Falcon Player (FPP) User Manual

Version 10.x

The FPP Community

2026

Table of Contents

About This Manual

This manual covers **Falcon Player (FPP) version 10**. FPP 10 introduces a significant number of changes over the 9.x series, including a refreshed user interface, a reorganized Settings page, a new audio/video pipeline based on PipeWire and GStreamer, and expanded MultiSync and health-monitoring features.

If you need a manual for an older version, see the Falcon Christmas.github repository.

The screenshots in this manual were taken from a running FPP 10 system. Your screens may differ slightly depending on your hardware platform (Raspberry Pi, BeagleBone, or a generic Linux/Docker host), the capes or hats you have installed, and your **UI Level** setting (see the FPP Settings → UI section). Screens that require specialised cape hardware are noted where they appear.

> **Tip:** Press **F1** on any FPP page to open context-sensitive help.

Conventions used in this manual

- **Bold** text indicates on-screen buttons, menu items, tab names, and settings.
- Menu paths are written as **Menu → Sub-item**, e.g. **Status/Control → FPP Settings**.
- Notes and warnings are called out in indented blocks.

What's new in FPP 10

The most visible and important changes since 9.x include:

- **Redesigned UI** – a new top navigation bar grouped into *Status/Control*, *Content Setup*, *Input/Output Setup*, and *Help*, with a persistent header showing host name, player state, CPU temperature, IP addresses and time.
- **Reorganised FPP Settings** – settings are now split across clearly labelled tabs (Playback, Audio/Video, Localization, UI, Email, MQTT, Privacy, Input/Output, Logging, Services, Storage, System, Developer). Each setting is tagged with a *UI Level* marker so you can choose how much detail to see.
- **New audio/video pipeline** – audio and video routing is now handled by **PipeWire** with **GStreamer**, adding audio output groups (per-output delay and EQ), a routing matrix, AES67 and Opus network audio, and video output/input groups. This is significant enough to have its own chapter, *The PipeWire Audio & Video Pipeline*.

- **Pixel Overlay Model improvements** – a model **preview**, plus support for xLights **submodels** and **model groups**, all addressable by name through the overlay commands.
- **Improved MultiSync** – faster remote discovery, clearer remote status, and more reliable synchronised playback across players and remotes.
- **System Health Check** – a consolidated health page that surfaces warnings about storage, services, audio/video, temperature and configuration issues.
- **Expanded plugin and package management** – a reworked Plugin Manager and a Packages page for installing optional software.
- **Interactive API explorer** – a built-in, browsable reference for FPP's REST API (**Help → REST API**), letting you inspect and try every endpoint on the device. It replaces the old static API help page, and plugins can now document their own HTTP routes alongside FPP's. See the *Protocols, Ports and the API* appendix.
- **Variables and Recurring Tasks** – FPP can now store named **Variables** and act on them: a **Set Variable** command stores values (fixed, counter, random, or a calculated expression), an **If** command branches on them, and any command's text field can substitute one with `%VAR:name%`. **Recurring Tasks** run a command or preset on a repeating interval and can capture its result into a variable. Both are on the *Content Setup* menu at the **Advanced** UI level and have their own chapters, Variables and Recurring Tasks.
- **Categorized FPP Commands** – the command list is now grouped (Audio, Effects, Events, Media, Playlist, Pixel Overlay, Outputs, System) and each command is tied to a UI Level, so the Basic level shows only everyday commands. Command arguments now carry their own help tooltips.
- **Temporary Advanced UI level** – you no longer have to change your UI Level permanently to reach one Advanced setting. A button on FPP Settings → UI unlocks the Advanced UI for 15 minutes, with an unlock icon in the header while it is active.
- **Reworked Plugins page** – a card/grid layout with plugin icons, categories, popularity, live counters and filtering, an *Installed / Available / Updates* tab strip, and a single search box that also accepts a `pluginInfo.json` URL. Plugins can now be loaded and unloaded without restarting FPP.
- **Playlist improvements** – the entry editor has been redesigned into labelled sections with help tooltips, and a media entry can now carry **companion media** that lives and dies with it.
- **Calendar view of the schedule** – alongside the weekly preview, the Scheduler offers a month/calendar view of what will actually run.

- **Support bundle** – Help → Troubleshooting Commands can package the logs, configuration and health-check data into a single zip to attach to a support request.

Each of these areas is covered in detail in the relevant chapter.

> **Note:** FPP 10 continues to be developed after release, so a very recent build may show items this manual does not yet describe. The version you are running is shown at the top-left of every page and on *Help → System Upgrade*.

1 Introduction

The **Falcon Player (FPP)** is a lightweight, optimized, feature-rich sequence player designed to run on low-cost Single Board Computers (SBCs). It was originally created to run on the $35 Raspberry Pi — hence the middle 'P' in the short name — but FPP now supports many more systems. The "FPP" shorthand is still used, but the software is now simply called **Falcon Player**. FPP is a software solution that you download and install onto hardware which can be purchased from numerous sources around the internet.

FPP aims to be controller-agnostic: it can output **E1.31**, **DDP**, **DMX**, **Pixelnet**, **Renard** and many other output types to hardware from multiple vendors. This includes controller hardware from Falcon Christmas (`http://pixelcontroller.com`) and Kulp Lights (`http://kulplights.com`), as well as others.

Up until the end of the 2015 Christmas season, most users ran FPP on the Raspberry Pi as the main player. Since then it has expanded, with the BeagleBone series of SBCs being widely used as well.

FPP can interface to a number of controllers. It can also play synchronised audio (via an audio port / FM transmitter) and synchronised video (via HDMI), it supports USB devices and external interfaces via the GPIO bus, and it can drive pixels directly via the GPIO bus. Many people use FPP as the show player by connecting it to one or more DDP/E1.31/DMX controllers and running their light show sequences and audio from it. Others use several FPP-based controllers operating in various modes to run their shows, play videos from a remote projector, control animatronics, or handle outside events — all synchronised to the main (Player) FPP.

A Raspberry Pi running FPP can be used with a cape to act as a controller for a small matrix (36 P10 or 15 P5 panels) or 2 strings of pixels using the standard Raspberry Pi outputs, or 24 strings using the DPIPixels outputs (approximately 1600 pixels per string at 20 fps, or 800 pixels per string at 40 fps; note that DPIPixels only supports 20 and 40 fps).

> **Note:** The DPIPixels string outputs require a license to control more than 50 pixels per port on more than 2 ports, The first two ports are fully functional without a license. See the Pixel Port Licensing chapter.

The BeagleBone series SBCs have been used extensively with a cape to drive up to 128 P10 or 64 P5 panels (depending on the cape and BeagleBone type — check with the vendor to determine capacity). The BeagleBone series can also support other capes and act as a controller, such as the K4-PB, F8-B/K8-B, F16-B/K16A-B, F32-B/K32A-B, F8-PB/K8-PB, F40D-PB, K40D-PB, OctoPlus, and so on.

This manual covers the functional aspects of installing, configuring and operating FPP — the most popular show player for animated holiday lighting displays. It has been updated for **FPP version 10**; see About This Manual → What's new in FPP 10 for a summary of the changes since the 9.x series.

1.1 Supported hardware

The current version of Falcon Player runs on the following hardware:

- Raspberry Pi 2 Model B
- Raspberry Pi 3 Model B / B+ / A+
- Raspberry Pi 4 Model B
- Raspberry Pi 5
- Raspberry Pi Zero (a micro-USB hub may be needed for network access)
- Raspberry Pi Zero-W / Zero-W2
- BeagleBone Black (Rev C) / Black Wireless
- BeagleBone Green / Green Wireless (Green Wireless is not recommended with capes)
- BeagleBone Gateway
- PocketBeagle / PocketBeagle 2

The philosophy of the FPP developers is to make FPP as easy to install and use as possible, while still providing much of the flexibility required by a diverse group of enthusiasts. The FPP software is free to download and use, and is provided and supported by a number of volunteers.

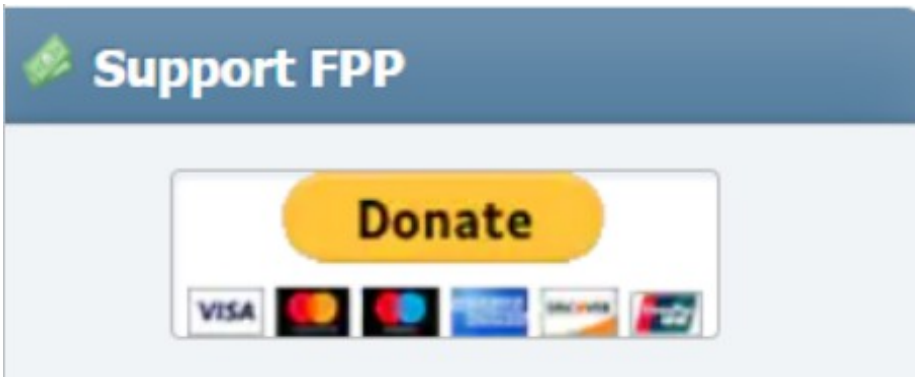
Please refer to the Falcon Christmas website https://FalconChristmas.com for the latest news and discussions. In particular, the FPP forum is a great resource for help.

1.2 Acknowledgments

FPP exists thanks to the work of a large community of volunteers. This manual builds on the Falcon Player Manual written by **Rick Harris (Poporacer)**

FPP's ongoing development is led by a small core team, including **Chris Pinkham (CaptainMurdoch)**, **Daniel Kulp (dkulp)**, **Stuart Ledingham (OnlineDynamic)**, **Daryl Collins (darylc)** , and **Pat Delaney (patdelaney)**.— along with the many contributors and users who report issues, test releases and help one another on the forums. This v10 edition would not have been possible without them.

The developers and authors of this software are volunteers. A very special thanks goes out to our families for supporting us in this hobby. Without their support we could not do this. If you find FPP useful please consider donating to help support future development of FPP. Here is a link: (Donate to FPP)



2 Hardware Needed

The Raspberry Pi and BeagleBone series SBCs have different requirements and setup instructions. Follow the instructions for your specific case. These are the basics to get your device(s) running; depending on your setup you will need additional items after the initial setup to actually run your show (power supply, network cables, wiring, and so on).

2.1 Raspberry Pi

Required items:

- A supported Raspberry Pi.
- A micro-SD memory card: 4 GB minimum, Class 10 or better, A1 recommended. 16 GB or larger is recommended. (Samsung Evo Select and SanDisk Ultra are recommended.)

Note: Some SanDisk High Endurance U3 cards have been reported as incompatible with recent FPP/OS versions.

- A power supply for the Pi:
  - 5 V DC 2.0 A micro-USB for the Pi Zero and Pi 2 series
  - 5 V DC 2.5 A micro-USB for the Pi 3 series
  - 5 V DC 3.0 A USB-C for the Pi 4 series
  - 5 V DC 5.0 A USB-C for the Pi 5 series (additional cooling may also be needed)

Optional items:

- A network cable (if connecting via Ethernet, or to use the Network Configuration process).
- A USB-to-micro-USB cable (for devices that support USB tethering; tethering is usually the easier process).
- A Wi-Fi USB adapter if using a Pi without built-in Wi-Fi (the Edimax Nano is recommended — *not* the V2 version; stick to 2.4 GHz-only adapters, as some 5 GHz cards have compatibility issues).
- A cape/hat, if you are using one.

2.2 BeagleBone series (BB)

It is recommended to use the BeagleBone Black, BeagleBone Green, PocketBeagle or PocketBeagle 2. The BeagleBone Green Wireless cannot be used with capes.

Required items:

- A supported BeagleBone SBC.
- A micro-SD memory card: 4 GB minimum, Class 10 or better; 16 GB or larger recommended. (Samsung Evo Select and SanDisk Ultra are recommended.)
- An appropriate power supply.
- Depending on your controller (typically PocketBeagle-based), a wireless USB adapter or USB Ethernet adapter may be required to connect FPP to your network. (The Edimax Nano is recommended. Some older Wi-Fi adapters that worked with older Linux versions are no longer supported.)

Optional items:

- A network cable (if connecting via Ethernet, or to use the Network Configuration process).
- A USB cable — for the USB-tether install method you need a USB-to-mini-USB cable (micro-USB for a PocketBeagle or BeagleBone

Green). USB tethering is probably the easiest install method for BeagleBone-based devices.
- An Octoscroller-type cape if connecting the BB to P10/P5 panels.
- Another cape, if you are using one.

3 FPP Quick Start Guide

This section gives you the basic configuration to get up and running. It may not be the ultimate configuration you need for your finished show — refer to the rest of this manual, and in particular General Overview and Navigation, for in-depth explanations of each function and setting.

The essential steps are:

1. **Prepare the hardware** – gather a supported board, a suitable micro-SD card and power supply (see Hardware Needed).
2. **Write the FPP image** to the SD card (see Installing the FPP Software). The easiest route is the **Raspberry Pi Imager**, which can download the correct FPP image for you.
3. **Boot and connect** – insert the card, power on, and browse to `http://fpp.local/` (or the device's IP address). The first boot takes a few minutes while the filesystem expands.
4. **Set the network** – configure the network settings so that you can communicate with your devices (see Initial Network Configuration).
5. **Choose the mode** – set **FPP Mode** to **Player** for your main controller, or **Remote** for devices that are going to be controlled from another device. (see The Status Page and MultiSync).
6. **Set the time zone** – on FPP Settings → Localization, so schedules run at the right time.
7. **Configure your outputs** – tell FPP how your lights are connected on *Input/Output Setup* → Channel Outputs.
8. **Add content** – (If needed) upload sequences and media on *Content Setup* → File Manager.
9. **Build and schedule a playlist** – create a playlist on *Content Setup* → Playlists and set it to run on *Content Setup* → Scheduler.

Installing the FPP software, initial network configuration, and updating are covered in detail in the sections below; the rest are covered in their own chapters elsewhere in this manual.

3.1 Installing the FPP Software

Operating-system files are called **images**. To install FPP you need a program for "burning" the image to the micro-SD card, and optionally one for formatting the card first.

> **Note:** You cannot simply copy the files to the card — the image must be written with imaging software.

3.1.1 Required programs

An SD card formatter (optional):

- (https://www.sdcard.org/downloads/) — versions for Windows and Mac.
- (https://gparted.org/) — for Linux systems.

An image-writer program:

- **Raspberry Pi Imager** — can download the FPP image for both Raspberry Pi and BeagleBone hardware directly, as part of the installation process.
- **Balena Etcher** (https://www.balena.io/etcher/) — Windows, Mac and Linux.
- **dotNet Disk Imager** (https://sourceforge.net/projects/dotnetdiskimager/) — a good option for Windows; it can also wipe the SD card, so you won't need a separate formatter.

3.1.2 Getting the FPP software

The software is available at (https://github.com/FalconChristmas/fpp/releases), where you can download the most current **image file** (not the application source code). The image file has `.img.zip` in its name.

Image files start with `FPP` and indicate the version and SBC image. Download the one that matches the SBC you are using:

- **Pi64** — for all Raspberry Pi 3/4/5, Zero 2 W, CM4/CM5 variants
- **Pi** — for all other Raspberry Pi variants.
- **BB64** — for the PocketBeagle 2.
- **BB** — for all other BeagleBone / PocketBeagle variants.

Several releases are listed on GitHub; not all have an image. Scroll down until you find the first version that provides images.

> **Note:** If you are going to use the Raspberry Pi Imager, you do not need to download the image file first.

Depending on your imaging program, you may have to unzip the file before you can use it (uncommon). If your imaging software cannot write directly from a `.zip`, make sure you flash the `.img` file, not the `.zip`; if unsure, unzip first.

The three most popular methods are covered below — Raspberry Pi Imager,dotNet Disk Imager, and Balena Etcher. Use the method for the software you have.

> **Note:** Raspbery Pi Imager can be used for BeableBone images as well.

3.1.3 Formatting the micro-SD card (optional)

Before writing the image you may format the card to remove any existing partitions. This is not usually needed with most imaging programs. Insert the card and do a **Quick Format** using the SD Card Formatter (not the Windows or Mac file manager).

3.1.4 Burning with Raspberry Pi Imager

The Raspberry Pi Imager is convenient because it downloads the chosen image as part of the process — you do not need to download the file separately. (Steps below match Imager 1.8.5; other versions differ slightly.)

1. Open Raspberry Pi Imager. Click **Choose OS** (do not use *Choose Device*).
2. Select **Other specific-purpose OS**, then **FPP OS**.
3. Choose the FPP version to image — the most current is recommended — matching your device (**Pi** or **BBB**).
4. Click **Choose Storage** and select your SD card, then click **Next**.
5. Click **Yes** to accept overwriting the card, and wait for the progress page.

> **Note:** If you are asked that the SD card is “not readable” and whether to format it, **do not** select yes.

When finished, click **Continue** and proceed to Software Installation.

3.1.5 Burning with dotNet Disk Imager

1. Open dotNet Disk Imager (allow it to make changes if prompted).
2. *(Optional wipe)* Select your SD card under **Device** and click **Wipe Device**; confirm and wait for completion.
3. Choose the downloaded image file via the file icon next to the **Image file** box, and select the SD card as the target.
4. Click **Write to Device** (confirm if prompted) and wait for the completion message.

As above, do **not** format the card after writing. Turn the device off and insert the card and proceed to Software Installation.

3.1.6 Burning with Balena Etcher

1. Open Balena Etcher and click **Select Image**; choose the downloaded image file.
2. Make sure the correct SD card is selected as the target.
3. Click **Flash!** (confirm if prompted) and wait for the completion message.

   Note: Some users have resolved Balena Etcher errors by running it as an Administrator, or by unblocking the image file in its properties.

The written image is not in a format Windows or Mac can read, so you may see an error after flashing. **Do not** format the card afterwards. Turn the Pi/BB off, insert the card, and proceed to Software Installation.

3.1.7 Software Installation

FPP is configured from a web interface — **you do not need to connect a monitor to the device.** You access it from a web browser on another computer.

> **Note:** Google Chrome is recommended. Some versions of Internet Explorer / Microsoft Edge have had trouble displaying the interface correctly.

Before you begin, decide how the device will ultimately connect to your network — Wi-Fi, Ethernet, or (in a few cases) both — and make sure the appropriate connection or adapter is fitted first. You will also need to know your home router's IP address (commonly `192.168.0.1` or `192.168.1.1`, among others).

> **Important:** If your home router uses a subnet of `192.168.6.x`, `192.168.7.x` or `192.168.8.x`, FPP will likely have communication problems — these are the default subnets used by Windows, Mac and Linux for USB tethering and by FPP's Wi-Fi tether, and can conflict. Change your home network to a different subnet to avoid problems.

There are three basic ways to install and configure FPP:

- **USB Tethering** — probably the easiest method: connect your computer directly to the device with a USB cable. Only a few devices support USB Tethering: Raspberry Pi Zero W, BeagleBone Black, PocketBeagle, BeagleBone Green, and BeagleBone Green Gateway. (The PocketBeagle 2 does **not** support USB tethering.)
- **Network Connection** — connect the device to your **router** with an Ethernet cable **(not directly to your computer)**. Any Pi or BeagleBone with an Ethernet port or adapter can use this method.

- **Wi-Fi Tethering** — for devices with Wi-Fi tethering capability (on-board or via a supporting adapter), useful when no other method is available.

Warning: If you are using the KulpLights **K4-PB v2.0** or **K40-PB v3.0** (produced in 2022), do **not** use USB tethering — it could destroy the USB circuitry.

3.1.7.1 USB Tethering Installation

Note: If the device needs a network adapter for its final connection, fit it before you start (e.g. a PocketBeagle you will later connect via Wi-Fi). Some capes draw more current than a USB connection can provide — you may need to remove the cape before connecting the USB cable.

Note: Only a few devices support USB Tethering: Raspberry Pi Zero W, BeagleBone Black, PocketBeagle, BeagleBone Green, and BeagleBone Green Gateway. (The PocketBeagle 2 does not support USB tethering.) 1. Make sure the SD card with the correct image is inserted. 2. Fit any network adapters you will need for your final configuration. 3. Connect one end of the USB cable to your computer. (Do **not** also connect a power supply to the device.) 4. On some older BeagleBone Black boards, for the first install you may need to press and hold the **S2** button (near the SD card) and hold it for 5 seconds after connecting. 5. All other devices — plug the USB cable into the device. (On the Pi Zero, use the **USB** port, not the power-only port.) 6. Wait about one minute (a Pi Zero may take slightly longer). 7. Open a browser and go to `192.168.7.2` (Windows) or `192.168.6.2` (Mac/Linux). For a PocketBeagle 2 the tether IP is `192.168.7.2` on all computers. 8. Continue to Initial Configuration.

3.1.7.2 Network Connection Installation

Note: Some capes have RJ45 ports that are **not** Ethernet — they are for DMX or differential receivers and cannot be used for setup. Fit any required network adapter before starting.

1. Insert the SD card with the correct image.
2. Fit any network adapters you will need.
3. Connect the Pi/BB to your router with an Ethernet cable.
4. On some older BeagleBone Black boards, hold **S2** for 5 seconds after connecting power for the first install.
5. All other devices — connect power.
6. Wait about one minute.
7. Browse to `http://fpp/` or `http://fpp.local/`. (If you cannot connect, see the Help and Troubleshooting chapter.)
8. Continue to Initial Configuration.

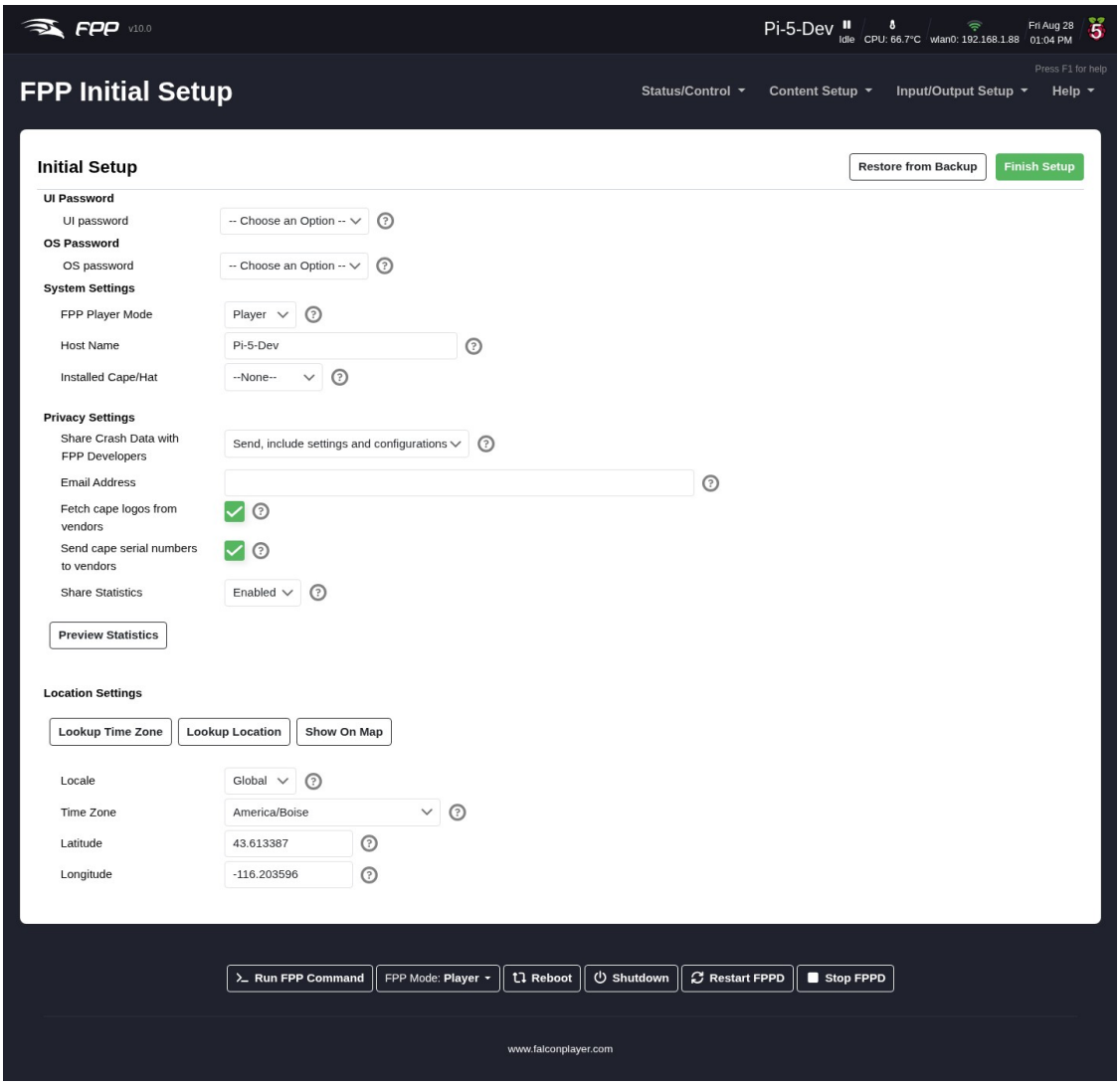
3.1.7.3 Wi-Fi Tethering Installation

Use this if your device supports Wi-Fi tethering, or to make changes when the device cannot otherwise connect to your network (without re-imaging). You need a computer with a wireless connection. > **Note:** Many USB Wi-Fi adapters do **not** support Wi-Fi tethering > (on-board Wi-Fi on Raspberry Pis usually does).

1. Insert the SD card with the correct image.
2. Fit any network adapters you will need.
3. On some older BeagleBone Black boards, hold **S2** for 5 seconds after connecting power for the first install.
4. All other devices — connect power.
5. Wait about one minute.
6. On your computer, connect to the wireless network named **FPP**; the password is **Christmas**.
7. Browse to `192.168.8.1`.
8. Continue to Initial Configuration.

3.1.8 Initial Configuration

Once FPP is installed, the **Initial Setup** page provides a convenient place to configure common or required settings:



The Initial Setup page.

- **UI Password** *(required choice)* — whether to set a web-UI password. This is an advanced setting; the recommended choice is **No Password (default)**. See FPP Settings → UI.
- **OS Password** *(required choice)* — used for SSH and similar access. This is an advanced setting; the recommended choice is **falcon (Default)**. See FPP Settings → System.
- **FPP Player Mode** — set the mode this device will run in. If unsure, leave it at **Player**. See The Status Page → FPP Mode.
- **Host Name** — a meaningful name for this device. See *Network → Host Settings*.
- **Installed Cape/Hat** — if your device has a cape/hat with no EEPROM, define the Virtual EEPROM here. See Pixel Port Licensing.

- **Share Crash Data with FPP Developers** — helps developers find and fix crash bugs. The recommended setting is *Include settings and configurations*. See FPP Settings → Privacy.
- **Email Address** — if you share crash data, providing an email lets the developers follow up if needed.

The Initial Setup page also offers to **restore a previous configuration**, which saves time in setting a replacement or rebuilt device up from scratch. As well as restoring from an FPP backup file, FPP 10 adds a **File Copy Restore** option that brings the configuration across directly. See Backup and Restore.

After completing initial setup, work through the Initial Network Configuration and the rest of this manual to finish setting up your show.

3.2 Initial Network Configuration

After completing Initial Setup, click **Finish Setup** (top-right). Back on the main screen you will see a reboot warning — **do not reboot yet**. Instead open **Status/Control → Network** to configure how the device connects.

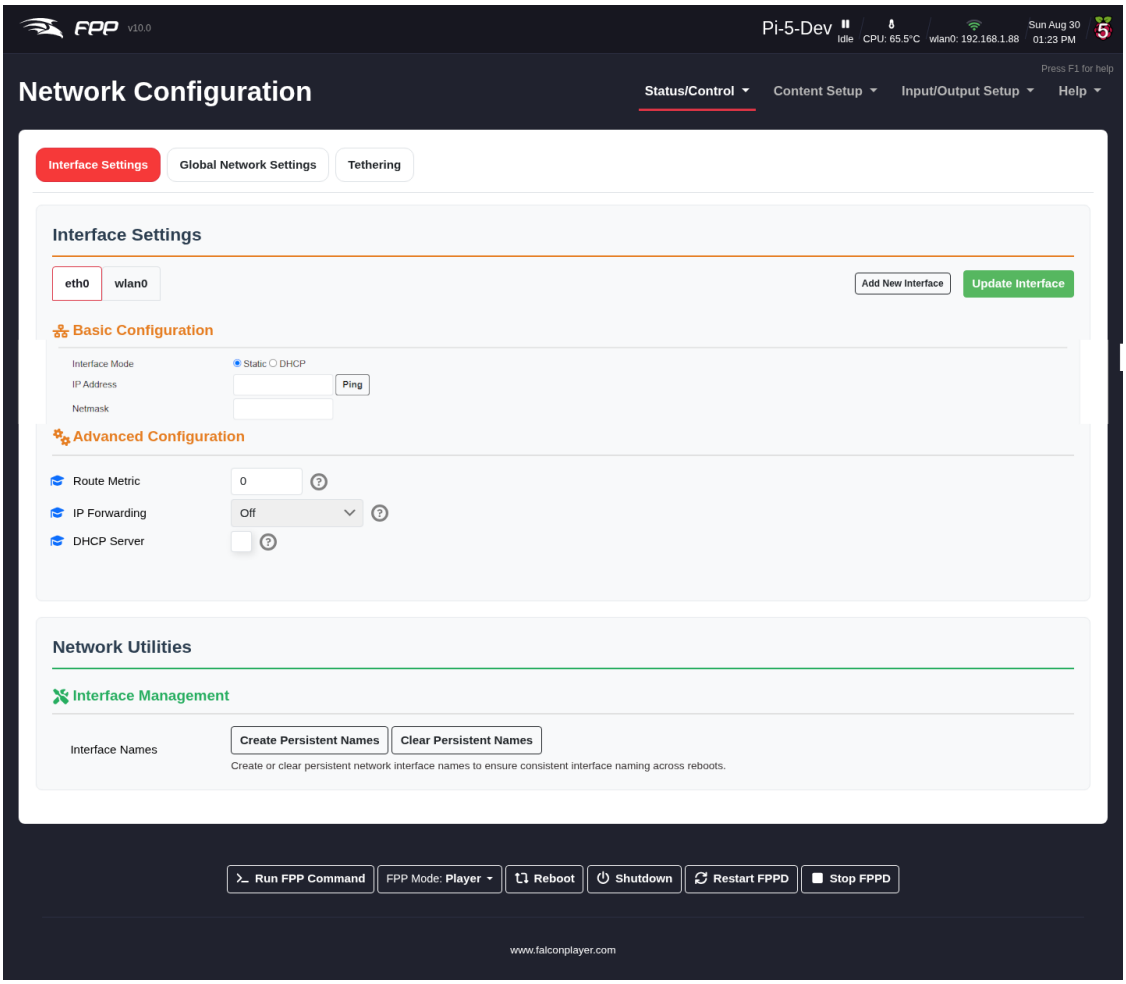
You should decide how you want your network configured before editing these settings (see Networking → Overview if you are unsure). For a temporary arrangement you can use a **wired-on-home-network** configuration so you can update the software and finish configuration before deploying the device in its final location — this is also a good testing configuration. If your device has no Ethernet port but has a Wi-Fi adapter, a **separate show network** may be a better option.

> **Note:** The Network page is covered in full in the Network chapter; this section walks through the first-time setup.

3.2.1 Wi-Fi network settings

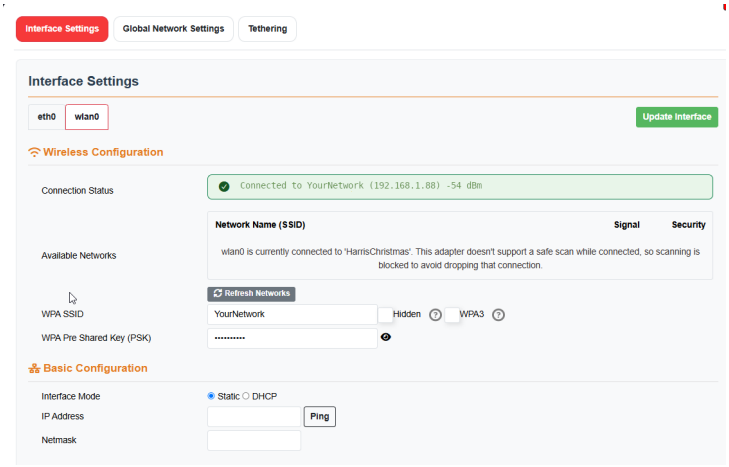
(Skip to Ethernet settings if you do not need Wi-Fi.)

> **Note:** Many USB Wi-Fi adapters do not support 5 GHz; 2.4 GHz networks are recommended for their better range. To configure Wi-Fi with no adapter fitted, see Network → Interface Settings.



The Network Configuration page.

1. Click the **wlan0** interface (the wireless interface).



Wlan0 Settings

2. Enter your **WPA SSID** exactly as configured in your router (including capitalization); you can usually pick it from the available networks.

