mode), tun (TUN type tunneling), tap (TAP type bridging). Required for each stream
WFB_STREAM_ID	0-255	Wi-Fi broadcast unique stream identifier, maps data streams on Tx and Rx sides. Required for each stream
WFB_LISTEN_PORT	1-65535	UDP port number to listen for data on. Required for tx mode
WFB_FORWARD_ADDR	IP address	Destintaion IP address to forward data to. Required for rx mode
WFB_FORWARD_PORT	1-65535	Destination UDP port to forward data to. Required for rx mode
WFB_TUN_IP_ADDR	IP address	IP address for TUN interface. Effective in tun,tap modes
WFB_TUN_IP_MASK	0-32	Network mask for TUN interface. Effective in tun,tap modes
WFB_TUN_MTU	1-65535	MTU size for TUN/TAP interface. Effective in tun,tap modes
WFB_TUN_KA	0, 1, ...	Keep-alive interval in miliseconds. Effective in tun,tap modes
WFB_TUN_IF_NAME	interface name	Tunnel interface name to assign to. Effective in tun,tap mode
WFB_TUN_IF_BRIDGE	bridge name	Bridge interface to add tunner to. Effective in tap mode

8.6.1 Wi-Fi Broadcast Stream configuration

Each enabled stream (listed in `WFB_STREAMS`) requires individual configuration with stream-specific parameters. Parameters are specified using the format `PARAMETER_name`.

Every stream requires the following mandatory parameters: - `WFB_MODE_name`: Operation mode (tx, rx, tun, tap) - `WFB_STREAM_ID_name`: Unique stream identifier (0-255)

Additional parameters are required based on the selected mode:

TX mode parameters (for WFB_MODE_name=tx)

```
WFB_MODE_drone=tx
WFB_STREAM_ID_drone=1
WFB_LISTEN_PORT_drone=5600
```

RX mode parameters (for WFB_MODE_name=rx)

```
WFB_MODE_gs=rx
WFB_STREAM_ID_gs=1
WFB_FORWARD_ADDR_gs=127.0.0.1
WFB_FORWARD_PORT_gs=5600
```

TUN mode parameters (for WFB_MODE_name=tun)

```
WFB_MODE_dial=tun
WFB_STREAM_dial=0
WFB_TUN_IP_ADDR_dial=10.20.30.1
WFB_TUN_IP_MASK_dial=24
WFB_TUN_MTU_dial=1516
WFB_TUN_KA_dial=0
```

TAP mode parameters (for WFB_MODE_name=tap)

```
WFB_MODE_bridge=tap
WFB_STREAM_ID_bridge=2
WFB_TUN_IF_BRIDGE_bridge=br0
WFB_TUN_MTU_bridge=1544
WFB_TUN_KA_bridge=0
```

8.7 NAWDS settings

```
WIFI0_NAWDS_MAC=""
```

Configures radio0 NAWDS peer address(-es). This should be VAP interface MAC address from remote NAWDS device.

8.8 BSS settings

```
WIFI0_VAP_MODE=ap
WIFI0_VAP_SSID=
WIFI0_VAP_PSK=

WIFI1_VAP_MODE=ap
WIFI1_VAP_SSID=
WIFI1_VAP_PSK=
```

Configures single VAP either in AP or STA mode for BSS type connection.

Parameter	Value options	Description
WIFIx_VAP_MODE	ap, sta	Radio VAP operation mode
WIFIx_VAP_SSID	SSID name	Radio VAP SSID, either to connect STA to or start AP service
WIFIx_VAP_PSK	WPA passphrase	Radio VAP WPA2-PSK security passphrase, when empty Open security

8.9 Video settings

```
VIDEO_INPUT=usb
VIDEO_HEIGHT=640
VIDEO_WIDTH=480
VIDEO_BITRATE=1500
VIDEO_WFB_STREAM=drone
```

Video settings allows to enable direct video input encoding and stream forwarding parameters.

Note: Only cameras capable of outputting raw/uncompressed video streams (YUYV format) are supported. Please verify that used camera supports the specified resolution in YUYV mode.

Parameter	Value options	Description
VIDEO_INPUT	usb, dummy, none	Video input source
VIDEO_HEIGHT	pixels	Video height in pixels
VIDEO_WIDTH	pixels	Video width in pixels
VIDEO_BITRATE	bits/s	Video bitrate in bits per second
VIDEO_WFB_STREAM	name	WFB video stream name

8.10 UART settings

```
UART_DEVS="ttyHS0"
UART_ACCESS_PORT_ttyHS0=
UART_ACCESS_TYPE_ttyHS0=
UART_TIMEOUT_ttyHS0=
UART_SPEED_ttyHS0=
UART_PARAMS_ttyHS0=
```

Configures UART ports forwarding over network using ser2net service. Further available configuration options can be found in ser2net reference configuration: https://github.com/chargebyte/ser2net/blob/master/ser2net.conf

Parameter	Value options	Description
UART_DEVS	ttyHS0, ttyHSx ...	List of TTY devices to enable network access. Each TTY device may have additional override settings as listed below
UART_ACCESS_PORT_{TTY}	1 - 65535	Network port number for TCP serial access, default 3000 + tty device index
UART_ACCESS_TYPE_{TTY}	raw, rawlp, telnet	Service access mode, default is raw
UART_TIMEOUT_{TTY}	0, ...	Connection activity timeout in seconds, default is 0 (disabled timeout)
UART_SPEED_{TTY}	9600, 19200, 38400, 57600, 115200, ...	UART port baud rate, default is 115200.
UART_PARAMS_{TTY}	Ser2Net UART config params	List of UART parameters for additional port configuration.

8.11 RTP pipeline settings

Supported since version: 0.3.1

```
RTP_MONITOR=on
RTP_PIPELINE_drone=off
RTP_DUP_DELAY_drone=100
RTP_LISTEN_PORT_drone=5900
RTP_PIPELINE_gs=off
RTP_UNDUP_WINDOW_gs=150
RTP_FORWARD_ADDR_gs=127.0.0.1
RTP_FORWARD_PORT_gs=5900
```

RTP (Real-time Transport Protocol) packet duplication/deduplication is dedicated to improve video streaming reliability over lossy wireless links. The RTP pipeline integrates with WFB streams to add configurable packet redundancy.

Each RTP pipeline is mapped 1:1 to a WFB stream using the same stream name suffix (e.g., `drone`, `gs`). RTP pipelines only work with WFB streams configured in `tx` or `rx` mode.

8.11.1 Global RTP Settings

Parameter	Value options	Description
RTP_MONITOR	on, off	Enable detailed JSON statistics output for all RTP streams.

8.11.2 Tx stream parameters (for WFB_MODE_name=tx)

Parameter	Value options	Description
RTP_PIPELINE_name	on, off	Enable/disable RTP pipeline for the named stream. Default is off
RTP_DUP_DELAY_name	1-1000	The delay in milliseconds before transmitting duplicate packets. Default: 100ms. Lower values reduce latency but may reduce recovery effectiveness
RTP_LISTEN_PORT_name	1-65535	The port number to listen for incoming RTP stream. Required when RTP pipeline is enabled for a TX mode WFB stream

8.11.3 Rx stream parameters (for WFB_MODE_name=rx)

Parameter	Value options	Description
RTP_UNDUP_WINDOW_name	1-1000	The time window in milliseconds for buffering and deduplicating packets. Should be >= duplication delay on TX side. Default: 150ms
RTP_FORWARD_ADDR_name	IP address	The destination address to forward deduplicated RTP stream to. Required when RTP pipeline is enabled for an RX mode WFB stream
RTP_FORWARD_PORT_name	1-65535	The destination port to forward deduplicated RTP stream to. Required when RTP pipeline is enabled for an RX mode WFB stream

8.11.4 RTP Configuration Example

Complete example for drone video streaming with RTP duplication:

```
# WFB stream configuration
WFB_STREAMS="drone gs"
WFB_MODE_drone=tx
WFB_LISTEN_PORT_drone=5600
WFB_MODE_gs=rx
WFB_FORWARD_ADDR_gs=127.0.0.1
WFB_FORWARD_PORT_gs=5601

# Enable RTP for drone TX stream
RTP_PIPELINE_drone=on
RTP_LISTEN_PORT_drone=5010     # Video source sends to this port
RTP_DUP_DELAY_drone=50          # 50ms delay between duplicates

# Enable RTP for ground station RX stream
RTP_PIPELINE_gs=on
RTP_FORWARD_ADDR_gs=192.168.1.100  # Forward to video player
RTP_FORWARD_PORT_gs=5011           # Video player listens here
RTP_UNDUP_WINDOW_gs=100            # 100ms deduplication window

# Enable statistics monitoring
RTP_MONITOR=on
```

8.12 Parallel monitoring interface

Supported since version: 0.3.1

```
WIFI0_MONITOR=on
```

Configures whether a monitoring interface should be added and enabled when the radio0 is set to either BSS or NAWDS modes.