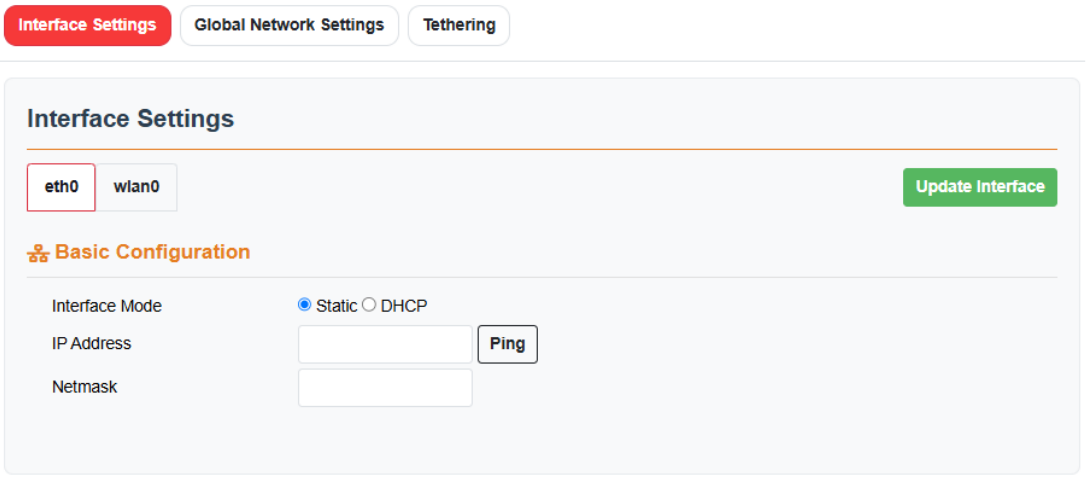
3. Enter the **WPA Pre-Shared Key (PSK)** — your wireless password — exactly as configured (use the show/hide button to check it).
4. Select your **interface mode** (Static or DHCP)

   Note: With DHCP, a correctly configured **host name** and **DNS server** become important.
5. If you are going to use a **Static** IP address, enter an **IP address** unique to this interface.
6. **Netmask** — for most consumer networks this is `255.255.255.0` (match your router).
7. Click on the Green **Update Interface** button.

3.2.2 Ethernet network settings

(Skip to Finishing up if you do not need Ethernet.)

1. Click the **eth0** interface (the wired interface).

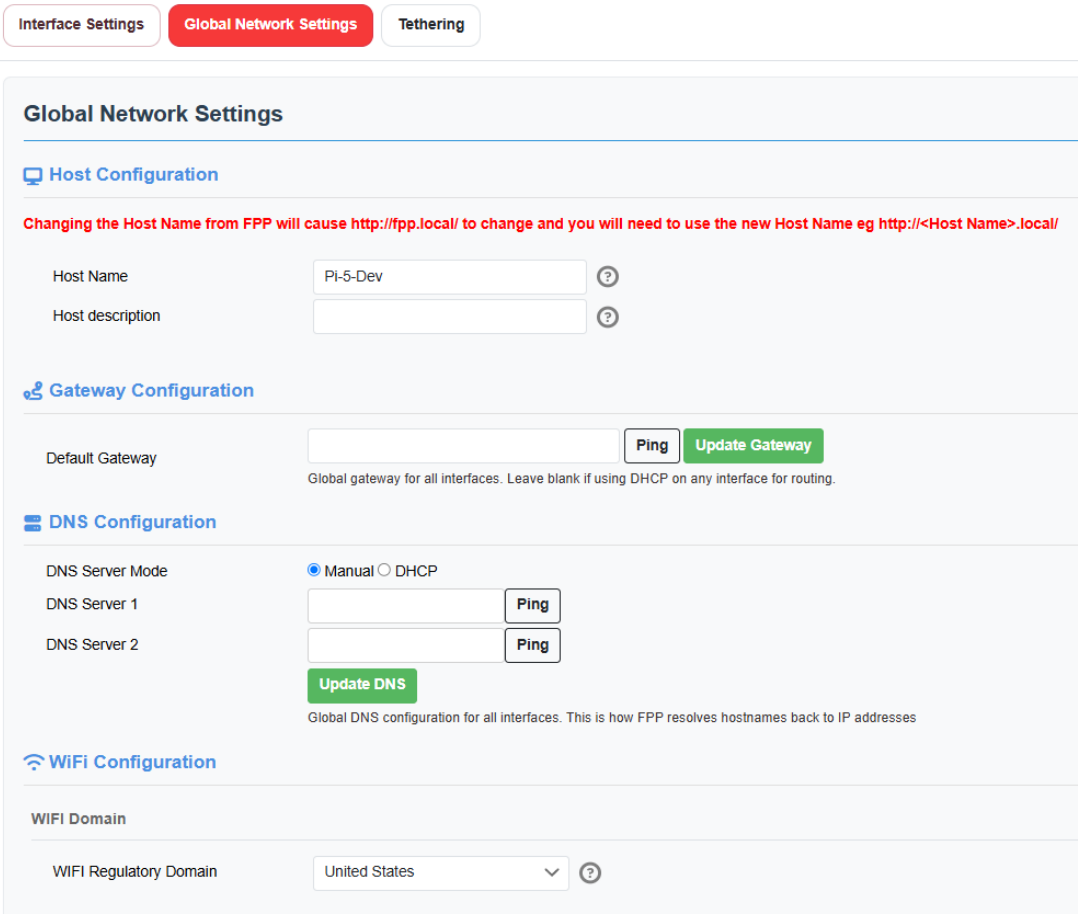


2. Select your **interface mode** (Static or DHCP)

   Note: With DHCP, a correctly configured **host name** and **DNS server** become important.
3. If you are going to use a **Static** IP address, enter an **IP address** unique to this interface.
4. **Netmask** — usually `255.255.255.0` (match your router).
5. Click on the Green **Update Interface** button.

3.2.3 Finishing up

1. Click on the Global Network settings button on the top of the page



2. **Host Name** Change the Host name if you need to.
3. **Default Gateway** — usually the IP address of your home/show router; it is often filled in automatically, but check it. Click on **Update Gateway**.If you are using DHCP, leave it blank.
4. **DNS Server Mode** — if any interface uses a static IP, set this to **Manual** and configure the addresses. Click on the **Update DNS** button
5. Select the **WiFI Regulatory Domain** to the country where you will use the device. Once eth0, wlan0, host and DNS are all configured, double-check them. When correct, click **Reboot** (from the red banner at the top or the button at the bottom).

At this point the device must be connected to your network according to the configuration you set, since you may not be able to reach it otherwise.

3.3 Final Configuration and Updating

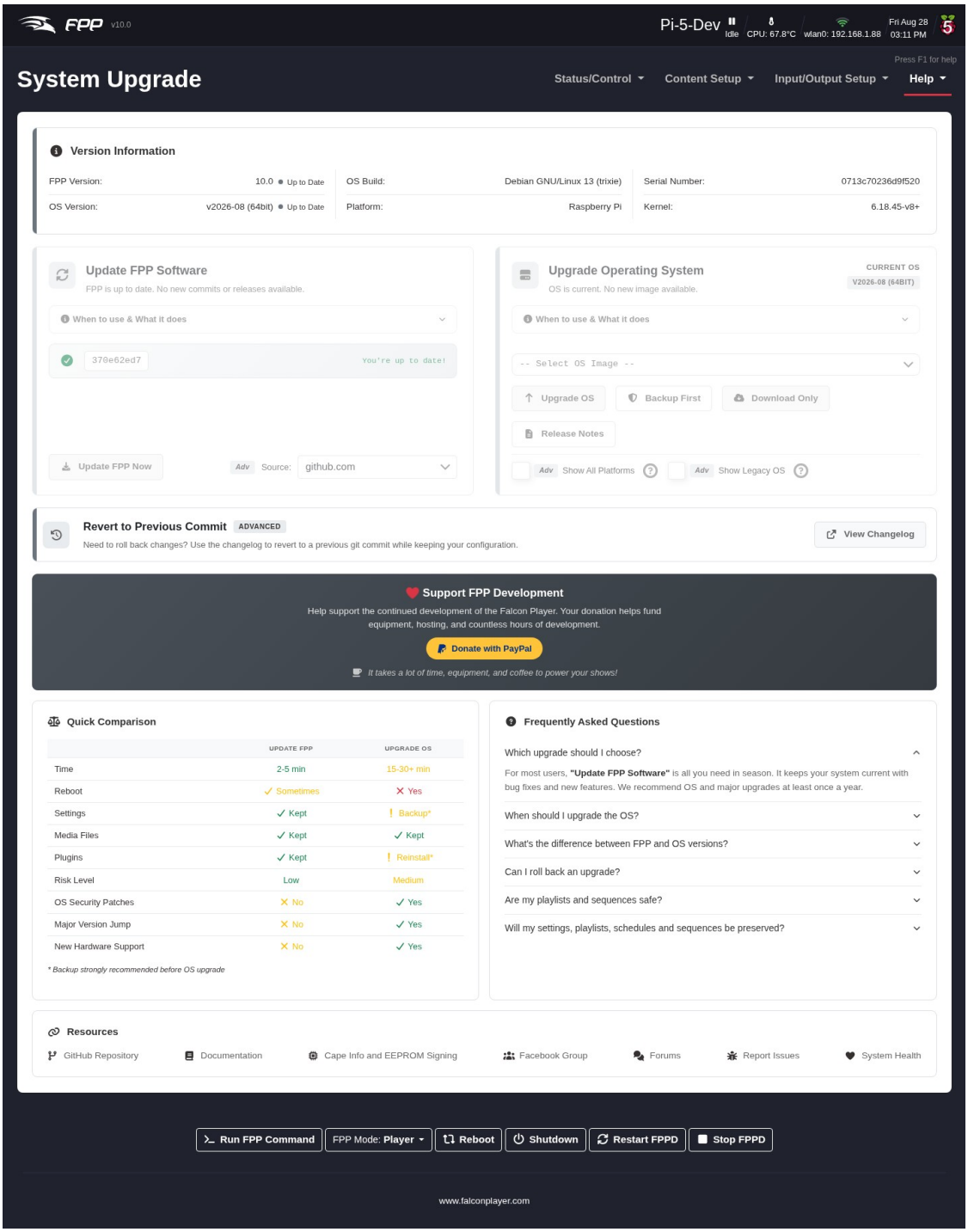
3.3.1 Reconnecting after the network change

Once the device is connected to your network using the settings you entered, open its web page by using either the static IP address that you configured or using the host name you configured. To connect to the device via host name, enter the address in this format: devicehostname.local The page currently on your screen may no longer reach your FPP device. If you cannot reach the FPP page, see the Help and Troubleshooting chapter.

In most configurations FPP has internet access and will keep the correct time automatically. If your device will **not** have internet access and has a Real-Time Clock (RTC) installed, see (FPP Settings → Localization → Time Config).

3.3.2 Updating the software

You should update to the current version of the software. Open **Help → System Upgrade**.



The System Upgrade / About page.

This screen shows the FPP version you are running and, if an update is available, a notice. Click **Upgrade FPP**.

> **Note:** If the **Remote Git Version** shows *Unknown*, FPP usually cannot reach the internet — most often a network/DNS configuration problem. See the Help and Troubleshooting chapter.

You will get a progress screen; the update can take several minutes. When it finishes, click **Close** at the bottom. In FPP 10 the progress pop-up also shows the current stage in its **title**, so you can tell how far along an FPP or FPPOS upgrade is at a glance — useful when the window is in the background.

Sometimes an additional **major** update is offered — if so, click **Upgrade**. You will usually see a **Release Notes** page; some updates need a matching Operating System (OS) update to gain full functionality, and that will be noted there.

> **Note:** If the release notes indicate an **FPPOS** upgrade may be required, clicking **Upgrade** will **not** upgrade the OS — that is a separate step. See *Help → System Upgrade* for details.

Confirm any prompts and let the update run, then click **Close**. When complete, the screen returns to the About page and you can verify the version. When fully up to date, the **Local Git Version** matches the **Remote Git Version**.

Your FPP software is now installed and up to date. There are many ways to use FPP, and the settings needed to run your show vary with your particular setup — refer to the appropriate chapters that follow for details.

> **Note:** After an **FPPOS** upgrade, FPP 10 prompts you to reinstall your plugins — an OS upgrade replaces the underlying system, so plugins need to be put back. Accept the prompt, or reinstall them yourself from the *Plugins* page (the **Reinstall All** button does the lot).
>
> **Tip:** Always take an **FPP Backup** (see the Backup and Restore chapter) before a major upgrade, so you can roll back if needed.

4 General Overview and Navigation

4.1 General Overview

The developers have built in several aids to explain the functionality of different settings. Most settings load based on the device FPP is running on, so you only see what is relevant and things are less confusing. FPP also has different **UI Levels** so that more advanced settings and functions do not clutter the screens (see FPP Settings → UI).

The four levels are **Basic**, **Advanced**, **Experimental** and **Developer**. Basic shows the settings most users need; Advanced exposes most of FPP's features; Experimental is for experts only and can expose settings that hurt performance or cause problems if misused; Developer is for developers and

beta testers. In FPP 10 the level also governs which **FPP Commands** and which menu entries you see, not just which settings.

> **Tip:** If you only need to configure one setting that requires an Advanced level setting, you do not have to change your Level permanently. On FPP Settings → UI the **Change to Advanced UI for 15 Minutes** button temporarily raises the level. While it is active, an **unlock** icon appears in the header showing roughly how long is left; click it (or **Exit Advanced Mode** on the UI settings page) to drop back immediately.

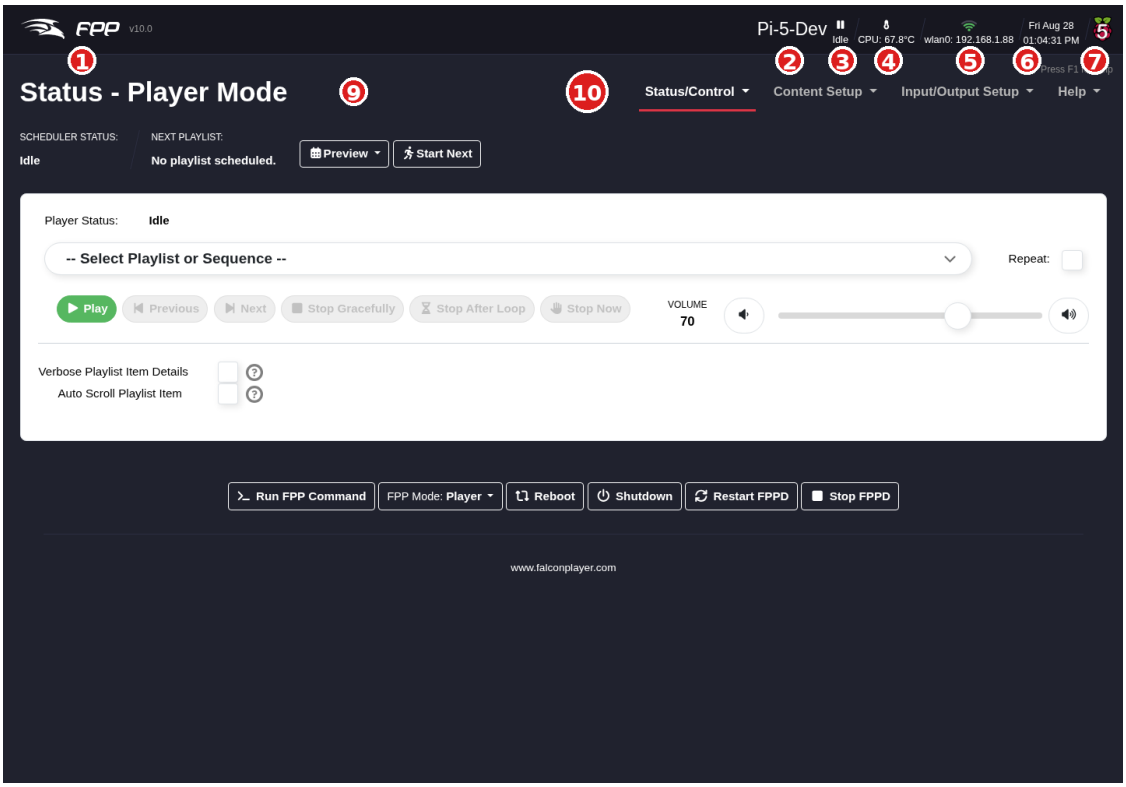
Many settings have a **Help popup**, identified by a blue question-mark icon. Hovering over the question mark brings up additional information specific to that item. Individual **FPP Command arguments** carry these tooltips too.

4.2 Program Settings and Navigation

The following chapters explain each program area and its settings, and how they work. Depending on your setup or FPP version, the screens may look slightly different.

The main page is reached from a web browser by entering the IP address or host name of the device you set up — for example `192.168.1.101` or `http://YardPi.local/` (yours will differ).

At the top of every page is a status/navigation section:



The header and navigation section (Status page shown).

1. **FPP Logo and version** – the current version is shown, and links to the *System Upgrade* (About) page to check for updates.
2. **Host name** – this device's host name; it can be used to reach the device (e.g. `http://Pi-5-Dev.local/`) and links to the *Host & DNS Settings* page. (Not all networks support host-name access.)
3. **Player status** – the current state. *Idle* when nothing is playing; *Playing* (hover to see the current sequence); or a *Stopped* icon when FPP is stopped.
4. **Sensors** – if the device has sensors, their readings appear here. Change the units under FPP Settings → Localization. With more than one sensor, click to toggle between them.
5. **Network interfaces** – a graphical status of each interface; hover for details. For example: -a. A live Ethernet connection with an IP address (this does not by itself guarantee the connection is valid). -b. A DHCP Ethernet connection that received **no** address and gave itself a link-local address (e.g. `169.254.x.x`) — usually because it is connected to a controller or directly to a computer. -c. A wlan0 connection with an IP address and signal strength (e.g. −57 dBm), also shown by the number of green bars. -d. The device operating as a wireless access point.
6. **Date and time** – the current system date and time.

7. **Platform graphic** – the hardware platform and variant (Raspberry or Beagle).
8. **Vendor logo** – may show your cape vendor's logo.
9. **Section indicator** – which area of FPP you are in.
10. **Main navigation toolbar** – on every page; clicking a heading reveals the options for that category (each is covered in its own chapter). The headings are *Status/Control*, *Content Setup*, *Input/Output Setup* and *Help*.

Some menu entries only appear in certain circumstances:

- **Variables** and **Recurring Tasks** (under *Content Setup*) require the **Advanced** UI Level or higher — see Variables and Recurring Tasks.
- **Port Status** (under *Status/Control*) only appears on devices whose cape reports current monitoring. It was called *Current Monitor* before FPP 10.
- **Cape Info** (under *Help*) only appears when a cape or hat is fitted.
- Plugins add their own entries at the bottom of whichever menu their author chose.

In the upper-right corner, **Press F1 for help** (or the F1 key) opens help specific to the current page.

4.3 Bottom control bar

At the bottom of every page are shortcuts to commonly used device functions:

- **Run FPP Command** – opens a popup to run one of the pre-programmed FPP Commands (see *Commands, Effects and Testing*).
- **FPP Mode** – change the FPP mode (see below).
- **Reboot / Shutdown** – restart or power down the device.
- **Restart FPPD** – stop and restart the FPP daemon, reloading many configuration changes without a full reboot.
- **Stop FPPD** – stop the FPP daemon.

4.4 FPP Mode

FPP runs in one of two primary modes:

1. **Player** – used when this device runs the show itself and/or sends show data to other controllers. A player can communicate with other devices in two ways:
    - **a. Output E1.31/DDP/ArtNet** to other controllers or FPP devices — sending the full pixel data frame by frame. Common

on a wired network driving one or more controllers. This requires the appropriate **Channel Outputs** to be configured.

   - **b. Send MultiSync packets** — used when you have more than one FPP device and want them synchronized via small sync packets. Each **remote** needs only a copy of the sequence (`.fseq`) files; the **player** holds the playlists and schedules.

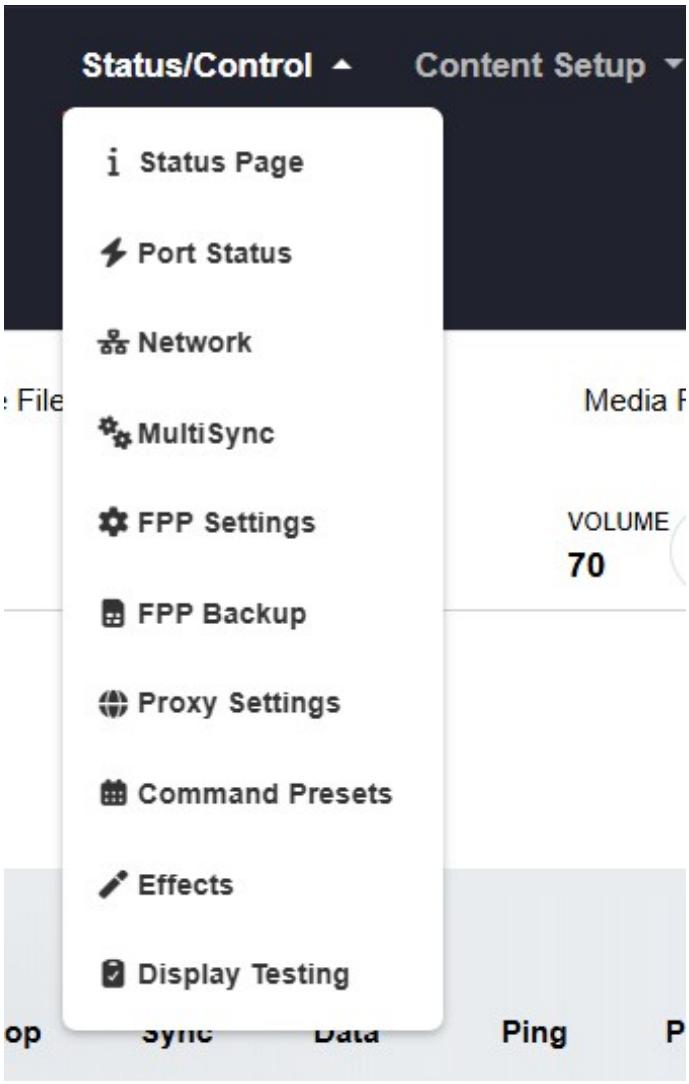
2. **Remote** – used to synchronize this device (and its attached controller or cape) to a player, in one of two ways:
   - **a. MultiSync** — the remote listens for MultiSync packets from a player and syncs its output to match. The remote needs a copy of every sequence (`.fseq`) that will be played (see the MultiSync chapter).
   - **b. E1.31/DDP/ArtNet** — the remote receives streamed data instead; this requires the input to be configured (see Channel Inputs and Separate Network).

4.5 Update indicators

- A **minor** upgrade (same development branch) shows an orange icon in the header; click it to go to *System Upgrade* and update.
- A **major** upgrade (new branch) shows an orange bar offering the upgrade; clicking the button takes you to *System Upgrade*.

5 Status/Control Section

The **Status/Control** menu holds the day-to-day settings, status reports and playback control of your FPP device. Its first entry — and the default page that loads when you log into FPP — is the **Status Page**. Depending on which mode FPP is running in, **Player** or **Remote**, the screen looks a little different.

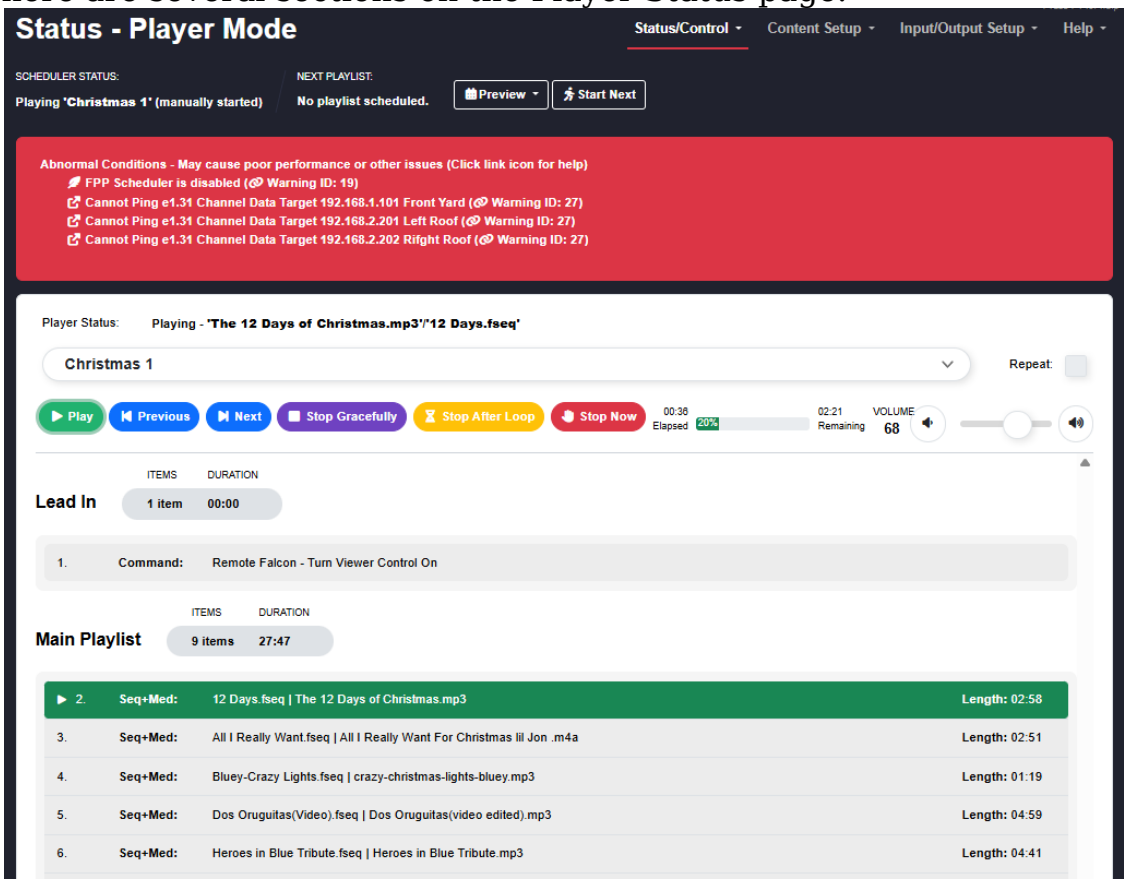


The Status page in Player mode.

6 Status Page

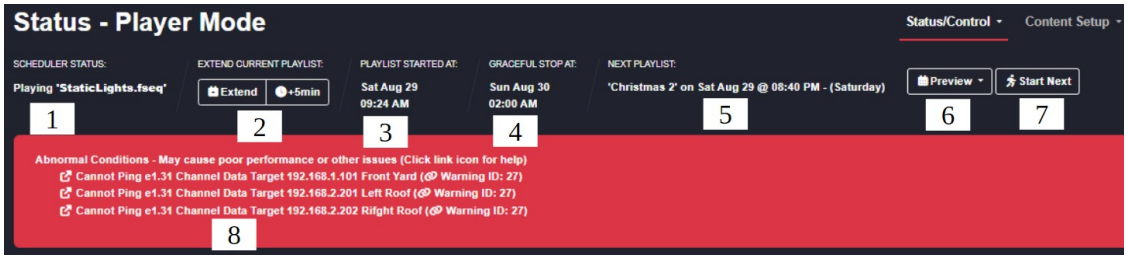
6.1 Player Status Page

There are several sections on the Player Status page.



6.1.1 Scheduler Status

This section shows the status of your Scheduler and options to control a playlist that is playing. **Note: This screenshot is showing Abnormal Conditions and this is not a normal status. You should remedy any abnormal conditions.**

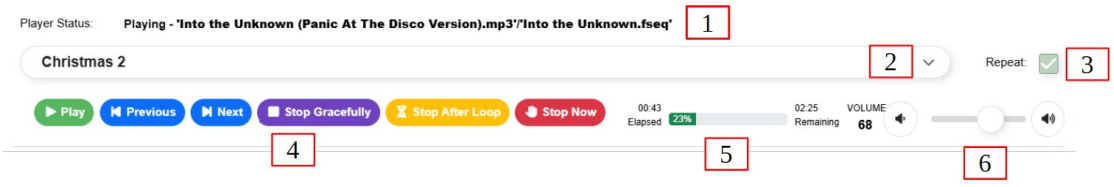


Scheduler Status in Player mode.

1. **Current Playlist** – shows the currently playing playlist. If nothing is playing it shows **Idle**. If the playlist was started by hand it shows *(Manually Started)* after the name.
2. **Playlist time extension** – you can manually extend (or reduce) a *scheduled* playlist that is running. Click **Extend** to change the scheduled end time in minutes (use a negative number to shorten it); a second button extends in 5-minute increments. You can extend the end time by at most **720 minutes (12 hours)** or reduce it by **360 minutes (3 hours)**. Once a playlist has reached its scheduled end time this option is no longer available, even if a song is still finishing (a graceful shutdown).
3. **Playlist Started at** – the time the scheduled show started.
4. **Stop Type** – the stop strategy for the scheduled playlist that is currently playing, and the time it is scheduled to stop.
5. **Next Playlist** – the next scheduled playlist, with the start time and day it will begin.
6. **Preview** – shows a graphical representation of your schedule for the next 4 weeks (extendable via an advanced setting in FPP Settings). This is a drop-down with a **Table View** or **Calendar View** option. (see the Scheduler chapter).
7. **Start Next** – ends the current playlist immediately and starts the next scheduled playlist. That playlist still ends at its normally scheduled time.
8. **Abnormal Conditions** – if FPP detects conditions that can affect performance, the messages are listed here. These almost always need to be remedied for your show to run properly; see the Help and Troubleshooting chapter for common messages and fixes.

Note: The status shown on this page is **pushed** to your browser over a WebSocket as it changes, rather than the browser polling for it every second. The page reacts faster and puts less load on the device; if the connection drops it reconnects on its own.

6.1.2 Player Status



Player Status in Player mode.

1. **Player Status** – lists the sequence/song currently playing. It also shows a “breadcrumb” when an inserted playlist is playing — for example when using Remote Falcon, or an FPP Command such as a push-button that inserts a playlist into your normal playlist.

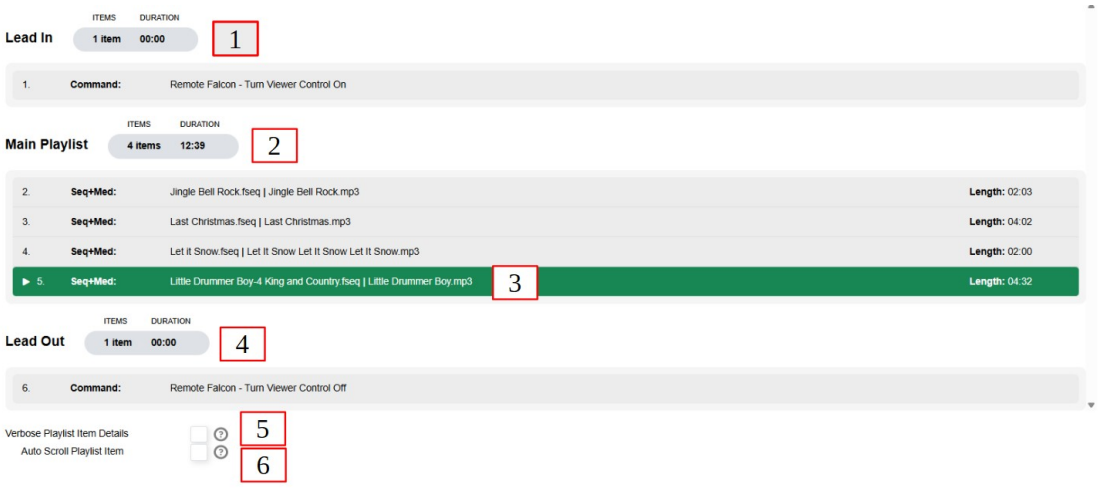
2. **Playlist / Sequence Selector** – shows the playlist that is playing; otherwise use it to select a playlist, an individual sequence, or an individual media file to play manually.
3. **Repeat** – if ticked when you manually start a playlist or sequence, it keeps playing until stopped manually.
4. **Player controls** – control the currently queued playlist:
    a. **Play** – play the queued playlist from the selected element. If **Repeat** is ticked it keeps playing until stopped manually or a scheduled playlist starts.
    b. **Previous** – step to the previous playlist item.
    c. **Next** – step to the next playlist item.
    d. **Stop Gracefully** – finish the current song, then stop.
    e. **Stop After Loop** – stop when the end of the current playlist loop is reached.
    f. **Stop Now** – stop immediately.
5. **Song Status** – shows how long the current song has been playing and how much time remains; it also indicates if the playlist is set to random order.