9. System networking

This document describes lower-level Wi-Fi radio and network configuration using CLI utilities. Wi-Fi connections are managed by platform utilities and `wpa_supplicant` and `hostapd` services.

9.1 QCA-wifi radio controls

Radio settings and network configuration are managed by combination of custom CLI utilities and `systemd` units.

Radio and VAP configuration parameters are persisted in radio configuration file and overlay enabled `systemd` services unit files.

9.1.1 Configuration file

Persistent radio configuration file is located at `/etc/qcawifi.conf`. In combination with enabled `systemd` units (`hostapd` or `wpa_supplicant`) used for VAP setup allows to preserve system configuration state across reboots.

Default configuration files format and parameters:

```
MODE=ap
BAND=5
COUNTRY=US
```

Parameter	Allowed values	Description
MODE	ap, sta	System managed settings to preserve desired operation mode
BAND	2, 5	Radio operation band to set up
COUNTRY	Country codes	Alpha2 format country code available in regulatory database
CHWIDTH	5, 10, 20, 40, 80, 160	Radio channel width
CHANNEL	Channel or frequency	Radio channel number or frequency
TXPOWER	Tx power, dBm	Radio Tx power

9.1.2 Systemd units

A `systemd` units sets up system during boot and updates configuration on demand during run-time. Available qca-wifi radio control service units:

Unit	Description
`qcawifi_init`	One-shot service unit to set up or tear down qca-wifi radio subsystem
`hostapd@`	Template service unit to start or stop AP mode VAP, requires interface name
`wpa_supplicant@`	Template service unit to start or stop STA mode VAP, requires interface name

A hostap and WPA supplicant template services accepts interface name to set up. Service unit on when setting up will:

- Create VAP interface of appropriate type either "master" or "station" (if does not exits).
- Initialise VAP default configuration for `hostapd` or `wpa_supplicant` (if does not exist).
- Start VAP manager service (`hostapd` or `wpa_supplicant`).

VAP services configuration files are stored at `/etc/hostapd` or `/etc/wpa_supplicant` depending on requested VAP operation mode. Each configuration file is named after corresponding interface name, i.e. `wlan0.conf`.

Service unit ensures that initialised VAP operation mode is compatible with started VAP service. Therefore either `hostapd` or `wpa_supplicant` may be running at the same time for the same interface name. Starting the `hostapd` would automatically stop the `wpa_supplicant` and vice versa, `systemd` takes care of that.

9.1.3 Radio controls

Basic QCA-wifi radio controls are available using `qcawifi` utility.

Available utility commands:

Command	Allowed values	Description
init	-	Initialise qca-wifi radio subsystem
purge	-	Shutdown qca-wifi radio subsystem
band	2, 5	Changes radio operation band and persists parameter
country	Country code	Changes radio country code and persists parameter
chwidth	160, 80, 40, 20, 10, 5	Changes radio channel width and persists parameter
channel	Channel number or frequency	Changes radio channel/frequency and persists parameter
txpower	Tx power in dBm	Changes radio Tx power and persists parameter
proto	ht, vht, he, auto	Changes radio protocol/modulation and persists parameter
gi	400, 800, 1600, 3200	Changes guard interval in ns and persists parameter
nss	NSS count, auto	Changes number of spatial streams and persists parameter
mcs	MCS index, auto	Changes modulation and coding scheme index and persists parameter
add-ap	Interface name	Creates AP mode VAP
add-sta	Interface name	Creates STA mode VAP
del-vap	Interface name	Deletes VAP

NOTE: radio band change, reinitialises qca-wifi subsystem, therefore VAP interfaces must be set up again. A `systemd` service units may be used. If requested band matches current band no reconfiguration is done.

NOTE: Creating AP or STA mode VAP does not include setting up `hostapd` or `wpa_supplicant` services.

Radio channel width, channel number, frequency and Tx power change, requires VAP interface to be created. If radio VAP is not yet available, the parameters apply will be deferred after first VAP is created.

Created VAP automatically has WDS and `4addr` mode enabled implying interfaces bridging. If VAP used in non-bridged scenarios `4addr` mode for STA should be disabled.

9.1.4 VAP controls

Basic QCA-wifi radio VAP configuration controls are available using `qcavap` utility. Utility initialises or updates VAP services config files located in `/etc/hostapd` and `/etc/wpa_supplicant` directories.

Available utility commands:

Command	Allowed values	Description
init	-	Initialise new VAP config of specified type
band	2, 5	Overrides radio operation band found in config
freqs	no, frequencies	Specifies operation frequencies restrictions, multiple comma separated frequencies may be specified
narrow	no, 10, 5	Overrides narrow channel width parameters found in config
ssid	no, SSID name	Specifies SSID name or generates random SSID when "no"
wpapsk	SSID passphrase	Specifies SSID WPA-PSK security, otherwise open
hidden	-	Disables SSID broadcast

When running utility mandatory positional arguments are required, an interface mode (ap or sta) and interface name to set up.

NOTE: modifying existing or creating new VAP services config does not reloads VAP services itself. A `systemd` service units may be used to do that.

9.1.5 Usage examples

Start AP mode VAP

```
systemctl start hostapd@ath0
```

Start STA mode VAP

```
systemctl start wpa_supplicant@ath0
```

When staring STA mode VAP, AP mode VAP is automatically stopped and vice versa.

Set up additional AP VAP

```
qcawifi add-ap ath1
systemctl start hostapd@ath1
```

Change AP mode VAP settings

```
qcavap ap ath1 ssid New-SSID
systemctl restart hostapd@ath1
```

Set up customized AP mode VAP

```
qcawifi add-ap ath2
qcavap ap ath2 init ssid My-SSID wpapsk My-passphrase
systemctl start hostapd@ath2
```

Initialised configuration is persisted between reboots, therefore boot time VAP auto-configuration may be enabled.

```
systemctl enable hostapd@ath2
```

Restrict AP mode VAP operation frequencies

```
qcavap ap ath1 freqs 5180,5200,5220
systemctl restart hostapd@ath1
```

Restrict STA mode VAP operation frequencies

```
qcavap sta ath1 freqs 5180,5200,5220
systemctl restart wpa_supplicant@ath1
```

Enable STA VAP and disable AP VAP on boot

```
systemctl enable wpa_supplicant@ath0
systemctl disable hostapd@ath0
```

Change radio band to 2GHz

```
qcawifi band 2
systemctl restart hostapd@ath0
```

Band change is persisted between reboots.

Change radio country

```
qcawifi country LT
```

Country change is persisted between reboots. To change without persistence use

```
cfg80211tool wifi0 setCountry LT
```

Change radio Tx power

```
qcawifi txpower 10
```

Tx power is persisted between reboots. To change without persistence use

```
iwconfig ath0 txpower 10
```

Change radio frequency

```
qcawifi channel 60
qcawifi channel 5300
```

Channel may be specified using numeric Wi-Fi channels or channel frequency. Frequency is persisted between reboots. To change without persistence use

```
iwconfig ath0 channel 60
iwconfig ath0 frequency 5300M
```

Change radio channel width

```
qcawifi chwidth 80
```

Channel width is persisted between reboots. To change without persistence use

```
cfg80211tool ath0 mode 11AHE80
cfg80211tool ath0 mode 11GHE80
```

Mode must be set according radio operation band, for 5GHz band mode prefix should be `11AHE`, for 2GHz band mode prefix should be `11GHE`. Suffix is desired channel width. To further change to narrow channel width (5MHz or 10MHz) radio mode should be set to 20MHz and additionally set

```
cfg80211tool ath0 chanbw 2
```

Possible `chanbw` values

chanbw	Channel width
0	20 MHz
1	10 MHz
2	5 MHz

Change radio protocol/modulation

```
qcawifi proto he
```

Available options: `ht`, `vht`, `he`, `auto`

When set to `auto`, modulation won't be enforced thereby the system will negotiate the best protocol with the client device based on capabilities and conditions.

Protocol is persisted between reboots. To change without persistence use:

```
cfg80211tool ath0 puren 0
cfg80211tool ath0 pure11ac 0
cfg80211tool ath0 pure11ax 1
```

Change guard interval

```
qcawifi gi 800
```

Available options: `400`, `800`, `1600`, `3200`

Guard interval is persisted between reboots. To change without persistence use:

```
cfg80211tool ath0 shortgi 0
```

Guard interval	`shortgi`
800ns	0
400ns	1
1600ns	2
3200ns	3

Change NSS

```
qcawifi nss 2
```

Available options: `1`, `2`, or `auto`

NSS is only applicable when using VHT (802.11ac) or HE (802.11ax) protocols. When set to `auto`, the system will use the maximum number of spatial streams available based on current conditions.

NSS is persisted between reboots. To change without persistence use:

```
cfg80211tool ath0 nss 2
```

Setup fixed MCS index

```
qcawifi mcs 2
```

Available options depend on the protocol in use: - HT (802.11n): 0-7 for single stream, 0-15 for two streams - VHT (802.11ac): 0-9 - HE (802.11ax): 0-11

When set to `auto`, the system will dynamically select the appropriate MCS index based on signal quality and conditions.

MCS index is persisted between reboots. To change without persistence use:

```
cfg80211tool ath0 set11NRates <value>
cfg80211tool ath0 vhtmcs 7
cfg80211tool ath0 he_mcs 7
```

9.2 QCACLD radio controls

Radio settings and network configuration are managed by combination of custom CLI utilities and `systemd` units.

Radio and VAP configuration parameters are persisted in radio configuration file and overlay enabled `systemd` services unit files.

9.2.1 Configuration file

Persistent radio configuration file is located at `/etc/qcacld.conf`. In combination with enabled `systemd` units (´`ihostapd` or `iwpa_supplicant`) used for VAP setup allows to preserve system configuration state across reboots.

Default configuration files format and parameters:

```
MODE=ap
COUNTRY=US
```

Parameter	Allowed values	Description
MODE	ap, sta	System managed settings to preserve desired operation mode
COUNTRY	Country codes	Alpha2 format country code available in regulatory database
CHWIDTH	20, 40, 80	Radio channel width
CHANNEL	Channel or frequency	Radio channel number or frequency
TXPOWER	Tx power, dBm	Radio Tx power

9.2.2 Systemd units

A `systemd` units sets up system during boot and updates configuration on demand during run-time. Available qcacld radio control service units:

Unit	Description
`qcacld_init`	One-shot service unit to set up or tear down qcacld radio subsystem
`ihostapd@`	Template service unit to start or stop AP mode VAP, requires interface name
`iwpa_supplicant@`	Template service unit to start or stop STA mode VAP, requires interface name

A hostap and WPA supplicant template services accepts interface name to set up. Service unit on when setting up will:

- Create VAP interface of appropriate type either "master" or "station" (if does not exits).
- Initialise VAP default configuration for `hostapd` or `wpa_supplicant` (if does not exist).
- Start VAP manager service (`ihostapd` or `iwpa_supplicant`).

VAP services configuration files are stored at `/etc/hostapd` or `/etc/wpa_supplicant` depending on requested VAP operation mode. Each configuration file is named after corresponding interface name, i.e. `wlan0.conf`.

Service unit ensures that initialised VAP operation mode is compatible with started VAP service. Therefore either `ihostapd` or `iwpa_supplicant` may be running at the same time for the same interface name. Starting the `ihostapd` would automatically stop the `iwpa_supplicant` and vice versa, `systemd` takes care of that.

9.2.3 Radio controls

Basic QCACLD radio controls are available using `qcacld` utility.

Available utility commands:

Command	Allowed values	Description
init	-	Initialise qcacld radio subsystem
purge	-	Shutdown qcacld radio subsystem
country	Country code	Saves radio country code in config
chwidth	80, 40, 20	Saves radio channel width in config
channel	Channel number or frequency	Saves radio channel/frequency in config
txpower	Tx power in dBm	Saves radio Tx power in config
add-ap	Interface name	Creates AP mode VAP
add-sta	Interface name	Creates STA mode VAP
del-vap	Interface name	Deletes VAP

NOTE: Creating AP or STA mode VAP does not include setting up `hostapd` or `wpa_supplicant` services.

Radio channel width, channel number, frequency and Tx power are saved to config and would be applied in VAP config.

Created VAP automatically has `4addr` mode enabled implying interfaces bridging. If VAP used in non-bridged scenarios `4addr` mode for STA should be disabled.

9.2.4 VAP controls

Basic QCACLD radio VAP configuration controls are available using `qtivap` utility. Utility initialises or updates VAP services config files located in `/etc/hostapd` and `/etc/wpa_supplicant` directories.

Available utility commands:

Command	Allowed values	Description
init	-	Initialise new VAP config of specified type
country	Country code	Specifies radio country code
chwidth	80, 40, 20	Specified radio operating channel width
channel	Channel number or frequency	Specified radio channel number or frequency
freqs	no, frequencies	Specifies operation frequencies restrictions, multiple comma separated frequencies may be specified
ssid	no, SSID name	Specifies SSID name or generates random SSID when "no"
wpapsk	SSID passphrase	Specifies SSID WPA-PSK security, otherwise open
hidden	-	Disables SSID broadcast

When running utility mandatory positional arguments are required, an interface mode (ap or sta) and interface name to set up.

NOTE: modifying existing or creating new VAP services config does not reloads VAP services itself. A `systemd` service units may be used to do that.

9.2.5 Usage examples

Start AP mode VAP

```
systemctl start ihostapd@wlan0
```

Start STA mode VAP

```
systemctl start iwpa_supplicant@wlan0
```

When staring STA mode VAP, AP mode VAP is automatically stopped and vice versa.

Set up additional AP VAP

```
qcacld add-ap wlan1
systemctl start ihostapd@wlan1
```

Change AP mode VAP settings

```
qtivap ap wlan1 ssid New-SSID
systemctl restart hostapd@wlan1
```

Set up customized AP mode VAP

```
qcacld add-ap wlan2
qtivap ap wlan2 init ssid My-SSID wpapsk My-passphrase
systemctl start hostapd@wlan2
```

Initialised configuration is persisted between reboots, therefore boot time VAP auto-configuration may be enabled.

```
systemctl enable hostapd@wlan2
```

Restrict AP mode VAP operation frequencies

```
qtivap ap wlan1 freqs 5180,5200,5220
systemctl restart hostapd@wlan1
```

Restrict STA mode VAP operation frequencies

```
qcavap sta wlan1 freqs 5180,5200,5220
systemctl restart wpa_supplicant@wlan1
```

Enable STA VAP and disable AP VAP on boot

```
systemctl enable wpa_supplicant@wlan0
systemctl disable hostapd@wlan0
```

Change radio country

```
qtivap ap wlan0 country LT
```

Country change is persisted between reboots.

Change radio Tx power

```
qcacld txpower 10
```

Tx power is persisted between reboots.

Change radio frequency

```
qtivap channel 60
qtivap channel 5300
```

Channel may be specified using numeric Wi-Fi channels or channel frequency. Frequency value is persisted between reboots.

Change radio channel width

```
qtivap chwidth 80
```

Channel width is persisted between reboots.

9.3 DHCP service

DHCP server is managed by CLI utilities and `systemd` unit. Either DHCP server or client service may be enabled through `systemd` unit files, Server automatically initialises DHCP server config and stores unit files in overlay.

9.3.1 Systemd units

A `systemd` units sets up system during boot and updates configuration on demand during run-time. A template unit services are available to start DHCP server or client on specified interface. Multiple instances may be started and enabled on separate interfaces.

DHCP server unit when setting up will do:

- Initialise DHCP server config for provided interface (if does not exits).
- Start DHCP server service (`udhcpd`).

DHCP service configuration files are stored at `/etc/udhcpd`. Each configuration file is named after corresponding interface name, i.e. `br0.conf`.

9.3.2 DHCP controls

Basic DHCP service config controls are available using `udhcpd-conf` utility. Available utility commands:

Command	Allowed values	Description
init	-	Initialise DHCP service config for interface
from	IP address or start number	Override IP pool range start address
to	IP address or end number	Overrides IP pool range end address
gw	IP address or number	Specifies default gateway IP address
dns	IP address or number	Specifies DNS server IP address
lease	Lease seconds	Specifies DHCP lease time in seconds

NOTE: Creating DHCP server config does not include setting up `udhcpd` service.

9.3.3 Usage examples

Start DHCP server

```
systemctl start udhcpd@br0
```

Interface is required to have IP address assigned before starting the service. If not explicitly configured prior starting up service default DHCP pool params are initialised.

Start DHCP client

```
systemctl start udhcpc@br0
```

When staring DHCP client, server is automatically stopped and vice versa.