6. **Volume** – controls the output volume for the currently playing sequence. Useful for setting the level fed to an FM transmitter or external speakers.

If nothing is playing the status is **Idle**. It also indicates when a playlist is shutting down gracefully (finishing the song, then stopping).

6.1.3 Playlist Details

This section shows the details of the currently selected/playing playlist (see the Playlists chapter for more).

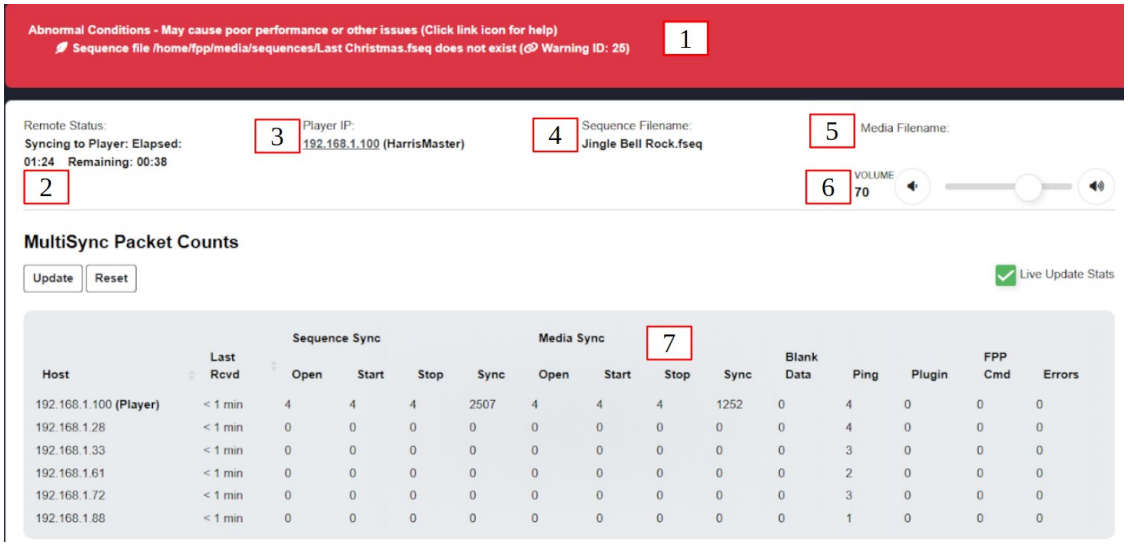


Player Status in Player mode.

1. **Lead In** – any Lead In items, with total items and per-item durations. If there are no Lead In items, this section is not shown.
2. **Main Playlist** – an overview of the Main playlist: number of items and total duration.
3. **Playlist Details** – every item in the Main playlist, showing sequence name and associated audio file; the currently playing item is highlighted.
4. **Lead Out** – any Lead Out items, with total items and per-item durations. If there are no Lead out items, this section is not shown.
5. **Verbose Playlist Item Details** – shows much more information for each playlist item (helpful for seeing all the arguments of scripts or FPP Commands).
6. **Auto Scroll Playlist Item** – If your playlist has several items, this will keep the currently playing item visible in the window.

6.2 Remote Mode Status Page

When **FPP Mode** is set to **Remote**, the Status page instead reflects synchronisation with a player:



Player Status in Remote mode.

1. **Abnormal Conditions** – as above, any conditions that need remedying.
2. **Remote Status** – the Remote-mode sync status: whether it is actively syncing to a player, elapsed time for the current sequence, and time remaining.
3. **Player IP** – which device is sending the sync packets, with its IP address (a hyperlink to that device) and host name.
4. **Sequence Filename** – the currently playing sequence.
5. **Media Filename** – the media (audio/video) file being played, if any.

6. **Volume** – controls the volume of media played on this remote.
7. **MultiSync Packet Counts** – all of the sync messages received from other devices.
   - **Live Update Stats** refreshes every second.
   - **Update** refreshes once.
   - **Reset** clears the history. Columns include:
   -
      a. **Host** – IP addresses of devices that have communicated with this one.
   -
      b. **Last Received** – the last day/time communication was received.
   -
      c. **Sequence Sync** / **Media Sync** – stats for sequence and media sync messages.
   -
      d. **Blank Data** – stats for blanking data received.
   -
      e. **Ping** – devices that have pinged this remote.
   -
      f. **Plugin** / **FPP Cmd** – stats for plugin and FPP-command messages.
   -
      g. **Errors** – any errors encountered.

6.3 Channel Inputs Status Section

If you have **Channel Inputs** enabled (see the Channel Inputs chapter) and FPP has received input data, an additional panel appears in the lower part of the Status page

Player Status Channel Inputs.

It shows the configured universes / DDP data and packet statistics per row: **Universe**, **Start Address**, **Packets**, **Bytes** and **Errors**.

Live Update Stats refreshes every second.

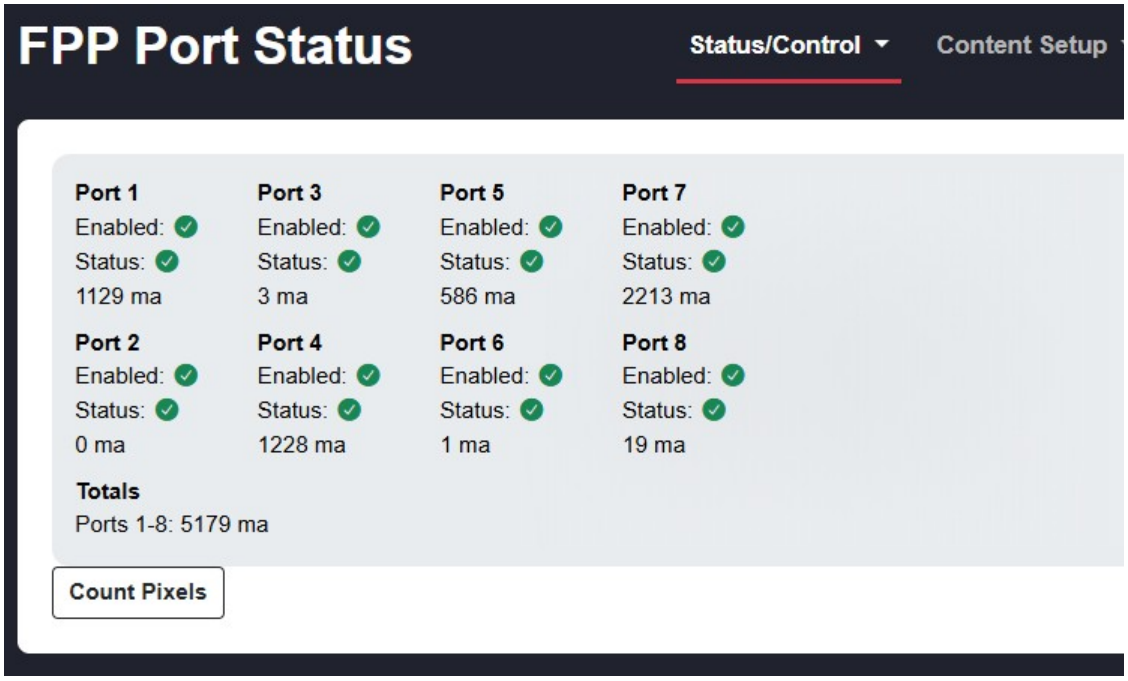
Update refreshes once.

Reset clears the counters.

> **Note:** When a change requires it, a banner appears prompting you to **Restart FPPD** or **Reboot**. Output does not resume until the requested action is done.
>
> **Note:** A playlist you started **manually** (that is, one the scheduler did not start) is resumed after FPPD restarts, rather than leaving the device idle. Scheduled playlists continue to be governed by the schedule.

7 Port Status

If your FPP device has eFuses (electronic fuses) and supports current-monitoring, a **Port Status** entry is available under *Status/Control*.

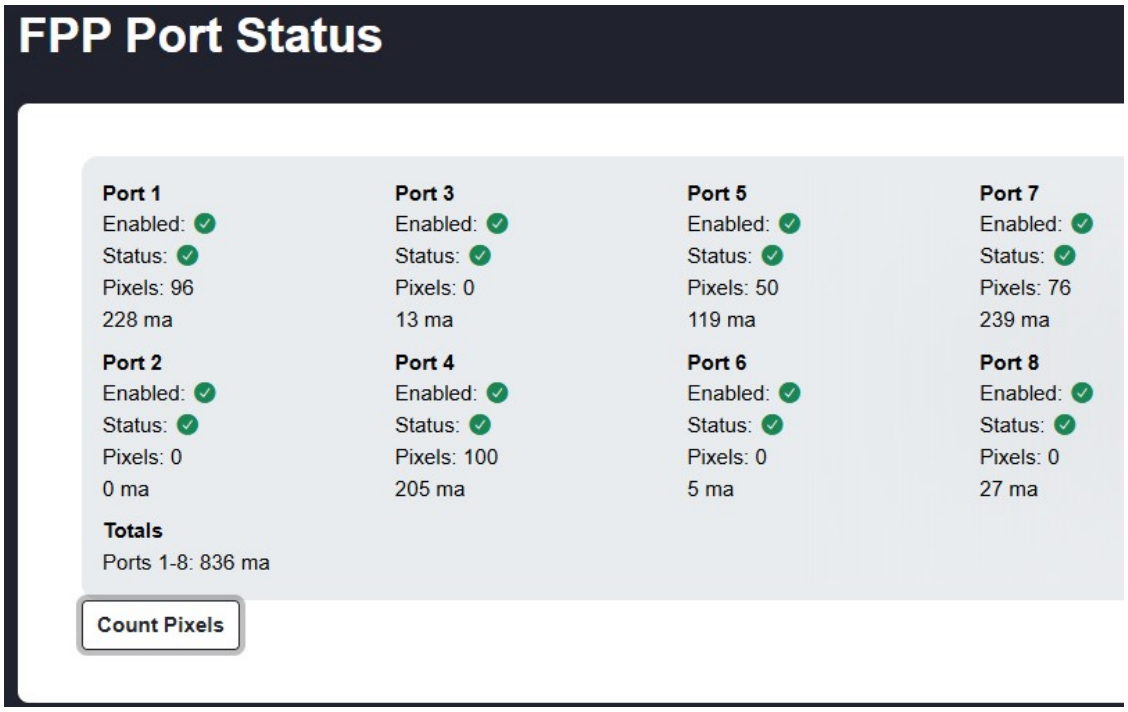


Port Status

This will list all of the ports along with the status and current being passed through the fuses. - Enabled- The icon next to this will indicate the state of that port. - Green circle with check mark- Port is enabled and outputting voltage - Blue circle with x- The port is disabled and not outputting voltage - Red circle with x- this indicates that the port is disabled due to an eFuse overload. - Status- The icon next to this will indicate the state of the eFuse on that port. - Green circle with check mark- eFuse is normal - Red circle

with x- this indicates that the eFuse is in a tripped state. - ma- This will indicate the number of milliamps the fuse is monitoring.

Count Pixels- Clicking this button will send a test pattern to the pixel string on each port and count how many pixels are connected to that port and then leave the last identified pixel lit white. If you have a bad pixel in the string or using power injection, the pixel count could be incorrect. With some of the newer low current draw pixels, the counting might be a little bit off.

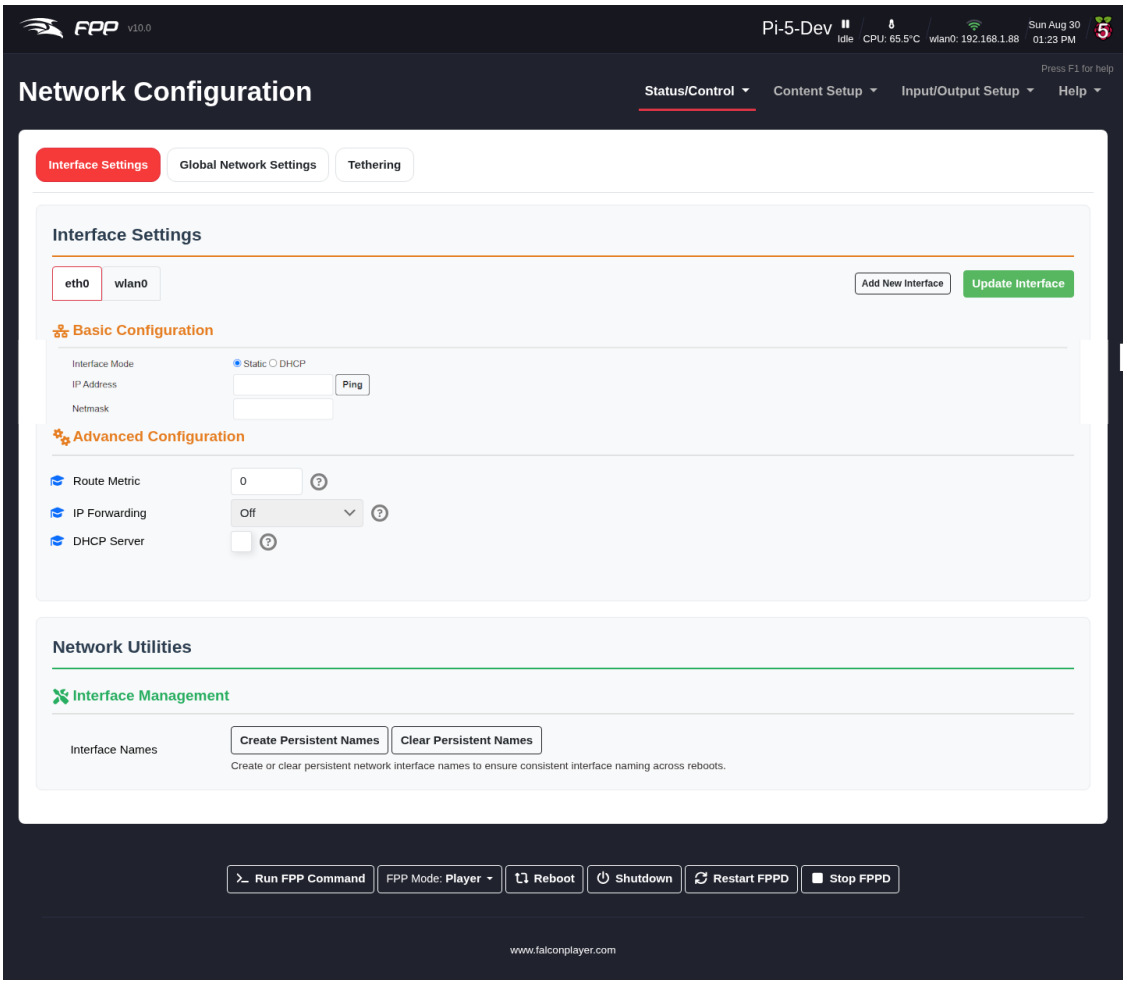


Pixel Counts

> **Note:** This page was called **Current Monitor** in earlier releases; it was renamed to **Port Status** in FPP 10. The entry only appears if the fitted cape advertises current monitoring, so on many devices it is not in the menu at all.

8 Network

The **Network** page (**Status/Control → Network**) is where you set up networking so your FPP devices and controllers can all communicate as needed. Networking works similarly whether wired or wireless, and the two work together. The page has three tabs: **Interface Settings**, **Global Network Settings** (host and DNS), and **Tethering**.



Network — Interface Settings.

> **Note:** This page shows some Advanced Level settings that you probably won't use. see more information below.

Because there are so many ways to configure a network, these settings cause many people difficulty. The basic setup earlier in this manual will get you running, though it may not be the best long-term arrangement; the sections below should give you a better understanding for your situation. For deeper background on IP addressing, see Networking → Overview.

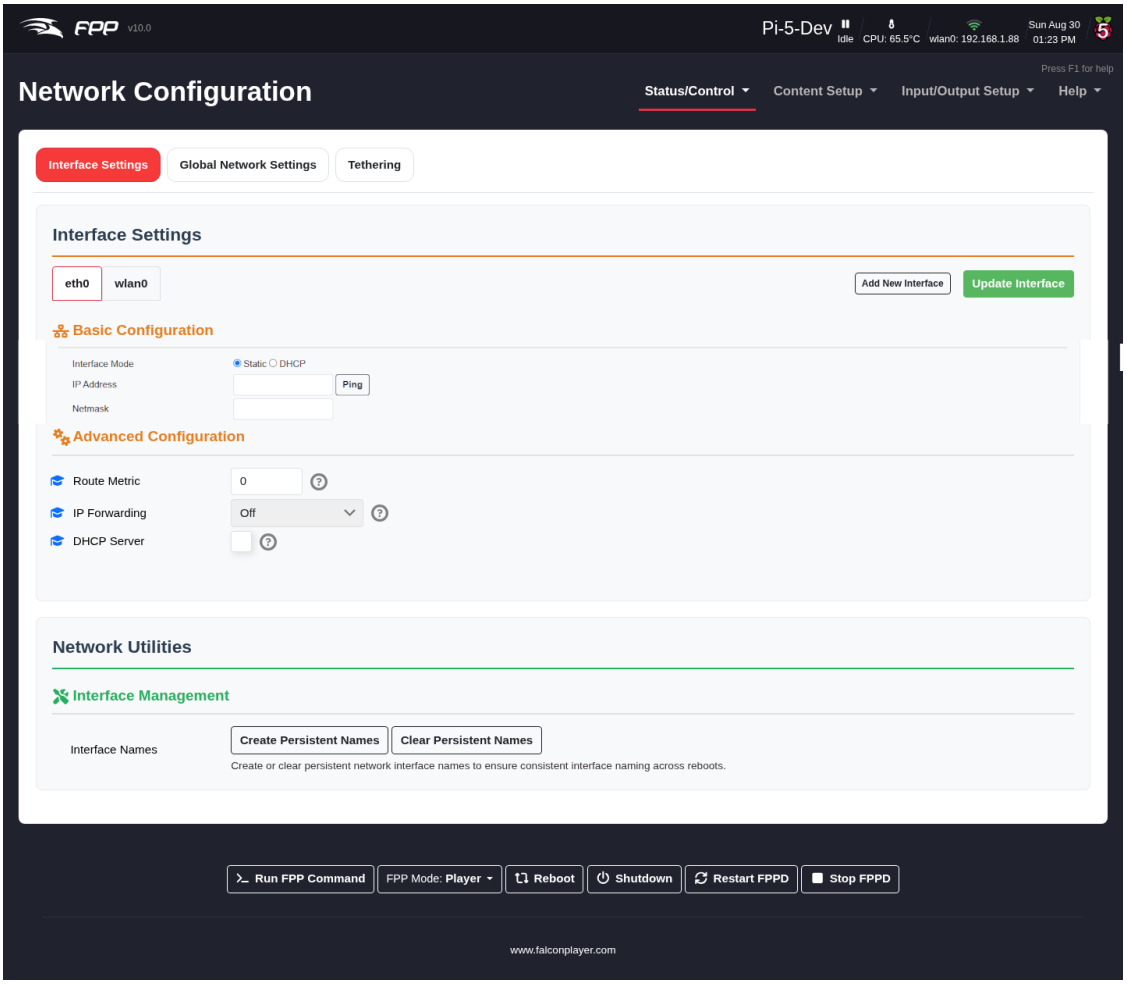
If you want a temporary network configuration you can use the Wired on Home Network configuration so that you can update the software and make final configurations before using FPP in your final network configuration. This would be a good configuration for testing purposes as well. But if your FPP device doesn't have an Ethernet port but has a Wi-Fi adapter then Separate Show Network might be a better option.

There is a good video here and Keith Westley, one of the xLights developers, has a good video as well here

8.1 Interface Settings

Depending on the device, FPP may have up to two built-in network interfaces (more if you add adapters, though that is uncommon): **eth0** for wired Ethernet and **wlan0** for wireless. If you have both, configure each separately.

eth0 Interface



- **Interface Mode:**
  - **Static** – you assign the IP address. You must ensure each address is unique and does not clash with one your router has already handed out via DHCP. You also need to make sure that your router does not assign this IP address to another device down the road. Many routers assign DHCP addresses from the low end of the range (not always), and some let you limit the DHCP range to avoid conflicts.
  - **DHCP** – your router assigns and manages the IP address and gateway. This is the easiest method, but your router may not retain the IP address if the device is disconnected for a long

time (you can usually still reach FPP by host name), and the interface must be on a network with a DHCP server.

- **IP address** – unique to this device/interface, in the same subnet as the network it talks to (usually the first three number groups). The **Ping** button can be used to check whether an address is already in use.
- **Netmask** – defines the network size; most consumer networks use `255.255.255.0`.

Note: If you use both interfaces they should be on **different subnets**, and only **one** interface should have a gateway — normally the one connected to your home network.

Create Persistent Name-If you are using more than one Ethernet interface (common for users with a Color-Light board) and you need the Ethernet adapter to keep the configuration order, then you can create a Persistent Name. The best practice would be:

- Power down the FPP device.
- Make sure that only the primary Ethernet interface is installed.
- Power up the FPP device.
- Plug in the USB Ethernet adapter.
- Configure the eth0 and eth1 devices
- Click on Update Interface
- Click on Create Persistent Name

This will save your eth0 and eth1 configurations so that they will load up in the correct order.

- **Route Metric** – leave at default for most setups. If more than one interface has a gateway (unusual), give your primary interface a **lower** number. *(This is only visible with an Advanced or higher level setting in the UI tab.)*

- **IP Forwarding** – This is typacally used when FPP connects to Wi-Fi and also feeds a controller/switch over Ethernet and you are not using a Proxy Host (not needed with a cape/hat or in standalone mode):

  - **Off** – no forwarding.
  - **Forwarding** – forwarding within the local network only; forwarded devices may not have internet access.
  - **Masquerading/NAT** – forwarding with NAT, giving forwarded devices internet access without complex static routing. Configure this on the interface connected to your home network. *(This is only visible with an Advanced or higher level setting in the UI tab.)*

Note: This replaces the old “Enable Routing between network interfaces” option on the former Interface Routing tab and is typically used on the wlan0 interface **NOT the eth0**

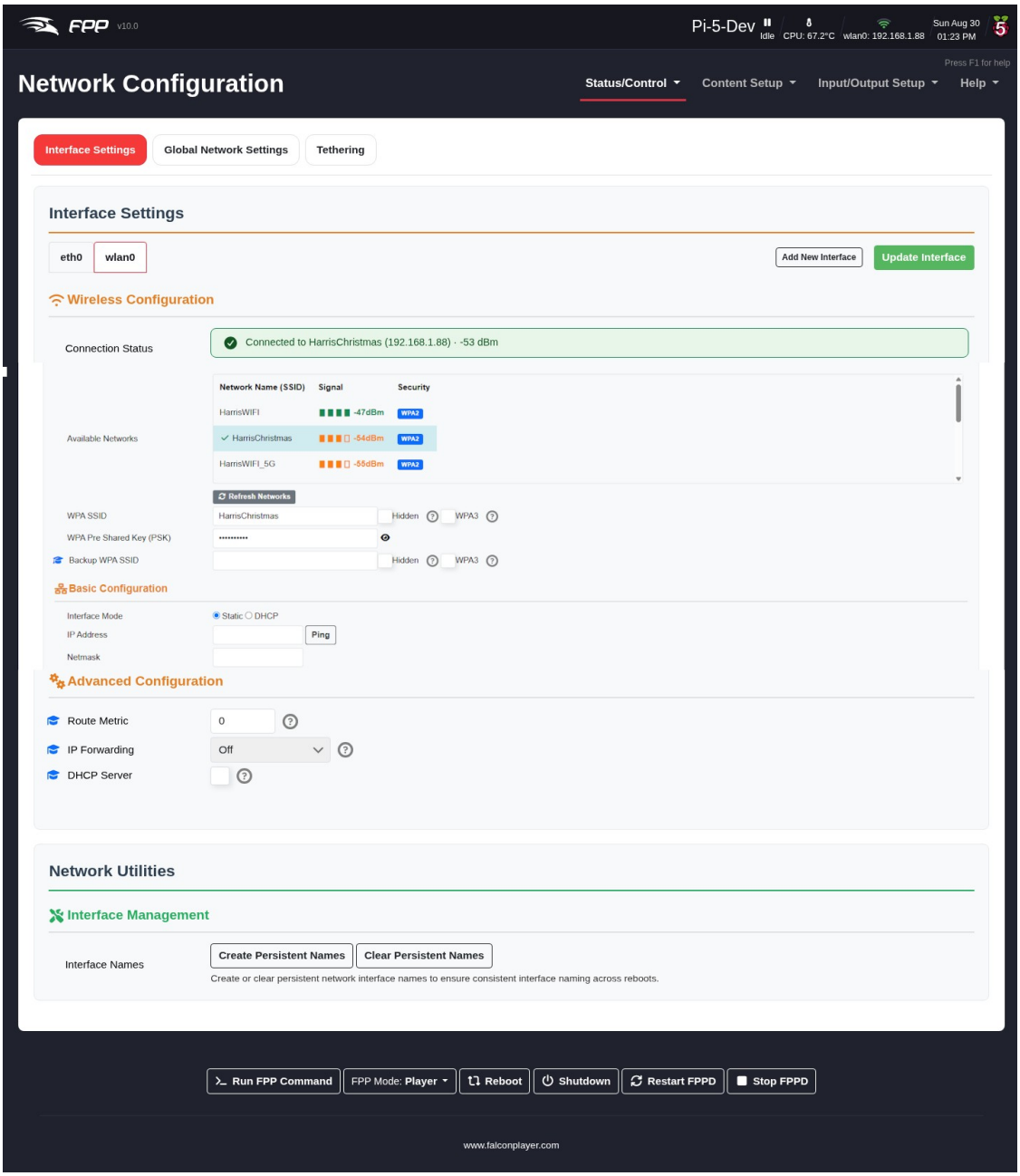
- **DHCP Server** – let FPP hand out IP addresses to connected devices — useful on a *Separate Show Network* with no router. Only **one** device on a network should issue DHCP, and it must have a static IP. *(This is only visible with an Advanced or higher level setting in the UI tab.)*

  Caution: If you connect this interface to your home network with the DHCP server enabled, devices may get incorrect addresses. Assigned devices appear under **Static Leases**, where you can create a reservation. *(This is only visible with an Advanced or higher level setting in the UI tab.)*

- **DHCP Pool Offset / Size** – the starting address and number of addresses the DHCP server may assign. *(*This is only visible with an Advanced or higher level setting in the UI tab and with DHCP enabled.)*

  Note: In FPP 10 — a change that comes from the move to a newer Debian base — **DHCP leases survive a reboot**. Restarting FPP no longer clears the leases it has handed out, so a device that had an address keeps it. Lease times are set to sensible values by default. To start over with a clean pool, run **FPP Settings → System → Reset FPP Config** with **Network Config Files** ticked; the saved leases are then cleared on the **next reboot** rather than immediately, which avoids disturbing addresses on interfaces that are still in use.

wlan0 Interface



Network — Interface Settings, wlan0.

This page will have information about your current Wi-Fi connection such as **Connection Status** It will also list Available Networks if your Wi-Fi adapter can scan the networks.

- **WPA SSID** – your wireless network name (tick **Hidden** if you are connecting to a hidden SSID).
- **WPA Pre-Shared Key** – the password; use the eye icon to reveal it and check it is correct.

- **Backup WPA SSID / PSK** – an alternate network that FPP tries if it cannot reach the primary. *(This is only visible with an Advanced or higher level setting in the UI tab.)*
- **Route Metric** – leave at default for most setups. If more than one interface has a gateway (unusual), give your primary interface a **lower** number. *(This is only visible with an Advanced or higher level setting in the UI tab.)*
- **IP Forwarding** – This is typacally used when FPP connects to Wi-Fi and also feeds a controller/switch over Ethernet and you are not using a Proxy Host (not needed with a cape/hat or in standalone mode):
  - **Off** – no forwarding.
  - **Forwarding** – forwarding within the local network only; forwarded devices may not have internet access.
  - **Masquerading/NAT** – forwarding with NAT, giving forwarded devices internet access without complex static routing. Configure this on the interface connected to your home network. *(This is only visible with an Advanced or higher level setting in the UI tab.)*

  Note: This replaces the old "Enable Routing between network interfaces" option on the former Interface Routing tab and is typically used on the wlan0 interface **NOT the eth0**
- **DHCP Server** – let FPP hand out addresses to connected devices — useful on a *Separate Show Network* with no router. Only **one** device on a network should issue DHCP, and it must have a static IP.

  Caution: If you connect this interface to your home network with the DHCP server enabled, devices may get incorrect addresses. Assigned devices appear under **Static Leases**, where you can create a reservation. *(This is only visible with an Advanced or higher level setting in the UI tab.)*
- **DHCP Pool Offset / Size** – the starting address and number of addresses the DHCP server may assign. *(Advanced, with DHCP enabled.)*

  Note: In FPP 10 — a change that comes from the move to a newer Debian base — **DHCP leases survive a reboot**. Restarting FPP no longer clears the leases it has handed out, so a device that had an address keeps it. Lease times are set to sensible values by default. To start over with a clean pool, run **FPP Settings → System → Reset FPP Config** with **Network Config Files** ticked; the saved leases are then cleared on the **next reboot** rather than immediately, which avoids disturbing addresses on interfaces that are still in use.

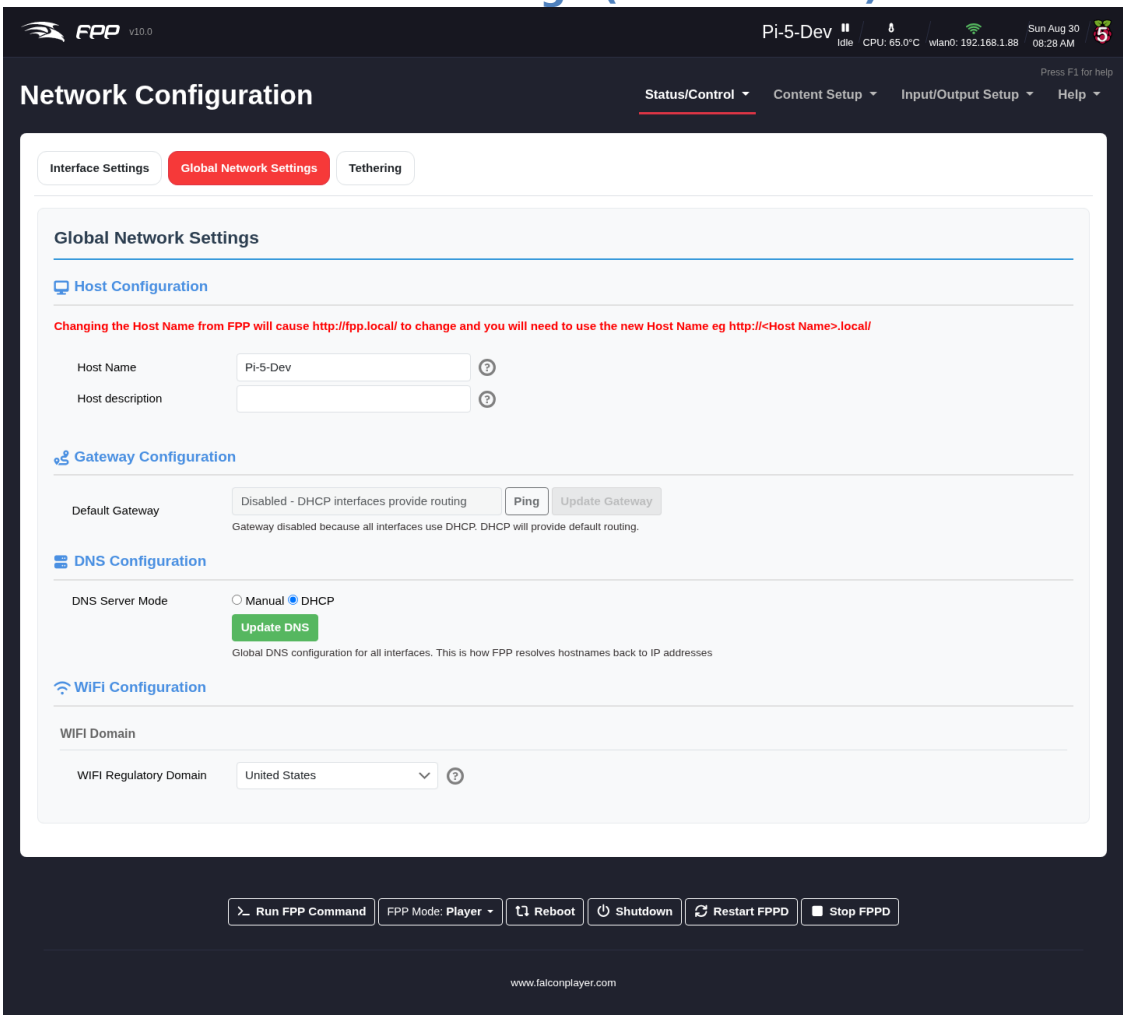
- **Update Interface** – saves the settings for the current interface. Click it before moving to another interface, and again when finished; then reboot.
- **Add New Interface** – configure an `eth0` or `wlan0` interface even when the physical hardware is not yet present (e.g. configuring wlan0 for a BeagleBone before its Wi-Fi adapter/cape is attached). A small dialog asks for the interface name — enter it exactly as the system will name it, such as `wlan0` or `eth1` — and the new interface then appears as its own tab to configure. *(This is only visible with an Advanced or higher level setting in the UI tab.)*
- **Create Persistent Name** – when using more than one Wi-fi interface (Not common) and you need the adapters to keep their order, create persistent names.