Enable DHCP client on boot

```
systemctl disable udhcpc@br0
systemctl disable udhcpd@br0
```

Set up and start DHCP server

```
udhcpd-conf br1 init from 192.168.9.10 to 192.168.9.20 gw 192.168.9.254
systemctl start udhcpd@br1
```

10. Wi-Fi broadcast

Wi-Fi broadcast enables long range ACK-less data transmission using Wi-Fi protocol. Transmission reliability in this case is backed using redundancy encoding and error correction -- FEC.

10.1 WFB-NG

WFB-NG sets up a TAP link for L2 network forwarding over Wi-Fi broadcast link. This enables bi-directional pass-through networking.

10.1.1 Operation

WFB-NG enables communication between two device nodes. Service provides data integrity and authentication facilities using pair of TX and RX keys to encode and decode the data. Same pair of keys must be installed on both machines to establish the data link.

NOTE: Only two devices using the same pair of keys may work at the same frequency and channel width. Having more than two devices causes link disruptions.

10.1.2 Statistics

WFB-NG automatically collects and stores performance statistics to help monitor connection quality and troubleshoot issues. The system tracks real-time metrics from both transmit and receive sides, maintaining both current and historical data. There are two statistics types, described below.

Text Table Statistics

The text table statistics contains human-readable chronological performance data in tabular format, which provides at-a-glance view of link performance that's useful during setup and troubleshooting. Statistics files are located at:

```
/var/stats/wfb-<profile>-tx.txt   # Transmitter statistics
/var/stats/wfb-<profile>-rx.txt   # Receiver statistics
```

Where `<profile>` is the WFB profile name (e.g., "video", "telemetry").

To monitor these statistics in real-time:

```
tail -f /var/stats/wfb-video-rx.txt
```

JSON Statistics

The JSON files makes it easy to integrate with monitoring tools or custom dashboards that can track link quality over time. For programmatic access or integration with monitoring tools, JSON statistics are stored at:

```
/var/stats/wfb-<profile>-tx.json   # Transmitter statistics
/var/stats/wfb-<profile>-rx.json   # Receiver statistics
```

RX STATISTICS FIELDS

Field	Description
rx_packets_all	Total number of packets received by WiFi hardware
rx_bytes_all	Total bytes received by WiFi hardware
rx_corrupt	Number of corrupted packets that couldn't be processed
rx_error	Number of packets with errors detected
rx_packets	Number of packets successfully processed
rx_recovered	Number of packets recovered using FEC
rx_lost	Number of packets that couldn't be recovered
rx_invalid	Number of packets that failed integrity/authentication checks
out_packets	Number of packets forwarded to the application/network
out_bytes	Number of bytes forwarded to the application/network
proto	WiFi protocol being used (HT, VHT, HE)
mcs	Modulation and Coding Scheme index
nss	Number of spatial streams
gi	Guard interval setting
rssi	Received Signal Strength Indicator (in dBm), minimal, average, maximal values for given period
snr	Signal-to-Noise Ratio, minimal, average and maximal values for given period

TX STATISTICS FIELDS

Field	Description
tx_packets	Number of packets transmitted over Wi-Fi
tx_bytes	Number of bytes transmitted over Wi-Fi
tx_dropped	Packets dropped due to queue overflow or timing issues
tx_truncated	Packets truncated because they exceeded maximum size
in_timeout	Timeout events while waiting for input data
in_packets	Number of packets received from the source application
in_bytes	Number of bytes received from the source application

10.1.3 Configuration

Configuration files should be available when starting WFB-NG service. Changing configuration requires WFB-NG service to be restarted.

File	Description
/etc/wfb-ng/tx.key	Tx security key
/etc/wfb-ng/rx.key	Rx security key
/etc/wfb-ng/default.conf	Common defaults configuration file
/etc/wfb-ng/.conf	Profile-specific configuration file

Parameters

Supported configuration file parameters stored in `/etc/wfb-ng/*.conf`. The parameters may be modified directly or through `wfb` configuration controls described below.

Parameter	Value options	Description
mode	tx, rx, tun, tap	Operation mode for the WFB stream
stream_id	0 - 255	Unique stream identifier for matching Tx/Rx streams
security	on, off	Enable/disable encryption
fec_k		FEC k specifies number of chunks required to decode data
fec_n		FEC n specifies number of chunks to produce from data
fec_delay		Delay between packets to improve interleaving and robustness
fec_timeout		Timeout for FEC decoding
proto	ht, vht, he	Transmission protocol/modulation to use
mcs	0 - 12	MCS index, ranges depends on used modulation
nss	1 - 2	Number of spatial streams (for VHT/HE protocols)
sgi	400, 800, 1600, 3200	Guard interval in nanoseconds
listen_port		Port to listen for data on in 'tx' mode
forward_addr		Address to forward data to in 'rx' mode
forward_port		Port to forward data to in 'rx' mode
tapbr		Bridge interface name for TAP mode
tundev		Name of the tunnel interface
tunaddr		IP address for the tunnel interface
tunmask		Network mask for the tunnel interface
tunmtu		Preferred MTU for the tunnel interface
tunka		Keepalive interval for tunnel

10.1.4 Startup

Service automatically starts up on boot when the device mode is set to `wfb` in the system configuration. By default, WFB is configured in TAP mode to provide seamless network traffic forwarding. The system has pre-initialized `tx.key` and `rx.key` security keys for instant operation.

Key generation

Generate new WFB-NG key pair:

```
wfb_keygen
```

This creates `tx.key` and `rx.key` files in `/etc/wfb-ng/` directory. To establish communication between devices:

- Copy generated `rx.key` as `tx.key` to remote device
- Restart WFB-NG service on both devices

10.1.5 Configuration controls

The `wfb` utility provides a convenient way to dynamically update WFB parameters without manually editing configuration files. Changes apply immediately to running services.

Initialize Profile

Initialises new WFB profile/stream with default settings. Creates WFB configuration file at `/etc/wfb-ng/<profile>.conf`. Once initialised, profile configuration parameters may be modified as documented below.

Note: Both `mode` and `stream-id` parameters are mandatory when initializing any new profile.

```
wfb <profile> init mode <tx|rx|tun|tap> stream-id <0-255>
```

Setup Profile

Applies system configuration settings to an existing profile and restarts the associated service. This command loads configuration from `/etc/robonode/system.cfg` and updates the profile accordingly.

```
wfb <profile> setup
```

Remove Profile

Removes a profile and its configuration. Operation includes stopping wfb services (if running), removing profile config and pruning system configuration.

```
wfb prune <profile>
```

Enable Profile

Starts WFB stream and enables WFB stream automatics startup.

```
wfb enable <profile>
```

Disable Profile

Stops WFB stream (if running) and disables WFB stream automatic startup.

```
wfb disable <profile>
```

Operation Mode

Sets the operation mode for the WFB stream:

```
wfb <profile> mode <tx|rx|tun|tap>
```

Mode	Description
tx	Transmit mode (requires LISTEN_PORT)
rx	Receive mode (requires FORWARD_ADDR/PORT)
tun	TUN tunnel mode (L3)
tap	TAP tunnel mode (L2)

When Tx mode additional mandatory argument is required. It specifies UDP port number to listen for data.

```
wfb <profile> mode tx <listen-port>
```

When Rx mode additional mandatory arguments are required. They specify destination UDP IP address and port number to forward the data.

```
wfb <profile> mode rx <forward-addr> <forward-port>
```

Stream ID

Sets the unique stream identifier used to match Tx and Rx streams:

```
wfb <profile> stream-id <0-255>
```

Security Mode

Enables or disables transfer encryption:

```
wfb <profile> security <on|off>
```

FEC parameters

Wi-Fi broadcast FEC (Forward Error Correction) parameters configure redundancy encoding for error correction. The parameters specify absolute numbers of data segments to produce and minimal amount of data segments required to recover the data. Higher redundancy (larger N relative to K) provides better error correction at the cost of bandwidth. For example, K=8, N=12 means 4 chunks (33%) can be lost without data loss.

```
wfb <profile> fec <K> <N>
```

Parameter	Description
K	Number of chunks required to decode data
N	Total number of chunks to produce from data (N>K)

FEC Delay and Timeout

Sets the FEC transmission delay and timeout values:

- FEC delay configures the time in milliseconds to wait before transmitting FEC redundancy packets. This parameter might be used to fine-tune transmission timings.
- FEC timeout configures the time in milliseconds to wait for FEC block completion before closing the block. When timeout expires, missing data packets will be backfilled with dummy data to transmit the FEC redundancy packets. This parameter might be used to tune latency vs. transfer efficiency radio.

```
wfb <profile> fec-delay <delay>
wfb <profile> fec-timeout <timeout>
```

Protocol

Protocol controls Wi-Fi transmission frame format and supported modulations.

```
wfb <profile> proto <ht|vht|he|auto>
```

Value	Description
ht	High Throughput (802.11n) protocol
vht	Very High Throughput (802.11ac) protocol, 5GHz only
he	High Efficiency (802.11ax) protocol
auto	Fallbacks to default `ht` mode

MCS index

MCS controls data transmission rate and modulation. The available MCS indexes and corresponding data rates depend on the selected protocol (HT, VHT, or HE).

```
wfb <profile> mcs <MCS|auto>
```

Protocol	Range	Description
HT (11n)	0-7,8-15	Single/Dual stream MCS indexes
VHT (11ac)	0-9	Per stream MCS indexes
HE (11ax)	0-11	Per stream MCS indexes
auto	Fallbacks to default MCS 3	

NSS count

NSS controls the number of spatial streams used for transmission. Allowed values are 1 or 2. Control is available for VHT (802.11ac) and HE (802.11ax) protocols. Auto fallbacks to default NSS 1.

```
wfb <profile> nss <NSS|auto>
```

Guard interval

Guard interval controls the time between transmitted symbols. Available options depend on the protocol in use.

```
wfb <profile> gi <400|800|1600|3200|auto>
```

Value	Description
400	0.4µs guard interval (short GI, HT/VHT only)
800	0.8µs guard interval (normal GI)
1600	1.6µs guard interval (HE only)
3200	3.2µs guard interval (HE only)
auto	Fallbacks to default GI 0.8us

Tunnel Interface Configuration

Sets TUN/TAP tunnel interface parameters:

```
wfb <profile> tundev <interface>
wfb <profile> tunaddr <ip> [mask]
wfb <profile> tunmtu <mtu>
wfb <profile> tunka <msec>
wfb <profile> tapbr <bridge>
```

Parameter	Description
tundev	Specifies the name of the tunnel interface to be created (e.g., 'wfb0'). Auto-generated when not specified.
tunaddr	Sets the IP address for the tunnel interface, optionally followed by a network mask.
tunmtu	Defines the Maximum Transmission Unit (MTU) for the tunnel interface in bytes. Defaults to 1516 for TUN mode and 1544 for TAP mode.
tunka	Sets the keep-alive interval in milliseconds. The system will send keep-alive ping packets, to maintain the connection. Defaults to 0 (disabled).
tapbr	Specifies the bridge interface name to which the TAP interface should be attached when operating in TAP mode.

10.1.6 Applying multiple parameters

Multiple parameters can be applied at once:

```
wfb <profile> proto vht mcs 3 nss 1 gi 800 fec 6 12
```

10.1.7 Getting parameters

Active WFB parameters can be retrieved using the `get` command. Caller may retrieve either single, multiple or all parameters.

```
wfb <profile> get [param1 [param2] ...]
```

Available parameters: `mode`, `stream_id`, `security`, `fec_k`, `fec_n`, `fec_delay`, `fec_timeout`, `proto`, `mcs`, `nss`, `sgi`, `stbc`, `ldpc`, `listen_port`, `forward_addr`, `forward_port`, `tundev`, `tunaddr`, `tunmtu`, `tunka`, `tapbr`.

Examples:

Get single parameter:

```
wfb video get mode
```

Get multiple parameters:

```
wfb video get mode fec_k fec_n
```

Get all parameters:

```
wfb video get
```

10.1.8 Service control

Before starting WFB-NG service, radio configuration should be set up for this operation mode. Refer to WFB mode setup and system config sections for configuration details.

The WFB system uses two service units: - `wfb@<profile>.service` - Management plane unit that sets everyting up - `wifibroadcast@<profile>.service` - The actual WFB process execution unit

Start service

Before WFB service can be started it must have WFB stream enabled in `WFB_STREAMS`. The service uses system config parameters to set up and launch the service.

```
systemctl start wfb@<profile>.service
```

Stop service

```
systemctl stop wfb@<profile>.service
```

Start WFB gstreamer

The WFB gstreamer provides local USB video camera streaming service. It depends on corresponding profile wfb service which is started automatically before WFB gstreamer.

```
systemctl start wfb-gstreamer@<profile>.service
```

Advanced services control

Start WFB process directly

```
systemctl start wifibroadcast@<profile>.service
```

Stop WFB process

```
systemctl stop wifibroadcast@<profile>.service
```

Stop all WFB service instances

```
systemctl stop wifibroadcast.service
```

10.1.9 Example

- Setup `WFB_STREAMS` variable in `/etc/robonode/system.cfg`

```
WFB_STREAMS="video telemetry"
```

- Configure stream parameters

```
# Video stream configuration
WFB_MODE_video=tx
WFB_STREAM_ID_video=0
WFB_FEC_K_video=8
WFB_FEC_N_video=12
WFB_LISTEN_PORT_video=5600

# Telemetry stream configuration
WFB_MODE_telemetry=rx
WFB_STREAM_ID_telemetry=1
WFB_FEC_K_telemetry=1
WFB_FEC_N_telemetry=2
WFB_FORWARD_ADDR_telemetry=127.0.0.1
WFB_FORWARD_PORT_telemetry=14550
```

- Start the services

```
systemctl start wfb@video.service
systemctl start wfb@telemetry.service
```

Switching Preconfigured Profiles

The system includes four preconfigured WFB profiles for different use cases:

Profile	Mode	Stream ID	Purpose	Default configuration
`bridge`	TAP	2	Network bridging	Bridge interface: `br0`, MTU: 1544
`dial`	TUN	0	Point-to-point tunneling	IP: 10.20.30.1/24, MTU: 1516
`drone`	TX	1	Drone transmission	Listen port: 5600
`gs`	RX	1	Ground station reception	Forward to: 127.0.0.1:5600

SWITCHING FROM BRIDGE TO DRONE PROFILE

```
wfb bridge disable
wfb drone setup enable
```

SWITCHING FROM DRONE TO GROUND STATION PROFILE

```
wfb drone disable
wfb gs setup enable
```

CHECK CURRENT PROFILE STATUS

```
# List active profiles
wfb list

# Check profile configuration
wfb gs get
```

10.2 RTP duplicator pipeline

WiFi Broadcast supports RTP (Real-time Transport Protocol) packet duplication and deduplication to improve video streaming reliability over lossy wireless links. This feature adds configurable packet redundancy by duplicating packets at the transmitter and removing duplicates at the receiver.

10.2.1 Data Flow with RTP

TX Pipeline (Transmitter/Drone)

```
Video Source → [RTP_LISTEN_PORT] → RTP duplication → [WFB_LISTEN_PORT] → WFB Tx → WiFi
```

RX Pipeline (Receiver/Ground Station)

```
WiFi → WFB Rx → [WFB_FORWARD_PORT] → RTP deduplication → [RTP_FORWARD_PORT] → Video Output
```

10.2.2 Configuration

RTP pipelines are configured per-stream in the system configuration. Each RTP pipeline is mapped 1:1 to a WFB stream using the same stream name suffix.

Example configuration for drone video streaming:

```
# Enable RTP for drone stream (TX mode)
RTP_PIPELINE_drone=on
RTP_LISTEN_PORT_drone=5010
RTP_DUP_DELAY_drone=50

# Enable RTP for ground station stream (RX mode)
RTP_PIPELINE_gs=on
RTP_FORWARD_ADDR_gs=192.168.1.100
RTP_FORWARD_PORT_gs=5011
RTP_UNDUP_WINDOW_gs=100
```

10.2.3 Performance Impact

The duplication delay and deduplication window settings control the trade-off between latency and packet loss resilience:

Duplication Delay	Deduplication Window	Characteristics
50ms	100ms	Lower latency, handles shorter packet loss series
100ms	150ms	Balanced setting, handles typical packet losses
200ms	250ms	Higher latency, improves resilience to prolonged packet loss series

Recommended: 100ms duplication delay provides good results in most scenarios, effectively handling multiple consecutive packet losses. The deduplication window should be slightly higher than the duplication delay to account for data transmission and processing duration.

10.2.4 Statistics

When `RTP_MONITOR=on` is configured, detailed JSON statistics are written to: - `/var/stats/rtpdup-<stream>.json` - Transmitter duplication statistics - `/var/stats/rtpundup-<stream>.json` - Receiver deduplication statistics - `/var/stats/rtpmon-<mode>-<stream>.json` - RTP monitoring statistics

11. USB system update

System resources can be updated through USB using either supported method:

- Connecting device to PC as mass storage device.
- Attaching USB flash drive or disk to the device.

Each of methods has different procedure defined below.

11.1 Configuration files

Recognised configuration files expected to be stored in USB storage.