  Best practice:

  1. Power down the device.
  2. Ensure only the primary Wi-fi interface is installed.
  3. Power up the device.
  4. Plug in the USB Wi-fi adapter.
  5. Configure `wlan0` and `wlan1`.
  6. Click **Update Interface**, then **Create Persistent Name**.

  This saves the configurations so they load in the correct order.

8.2 Global Network Settings (Host & DNS)



Network — Global Network Settings.

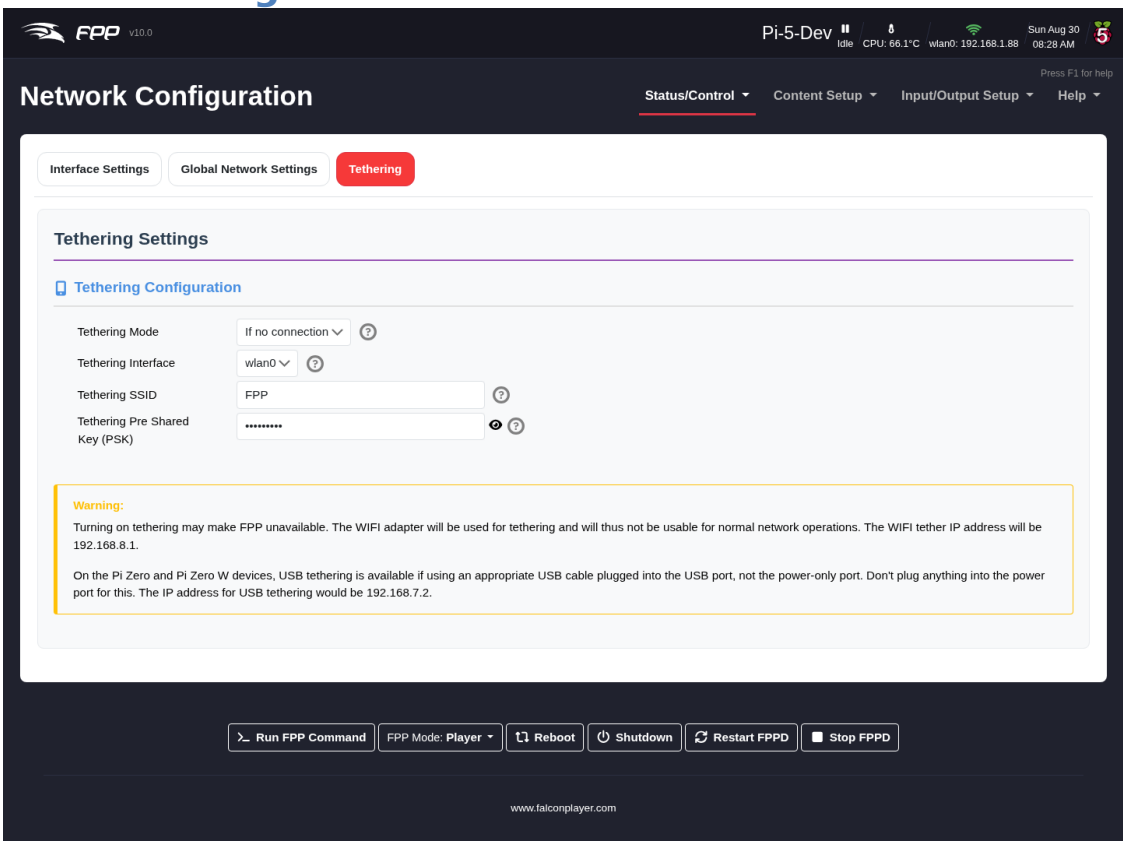
This tab assigns the device's **host name** and DNS settings.

- **Host Name** – the "human" name used to reach the device (like typing a domain instead of an IP). If DNS or another part of your network is misconfigured the host name may not resolve, but you can still reach FPP by IP address. Choose something meaningful and **unique** among your devices — e.g. `FPPMaster`, `FrontLawn`, `HouseOutline`. Names may contain only letters, numbers and hyphens (`-`), may not begin or end with a hyphen, and cannot contain spaces. After changing it you can no longer use `http://fpp.local/`; use the new name (e.g. `http://YardProps.local/`) or the IP address. Save after entering it.

  Keeping the default `FPP` is possible if you will never add another instance, but renaming is strongly recommended — shows tend to grow, and duplicate names cause confusion.

- **Host Description** – additional, free-form text (no host-name restrictions) shown on the MultiSync page and the xLights FPP Connect screen.
- **Default Gateway** – You will typically set this to your router's IP address.
- **DNS servers** – with any static interface, set DNS to **Manual** and enter servers; typically your router's IP for one and an internet server such as `8.8.8.8` for the other.
- **Wi-Fi Regulatory Domain** – enter your country. Some jurisdictions have regulations, and Wi-Fi will not work correctly unless this is set correctly.

8.3 Tethering



Network — Tethering.

FPP supports two kinds of tethering: **Wi-Fi Tethering**, where FPP acts as its own access point, and **USB Tethering**, where FPP connects directly to a computer by USB cable.

8.3.1 Wi-Fi Tethering

> **Note:** Due to the number of various USB Wi-Fi adapters, the Wi-Fi Tethering might not work using some USB Wi-Fi adapters. Thay

have to support AP mode. (this does not apply to the **internal** Wi-Fi adapters on the Raspberry Pis or BeagleBones).

Wi-Fi Tethering lets you reach FPP when nothing is connected to the Ethernet or Wi-Fi interfaces — especially useful on a Raspberry Pi whose on-board Wi-Fi supports AP mode (not all adapters do). Bring your computer near the device, connect to the **FPP** wireless network (password **Christmas**), and browse to `192.168.8.1`.

Note: If the device has an OLED screen it will display a QR code you can scan to reach it.

There are three Wi-Fi tethering modes:

- **If no connection** *(default)* – FPP starts the **FPP** access point at boot only if it detects no network on any interface. (A device connected to an Ethernet port usually counts as a connection, so AP mode will not start.)
- **Enabled** – FPP always starts the access point at boot.
- **Off** – the access point is never started.
- **Note:** It is not recommended to turn off the Wi-Fi tethering. This is a useful setting that can often time save you from reformatting your SD card if you mis-configure your network settings

Warning: If you set the Tethering to Enabled, the **only** way you will be able to connect to the Wi-Fi interface of your FPP would be through the FPP AP it will NOT conect to the configured wlan0 interface.

USB Tethering

Note: Only a few devices support USB Tethering: Raspberry Pi Zero W, BeagleBone Black, PocketBeagle, BeagleBone Green, and BeagleBone Green Gateway. (The PocketBeagle 2 does not support USB tethering.)

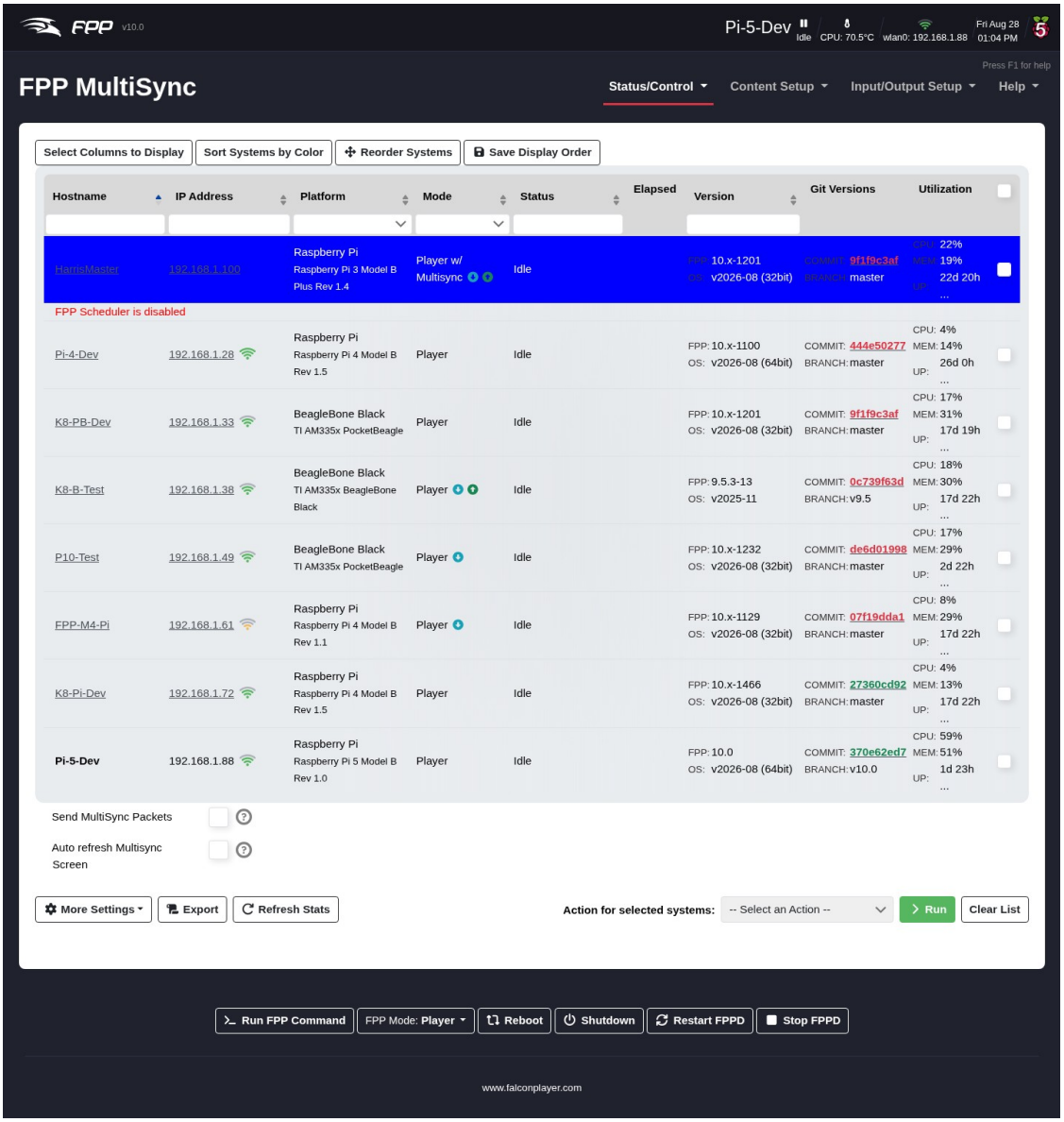
USB Tethering connects FPP directly to your computer with a USB cable, as described in Installing the FPP Software → USB Tethering Installation. It is often the easiest way to reach a device for setup, on the hardware that supports it.

9 MultiSync

The **MultiSync** page (**Status/Control → MultiSync**) is where you set up a MultiSync/Remote layout, but it has evolved into a much more useful interface for all of your FPP devices and most controllers. A MultiSync/Remote layout can eliminate the Ethernet cabling between your FPPs/controllers and allows widespread model placement; one FPP **player** keeps any number of FPP **remotes** playing in perfect time by broadcasting

small timing packets. You can also view status and system information for every device, and upgrade them all from one interface.

You can use MultiSync on a wired setup too, to save network traffic, though the benefit there is usually small. For deeper background on layout and function, see Networking → Overview.



The MultiSync page listing discovered FPP systems.

> **Note:** The MultiSync page appears differently depending on your UI Level and the device's mode. It also shows the UI header colours for easy identification. (In the example image, several errors are shown deliberately to illustrate potential problems — this is not normal.)

9.1 The systems table

FPP automatically discovers other FPP instances on the network. Use **Select Columns to Display** to choose columns, **Sort Systems by Color** to group by header colour, and the header fields/arrows to filter and sort. Columns include:

- **Hostname** – every discovered device (the current device is in **bold**). If two devices share a host name the display may be inaccurate.
- **IP Address** – all configured addresses for each device, as hyperlinks to that device. If a remote has a Wi-Fi connection, a checkbox appears next to the Wi-Fi icon to configure it for **Unicast Sync**. *Multicast* sync is recommended when your network supports IP Multicast and IGMP Snooping; if it does not, or you get dropped/unreliable connections, set your remotes to **Unicast** and disable "Send MultiSync to all remotes via Multicast".
- **Platform** – the SBC model and other information.
- **Mode** – the FPP mode each device is running.
- **Status** – the current status; if playing, the sequence name.
- **Elapsed** – elapsed time of the playing sequence on player and remotes, or an indication that the device is bridging.
- **Version** – the FPP branch, version and OS installed.
- **Git Version** – software status: **red** = an upgrade is available, **green** = up to date, **black** = the device could not report (usually a network/DNS configuration error).
- Utilization- This will show the utilization of various system resources like CPU,
- Memory and uptime.

9.2 Display and sending options

- **Send MultiSync Packets** *(Player mode)* – send MultiSync packets to devices configured as remotes.
- **Auto refresh Multisync Screen** – refresh the systems table automatically (every 2 seconds) so status, elapsed time and version stay current without reloading the page.
- **More Settings** – additional display and sending options:
  - **Send MultiSync to all Remotes via Multicast** *(Player, with Send enabled)* – send via IP Multicast.
  - **Send MultiSync to all Remotes via Broadcast** *(Player)* – send via network broadcast; helpful on networks without IP Multicast / IGMP Snooping.
  - **Send MultiSync to ALL KNOWN remotes via Unicast** This will send the MultySync packets via unicast.

  - **MultiSync Unicast Discovery IPs (CSV list)** – for instances/controllers not found by normal discovery. *(Advanced.)*
  - **HTTP Discovery IPs & subnets** – add devices on a different subnet so they appear on the MultiSync page. *(Advanced.)*
  - **Hide 10.x.x.x/8**, **Hide 172.16.x.x/12** and **Hide 192.168.x.x/16 Subnets** – three separate toggles that hide those private ranges from the systems list, useful when a device sits on both a show network and your home network and you only want to see one of them.
  - **External IP Address For MultiSync Packets** – the address other hosts see for this FPP instance. Leave blank unless NAT or a router means this device is reached on a different address from the one it knows about.
- **Export** – download a spreadsheet of all connected devices and their stats at the time of export.

Important: FPP 10 defaults new installs to Unicast, not Multicast. Multicast needs both IP Multicast and IGMP Snooping to be working across your network, which many home switches and access points do not handle well, so unicast is the more reliable starting point. If you *upgraded* an existing device — whether by an FPP upgrade or an FPPOS upgrade — your previous multicast setting is preserved, so nothing changes underneath a working show.

Tip: In FPP 10 a change to the sync destinations is applied without restarting FPPD, so you can switch between unicast and multicast and see the effect immediately.

Note: At the time of writing, Falcon controllers in remote mode do not support Multicast (expected in a future controller update), and not all home networking equipment supports Multicast.

9.3 Actions on selected systems

Tick one or more devices in the right-hand column, choose an **Action**, and click **Run** to apply it to all of them at once:

- **Upgrade FPP** – upgrade software on any or all devices simultaneously (red = out of date, green = up to date, black = could not determine, usually a DNS problem). A progress screen is shown; afterwards you can Reboot or Restart FPPD.
- **Restart FPPD** – restart the FPP software without a full OS reboot.
- **Reboot / Shutdown** – full OS reboot or shutdown of the selected devices.
- **Copy Show Files** – copy show files from this device to others. You cannot pick individual files; all files of the selected **type** are copied.

Tick the types to include: **Copy Sequences**, **Copy Music**, **Copy Videos**, **Copy Effects**, **Copy Scripts** and **Copy Events**.

- **Compress files during copy to Remotes** – compress files before sending, mainly useful for older V1 FSEQ files. Newer xLights versions already write a compressed FSEQ format, so for those this option can actually *slow the transfer down* as FPP tries to recompress already-compressed data.

Warning: Take care when copying **sequences**. Newer xLights versions create host-specific `.fseq` files containing only the channels each individual FPP instance needs — so the `.fseq` sitting on your player may not hold all the channel data a remote requires to play correctly. Where you use host-specific files, upload to each device from xLights instead of copying them between devices here.

Note: `rsync` must be enabled on any remote you send files to.

- **Copy OS Upgrade Files** – copy `.fppos` OS-upgrade files to other devices; only files appropriate to each platform are copied (Pi files won't go to BB systems, and vice versa). `rsync` must be enabled on the target.
- **Set to Player** / **Set to Remote** – change the mode of the selected devices.
- **Add as Proxy** – configure this device to act as a Proxy Host for the selected device (see Proxy Settings).

Note: FPP Connect in xLights is a more robust way to upload files to your FPP devices — it can upload in a sparse format that greatly reduces file size, and can also configure inputs, outputs and other settings.

10 FPP Settings

The **FPP Settings** page (**Status/Control → FPP Settings**) is where you set up administrative functions and settings. The settings are organized into a row of tabs across the top of the page: **Playback**, **Audio/Video**, **Localization**, **UI**, **Email**, **MQTT**, **Privacy**, **Input/Output**, **Logging**, **Services**, **Storage**, **System** and **Developer**.

Note: The FPP Settings page displays differently depending on your UI Level, hardware and mode. Most settings save immediately; some prompt you to **Restart FPPD** or **Reboot** before they take effect.

10.1 UI Levels

FPP has several **UI Levels** that show more or fewer settings so that advanced options do not clutter the screen (set on the **UI** tab — see below).

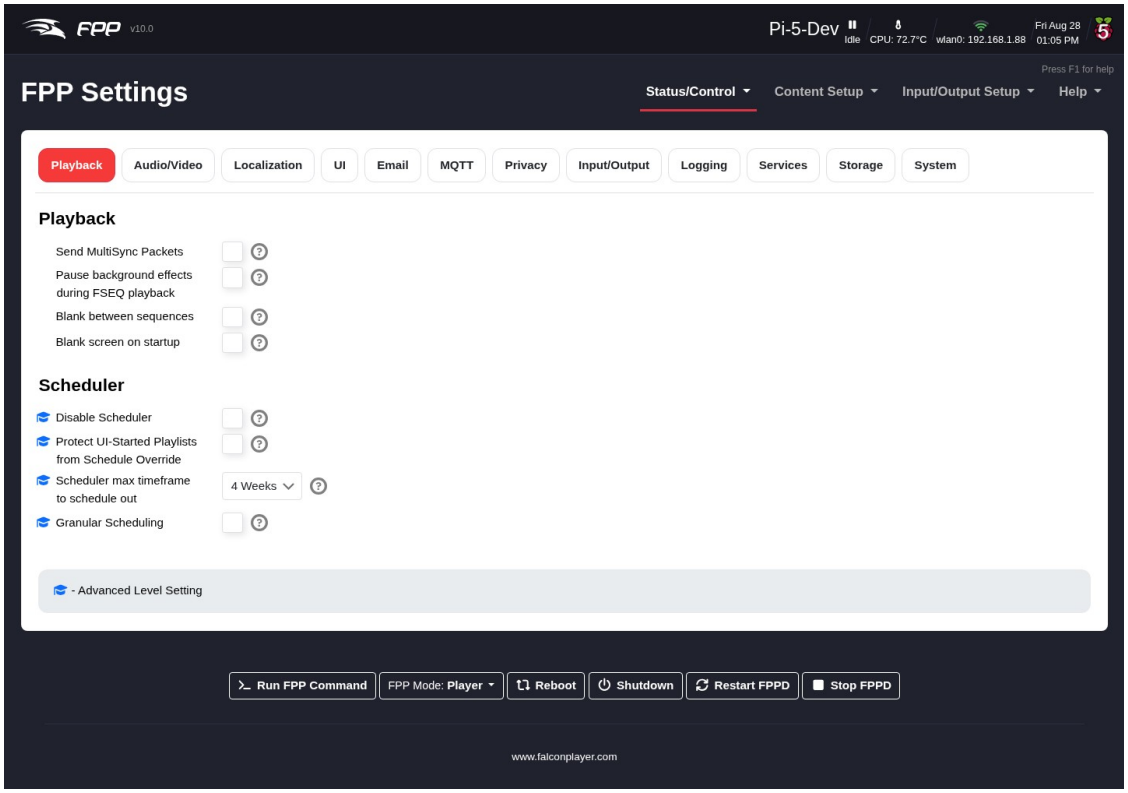
In addition, individual settings are tagged with an icon indicating the minimum level at which they appear:

- 🎓 **Advanced Level Setting**
- 🧪 **Experimental Level Setting**
- </> **Developer Level Setting**

Settings with no icon appear at all levels. Throughout this chapter, items marked *(Advanced)*, *(Experimental)* or *(Developer)* are only visible at that UI Level or higher. If a setting described here is not visible, raise your UI Level.

10.2 Playback

Configures general playback behaviour.



Settings — Playback tab.

- **Send MultiSync Packets** *(Player mode only)* – send MultiSync packets to remote devices (see MultiSync).
- **Pause Background Effect Sequence during FSEQ playback** – effect sequences normally take priority over FSEQ files; select this if you want the FSEQ file to take priority over a background effect sequence.

- **Blank between sequences** – send blanking data to turn the pixels off between items.
- **Blank screen on startup** – By default, FPP will display system data to the HDMI port when it boots up and will remain until other data is sent out to the HDMI port. The screen blanking will turn the text console off after one minute so that it does not remain on the monitor. If you are using this FPP to play video through the HDMI port, then you probably want to enable this setting. (This option is only available on Pi based FPP devices as the HDMI port is disabled on BeagleBone based systems)
- **Inactivity timeout for screen blanking** – when *Blank screen on startup* is enabled, blank the screen after this many minutes with no activity. *(Developer.)*
- **Open/Start Delay** – a delay (ms) before playback begins.
- **Local Media/Sequence Offset** – trims the synchronization between media and sequences on *this* device, in milliseconds. A positive value moves the media ahead, a negative value moves it back. Requires an FPPD restart. *(Advanced.)*
- **Remote Media/Sequence Offset** – the same trim applied to an FPP **remote**, in milliseconds. Requires an FPPD restart on the remote. *(Advanced.)*

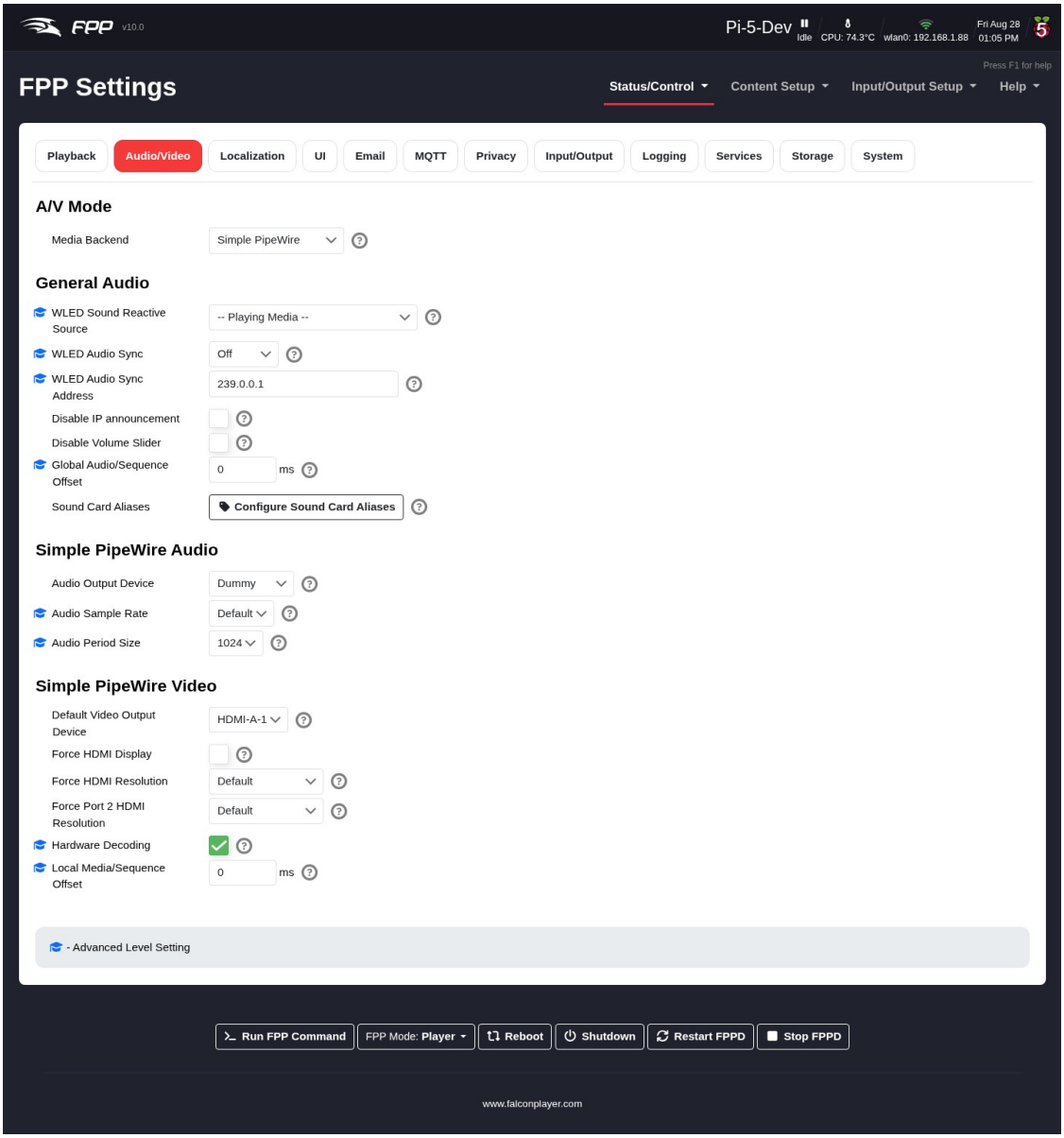
  Warning: Both offsets apply to **every** media file, not one at a time. If individual files need different offsets, fix the audio files or sequences themselves rather than using these settings.
- **Ignore Media Sync Packets** – on a remote, start and stop media when the player says so, but make no attempt to keep it in sync during playback. The media runs more smoothly but may drift away from the player. *(Advanced.)*

Scheduler sub-settings:

- **Disable Scheduler** – globally turn scheduling off.
- **Protect UI-Started Playlists from Schedule Override** – stop a scheduled item interrupting a playlist you started by hand.
- **Scheduler max timeframe to schedule out** – how far ahead the schedule is calculated (this governs the Status page's schedule *Preview* range).
- **Granular Scheduling** – finer-grained schedule control.

10.3 Audio/Video

The **Audio/Video** tab is substantially expanded in FPP 10, which uses a **PipeWire**-based audio/video pipeline (with GStreamer) for flexible routing.



Settings — Audio/Video tab (PipeWire pipeline).

- **A/V Mode → Media Backend** – selects the media backend. **Simple Pipewire** is the default setting for most systems. **PipeWire (Advanced)** enables the full routing capabilities but is an advanced concept. See PipeWire
- **Audio Output Device** – which device audio plays through: on-board analogue audio, a Raspberry Pi's HDMI output, or a USB sound card. FPP stores the choice by the sound card's stable ALSA name rather

than its index, so it survives cards being probed in a different order (for example a USB device being added or removed).

- **Default Video Output Device** – where video plays by defaul (Advanced)** enables the full routing capabilities below.
- **Audio Output Device** – which device audio plays through: on-board analogue audio, a Raspberry Pi's HDMI output, or a USB sound card. FPP stores the choice by the sound card's stable ALSA name rather than its index, so it survives cards being probed in a different order (for example a USB device being added or removed).
- **Default Video Output Device** – where video plays by default. On every FPP system a video in a playlist can be shown on a Pixel Overlay model; on a Raspberry Pi it can also go to the HDMI or composite outputs.
- **Audio Sample Rate** – the rate PipeWire's audio graph runs at. The **Default** uses 44100 unless the selected sound card clocks at something else; 44100, 48000 and 96000 can be forced. The bit depth is not set here — FPP probes the card and uses the widest format that holds the rate. *(Advanced.)*
- **Audio Period Size** – how many samples PipeWire hands the sound card at a time (the graph quantum), from 1024 up to 8192. Smaller values lower latency; larger values are less likely to glitch on a busy player. *(Advanced.)*

Note: In **PipeWire (Advanced)** mode these two are set *per card* in the PipeWire Audio Groups settings instead — see PipeWire Audio & Video Pipeline

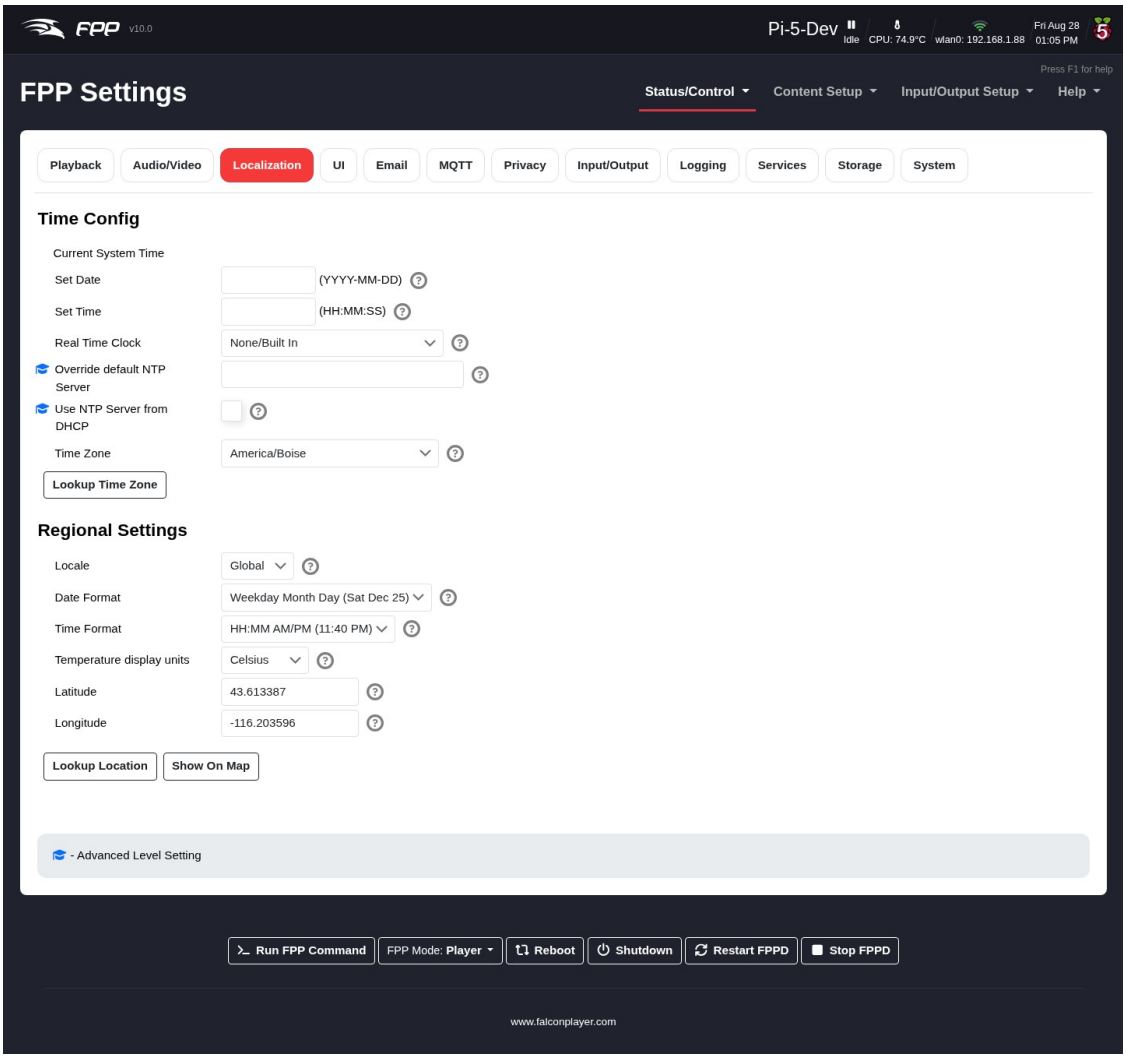
- **Force Audio Card ID** – override the card ID that FPP normally reads from the sound card's `id` file. Occasionally that ID is wrong; setting it by hand can clear the error *"Could not open audio device — ALSA: Couldn't open audio device: Invalid argument"*. *(Experimental.)*
- **Hardware Decoding** – use the hardware video decoder. Turning it off greatly increases CPU use but can give finer control over the video output. *(Advanced.)*
- **Force HDMI Display** – force a Raspberry Pi to treat HDMI as the default display. Sometimes needed when the display is not detected properly, or is not powered on when the Pi boots. While this is on, the Pi's composite output cannot be used.
- **Force HDMI Resolution** (and **Force Port 2 HDMI Resolution** on a Pi with two HDMI ports) – pin the output to a given resolution instead of whatever the monitor reports at boot.
- **General Audio** – master audio behaviour, including **Global Audio/Sequence Offset** (a fine sync trim in ms) and **Disable Volume Slider**. **Configure Sound Card Aliases** gives friendly names to audio devices.

- **WLED Sound Reactive** – drives sound-reactive effects on WLED devices from FPP's audio:
  - **WLED Sound Reactive Source** – where the sound-reactive data comes from; the default is the media FPP is currently playing. *(Advanced.)*
  - **WLED Audio Sync Address** – where sync packets are sent, and in receive mode the multicast group to join. **239.0.0.1** is WLED's default multicast group; use a unicast address (e.g. `192.168.1.70`) or a broadcast address (e.g. `192.168.1.255`) on networks where multicast is filtered. *(Advanced.)*
  - **WLED Audio Sync Port** – the UDP port, default **11988**. Only change it if you have reconfigured your WLED devices. *(Experimental.)*
- **Suspend Audio Device When Idle** – lets PipeWire suspend the sound card while nothing is playing. Left on (the default), it saves roughly 4–5% of a CPU core on single-core boards, because the audio graph would otherwise run continuously and keep the card clocked. Turn it off if your sound card does not resume cleanly when playback starts. *(Advanced; requires a reboot.)*
- **PipeWire Routing** – **Open Routing Matrix** to patch audio sources to outputs, and **Visualise Current Pipeline** to see a live PipeWire graph.
- **PipeWire Audio** – **Configure Input Mixing (Mix Buses)** and **Configure Output Audio Groups** to combine and split audio across multiple outputs.
- **PipeWire Network Streams** – **AES67 Audio-over-IP** and **Opus RTP Audio Streaming** for sending/receiving audio over the network. The AES67 page also holds the device-wide **PTP clock** settings (enable, interface, domain and clock role), and both pages show live per-stream status.
- **PipeWire Video** – **Configure Video Input Sources** and **Configure Video Output Groups** for HDMI/video routing and video-to-pixel mapping.

Note: Each of these buttons opens a dedicated configuration page with substantial functionality of its own. The whole pipeline — sound card aliases, audio output groups (delay/EQ per card), input mixing, the routing matrix, the live pipeline graph, AES67 and Opus RTP audio streaming, and video input/output groups — is documented in its own chapter, **The PipeWire Audio & Video Pipeline**, which immediately follows this one. (It replaces the simpler Audio/Video settings of FPP 9.x.)

10.4 Localization

Configures time and location. For playlists to start automatically at scheduled times, the **scheduling** FPP (not the remotes) must keep accurate time. Without internet access you can set the date and time manually, but without a Real-Time Clock (RTC) or internet the time resets on reboot.



Settings — Localization tab.

Time Config:

- **Current System Time** – the current date, time and configured time zone.
- **Set Date / Set Time** – set these manually when there is no network.
- **Real Time Clock** – if a cape/hat with an RTC is attached, select it from the list (FPP tries to detect it), reboot, then set the time here.

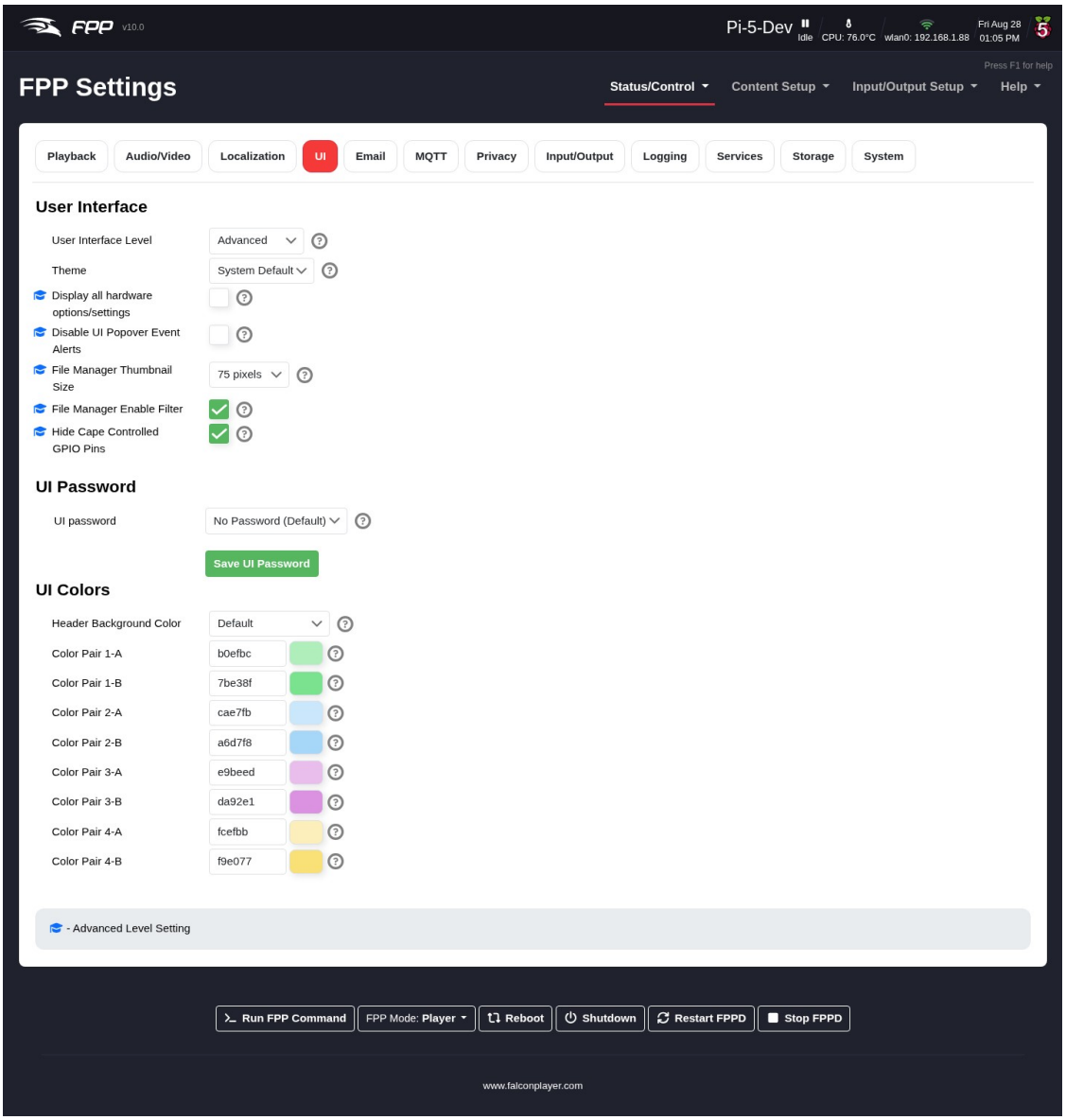
- **Use NTP Server from DHCP** – let your DHCP server supply the time server, overriding the one configured below. Off by default, in which case only the configured NTP server is used. *(Advanced.)*
- **Override default NTP Server** – normally left blank; enter a different time server's IP only in special cases. *(Advanced.)*
- **Time Zone** – required so an NTP-synced clock shows the correct local time.
- **Lookup Time Zone** – detect your time zone (requires internet).

Regional Settings:

- **Locale** – country-specific settings such as Holidays used in the Scheduler.
- **Date Format / Time Format** – how dates and times are displayed.
- **Temperature Display Units** – Fahrenheit or Celsius.
- **Latitude / Longitude** – required for sunrise/sunset scheduling. Use **Lookup Location** (verify with **Show on Map**), or obtain coordinates from `LatLong.net` or Google Maps (in Google Maps they follow the `@` in the address bar, latitude first; keep any minus sign).

10.5 UI

Changes the appearance and behaviour of the web interface.



Settings — UI tab.

User Interface:

- **Temporary User Interface Level** – shown when you are at the **Basic** level. The **Change to Advanced UI for 15 Minutes** button raises the interface to *Advanced* for the rest of your browser session, without permanently changing your UI Level — useful when you need to reach one Advanced setting and would rather not leave the extra clutter switched on. While the override is active an **unlock** icon appears in the page header showing roughly how many minutes are left; click it, or **Exit Advanced Mode** here, to return to Basic immediately. The override is per-browser and expires by itself.
- **User Interface Level** – four levels that tailor how much is shown:

  - **Basic** – all the settings most users need; the recommended setting.
  - **Advanced** – extra features/settings for unusual configurations.
  - **Experimental** – settings still in testing; changes may not work correctly until fully tested.
  - **Developer** – settings used by developers for testing; changing these can cause problems if misconfigured.
- **Display all hardware options/settings** – show settings for all devices, even those not detected. Enabling this lets you change settings that could cause problems. *(Advanced.)*
- **Disable restart/reboot UI Warnings** – for developer testing; you may then not be warned when a reboot/restart is required. *(Developer.)*
- **File Manager Thumbnail size** – size of image thumbnails in the File Manager. *(Advanced.)*
- **File Manager Enable Filter** – toggle the File Manager's sort/filter header to free up screen space.
- **Theme** – controls the web interface's appearance. **System Default** follows your browser or operating system's dark-mode preference; **Light** and **Dark** override it and always use that theme.
- **Disable UI Popover Event Alerts** – suppress the alert popovers that appear at the top right when an event fires. These alerts are useful feedback, so turn them off only for special cases such as a kiosk or a display screen. *(Advanced.)*

UI Password:

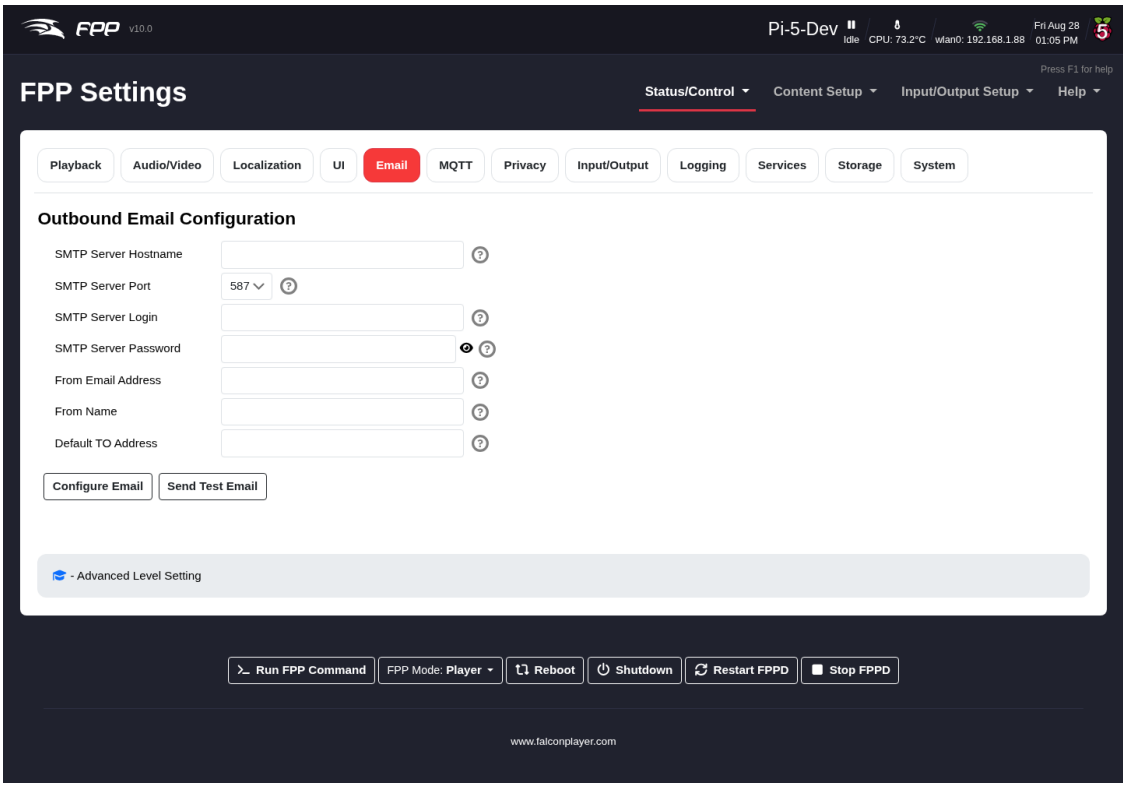
- **UI Password** – by default no password is required (the UI is only reachable from your local network). Setting one is for advanced users, as it can disable some FPP functionality without extra configuration. The password must be at least 8 characters; once set, log in with username **admin** and your password. (Defaults: username **admin**, password **falcon**.)

UI Colors:

- **Header Background Color** – colour the header to tell devices apart at a glance.
- **Color Pairs** – the colours used in tables such as the Schedule Preview, making scheduling problems easier to spot.

10.6 Email

Lets FPP send email (via FPP commands or a script).



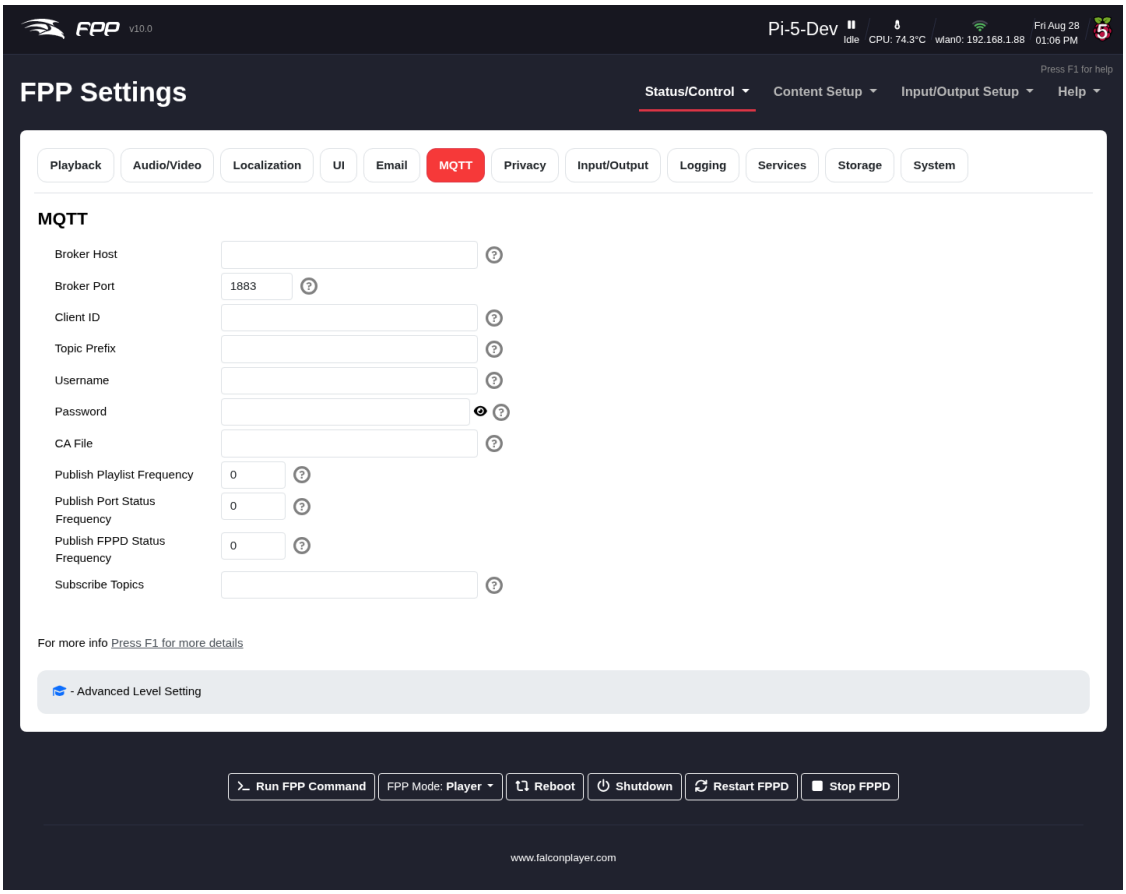
Settings — Email tab.

- **SMTP Server Hostname / Port** – your mail server and port (587 is most common; 465 and 25 are also used).
- **SMTP Server Login / Password** – credentials for the sending account.
- **From Email Address / From Name** – the sender shown on the email (the From Name could be the FPP host name).
- **Default TO Address** – the default recipient.
- **Configure Email** saves the settings; **Send Test Email** tests them.

Note: Some providers (e.g. Gmail, Yahoo) block third-party clients by default; you may need to adjust their security settings to allow FPP to send.

10.7 MQTT

Connects FPP to an MQTT broker for automation (e.g. a home-automation system).



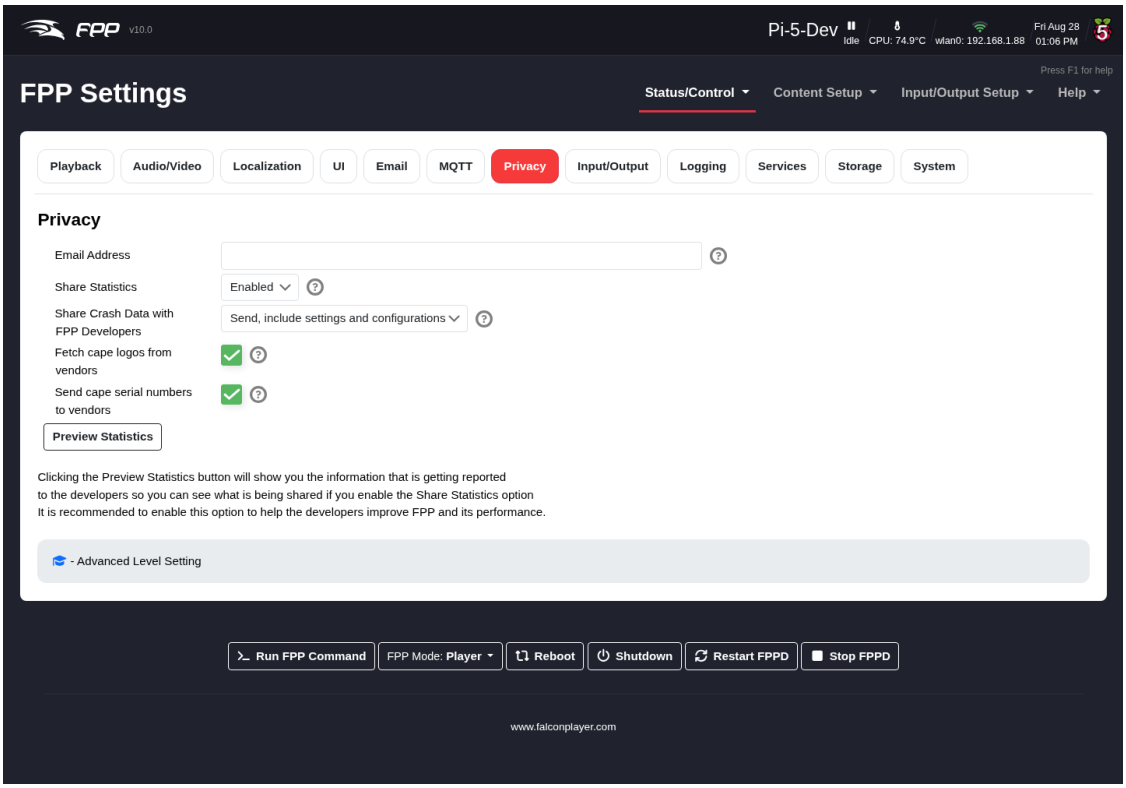
Settings — MQTT tab.

- **Broker Host / Port** – the broker's address and TCP port.
- **Client ID** – left blank, the broker assigns one.
- **Topic Prefix** – prefix used when publishing messages.
- **Username / Password** – broker authentication.
- **CA File** – optional CA to validate the broker's certificate (only for SSL with self-signed certificates).
- **Publish Playlist Frequency** – how often, in seconds, to publish playlist status to the broker. **0** means it is not published on a timer, but is still available on demand.
- **Publish FPPD Status Frequency** – how often, in seconds, to publish the FPPD status JSON. **0** disables periodic publishing.
- **Publish Port Status Frequency** – how often, in seconds, to publish port monitoring status JSON (on capes that report it). **0** disables it.
- **Subscribe Topics** – one or more additional topics to subscribe to. Use `#` on its own to subscribe to everything the broker publishes, a prefix filter such as `smartthings/#` for a whole tree, or an exact topic for just one. Separate several with a semicolon (`;`).

Tip: Anything received on a subscribed topic shows up on the **Variables** page under *MQTT Read-only Variables*, so you can act on it with an **If** command — see Variables.

10.8 Privacy

FPP's developers are cautious about privacy and let you customise what is shared.



Settings — Privacy tab.

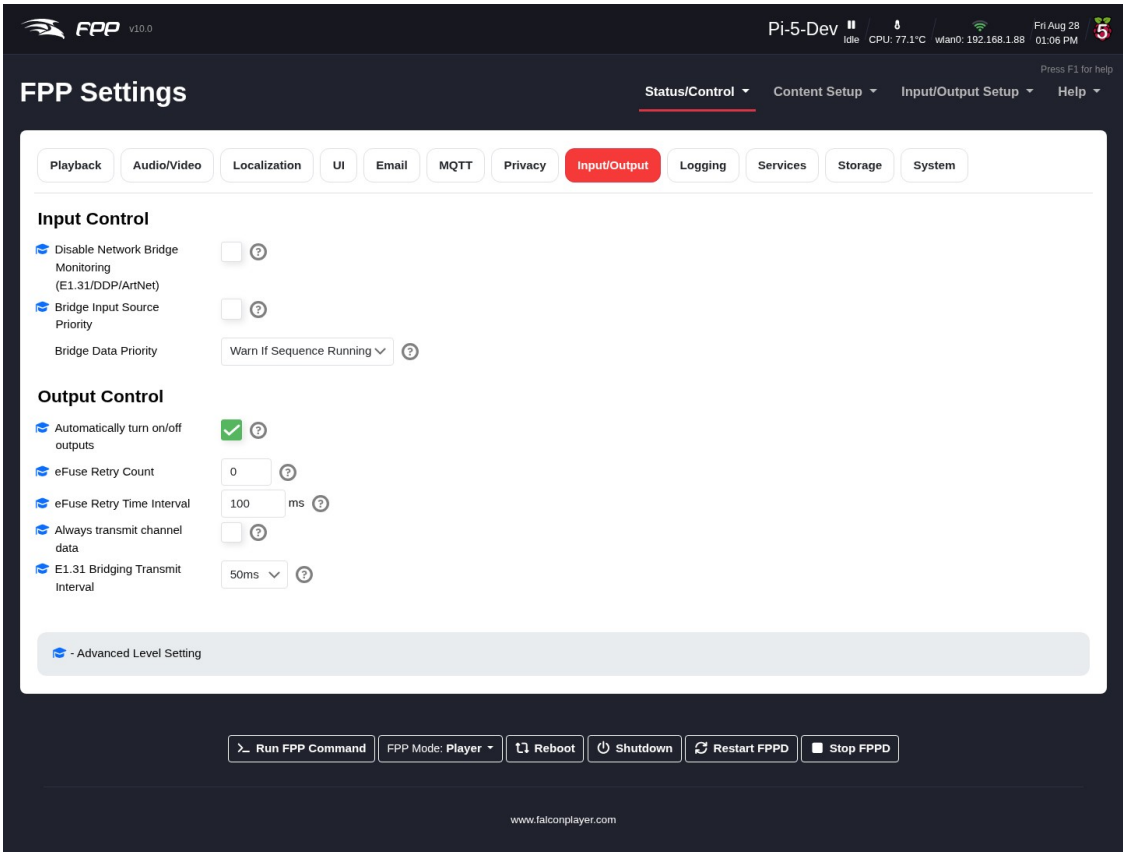
- **Email Address** – if you share crash data, this lets developers contact you for more information.
- **Share Statistics** – anonymous usage statistics (SBC type, installed plugins, FPP version, etc.) with no personally identifying information; click **Preview Statistics** to see exactly what is sent. Options: **Enabled** (recommended), **Disabled**, or **Banner** (prompt on the Status page).
- **Share Crash Data with FPP Developers** – choose what, if anything, is sent to help diagnose crashes. The default *Include settings and configurations* is recommended. FPP 10 also **keeps crash reports on the device** even when you choose not to send them, so you can inspect one yourself or attach it to a support bundle. The reports carry more useful detail than before — the context around the crash,

the versions of any installed plugins, and stack addresses resolved to file and line on the device itself.

- **Fetch cape logos from vendors** – a vendor logo shown in the header must be downloaded from the vendor, which exposes your IP to them (usually low risk).
- **Send Cape serial numbers to vendors** – could identify you from purchase history (usually not a security issue), but you can disable it.

10.9 Input/Output

Global input/output settings *(Advanced UI Level or higher)*.



Settings — Input/Output tab.

Input Control:

- **Disable Network Bridge Monitoring** – disable bridge monitoring (useful when developing your own bridge listener). *(Advanced.)*
- **Bridge Data Priority** – how FPP treats incoming bridge data versus local playback:
  - **Warn if Sequence is running** – warn but keep playing the local sequence.

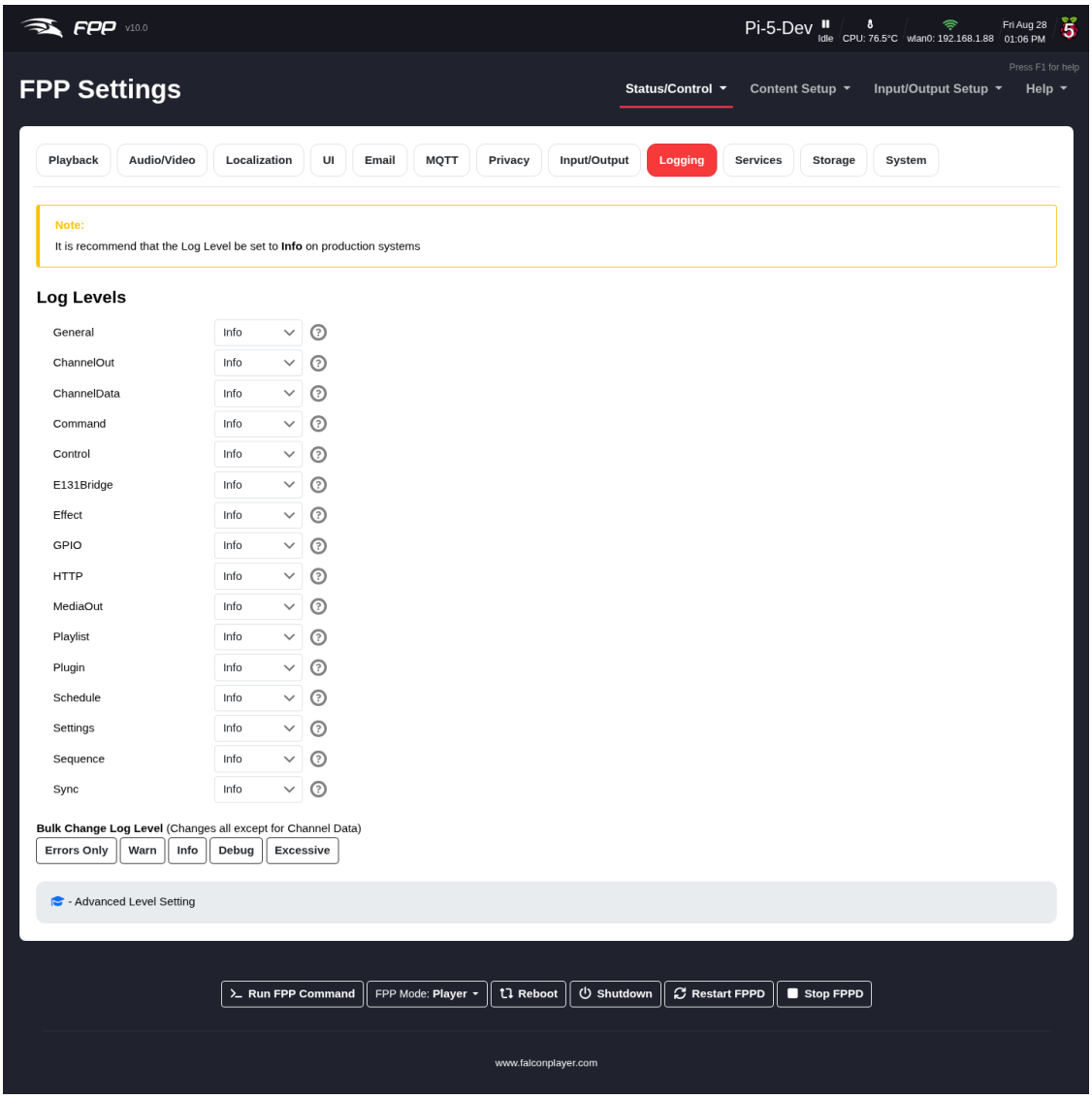
  - **Prioritize Bridge** – incoming bridge data overrides local sequences.
  - **Prioritize Sequence** – local sequences override bridge data; bridge data is used only when nothing is playing (usually what you want during show season, so bridging does not interrupt the show).
- **Bridge Input Source Priority** – when several senders transmit to the same input universe, lock onto one source and suppress the others. For **E1.31** the priority byte in the packet header is honoured, so a higher-priority source always wins; for **ArtNet**, which has no priority field, the first source seen wins. In both cases, if the active source stops sending for longer than the input timeout, the next packet from any source takes over. **DDP** is unaffected. *(Advanced.)*
- **Hide Cape Controlled GPIO Pins** – on by default. Hides GPIO pins the fitted cape is already using from the GPIO pin control page, so you cannot change a pin the cape depends on by accident. *(Advanced.)*

Output Control *(Advanced)*:

- **Automatically turn on/off outputs** – for controllers that can cut output power when idle.
- **Efuse Retry count / interval** – automatically reset tripped eFuses (good for intermittent trips, especially at startup), and the wait between retries.
- **Always transmit channel data** – force output whenever FPP is running (FPP normally transmits only when a sequence plays or an overlay model is enabled). Use only for older controllers that go into test mode without data.
- **E1.31 Bridging Transmit Interval** – timing interval in bridge mode (default 50 ms, recommended; some devices only support 50 ms).
- **Disable Colorlight outputs on link down** – by default FPP disables ColorLight outputs when the link is down, below 1 Gbps, or no receiver is detected (a restart re-enables it). Disabling this keeps the output active but you still get the warnings.
- **Colorlight Firmware Version** – manually select the ColorLight receiver firmware version if FPP cannot detect it.

10.10 Logging

Sets the logging criteria for the device. FPP creates several logs that help with troubleshooting. Normally leave this at **Info** unless the development team asks otherwise or you are an advanced user.



Settings — Logging tab.

You can set the level per subsystem. The subsystems include **ChannelData** (serial and LOR output), **ChannelOut** (channel testing and overlays), **E131Bridge** (E1.31 and DDP input bridging), **MediaOut** (audio and video output), and others covering the scheduler, playlists, plugins and MultiSync.