Filename	Description
`system.cfg`	System configuration file
`firmware.zip`	Software upgrade file
`tx.key`	Wi-Fi broadcast Tx encryption/decryption key
`rx.key`	Wi-Fi broadcast Rx encryption/decryption key

Please refer to system config reference section for system configuration file options.

11.2 System configuration and upgrade using PC (USB device mode)

This update method uses device as a storage device to maintain system configuration. Once device is connected to PC over USB cable it appears as a storage disk/partition. While connected to the PC, device will wait for the user to complete modifying configuration files or optionally uploading firmware. After unplugging from the PC, updated configuration is applied in the system. Upon finishing configuration update procedure, activity log is saved into storage for future inspection. Next, if a firmware image file is detected in storage, a software upgrade is attempted. Upon successful completion device is automatically rebooted, otherwise log file is saved into storage for future inspection.

11.2.1 Preparation

Refer to hardware reference document to find right peripherals on board. For this procedure you will need:

- USB gadget port
- Configuration LED

11.2.2 Update procedure

- Connect the PC to the device's "USB gadget" port.
- Once USB connected device enters configuration state and indicates this by turning on "configuration LED".
- Device appears on PC as mass storage device. If needed mount it as FAT32 file system.
- Storage will have current active configuration files, and active configuration stored in `system.cfg.ref`. Make desired configuration changes in `system.cfg` configuration file or add firmware upgrade file if firmware upgrade is desireable.
- Safely remove / eject / unmount the storage and unplug the USB cable. *Note:* Unplugging USB cable without ejecting storage might not save configuration changes or corrupt the data.*
- After unplugging the USB cable, the device automatically applies the configuration changes to the system and indicates operation by fast blinking "configuration LED". Applied configuration is moved to `system.cfg.ref` file. If the device detects firmware upgrade file in the storage, it will attempt to upgrade, indicated by slow blinking "configuration LED". At this point, the device will either reboot or exit the configuration state, indicated by shutting off the "configuration LED". If the device has not reboot, the upgrade log will be saved in the storage file named `import-config.log` for future inspection.

11.3 System configuration using USB flash disk (USB host)

This update method uses externally connected USB flash drive to retrieve the configuration data and apply settings in the system or perform software upgrade. Once plugged in device will automatically attempt to mount the drive storage, import settings, apply configuration to the system and if software image is found - upgrade the firmware. Configuration update activity log and software upgrade log are saved into the flash drive storage for future inspection.

11.3.1 Preparation

Refer to hardware reference document to find right peripherals on board. For this procedure you will need:

- USB host port
- Configuration LED

Ensure that flash drive has single partition and is formatted to FAT32 file system.

11.3.2 Update procedure

- Prepare and store configuration files and/or the software upgrade file on the root of USB flash disk.

- Plug in the USB flash disk into the device's "USB host" port.
- Once valid flash drive is detected device enters configuration state and indicates this by turning on "configuration LED".
- Device configuration import and apply is started automatically which is indicated by fast blinking "configuration LED". This is short operation thus LED may blink once or twice. Once complete, configuration log is saved to `import-config.log` on the root of USB flash disk.
- Software upgrade procedure will start automatically if software upgrade image is found on USB flash disk. This procedure is indicated by slow blinking "configuration LED". The device is rebooted automatically after successful upgrade. An upgrade log is saved to `upgrade.log` on the root of USB flash disk.
- Unplug USB flash disk to leave the configuration state and "configuration LED" is turned off.

12. USB Gadget setup

12.1 USB gadget configuration

USB gadget implementation uses the configfs framework for creating configurable composite USB gadgets. Gadgets are managed via the `usb-gadget` CLI utility installed at `/usr/bin/usb-gadget` and started on boot by `usb-gadget.service`.

The active USB mode is persisted in `/etc/default/usb-gadget` configuration file. Default mode is `single` (all functions on USB2 port).

Multiple gadgets may be created, however only one gadget may be bound to a USB Device Controller (UDC) at a time. Valid UDCs may be found at `/sys/class/udc`.

Currently implemented functions:

- RNDIS Ethernet;
- ACM Serial;
- Android Debug Bridge;
- Mass Storage;

Current implementation supports either single USB2 port mode, through which all USB gadget functions are functioning, and twin USB mode, in which both USB3 and USB2 ports are utilised as gadgets, and split function assignments in between. See Switching between twin and single USB gadget modes below on how to switch between these two modes.

12.1.1 User-editable values

Many of the identifying strings may be changed during runtime, though a USB cable re-plug may be necessary to see the changes.

Identifier	Location	Format
Name of the USB composite gadget	`<gadget_directory>/strings/0x409/product`	Any ASCII string
Vendor name (visible on USB lists, e.g. `lsusb`)	`<gadget_directory>/idVendor`	Hexadecimal number in `0x0000`
Android Debug Bridge unique device ID	`<gadget_directory>/strings/0x409/serialnumber`	Any ASCII string

Valid vendor IDs in decimal format may be found at usb.org;

12.2 Modification examples

12.2.1 Switching between twin and single USB gadget modes

Use the `usb-gadget` CLI utility to switch modes:

```
usb-gadget setup single
```

```
usb-gadget setup twin
```

Mode change takes effect immediately and is persisted across reboots.

To check the current mode and gadget state:

```
usb-gadget status
```

A temporary diagnostics mode is also available. It provides only ADB access and reverts to the persisted mode on reboot:

```
usb-gadget setup diag
```

12.2.2 Removing a function from a USB gadget

Gadget function composition is managed in the `usb-gadget` utility source. Functions may be removed by modifying the utility and rebuilding.

12.2.3 Moving a gadget function from one composite USB gadget to another

Note: this example assumes twin USB gadget configuration.

Gadget function assignments between USB2 and USB3 ports are managed in the `usb-gadget` utility source and may be changed by modifying the utility.

12.2.4 Adding additional mass storage function

- Create a new mass storage directory with a different instance name;

```
mkdir -p functions/mass_storage.example`
```

- Add or modify additional flags in that directory as required;

```
echo 1 > functions/mass_storage.example/lun.0/ro
echo 0 > functions/mass_storage.example/lun.0/cdrom
echo 1 > functions/mass_storage.example/lun.0/nofua
echo /dev/mmcblk0p38 > functions/mass_storage.example/lun.0/file
```

- Symlink the new directory to the gadget's configuration directory

```
ln -sf functions/mass_storage.example configs/c.1/ms_example
```

After applying the USB Device Controller to the gadget's `UDC` file, the new gadget will be applied.

12.3 Raw configuration example

This is a brief example on how to manually configure the USB gadget without the script. A more in-depth guide may be found at docs.kernel.org.

```
mkdir -p /sys/kernel/config/usb_gadget/g1
cd /sys/kernel/config/usb_gadget/g1
mkdir -p strings/0x409
echo "Product name" > strings/0x409/product
echo "Manufacturer name" > strings/0x409/manufacturer
echo "serial_number" > strings/0x409/serialnumber
echo "0xaabb" > idVendor
echo "0xaabb" > idProduct
mkdir -p configs/c.1
# mkdir -p functions/<function>.<instance_name>
mkdir -p functions/rndis.example
ln -sf functions/rndis.example configs/c.1/rndis
echo "7580000.dwc3" > UDC
```

Removing the gadget may be done by the following brief example below.

```
# Unbind the USB gadget by echoing 'none' to UDC
echo none > UDC

# Remove the directory
cd ..
rm -rf ./g1
```

13. TobuFi UART

Refer to TobuFi DVK board section for available UART ports.

13.1 Software configuration

UART ports are configured in TobuFi-DVK DTS files. The base configuration is in `qcs405-tobufi-dvk.dts` with revision-specific overrides in `qcs405-tobufi-dvk-rev3.dts`, `qcs405-tobufi-dvk-rev4.dts`, and `qcs405-tobufi-dvk-rev5.dts`. These can be found in `meta-8dev-bsp` layer `linux-msm` package files.

By default UART ports function is enabled. It may be disabled setting

```
&blsp1_uart2_hs {
        status = "disabled";
};
```

```
&blsp2_uart1_hs {
        status = "disabled";
};
```

DTS reference	Board identifier
`blsp1_uart1_hs`	UART/GPIO block A0 (rev4+ only)
`blsp1_uart2_hs`	UART/GPIO block A1
`blsp2_uart1_hs`	UART/GPIO block 5

Note: BLSP0 UART (block A0) is available only on rev4 and later boards. On rev3, BLSP0 GPIOs are not exported.

13.2 Software mapping

Linux drivers registers TTY character devices for each DTS configured UART node. The ttyMSM0 is primary debug serial console. Kernel uses it to log process.