The five levels are:

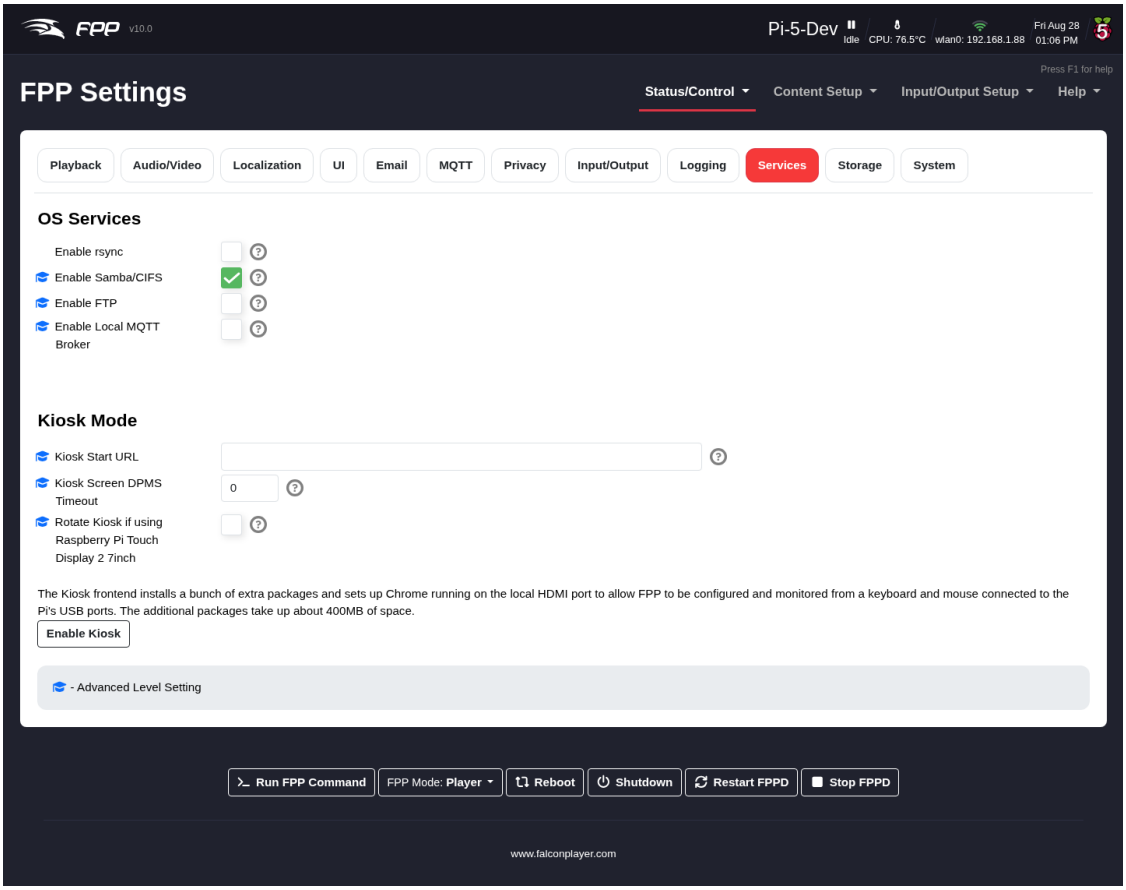
- **Errors Only** – only items identified as errors.
- **Warn** – only warnings.
- **Info** – basic information suitable for most troubleshooting (recommended for production systems).
- **Debug** – Info plus debug messages; use only when requested.

- **Excessive** – everything; can create very large log files and impact performance — use only when requested.

Note: Buttons at the bottom let you change all sections at once (except Channel Data).

10.11 Services

Configures optional system services; options depend on the SBC.



Settings — Services tab.

OS Services:

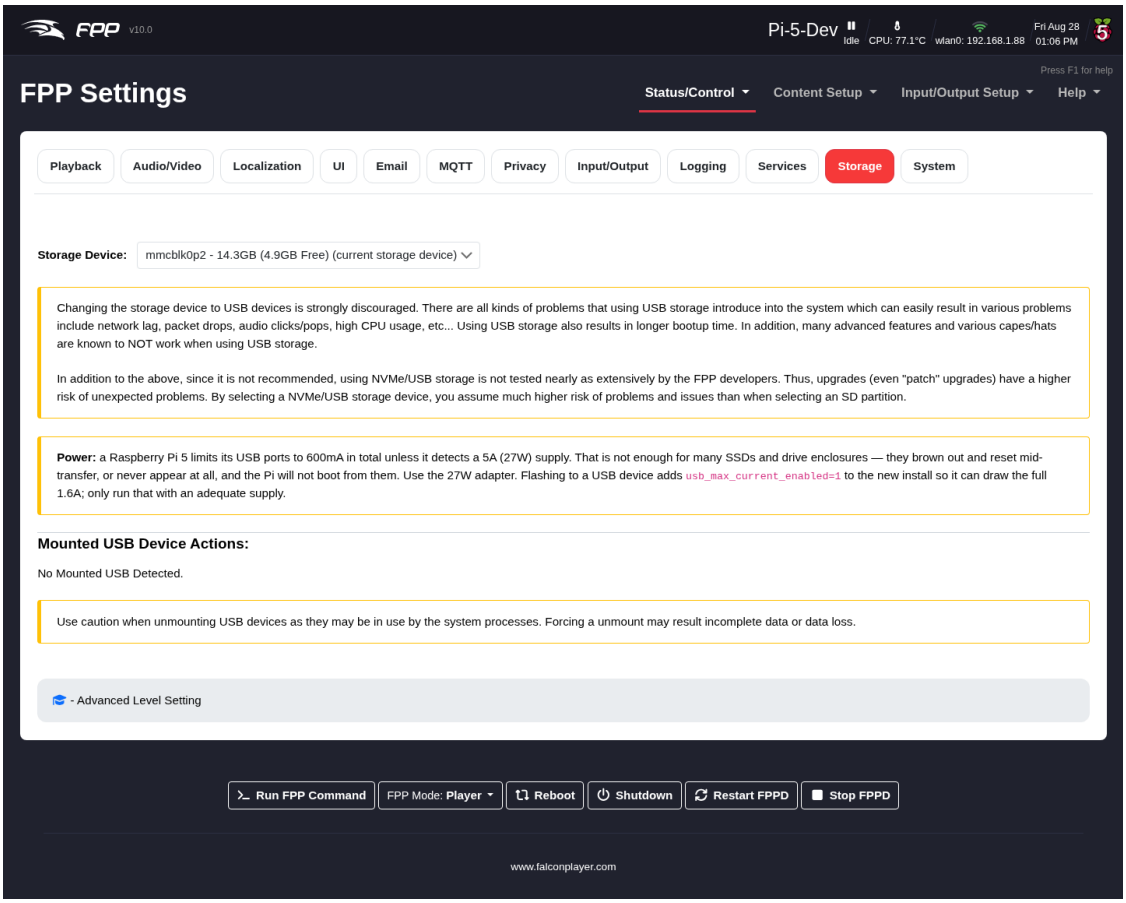
- **Enable rsync** – allow this device to receive files from other FPP devices (e.g. from the MultiSync page).
- **Enable Samba/CIFS** – access the media folder over SMB (e.g. Windows File Explorer). *(Advanced.)*
- **Enable FTP** – transfer files with an FTP client. *(Advanced.)*
- **Enable Local MQTT Broker** – run a local MQTT broker. *(Advanced.)*

Kiosk Mode *(Pi only, Advanced)* – for a fully standalone device with a connected touchscreen (an advanced configuration; not all touchscreens are supported):

- **Kiosk Start URL** – the page shown at startup (default: the Status page).
- **Kiosk Screen DPMS Timeout** – seconds before the display sleeps.
- **Rotate Kiosk** – tick this if you are using the **Raspberry Pi Touch Display 2** (7 inch), which needs the output rotated to appear the right way up.
- **Enable Kiosk** installs and enables Kiosk mode.

10.12 Storage

Configures where sequences and media are stored, and manages the device's storage media *(Advanced UI Level or higher)*.



Settings — Storage tab.

Flash FPP to Another Device:

FPP 10 replaced the old platform-specific eMMC/USB flashing pages with one flow that works on Raspberry Pi, BeagleBone and BeagleBone 64 alike. Each detected target device is listed with its own buttons:

- **Create** – write a clean FPP install to the target, leaving your media, sequences and settings behind.

- **Copy** – the same, but bring your media, sequences and settings along.
- **Create (BTRFS)** – as *Create*, using BTRFS, which compresses the file system but may slow the device slightly.

Either way the copy is given its **own SSH host keys**, so it can safely run as a separate player alongside this one.

> **Warning:** Flashing **erases everything** currently on the target device. Check you have selected the right one — the confirmation dialog names the device and its `/dev/` path.

Storage Device:

Selects which device holds the media directory (sequences, audio, video, images, logs). A device that is not yet mounted can be formatted from here, and when you change the location FPP offers to copy all existing files across.

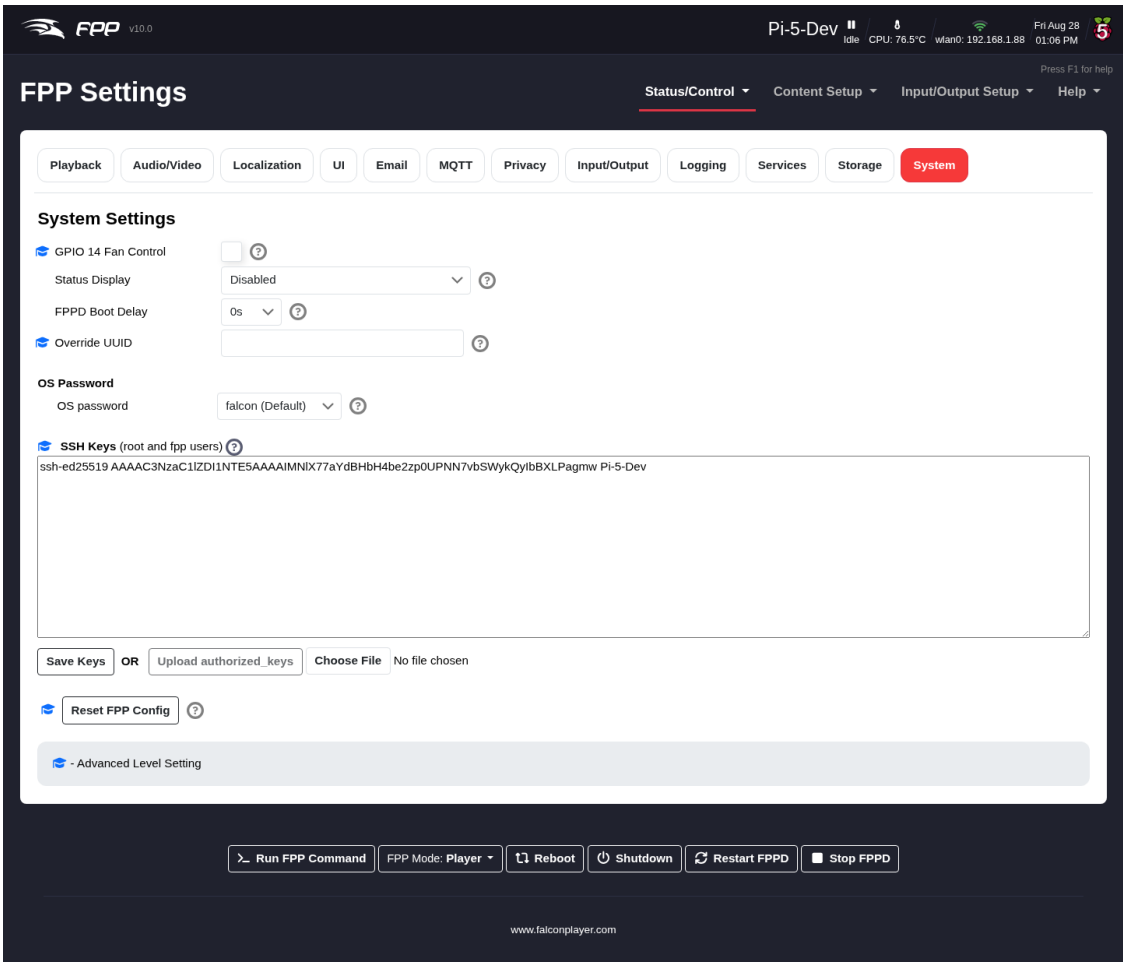
> **Warning:** On a **Raspberry Pi 4 or 5**, moving storage to a USB or NVMe device is **strongly discouraged**. It can introduce network lag, packet drops, audio clicks and pops, high CPU usage and longer boot times, and many advanced features and several capes/hats are known **not** to work with USB storage. It is also far less tested, so even patch upgrades carry a higher risk. Prefer an SD partition.
>
> **Note:** A Raspberry Pi 5 limits its USB ports to 600 mA in total unless it detects a 5 A (27 W) supply, which matters if you are powering storage from the USB ports.

Mounted USB Device Actions:

- **Force Unmount** – unmount a USB device that is still mounted. If files are open on it, FPP lists the processes holding them so you can see what is in the way.

10.13 System

System-wide settings; the page varies with the SBC and its hardware (there are separate Raspberry Pi and BeagleBone variants).



Settings — System tab.

- **GPIO 14 Fan Control** – PWM fan control on GPIO 14. *(Pi only, Advanced.)*
- **Fan On Temperature** – the temperature above which that cooling fan switches on (default 70 C). Fan state is reported on the System Health Check page. *(Advanced.)*
- **Disable IP announcement** – during boot FPP announces its IP addresses over the audio output. Turn this off for production use, when the audio output feeds an FM transmitter or your show speakers.
- **Override UUID** – set the UUID FPP uses for FPP Connect de-duplication, stats and services, instead of deriving it from the board's serial number. Only needed when two devices end up reporting the same identity. *(Advanced.)*
- **Status Display** – configure an OLED screen on the I2C bus (usually auto-detected at boot); it shows IP addresses, status and the playing sequence. Select your OLED model here.

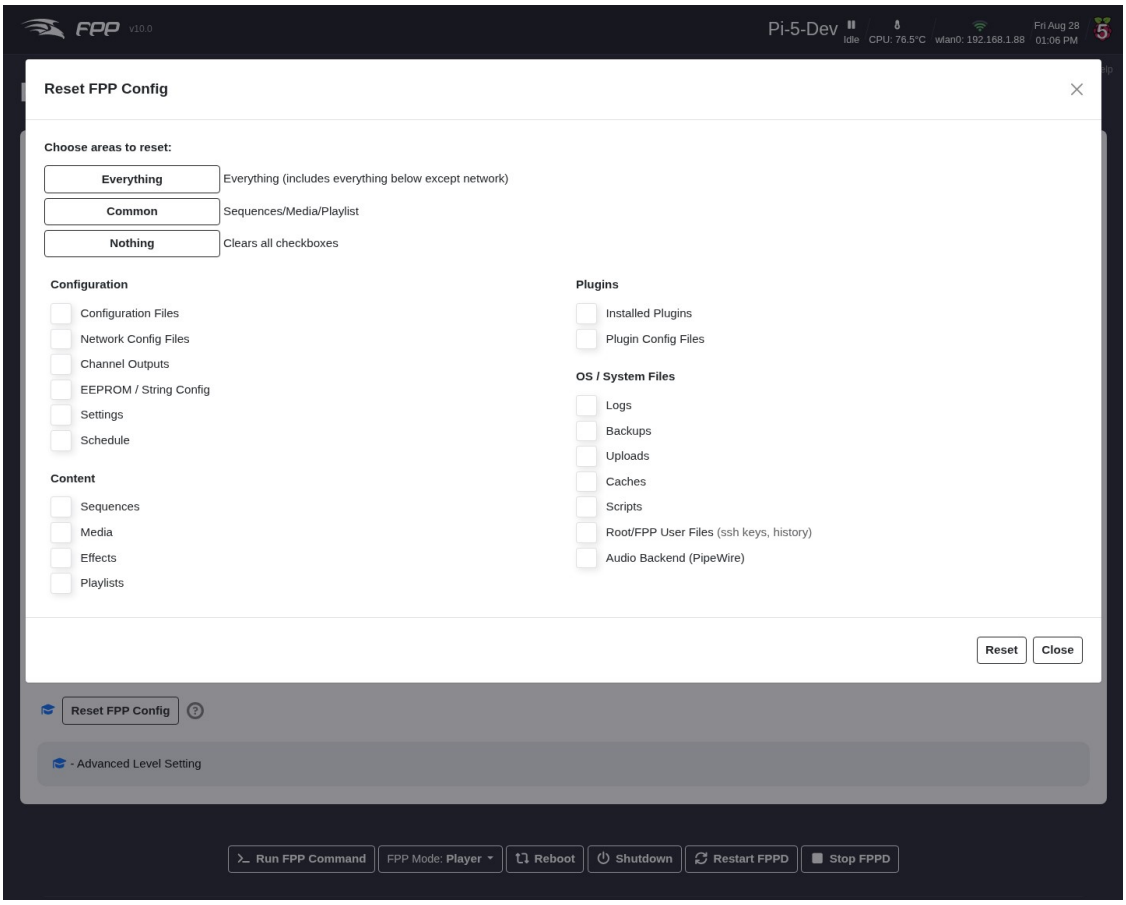
- **FPPD Boot Delay** – delay FPP's startup, useful if you power everything on together and want routers/switches to initialise first. The **Auto** setting waits for a valid time source before booting (falling back to internal time after 10 minutes).
- **Enable HDMI Display** – enable the HDMI port on a BeagleBone. *(BeagleBone only, Advanced.)*

  Warning: Enabling the BeagleBone's HDMI port **disables many GPIO pins**, which will stop most capes working. Only turn it on if you are not using a cape.
- **BeagleBone LEDs** – control or disable the five on-board LEDs (commonly disabled if distracting; defaults recommended). *(BeagleBone only.)*
- **Reboot If USB WiFi Adapter Fails** – on by default. Some USB Wi-Fi adapters intermittently lose their USB connection while booting: the adapter is detected and the interface appears, but the radio never connects, leaving FPP running normally yet unreachable over the network. When FPP detects this specific failure it reboots to recover. It will only do so when USB errors are present *and* no other connection has an IP address — so an access point being switched off or out of range will not trigger one — and it gives up after two attempts so it can never boot-loop. *(Advanced.)*
- **OS Password** – the password for SSH and similar access; the default `falcon` is recommended.
- **SSH Keys** – configure SSH keys to authenticate with a key instead of a password. *(Advanced.)*
- **Reset FPP Config** – reset FPP to factory settings, either all options or selected areas — useful if a configuration or an xLights upload has gone wrong. See The Reset FPP Config dialog below.

  Warning: Take an FPP Backup before **Reset FPP Config** (see Backup and Restore).

10.13.1 The Reset FPP Config dialog

Reset FPP Config opens a dialog where you choose exactly what to clear, rather than resetting everything blindly.



Choosing what to reset.

Three shortcut buttons set the tick boxes for you: **Everything** (everything below *except* network), **Common** (sequences, media and playlists), and **Nothing** (clears all the boxes so you can pick your own). The areas are:

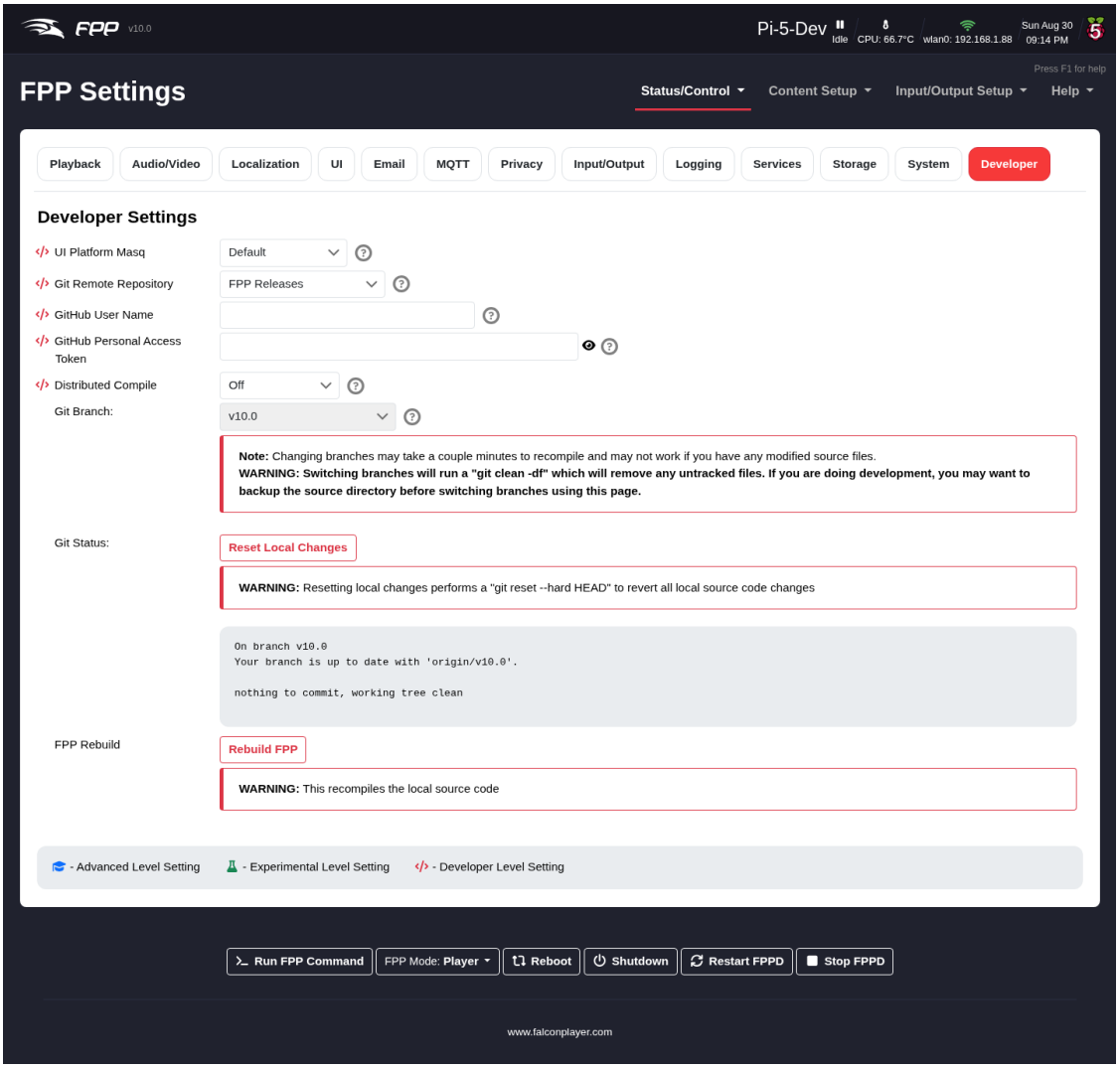
- **Configuration** – **Configuration Files**, **Network Config Files**, **Channel Outputs**, **EEPROM / String Config**, **Settings**, **Schedule**.
- **Content** – **Sequences**, **Media**, **Effects**, **Playlists**.
- **Plugins** – **Installed Plugins**, **Plugin Config Files**.
- **OS / System Files** – **Logs**, **Backups**, **Uploads**, **Caches**, **Scripts**, **Root/FPP User Files** (ssh keys, history), **Audio Backend (PipeWire)**.

Click **Reset** to apply, or **Close** to back out.

> **Note: Network Config Files** is deliberately excluded from *Everything*, so a reset cannot cut off your own access to the device. Tick it explicitly if you really do want the network configuration cleared — and note that any DHCP server leases this device has issued are then cleared on the **next reboot**, not immediately.

10.14 Developer

Only shown at the **Developer** UI Level; useful for switching FPP versions or developer testing.



Settings — Developer tab.

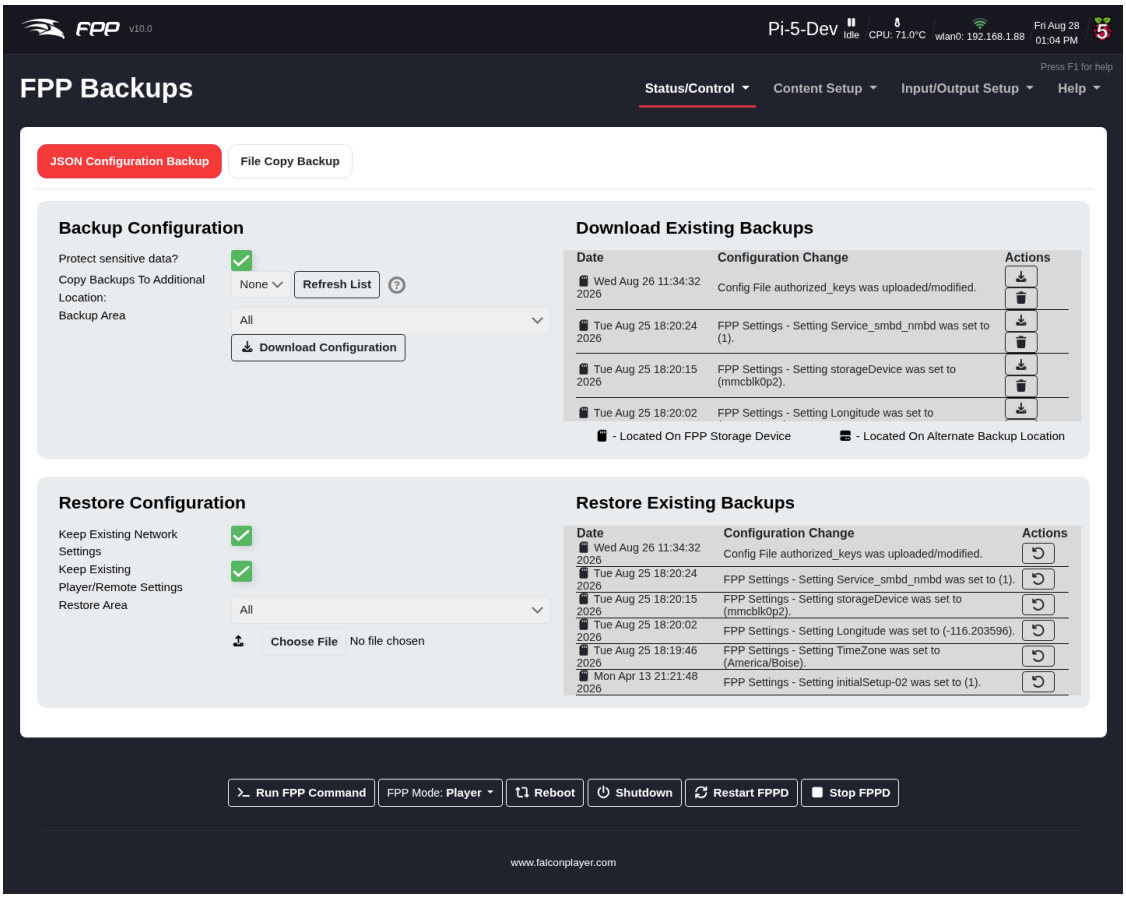
- **UI Platform Masq** – display settings for a different platform than the one detected (for plugin/feature development). Take care, as changes can adversely affect your device.
- **Git Remote Repository** – which git remote branch selection uses: **origin** is the main FPP repository, **newfeatures** is for new-feature testing.
- **Git Branch** – choose which FPP version/branch to run, e.g. **Master** for the latest improvements ahead of release. Note that some upgrades require an OS rebuild to get all benefits (see Final Configuration and Updating).

- **Reset Local Changes** – revert any manual code changes to the original code.
- **Git Status** – show the status of your local FPP version.
- **FPP Rebuild** – recompile all FPP files (useful after an interrupted install or corrupted files).
- **GitHub User Name** and **GitHub Personal Access Token** – credentials used to install or upgrade plugins hosted in **private** GitHub repositories. The token needs read access to those repositories and is stored locally in `/home/fpp/media/settings`. They are only used when *Use Credentials* is selected on the Plugins page.
- **Distributed Compile** – hand FPP's source compiles to a helper host to speed up a full rebuild, such as after switching branch. The options are **Off**, **distcc**, **distcc + zeroconf**, **nocc** and **nocc + mdns**. `distcc` preprocesses locally then compiles on the helper; `nocc` does *not* preprocess locally, which makes it far faster on slow single-core boards such as a BeagleBone Black or Pi Zero. The `zeroconf`/`mdns` variants discover helpers on the LAN automatically. The helper must run the same major compiler version — set one up with `scripts/setup_distcc_host.sh` or `scripts/setup_nocc_host.sh`. Any distributed mode disables the precompiled header.
- **Compile Hosts** – the space-separated list of helper hosts to compile on. For distcc use `host[:port][/jobs][,options]`, e.g. `fpppi5:3632/6,lzo`; for nocc use `host[:port]` (default port 43210), e.g. `fpppi5`. Leave it empty for the *zeroconf* / *mdns* modes, which discover their helpers automatically.

11 Backup and Restore

11.1 FPP Backup

FPP has several backup options. You can save just your configuration files (the **JSON Configuration Backup**), or your configuration **and** all relevant files (the **File Copy Backup**). FPP also creates a backup of your configuration every time you make a system change. Open **Status/Control → FPP Backup**.



The FPP Backup page.

11.1.1 JSON Configuration Backup

This saves all or part of your settings to your computer to restore later. It saves only the selected **configuration** files — **not** sequences, audio or video.

Backup Configuration (creating a backup):

- **Protect sensitive data** – when selected, your wlan0 network password is **not** saved, and you must re-enter it after restoring. When cleared, a complete backup is saved and the device should be fully functional when restored — but anyone with the backup file can read your wireless password from it.
- **Backup area** – which portion of the configuration to save; normally select **all**, or choose individual sections.
- **Download Configuration** – save the configuration to your computer. The file is named with the device name and a timestamp so you can identify the newest.
- **Download Existing Backups** – FPP keeps a backup after every system change; download any of them here.

> **Note:** This does **not** save media files such as sequences, music or videos.

Restore Configuration:

- **Keep Existing Network Settings** – if selected, the device's saved network settings are not overwritten by the backup.
- **Keep Existing Player/Remote Settings** – if selected, the Player/Remote settings are not overwritten.
- **Restore Area** – restore only a specific area; other settings are left alone.
- **Choose File** – select the backup file to restore from (check you have the right one if you keep several).
- **Restore Configuration** – restore the selected areas from the chosen file.
- **Restore Existing Backups** – restore from one of FPP's automatic backups.

11.1.2 File Copy Backup

The File Copy Backup copies **every item** stored on the device except the operating system — useful for keeping full copies of your FPP devices. You can save to several locations.

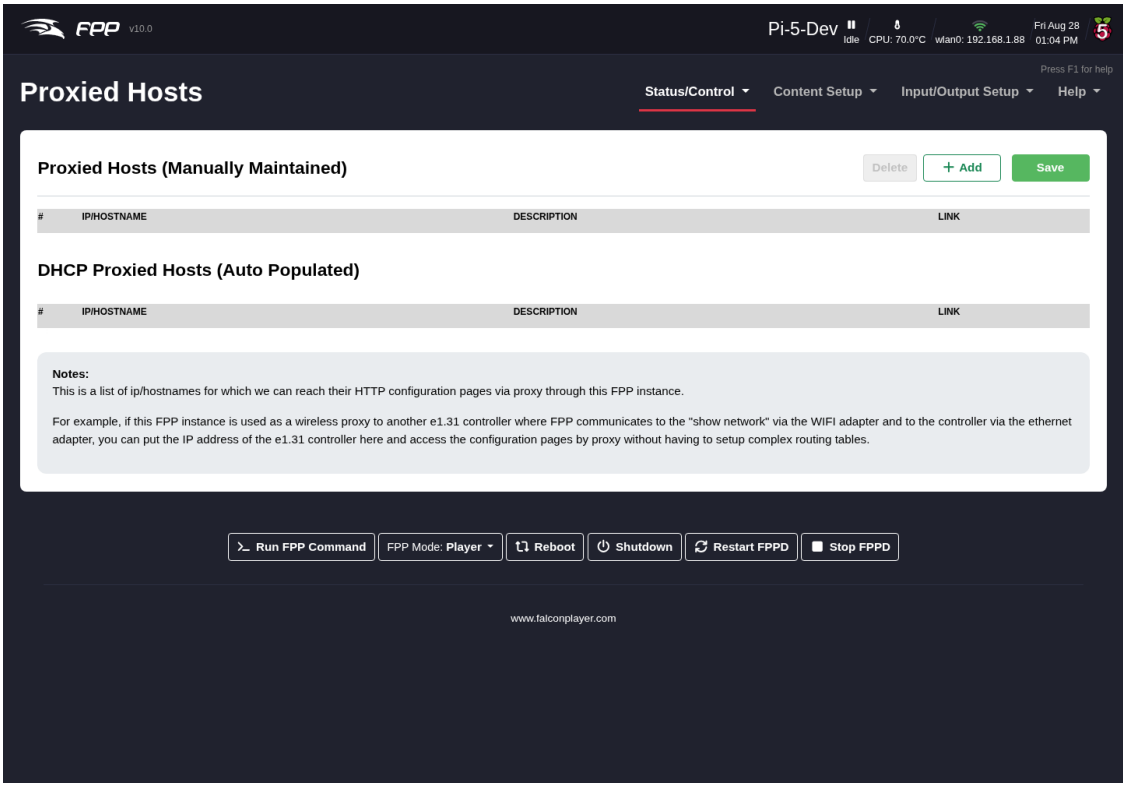
> **Note:** If you plug in a USB drive after the device has booted, press **Refresh List** to detect it.

- **Copy Type** – the operation to perform:
  - **Backup To USB / Restore From USB** – copy selected items to/from a USB drive on this device. *(Restore overwrites existing files.)*
  - **Backup To / Restore From Local FPP Backups Directory** – copy to/from a backup folder on this device's SD card.
  - **Backup To / Restore From Remote FPP Backups Directory** – copy to/from a backup folder on **another** FPP device on your show network (you enter its host name or IP).
- **USB Device** – appears for USB options; pick the drive (use **Refresh List** if it is not shown).
- **Hostname/IP** – appears for remote options; the remote device's address.
- **Backup Path** – for *Copy To*, defaults to this device's host name (change with care); for *Copy From*, lists the available backup directories.
- **What to copy** – the items to include.
- **Delete extras** – on restore, delete any files on the device before restoring from the backup folder.

> **Note:** There is no advance warning if there is not enough space for a backup; during the process you will see an `rsync ... No space left on device` error. An incomplete backup will not restore completely.

12 Proxy Settings

Proxy Settings route network traffic through an FPP device to a connected controller. Open **Status/Control → Proxy Settings**.



The Proxy Settings page.

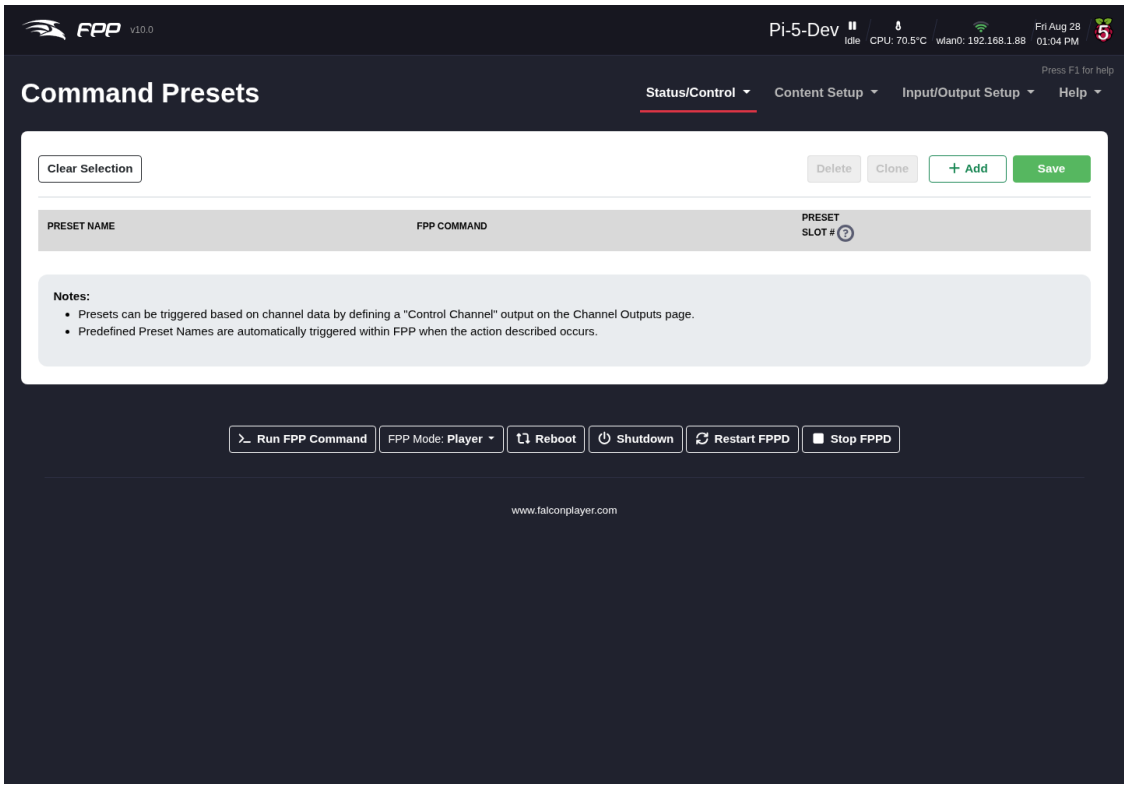
Configure FPP as a **Proxy Host** by entering the IP address of the controller(s) attached to it, so you do not need static routes on your computer or router (especially helpful on Macs, where routes are not persistent). To reach a proxied controller's web UI, click its link on the Proxied Hosts page, or enter the FPP device's IP followed by `/proxy/` and the controller's address — for example, if the FPP's wlan0 IP is `192.168.1.101` and the controller is `192.168.101.2`, browse to `192.168.1.101/proxy/192.168.101.2`.

> **Note:** Not all controllers support being proxied. Falcon controllers (with current firmware) and KulpLights controllers do.

Configuring proxies from xLights (recommended): in the example, an F16 uses a Raspberry Pi as its Proxy Host. The Pi has a Wi-Fi (wlan0) address of `192.168.1.200` on the home network and an eth0 address of `192.168.200.2`; the F16 is wired to the Pi at `192.168.200.3`. In xLights, define the controller that *needs* the proxy with the IP address of the FPP device acting as proxy — the more globally reachable interface, here the Pi's wlan0 address (`192.168.1.200`). Then in **FPP Connect**, tick **Upload** for the Proxy Host (the Pi) and enable **Add Proxies** (do **not** add proxies for the controller that needs the proxy), and set **UDP Out** to **Proxied**.

13 Command Presets

Command Presets replace the earlier *Events* feature (greatly enhanced). A preset is a saved way to run an **FPP Command**. Because FPP Commands can now be used directly in playlists and on GPIO inputs, you no longer *need* to create a preset first — but if you will use the same command in more than one place, a preset makes it easier to reuse. Open **Status/Control → Command Presets**.



The Command Presets page.

13.1 How presets are triggered

A Command Preset can be triggered in four ways:

- **Playlist** – as a playlist entry (e.g. a Lead In item that switches on a relay for a radio or prop at the start of the show, and off at the end).
- **Sequence** – in the middle of a sequence (e.g. triggering a countdown on a matrix at set points).
- **GPIO Input** – from a GPIO pin (e.g. a push button that starts your show).
- **Manual Trigger** – from the Command Presets page itself (useful for testing).

13.2 Creating a preset

- **Preset Name** – use a clear name that describes what it does, e.g. `StartMainPlaylist` or `StartOvernightPlaylist`.
- **FPP Command** – choose from the many built-in commands (plugins can add more).

13.3 How commands are organized

FPP 10 groups the command list into **categories**, and ties each command to a **UI Level**, so the list you see matches how much of FPP you have chosen to expose (see FPP Settings → UI). The categories are:

Category	What it covers
Audio	Volume control, per-slot volume, and audio routing
Effects	Start/stop effects and FSEQ-as-effect, All Lights Off
Events	Presets, scripts, URLs, and the *If* / *Set Variable* commands
Media	Play and stop media files, and per-slot media control
Outputs	Test mode, GPIO output and MQTT publishing
Pixel Overlay	Fill, clear, text and effects on overlay models
Playlist	Start, stop, pause, insert and navigate playlists
Plugins	Commands created by installed Plugins
System	Reboot, shutdown, and switching between Player and Remote mode

At the **Basic** level you see the everyday commands — starting and stopping playlists, effects and media, and volume. Raising the level to **Advanced** unlocks the rest, including *If*, *Set Variable*, the preset-slot and remote

triggers, GPIO and MQTT commands, testing, pixel-overlay commands, and the **Reboot** and **Shutdown** commands.

> **Tip:** Each command *argument* now has its own help tooltip, so you can hover a field in the FPP Command Editor to find out what it expects — including the newer *ifNotRunning* and MultiSync arguments.

13.4 Available FPP Commands (selection)

The command list is extensive; commonly used commands include:

- **All Lights Off** – turn all lights off.
- **Effect Start / Effect Stop / Effects Stop** – start a saved effect, stop one, or stop all.
- **Extend Schedule** – extend (or, with a negative number, shorten) the currently playing scheduled playlist by a number of minutes.
- **FSEQ Effect Start / Stop** – start or stop any stored `.fseq` file; can loop, and can run in the **Background** (resuming after a playlist finishes). An **ifNotRunning** option starts it only if that sequence is not already playing, and it can restart an already-running effect in place instead of starting a duplicate.
- **GPIO** – set GPIO pins on or off.
- **If** – run one set of commands when a condition is true and another when it is false. See Variables.
- **Insert Playlist After Current** – queue a playlist to run after the current one finishes (with optional start/stop items), then resume.
- **Insert Playlist Immediate** – start a playlist immediately, stopping the current one, then resume it afterwards.
- **Insert Random Item From Playlist** – insert a random item (immediately or after the current song).
- **Next / Prev / Restart Playlist Item** – navigate within a playing playlist.
- **Outputs On / Off** – turn outputs on/off (on capable devices).
- **Overlay Model Clear / Fill / State** – clear an overlay model, fill it with a colour, or set its state (Enabled, Disabled, Transparent, Transparent RGB).
- **Overlay Model Effect** – apply an effect to an overlay model: **Bars**, **Blink**, **Color Fade**, **Text**, **Images** (draw or scroll an image across the model — new in FPP 10), **WLED Effects** (some sound-reactive, marked with a musical note), or **Stop Effects**.
- **Pause / Resume Playlist** – pause or resume the current playlist.
- **Play Media** – play a media file (optionally onto an overlay model). It can also target a specific **video output**, so a non-primary slot can drive a second display.

- **Reboot / Shutdown** – restart or power down the device from a command, so a schedule entry or GPIO button can do it (Advanced UI level).
- **Remote Effect / FSEQ / Playlist / Script / Command Preset** – trigger effects, sequences, playlists, scripts or presets stored on a **remote** device (enter the name/slot exactly as stored on the remote).
- **Run Script** – run a script stored on this device.
- **Set Variable** – store a value (fixed, counter, random or calculated) in a named variable for other commands to read. See Variables.
- **Start Playlist** – start a stored playlist (also available directly in playlist entries and GPIO inputs).

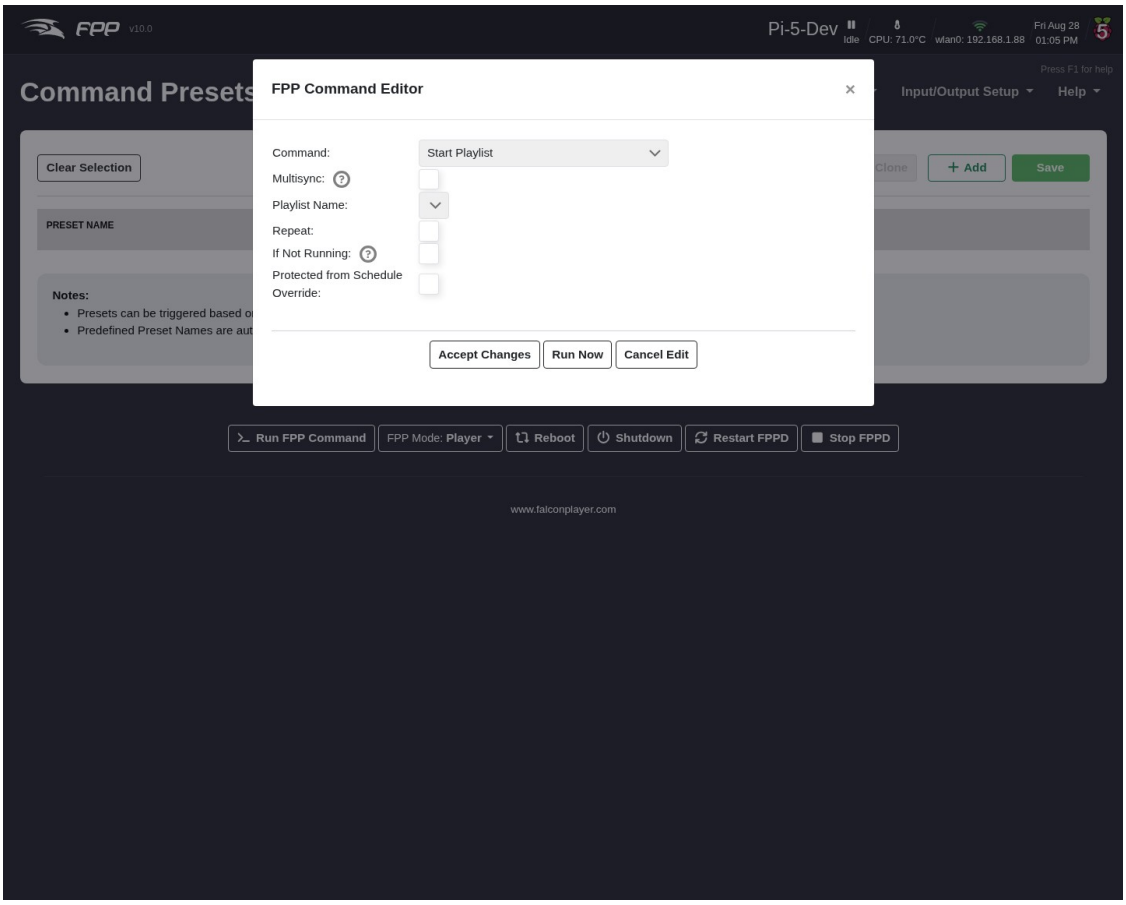
13.5 FPP System Event Commands

FPP has some built-in commands that are triggered by system events. These commands are available in the drop down list before you enter any text.

- FPPD_STARTED- This will trigger any time FPPD is started.
- FPPD_STOPPED- This will trigger any time FPPD is stopped.
- PLAYLIST_STARTED- This will trigger each time a playlist is started.
- PLAYLIST_STOPPED- This will trigger each time a playlist is stopped.
- PLAYLIST_START_TMINUS_xxx- This will trigger xxx seconds before a playlist is started.
- SEQUENCE_STARTED- This will trigger each time a sequence is started.
- SEQUENCE_STOPPED- This will trigger each time a sequence is stopped.
- MEDIA_STARTED- This will trigger each time a media file is started.
- MEDIA_STOPPED- This will trigger each time a media file is stopped.
- Outputs Enabled-If your FPP device has the ability to turn on the outputs, then you can use this command to trigger an action when the Outputs get enabled.
- Outputs Disabled-If your FPP device has the ability to turn off the outputs, then you can use this command to trigger an action when the Outputs get disabled.

13.6 The FPP Command Editor

Wherever FPP lets you choose a command — a preset, a playlist entry, a GPIO trigger, a scheduler entry — it opens the same **FPP Command Editor** dialog.

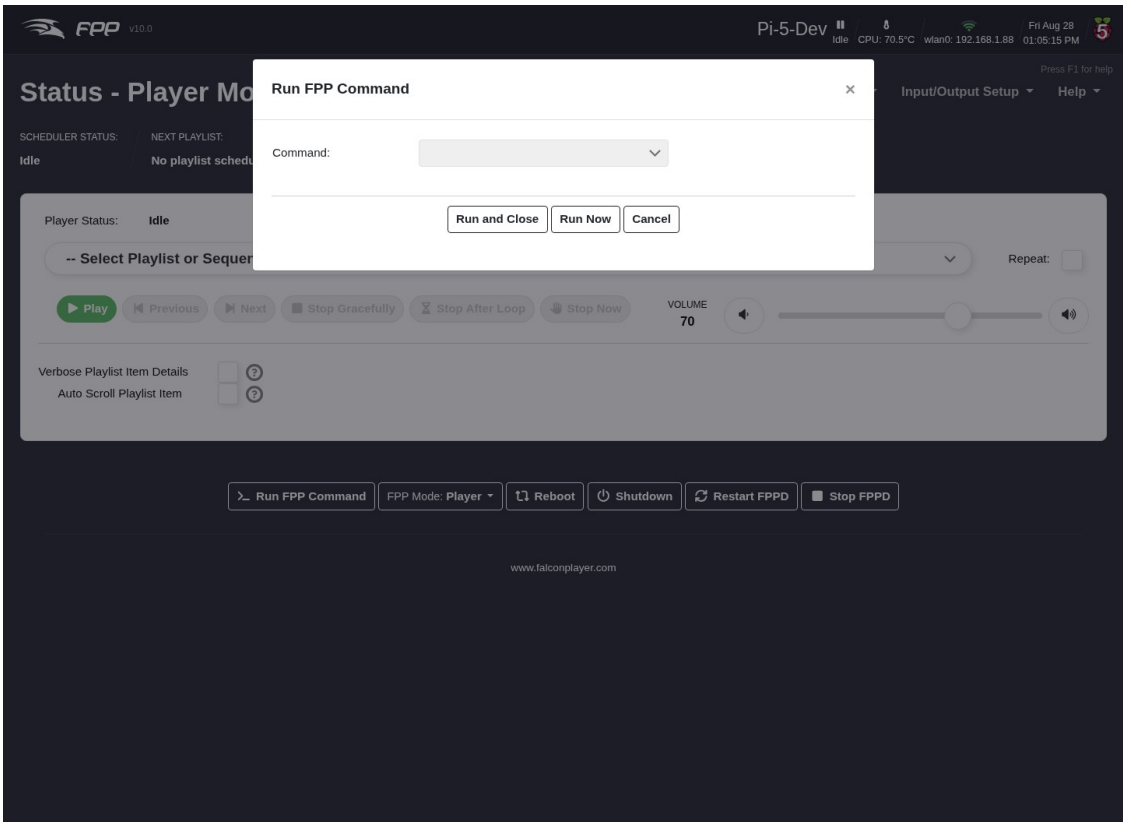


The FPP Command Editor, with the Start Playlist command selected.

Pick the **Command** at the top and the dialog rebuilds itself to show just that command's arguments — in the example above, *Start Playlist* offers **Playlist Name**, **Repeat**, **If Not Running** and **Protected from Schedule Override**. Hover the **?** beside any argument for help on what it expects.

- **Multisync** – also send this command to your remotes rather than running it only here. It appears on commands that can be multisynced.
- **Accept Changes** – keep the command and close the dialog.
- **Run Now** – execute it immediately without leaving the dialog, which is the quickest way to check a command does what you expect before saving it.
- **Cancel Edit** – discard and close.

Tip: The bottom bar's **Run FPP Command** button opens a cut-down version of this dialog on any page, for firing a command by hand.



The Run FPP Command popup.

Choose a **Command** and the same argument fields appear beneath it. **Run Now** runs it and leaves the popup open, **Run and Close** runs it and closes.

Preset Slot – a number from 1–255 that identifies the preset, so it can be triggered by slot (from GPIO, a remote, the API, etc.).

13.7 Example — a push button that starts a playlist

To make a GPIO button start a “Thank You” playlist:

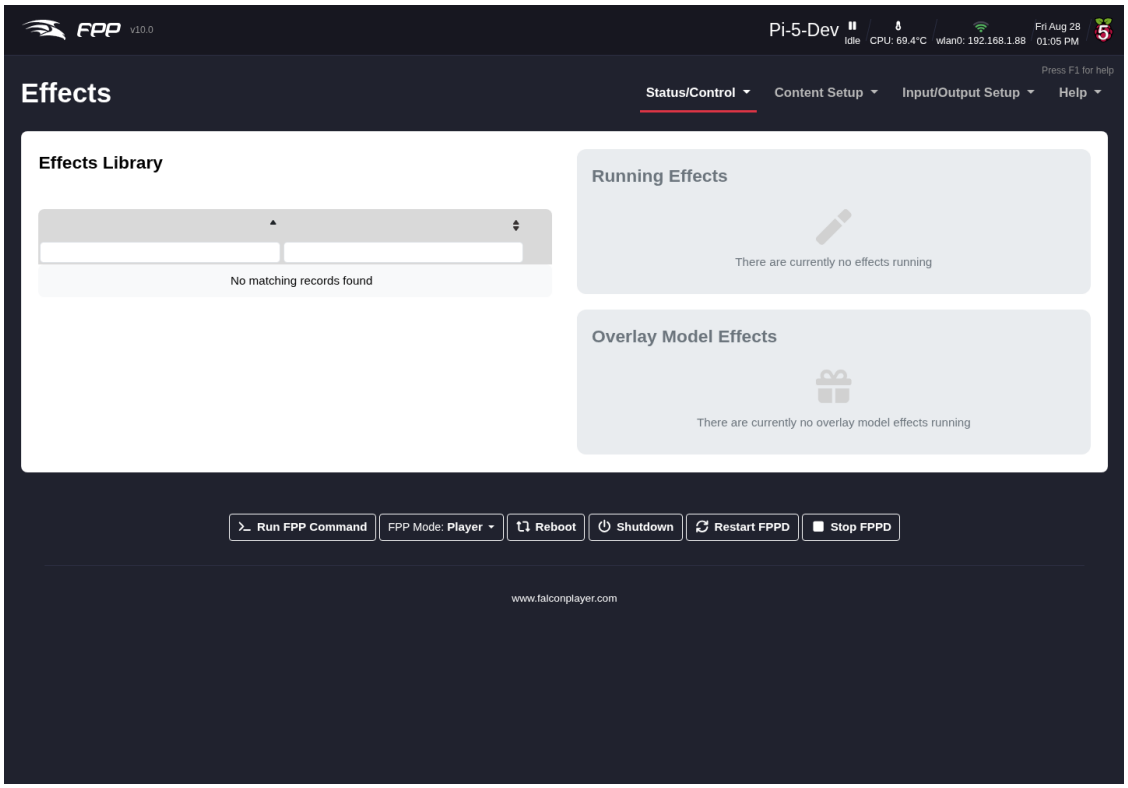
1. On the **Command Presets** page click **+ Add** to add a new preset.
2. Give it a **Preset Name** (e.g. *Start Thank You Playlist*).
3. Choose the **Start Playlist** command; in the FPP Command Editor select the *Thank You* playlist and click **Accept Changes**.
4. Enter a **Preset Slot** (1–255) — say **5**.
5. Go to **Input/Output Setup → GPIO Inputs** and click **+ Add GPIO Trigger**.
6. In the dialog, choose the **GPIO Pin** you are using (check a pinout chart for valid pins), tick **Enabled**, and give it a **Description**.

7. Set **Pull Up / Down** to match your wiring — *Pull Down* for a button wired to 3.3 V, *Pull Up* for one wired to ground, or *None / External Pull* if you fit your own resistor.
8. Under **Rising Edge Commands** (or **Falling Edge Commands**, depending on your wiring) click **Add Command**, choose **Trigger Command Preset slot**, and select slot **5**.
9. Click **Apply**, then **Save** on the page, then **Restart FPPD** when prompted.

Note: Because *Start Playlist* is itself an FPP Command, you could also enter it directly on the GPIO Inputs page and skip creating a preset — the preset is worth it when you reuse the same command in several places.

14 Effects

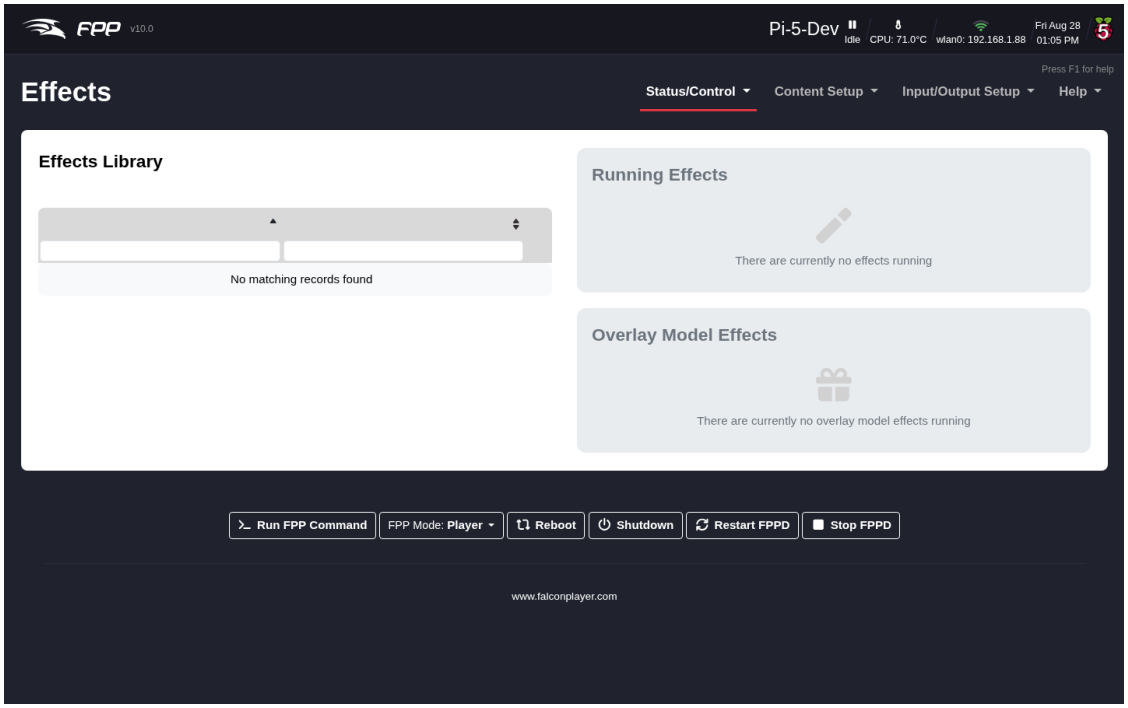
Effects (effect sequences) are used where you want part of your show to run independently of, or in parallel with, the main show sequences. Effect sequences (`.eseq` files) are normally created for a single model in your sequencing software, though you can use a full `.fseq` file as an effect (uncommon). Open **Status/Control → Effects** to start or stop effects manually.



The Effects page.

The **Effects Library** on the left lists every `.eseq` and `.fseq` on the device with a **Start** button; **Running Effects** and **Overlay Model Effects** on the right show what is currently playing, so you can stop them again.

Clicking **Start** opens a small dialog with the options for that effect:



Starting an effect.

- **Loop Effect** – repeat the effect until it is stopped, instead of playing once.
- **Run in Background** – let the effect keep running underneath a playlist, so it resumes when the playlist finishes rather than being cut off.
- **Start Channel Override** – play the effect at a different start channel from the one it was saved with, which is how you point an effect built for one model at another (see Redirecting an effect to another model).

A special use is a **Background Effect**: if you name an effect `background.eseq`, FPP plays it whenever it is not receiving sequence data. (Whether a background effect is paused during FSEQ playback is controlled on FPP Settings → Playback.)

14.1 Creating an effect sequence (in xLights)

1. In xLights, open the model you want the effect for. To use **all** effects on that model, right-click it and choose **Model → Render and Export**. To use only some, select those effects first and choose **Model → Render and Export Selected Model Effects**.

2. Choose **FPP Compressed Sub Sequence** and save it somewhere memorable.
3. Upload the resulting `.eseq` to the FPP device that will play it (see File Manager).

14.2 Triggering an effect from within a sequence

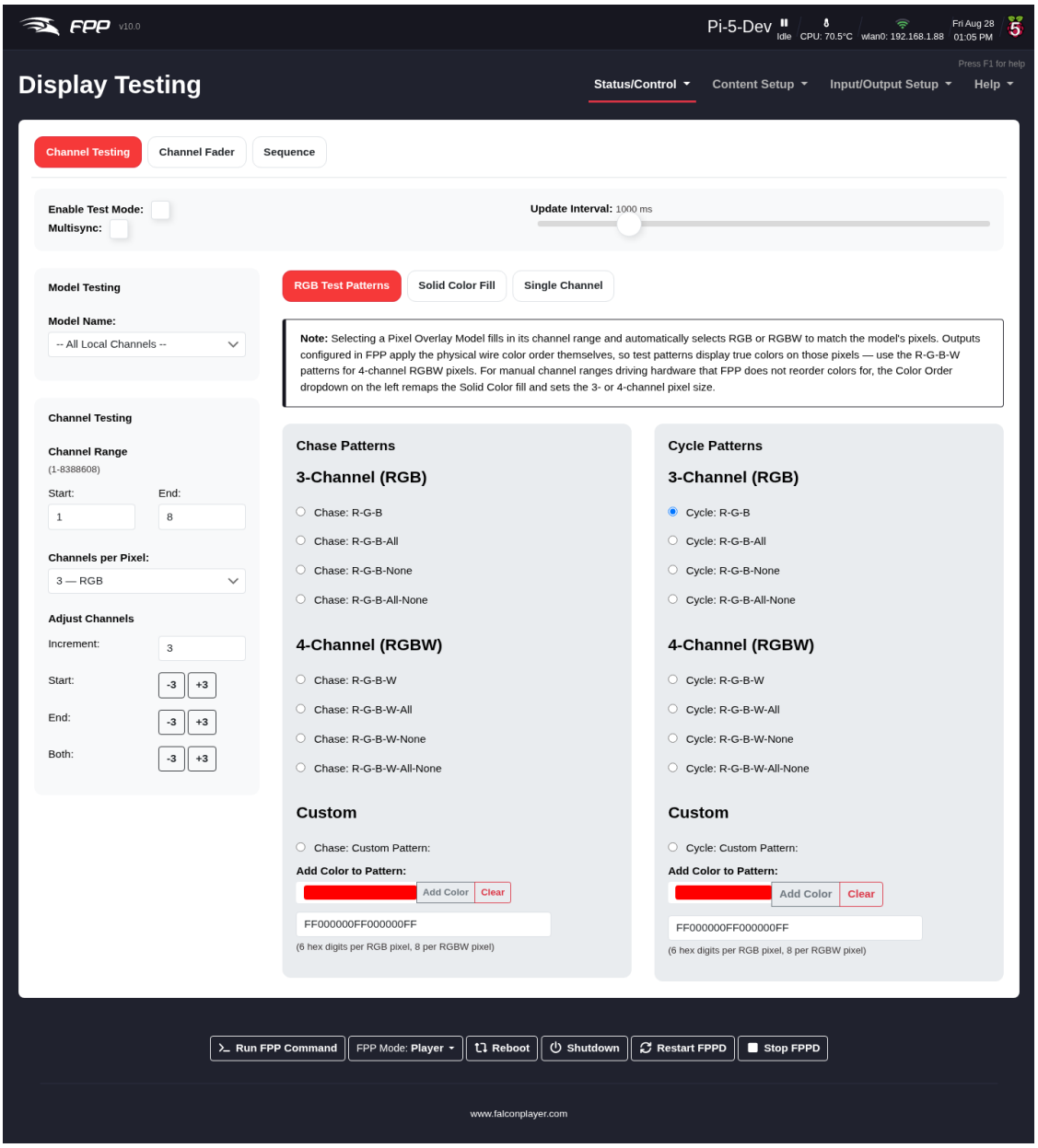
1. In the xLights **Sequencer**, open the sequence.
2. Right-click the **Timing Tracks** area, choose **Add Timing Track**, and select **FPP Effects** (the default name is fine).
3. With only the **FPP Effects** timing track selected, click the waveform where you want to trigger the effect and press **t** to add a timing mark; click slightly to the right and press **t** again to close the slot.
4. Enter the effect name into the slot (Shift+double-click or double-click the slot, depending on your xLights configuration) — using the **exact** name of your effect (e.g. *Thank You*). A yellow box with the effect name appears on the timeline.

14.3 Redirecting an effect to another model

You can play an effect on a different model with the same attributes by supplying a **channel offset** — the absolute start channel of the target model. For example, if an effect was built on a mini-tree starting at channel 1326 and you want it on an identical mini-tree starting at 1842, select the effect from the Effects Library, enter **1842** in the **Start Channel Override**, and click **Play Effect**.

15 Display Testing

The **Display Testing** page tests your channel outputs and lets you quickly test stored sequences without defining playlists — a very useful troubleshooting tool. Open **Status/Control → Display Testing**.



The Display Testing page.

The page has three tabs — **Channel Testing**, **Channel Fader** and **Sequence** — with the test controls common to all of them across the top.

15.1 Test controls

- **Enable Test Mode** – tick to start the test pattern. **Untick it when you are done**, or you will get unexpected results once a show tries to run.
- **Multisync** – also send the test pattern to other FPP devices set to remote mode.
- **Update Interval** – how quickly the pattern changes, in milliseconds.

> **Note:** Test mode takes over the outputs. In FPP 10 FPP tracks which browser or client owns the test, and warns you if another session is about to take it over — so two people testing the same device no longer silently fight over it.

15.2 Channel Testing

Model Testing lets you pick a **Model Name** from your Pixel Overlay models instead of working in raw channels; the channel range is filled in for you.

Channel Testing sets the range by hand:

- **Channel Range** – **Start** and **End** channels. By default FPP loads this device's configured channels, and shows the valid range beneath the heading.
- **Channels per Pixel** – **1 — Single Color**, **3 — RGB** or **4 — RGBW**. This sets the pixel size the patterns and the solid-colour fill work in.
- **Adjust Channels** – shift the range by the **Increment** value using the **−/+** buttons for **Start**, **End** or **Both**. With an increment of 3 (one RGB pixel) this is the quickest way to find the exact start or end of a string.

> **Note:** Selecting a Pixel Overlay Model fills in its channel range *and* automatically selects RGB or RGBW to match the model's pixels. Outputs configured in FPP apply the physical wire colour order themselves, so test patterns show true colours on those pixels — use the **R-G-B-W** patterns for 4-channel RGBW pixels. The manual channel-range path is for hardware FPP does not reorder colours for.

15.2.1 RGB Test Patterns

Test patterns light the string in colours you choose. They come in two families, each offering **3-Channel (RGB)**, **4-Channel (RGBW)** and **Custom** variants.

> **Note:** RGB **Chase** patterns do not take output settings into account, so colours may not display as true Red/Green/Blue — use a **Cycle** pattern instead in that case.

Chase Patterns light a repeating colour pattern that then shifts along the string:

- 3-Channel: **Chase R-G-B**, **R-G-B-All**, **R-G-B-None**, **R-G-B-All-None**.
- 4-Channel: **Chase R-G-B-W**, **R-G-B-W-All**, **R-G-B-W-None**, **R-G-B-W-All-None** — the same idea with the white element included.