TTY device	Board identifier
/dev/ttyMSM0	Serial console
/dev/ttyHS0	UART/GPIO block A0 (rev4+ only)
/dev/ttyHS1	UART/GPIO block A1
/dev/ttyHS2	Internal WCN39xx UART
/dev/ttyHS3	UART/GPIO block 5

Note: On rev3 boards, UART block A0 (`/dev/ttyHS0`) is not available. The TTY device numbering shifts accordingly: block A1 maps to `/dev/ttyHS0`, WCN39xx to `/dev/ttyHS1`, and block 5 to `/dev/ttyHS2`.

13.3 Examples

To enable serial console on TTY HS5 (UART block 5) connect serial console TTL cable to `UART/GPIO block 5` and execute in linux shell:

```
/sbin/agetty -n -a root --noclear ttyHS5
```

14. Robonode UART

Refer to Robonode board section for available UART ports.

14.1 Software configuration

UART ports are configured in Robonode DTS `qcs405-tobufi-robonode.dts`. It can be found in `meta-8dev-bsp` layer `linux-msm` package files.

By default UART ports function is enabled. It may be disabled setting

```
&blsp1_uart1_hs {
        status = "disabled";
};
```

```
&blsp1_uart2_hs {
        status = "disabled";
};
```

```
&blsp2_uart1_hs {
        status = "disabled";
};
```

DTS reference	Board identifier
`blsp1_uart1_hs`	UART/GPIO block#0
`blsp1_uart2_hs`	UART/GPIO block#1
`blsp2_uart1_hs`	UART/GPIO block#5

14.2 Software mapping

Linux drivers registers TTY character devices for each DTS configured UART node. The ttyMSM0 is primary debug serial console. Kernel uses it to log process.

TTY device	Board identifier
/dev/ttyMSM0	Serial console
/dev/ttyHS0	Internal WCN39xx UART
/dev/ttyHS1	UART/GPIO block A0
/dev/ttyHS2	UART/GPIO block A1
/dev/ttyHS3	UART/GPIO block 5

14.3 Examples

To enable serial console on TTY HS0 (UART block#5) connect serial console TTL cable to `UART/GPIO block#0` and execute in linux shell:

```
/sbin/agetty -n -a root --noclear ttyHS1
```

14.4 Serial to network

14.4.1 Service control

Before UART service can be started it must have TTY device enabled in `UART_DEVS`. The service uses system config parameters to set up and launch the service.

Start service

```
systemctl start uart@<tty>.service
```

Stop service:

```
systemctl stop uart@<tty>.service
```

Advanced services control

Start service

```
systemctl start ser2net@<tty>.service
```

Before starting the service it is required that configuration file for appropriate TTY device would be stored in `/etc/ser2net.d/<device>.conf`.

Stop service

```
systemctl stop ser2net@<tty>.service
```

Stop all services instances

```
systemctl stop ser2net.service
```

14.4.2 Example

- Setup `UART_DEVS` variable in `/etc/robonode/system.cfg`

```
UART_DEVS="ttyHS1"
```

- Configure device parameters

```
UART_SPEED_ttyHS1=115200
UART_ACCESS_PORT_ttyHS1=3100
```

- Start the services

```
systemctl start uart@ttyHS1.service
```

- Connect to the UART device using netcat

```
nc 127.0.0.1 3100
```

15. Software

15.1 Features

- Kernel v4.14
- SD card storage support
- HDMI and MIPI DSI display support
- Dual-band 2.4 GHz + 5 GHz or 2.4 GHz + 6 GHz Wi-Fi radios
- Single band operation (2.4 GHz or 5 GHz/6 GHz, non concurrent)
- Wi-Fi narrow channels
- Wi-Fi frequencies shifting
- Wi-Fi extended channels range
- Multiple Wi-Fi modes support: BSS, NAWDS, WFB
- USB device mode storage, ethernet, serial functions
- USB device ADB service access
- Single configuration file
- System update from USB flash drive or USB device storage
- Software dual-boot update
- Software versioning support
- Wi-Fi broadcast uni-directional streaming
- Wi-Fi broadcast bi-directional pass-through
- USB video input streaming support

16. Releases

16.1 v0.0.1

- TobuFi-DVK BSP
- Wi-Fi 6 Radio #1 (Pine)
- Wi-Fi 5 Radio #2 (Cherokee)
- 1000M Ethernet port
- USB host/gadget ports

- SD card support
- MIPI DSI support
- HDMI support
- Robonode BSP
- Wi-Fi 6 Radio #0 (Pine)
- Wi-Fi 5 Radio #1 (Cherokee)
- 1000M Ethernet port
- USB host/gadget ports
- QSDK SPF-12 qca-wifi driver
- Narrow (5MHz, 10MHz) channels support
- Extended frequency range support
- Compliance testing country support
- USB device mode storage, ethernet, serial functions support
- USB ADB service access
- Software versioning support
- Single configuration file
- System config sync from USB flash drive
- System config sync using USB device storage
- OpenHD unidirectional Wi-Fi broadcast streaming
- Video source USB camera
- Video source IPTV stream

16.2 v0.1.1

16.2.1 Fixes

- Fixed ADB service startup and detection on Windows
- Fixed ETH 2-pair 100M speed autoneg
- Fixed kernel warning during startup, shutdown and interface state change
- Fixed Yocto build warnings
- Fixed systemd hostapd startup order
- Fixed USB ethernet RNDIS gadget configuration

- Fixed USB gadget config sync cable detection
- Fixed USB storage system update log file reports

16.2.2 Improvements

- Updated user guide documentation
- Updated firmware versioning and version metadata in rootfs
- Added OpenHD forwarding IP addresses setting
- Improved LEDs indication during USB storage update
- Updated GPIO fan integration into linux thermal framework

16.2.3 Features

- Enabled OTA system update image build support
- Added OTA system update image software upgrade support
- Added device mode switch using HW switch button
- Added factory reset using HW push button
- Changed default device mode to `drone`
- Added software update support via USB storage
- Added EEPROM production data storage support
- Added EEPROM production information recovery

16.3 v0.1.2

16.3.1 Fixes

- Fixed upstream repositories migrated default branch handling

16.3.2 Improvements

- Update build environment and config initialisation
- Improved EDL recovery flash script image detection
- Updated Robosoft user guide documentation

16.4 v0.1.3

16.4.1 Improvements

- Refine USB configuration sync using device storage and flash disk
- Added USB gadget cable plug/unplug events handling

16.4.2 Features

- Improved calmode startup sanity checking
- Optimised calmode startup and reconfiguration time
- Added calmode option to start calibration services
- Added calmode active status reporting

16.5 v0.1.4

16.5.1 Fixes

- Fixed USB configuration sync handling using device storage
- Fixed Pine radio `netconfi` radio parameters controls
- Fixed device crash in `ground-station` mode when working in 2GHz band
- Fixed partition mount dependencies loop
- Fixed ADSP and CDSP firmware load

16.5.2 Improvements

- Increased HW button press delay to start factory reset
- Updated OpenHD unit service start/stop dependencies
- Improved OpenHD run-time statistics file logging
- Improved hostapd and wpa_supplicant service units startup
- Allow network IP address setup without carrier
- Updated default SSID name and radio settings

16.5.3 Features

- Added factory reset and device mode LEDs indication
- Added dual-band configuration control for Wi-Fi broadcast

- Added AP and STA device mode set up support
- Added Pine radio Tx power control
- Added dummy video input source for stream testing
- Added Pine radio general boot configuration file
- Added CLI utility to control Pine radio settings
- Added CLI utility to manage Pine VAP configs

16.6 v0.1.5

16.6.1 Fixes

- Patched systemd and lz4 upstream source fetch issues
- Fixed hostpad FREQLIST and CHANNEL configuration
- Fixed systemd startup permissions issues
- Fixed WPA-PSK security clients authentication

16.6.2 Improvements

- Introduced `plane` base layer for QSDK flavoured builds
- Extended version string formatting with labels
- Improved CI built SW images deployment
- Deploy latest release image to dedicated storage
- Moved in Robonode board support
- Restored original Wi-Fi driver for BSP builds

16.6.3 Features

- Added NAWDS configuration support for Pine radio
- Linux 6.6 kernel and ath10k/ath11k prototype support
- Added Pine radio band control support in ath11k driver
- Added hidden SSID and auto-generated SSID name config support

16.7 v0.2.0

16.7.1 Fixes

- Fixed GPIO fan thermal trigger activation

- Fixed frequency offset reconfiguration
- Fixed USB ethernet gadget interface bridging
- Patched NAWDS mode peer set up failures after band change
- Fixed calibration services startup over ADB
- Fixed mount warnings for initial overly mount
- Fixed HS UART ports power saving
- Fixed Pine radio signal level reporting
- Fixed USB ethernet gadget endpoint config on Windows
- Fixed HDMI port setup on TobuFi-DVK (HW fixes required)
- Fixed gtest/chrony packages source fetch failures

16.7.2 Improvements

- Updated to Yocto dunfell release
- Removed dual-role Wi-Fi broadcast setup
- Refactored system configuration file
- Removed excessive dual-boot partitions layout
- Moved sub-processor firmware images to rootfs
- Simplified fastboot flashing procedures
- Added fastboot flashing partitions validation
- Add run-time userdata/overlay initialisation
- Removed bash as default SDK shell interpreter requirement

16.7.3 Features

- Added UART forwarding to network using ser2net
- Added bi-directional WFB-NG streaming support
- Prototype HE mode Wi-Fi injection support
- Added internal Wi-Fi 5 radio VAP control utilities
- Added DHCP client and server control utilities

16.8 v0.2.1

16.8.1 Fixes

- Fixed SDK build warnings
- Fixed calibration services startup for Pine and internal radios
- Fixed calibration EEPROM read in production
- Fixed firmware images install and updated internal radio BDF

16.8.2 Improvements

- Updated documentation board images and configuration description
- Updated documentation ADB host tools setup instructions
- Added USB ports configuration description

16.9 v0.2.2

16.9.1 Fixes

- Fixed calibration services startup for internal radio

16.9.2 Improvements

- Added Pine radio modulation parameters controls
- Improved radios and network management utilities logging

16.9.3 Features

- Added Wi-Fi broadcast FEC parameters configuration
- Added Wi-Fi modulation parameters configuration for Pine radio
- Added WFB-NG configuration update utility
- Added Pine radio Wi-Fi modulation and WFB FEC parameters controls
- Added device-hosted management Web UI application support

16.10 v0.2.3

16.10.1 Fixes

- Fixed WFB custom interfaces configuration for all modes

- Fixed ADSP audio sub-processor firmware load
- Removed implicit Pine radio defaults setup

16.10.2 Improvements

- Updated WFB MCS and NSS configuration control
- Compatibility production eeprom udpate
- Optimised software package composition and boot duration

16.11 v0.2.4

16.11.1 Fixes

- Fixed internal radio startup order
- Fixed Pine radio startup order

16.12 v0.3.0

16.12.1 Fixes

- Fixed local device CORS request support
- Fixed DHCP client IP fallback setup
- Fixed oversize packets WFB forwarding
- Fixed USB serial port setup
- Fixed unstable bridge MAC address
- Fixed RAW packet injection radiotap defaults init
- Fixed RAW injection radiotap short GI, MCS, NSS, BW parse
- Fixed Pine radio extended channels setup crash
- Disabled USB flash disk boot-time provisioning

16.12.2 Improvements

- Bump to latest stable 25.01 WFB-ng version
- Simplified wfb-ng services launcher
- Improved WFB-ng services termination and reload
- Added tunnel MTU configuration control
- Updated user guide styling

- Updated system startup organisation
- Updated UART startup organisation and TTY devices order
- Optimised Pine radio band change
- Updated VAP services channel/frequency setup
- Updated configuration reload procedure
- Updated USB provision SW upgrade logging
- Switch to default MCS 3 NSS 1 on Pine radio
- Switch to DHCP client mode by default
- Switch to 2GHz radio band on Pine radio by default
- Enabled all UART devices network forward by default
- Updated USB storage provision LEDs indication
- Allow removing network IP address
- Improved internal radio BDF configuration
- Added RAW Tx injection bursting support
- Added RAW Tx injection QoS controls

16.12.3 Features

- Added system config versioning support
- Added Wi-Fi protocol (HT,VHT,HE) support
- Added Wi-Fi CCA threshold control
- Added multi WFB profiles/streams support
- Added TUN tunnel mode support
- Added WFB-ng unsecured transfer control option
- Added WFB-ng FEC redundancy disabling support
- Added WFB-ng session exchange interval control option
- Added WFB-ng linux qdisc configuration control
- Added WFB stats collection and reporting support
- Added radios and WFB information retrieval via CLI
- Added WFB link tester utility
- Restored local video input streaming over WFB stream

16.13 v0.3.1

16.13.1 Fixes

- Fixed configuration issues during boot
- Fixed CCA threshold not being applied during boot

16.13.2 Improvements

- Increased ACK timeout to 255 µs for NAWDS/BSS modes
- Added option to clean up overlay during software upgrades
- Added multi-board support
- Changed default kernel tick rate to 1000 Hz
- Added option to force `reload-config`

16.13.3 Features

- Added monitor interface support for NAWDS/BSS modes
- Added smart duplication of RTP stream data

16.13.4 Known Issues

- On rare occasions, the device may get stuck in the bootloader after a firmware upgrade or reboot
- Occasionally, Cherokee (radio1) startup may fail, preventing the AP from starting
- Station mode does not work on Cherokee (radio1) radio
- NAWS mode might be unstable in 5/10MHz modes

16.14 v0.3.2

16.14.1 Fixes

- Fixed MAC address handling for duplicate EEPROM entries
- Fixed enhanced Pine radio offload stats enabling
- Fixed occasional stuck in bootloader after device reboot
- Fixed unconfigured profile options output in WFB-NG
- Fixed radio mode setup after boot when WFB drone profile is set
- Fixed internal radio interfaces bridging for more intuitive non-WDS STA setup

- Fixed radio reconfig failures due to premature configuration state discard

16.14.2 Improvements

- Added default config fallback when missing system configurations
- Added atomic configuration file updates
- Refactored startup logging and config operations
- Relocated volatile tun-time system configuration state files
- Improved USB provisioning with empty file guards
- Relocated USB state files to dedicated state location

16.14.3 Features

- Added automatic background spectral analyzer support
- Spectral scan performed on current running frequency in 20MHz channel width
- Spectral scan results periodically dumped to volatile stats files (persisted until reboot)
- Added manual spectral analyzer capture with frequency range control
- Added interface throughput background monitoring support
- Added diagnostic troubleshooting package for basic system diagnostics

16.14.4 Known Issues

- Internal (radio1) station mode WDS bridging is not supported
- Internal (radio1) Tx power control does not work
- NAWDS mode might be unstable in 5/10MHz modes
- Exiting calibration mode leaves internal (radio1) unusable, reboot required.

16.15 v0.4.0

16.15.1 Fixes

- Clarified TobuFi radios operating frequencies rages and maximal Tx power
- Fixed radio statistics collect after reconfiguration
- Fixed Pine radio HE mode operation in narrow channels
- Fixed NAWDS setup to use 2 NSS chains by default
- Fixed NAWDS NSS and guard interval reconfig

- Fixed false radio parameters error when starting scan with negative freq shift

16.15.2 Improvements

- Added 8devices documentation branding
- Collect VAP interface pcap during diagnostic troubleshoot
- Speed up NAWDS local ARP cache update
- Added clibration BDF use from custom location support

16.15.3 Features

- Added premium features licensing
- Future compatibility v1.0 TVL eeprom structure support
- Unified TobuFi-DVK and Robonode Robosoft SW images
- Unified flash/recovery images for all TobuFi-DVK and Robonode HW revisions
- Unified Robosoft documentation for TobuFi-DVK and Robonode devices

16.15.4 Known Issues

- Internal (radio1) station mode WDS bridging is not supported
- Internal (radio1) Tx power control does not work
- Exiting calibration mode leaves internal (radio1) unusable, reboot required.
- CLI config tools shows invalid error message when passed invalid arguments
- CLI config tools via ADB works incorrectly
- Occasional driver crash in 2GHz band 40MHz channel width on crowded channel

16.16 v0.4.1

16.16.1 Fixes

- Fixed USB device initialization in twin mode
- Fixed USB device mode configuration failures caused by persistent state between reboots
- Fixed system crash when internal radio board data is invalid
- Fixed continuous radio reinitialization on radio service failures
- Fixed system crash on DSP subsystem failures
- Fixed CLI tools control parameters validation

- Fixed CLI config tools operation via ADB

16.16.2 Improvements

- Allow configurable tool paths in fastboot flashing utility
- Updated ADB service logging and connection handling
- Added data partition preservation option during EDL recovery flash
- Force NAWDS reconfiguration on country code change

16.16.3 Features

- Added TobuFi-DVK rev 05 hardware support

16.16.4 Known Issues

- Internal (radio1) station mode WDS bridging is not supported
- Internal (radio1) Tx power control does not work
- Exiting calibration mode leaves internal (radio1) unusable, reboot required.
- Occasional driver crash in 2GHz band 40MHz channel width on crowded channel