Cycle Patterns light the whole string one colour, then cycle to the next, with the same 3- and 4-channel variants (**Cycle R-G-B** … **Cycle R-G-B-W-All-None**).

Custom – **Chase: Custom Pattern** and **Cycle: Custom Pattern** take a list of colours in hex: **6 hex digits per RGB pixel, 8 per RGBW pixel**. Use the colour box and **Add Color** to append the selected colour to the pattern, or **Clear** to start again.

15.2.2 Solid Color Fill

Illuminate the whole range with one colour, set on the sliders or with the colour picker.

15.2.3 Single Channel

Tests a prop by channel value, where the **Channel Data Value** is the intensity. The fill option sends the test value to all configured channels. Selecting a **Chase Size** sends a packet of that size with the first channel at the test value and the rest at 0, then repeats.

15.3 Channel Fader

The **Channel Fader** tab gives you a grid of sliders — one per channel, each 0–255 — so you can drive individual channels by hand. It is the tool to reach for when troubleshooting moving heads, dumb RGB fixtures, fog machines, pixels and other devices where you need to work out what a particular channel actually does.

Channel numbers are absolute FPP channel numbers, counting from the configured **Start Channel**. Fixtures you have defined under *Input/Output Setup → Pixel Overlay Models* are highlighted with a coloured bar and labelled with the fixture name and the channel's position within it (for example *Ch 3 / 16*), so you can tell at a glance which function a given absolute channel maps to.

> **Note:** FPPD runs one test at a time, so a test left running on the **Channel Fader** tab and one on the **Channel Testing** tab will interrupt each other — FPP warns you when this is about to happen. Very large channel counts (matrices and similar props) are skipped rather than drawn as thousands of sliders.

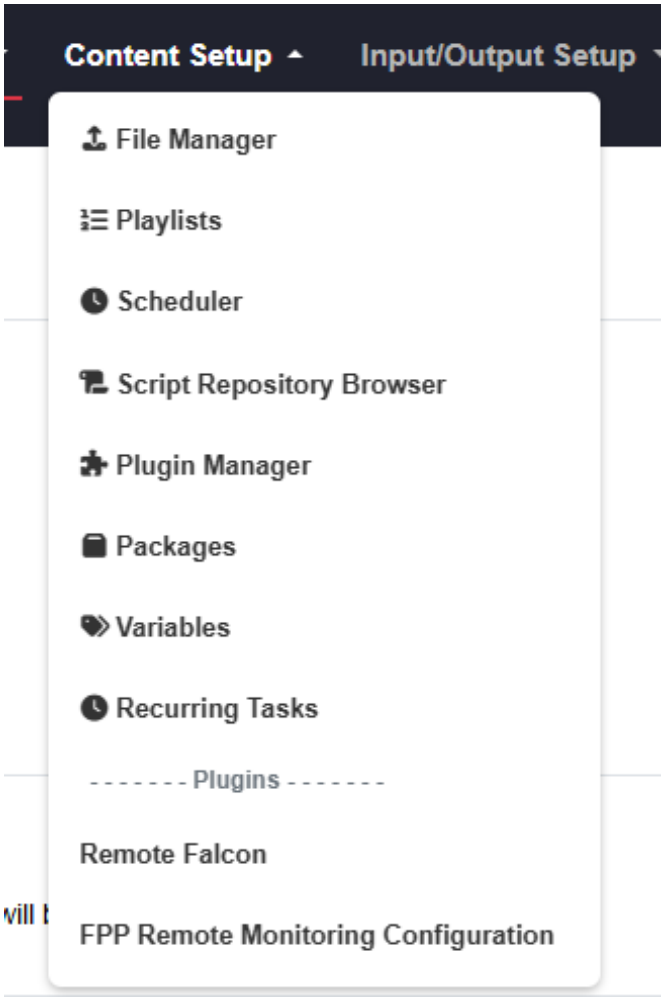
15.4 Sequence

The **Sequence** tab tests a stored sequence.

> **Note:** Only the sequence **data** is output on the local system — audio and video are not played. Network and channel configuration must be defined before testing, and this is only available in **Player** mode.

16 Content Setup Section

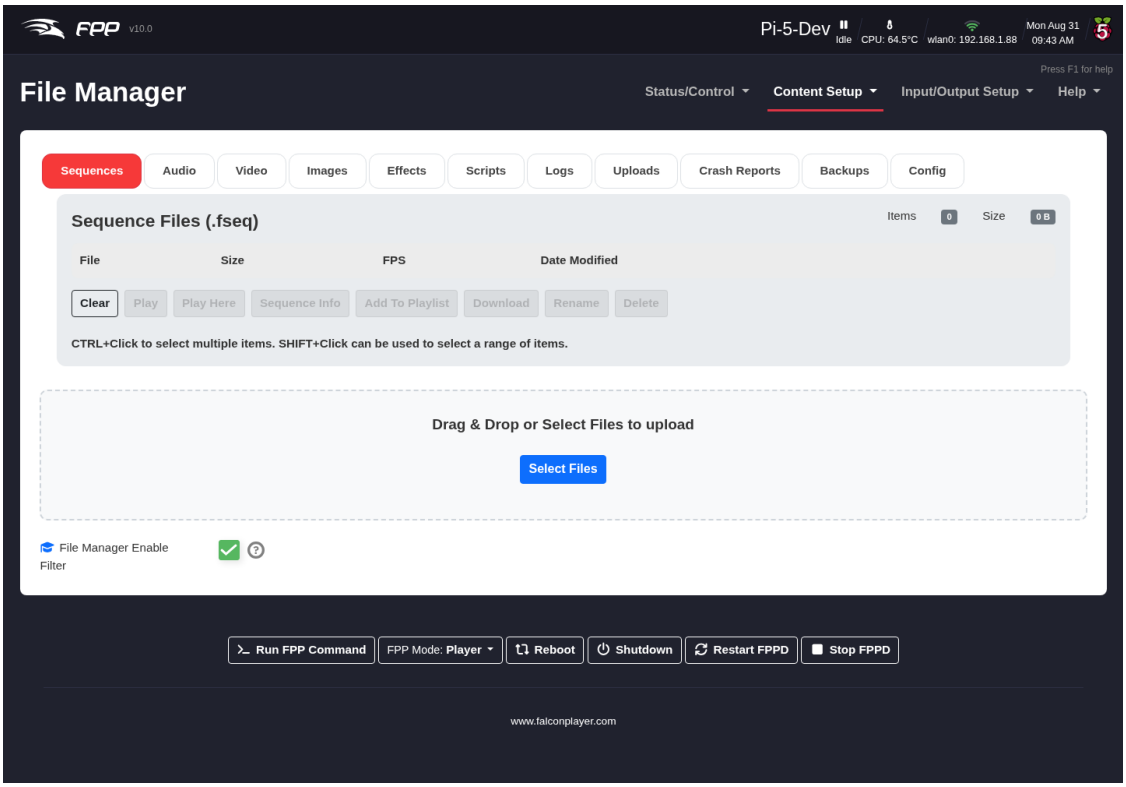
The **Content Setup** menu manages everything your show plays: uploaded media and sequences, playlists, the scheduler, scripts, plugins and optional packages, and the variables/recurring-tasks automation added in FPP 10. Its first entry is the **File Manager**, where sequences, audio, video and other show files are uploaded and organised.



Content Setup Navigation

17 File Manager

The **File Manager** is where you manage the personalized files on your FPP — uploading, downloading and, in some cases, editing them. Open **Content Setup → File Manager**.



The File Manager.

> **Note:** The available options depend on the FPP mode (Player or Remote).

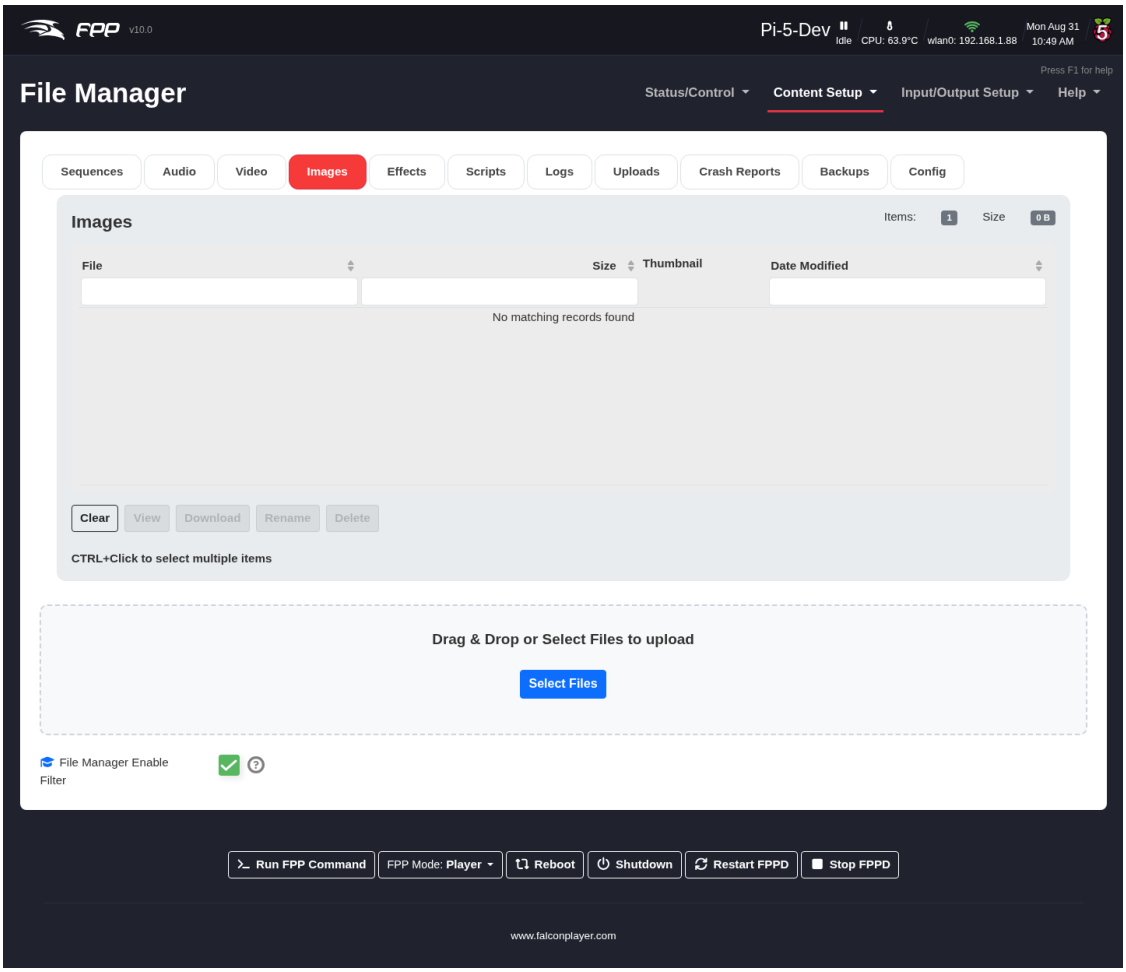
The upper-right of the screen shows the number of items of the file type you are viewing and total size of those files. You can sort the columns by clicking on the up or down arrows above the column or you can filter the list entering the search criteria in the boxes above the columns.

17.1 File categories

Files are grouped into tabs. Most tabs share **Clear**, **Download**, **Rename** and **Delete**; the type-specific options are noted below. You can multi-select files using shift-click or cmd/ctrl-click. **Clear** will clear your selection.

- **Sequences** – your `.fseq` files. Additional options include:
  - **Play / Play Here** (Player mode only) both will start the sequence.
  - **Sequence info** (version, compression, frame rate, etc,).
  - **Add To Playlist** appends to the selected playlist without checking for duplicates.
- **Audio** – audio files. Additional options include:
  - **Listen** plays the file on your computer.
  - **MP3Gain** normalizes the levels of the selected files

    - **Add To Playlist** appends to the selected playlist without checking for duplicates.
- **Video** – video files. Additional options include:
    - **View** plays the file on your computer.
    - **Video info** shows codec, aspect ratio, duration, etc.
    - **Add To Playlist** appends to the selected playlist without checking for duplicates.
- **Images** – image files. This will also display a thumbnail of the image and animated gifs will be animated. Additional options include:
    - **View** this will allow you to see the image with an option to view full size.



The File Manager-Images.

- **Effects** – effect sequences (`.eseq`), Additional options include:
    - **Sequence info** (version, compression, frame rate, etc,).
- **Scripts** – shell scripts, Additional options include:
    - **View** allows you to view the script text
    - **Run** useful for testing

  - **Edit** edit the code in the browser
  - **Copy** duplicate the script under a new name
  - **Add To Playlist**.

Note: Scripts will be deprecated in v11

- **Logs** – system logs for troubleshooting. Additional options include:
  - **Zip** bundles all logs and downloads them.
  - **View** shows the log.
  - **Tail** shows just the last 50 lines.
  - **Tail Follow** shows the last 50 lines and a live update as entries are added.
- **Uploads** – files that do not fit the standard formats, including `.fppos` upgrade files; Additional options include:
  - **Copy** duplicate the file under a new name
- **Crash Reports** – crash reports FPP has generated. FPP keeps these on the device even when you have chosen not to send crash data to the developers (see FPP Settings → Privacy), so you can inspect one yourself or attach it to a support request.
- **Backups** – any manual backups you have created (see Backup and Restore).
- **Config** – FPP's own configuration files. Additional options include:
  - **View** shows the file.
  - **Edit** opens it in an in-browser editor to change and save directly. This is the same content the JSON configuration backup captures.

Warning: Editing configuration files by hand bypasses every check the normal settings pages make, and a malformed file can cause catastrophic results. Take an FPP Backup first.

17.2 Uploading files

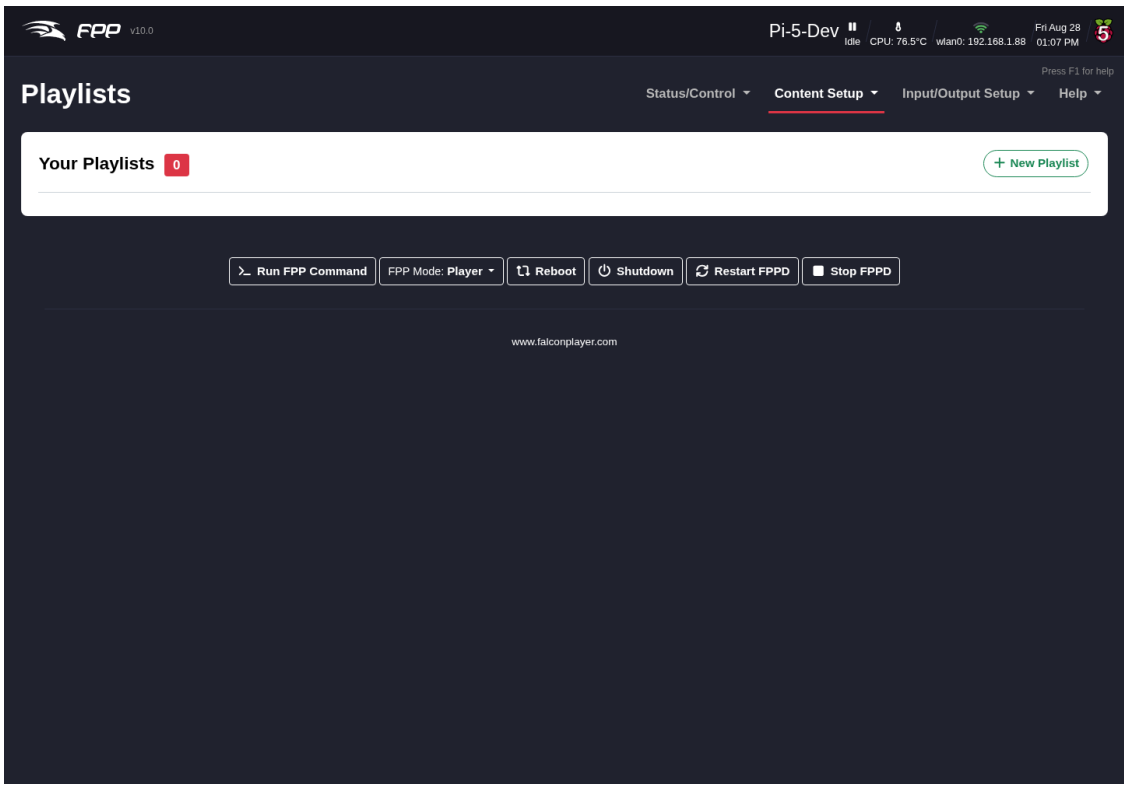
Upload files by dragging them from your computer's file manager onto the **Drag & Drop or Select Files to upload** area. FPP shows each file on the tab appropriate to its type; anything that does not match a standard type appears on the **Uploads** tab.

Tip: xLights **FPP Connect** is a more efficient method to upload sequences since is supports the sparse format. and media directly, so for everyday sequencing you often will not need the File Manager by hand.

Tip: Watch free space (shown in the header warnings and on *FPP Settings → Storage*). Large video files and many sequences can fill smaller SD cards.

18 Playlists

A playlist can be far more than a list of songs — it is one of FPP's most versatile features. A **playlist** is an organised group of sequences, commands, scripts, videos and more, played in a particular order, and it is where you combine everything to create your light show. You can keep several playlists for different time frames or days, and you can even nest a playlist within a playlist. Open **Content Setup → Playlists**.



The Playlists page.

Hover over a playlist to **edit** or **delete** it; click it to open the editor. Click **New Playlist** to create one, entering a name and an optional description.

> **Note:** A warning icon on a card indicates the playlist references media or sequences that are missing from this device.

18.1 Playlist options

Once a playlist is open, hover over its name for an **Edit** button, and use the options at the top right:

- **Settings** – edit saved settings such as the description, **Verbose** details, and **Randomize**:
  - **Off** – play in the configured order.
  - **Once Per Load** – randomise each time the playlist begins.

    - **Every Iteration** – randomise after each song (with logic to prevent the same song playing back-to-back).
- **Playlist Actions** – **Copy Playlist**, **Rename Playlist**, **Randomize Playlist** (reorder now), **Reset Playlist** (revert to the saved version), and **Delete Playlist**.

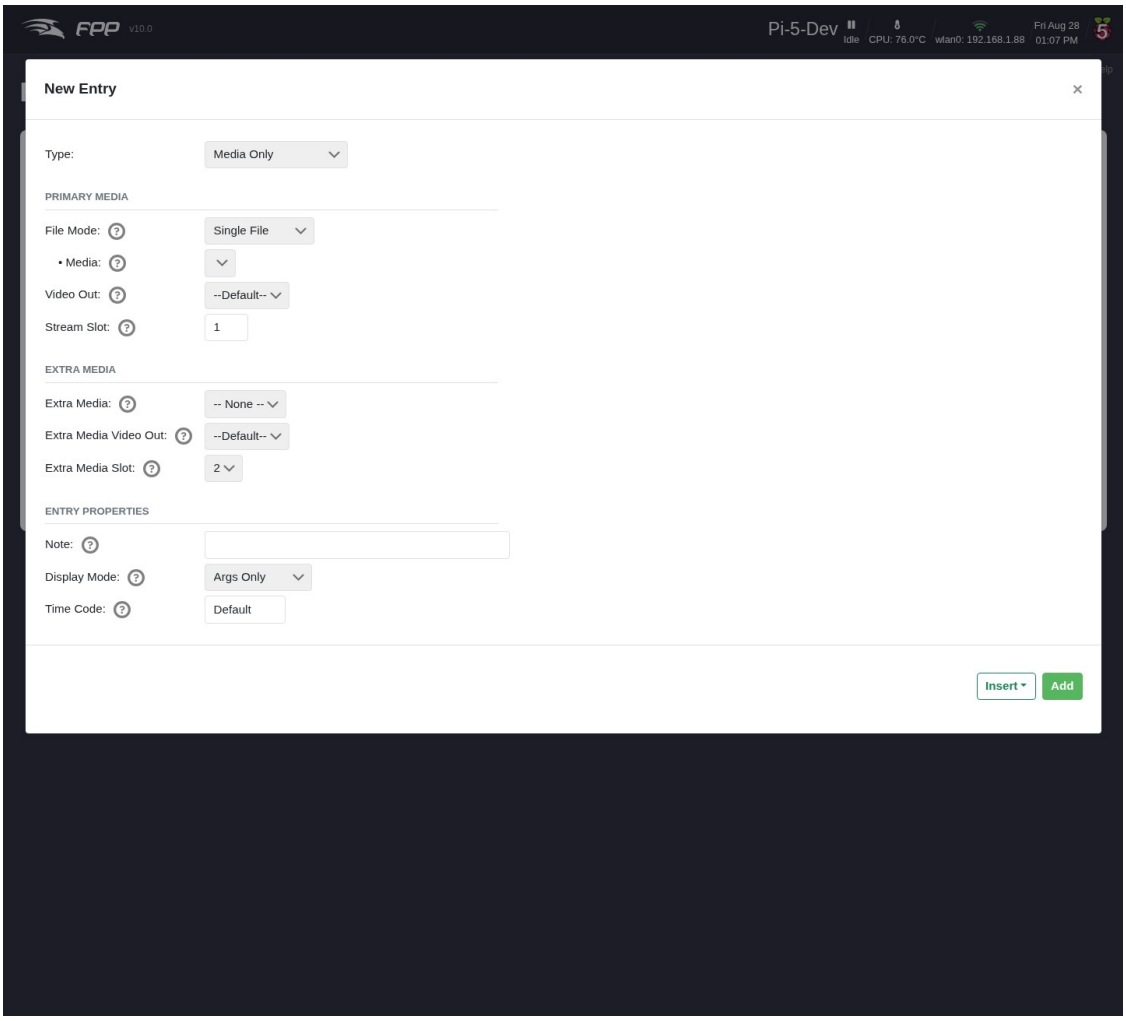
18.2 The three sections

A playlist has three sections, each showing its item count and total duration:

- **Lead In** – plays once when the playlist starts (e.g. to activate items before the show, or a one-time announcement). It does **not** repeat even when Repeat is selected.
- **Main Playlist** – your main items; this section repeats when Repeat is selected in the Scheduler or on the Status page.
- **Lead Out** – plays once when the playlist ends (e.g. to switch things off, or a closing announcement).

18.3 Adding entries

Adding to any section is the same: click **Add a Sequence/Entry**, which opens the entry dialog. FPP 10 redesigned it: instead of one long list of fields, the options are grouped into labelled **sections**, and individual fields carry help tooltips — hover the **?** beside a field to read it.



Adding a playlist entry.

Two options at the top make building a playlist quicker:

- **Auto-Select Matching Media/Sequence** – when you pick a sequence, FPP selects the media file with a matching name for you (and vice versa).
- **Hide sequences already in playlist** – filter out anything already added, so you can see what is left.

The sections shown depend on the **Type** you choose. For a *Sequence and Media* entry they are **Sequence**, **Primary Media**, **Extra Media** and **Entry Properties**:

- **Sequence** – the `.fseq` to play.
- **Primary Media** – the **Media** file, the **Video Out** it should use, and its **Stream Slot**.
- **Extra Media** – companion media that plays alongside the primary (see below).

- **Entry Properties** – a free-text **Note**, the **Display Mode** used when showing the entry in lists, and an optional **Time Code**.

Add appends the entry to the section; the **Insert** drop-down beside it places it at a chosen position instead.

The available types are:

- **Sequence and Media** – the most common entry: an `.fseq` plus its associated audio and/or video (with a **Video Out** option).
- **Branch** – change the playlist while it runs, branching to another position based on test conditions (e.g. lowering the volume at a certain time of day). Several test conditions and settings are available.
- **Dynamic** – items created on the fly by an outside script, plugin or process.
- **FPP Command** – run any FPP Command as a playlist item (see Command Presets).
- **Image** – display images through the HDMI port (a virtual "picture frame"). Enter `/home/fpp/media/images` to use all images (played in random order), or an individual file name; choose a transition, and add a **Pause** entry for how long each image shows.
- **Media Only** – play media with no lights controlled (e.g. pictures on a matrix or TV).
- **Pause** – wait for a set time.
- **Playlist** – embed a playlist within a playlist (e.g. a shared Lead In playlist reused by several daily playlists).
- **Remap** – remap channels to another range, handy if you move a prop to a different port and cannot rebuild the sequence.
- **Script** – run a script from the File Manager's *Scripts* tab (see *Plugins, Packages and Scripts*).
- **Sequence Only** – sequence data with no media (e.g. an animation).
- **URL** – send URL commands to outside programs (e.g. switch a smart power strip, or post the current song to a website).

Drag items to reorder them, including between sections. Hover over an item for **Edit** and **delete** options. Save the playlist when done; it is then available on the Status page and to the Scheduler.

18.4 Companion media

New in FPP 10, a media entry can carry **companion media** — extra audio or video that starts and stops with it, on its own stream slot. You configure it in the **Extra Media** section of the entry dialog:

- **Extra Media** – the additional media file to play, or – *None* –.

- **Extra Media Video Out** – which video output it should use, so a second video can drive a second display.
- **Extra Media Slot** – the stream slot it plays on (the primary media uses slot 1, so a companion defaults to 2).

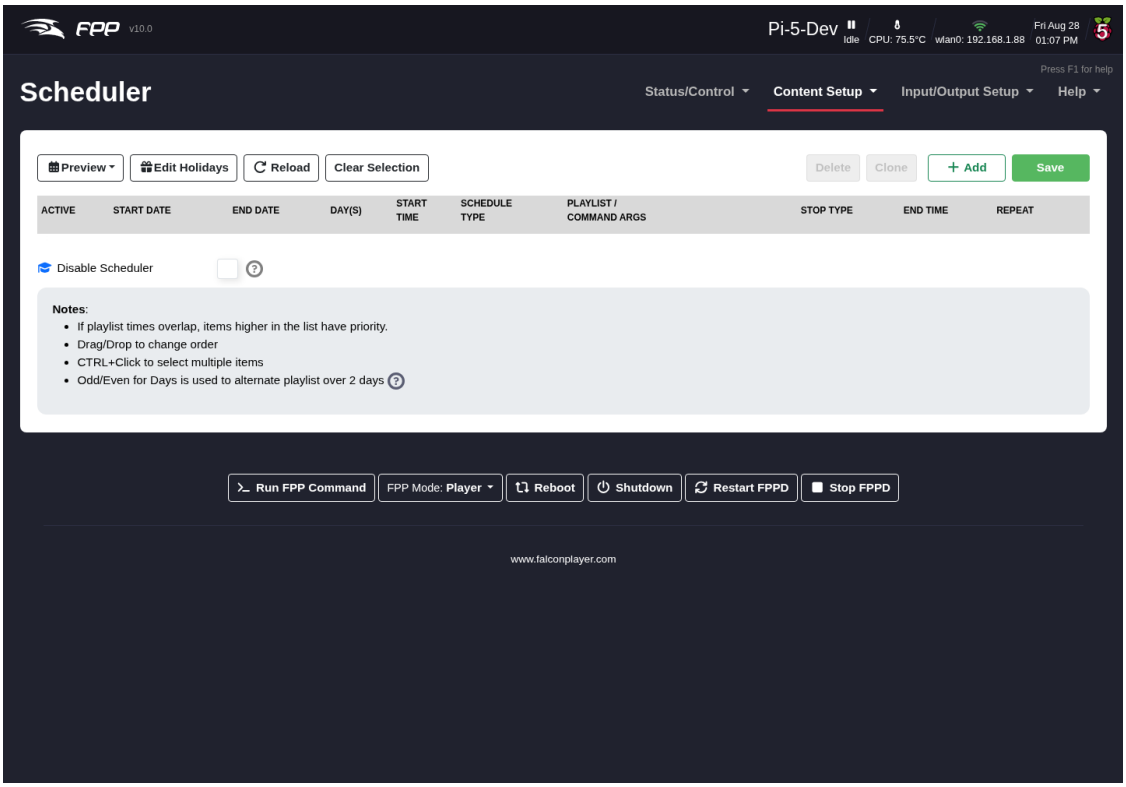
A playlist item carrying one shows an extra-media badge in the entry list.

The usual reasons to want this are a second audio feed — an alternate-language soundtrack, or a feed for a lobby or indoor speaker — or a second display running alongside the main show.

Before FPP 10 you could approximate this by firing a *Play Media* command from a start-of-entry hook and a matching *Stop Media Slot* from a stop hook — but anything that ended the entry another way (the media finishing by itself, **Stop Now**, a playlist change, or a pause) left the companion still playing. Declaring the companion on the entry gives it exactly the entry's lifetime instead: it stops when the entry stops however that happens, and it is restarted at the right offset when the entry is paused and resumed.

19 Scheduler

The **Scheduler** runs predefined playlists (or FPP Commands, or single sequences) automatically on a preset schedule — the heart of an unattended show. Open **Content Setup → Scheduler**.



The Scheduler page.

> **Note:** The scheduler is normally used only on a device in **Player** mode, not on remotes (which take their instructions from a player) — though you can schedule *FPP Commands only* on a remote for special cases.

You can have multiple playlists for different days, holidays, or even different times of the same day. The scheduler supports **priority scheduling**: an entry higher in the list has a higher priority and preempts lower ones.

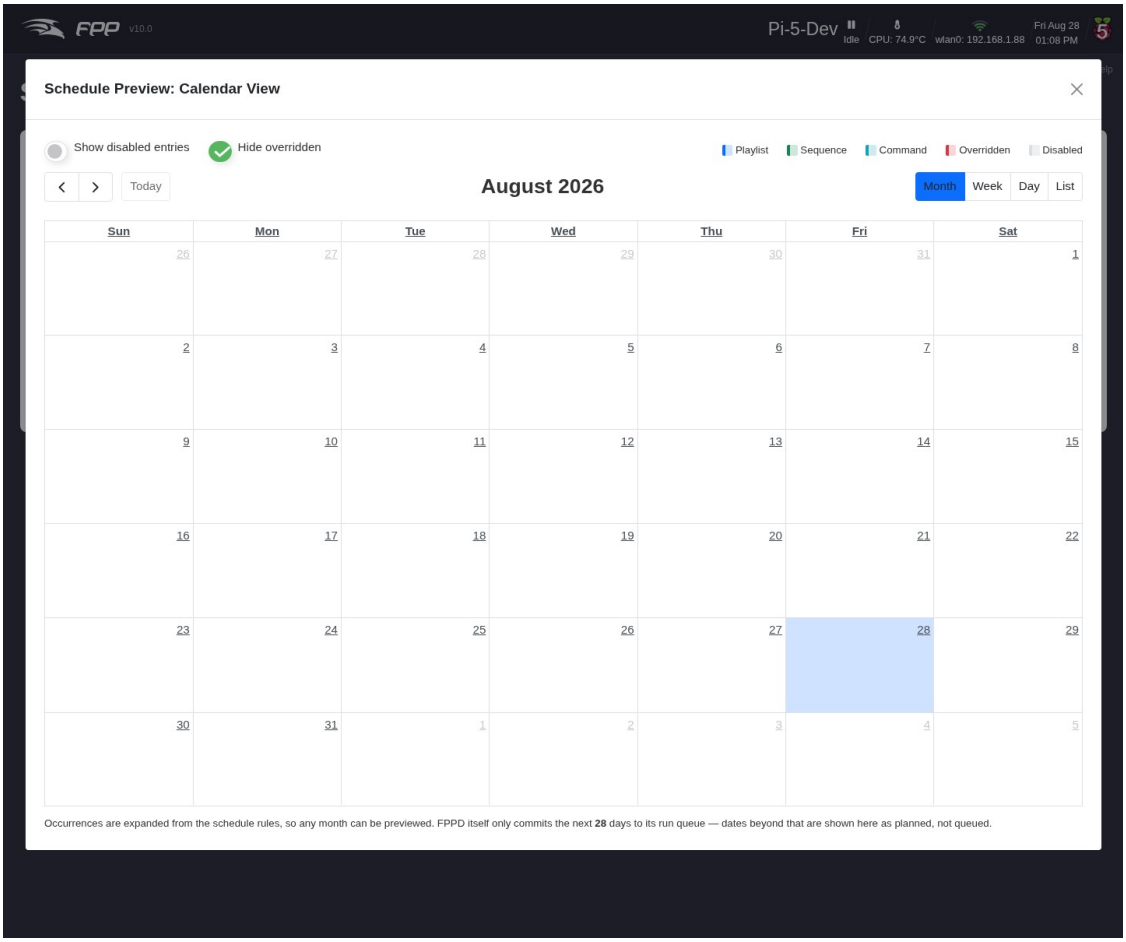
> **Important:** For schedules to start at the right time, your time settings must be correct — including Latitude/Longitude for sun-based times (see FPP Settings → Localization).

19.1 Understanding the schedule (Preview)

Preview shows a graphical view of the saved schedule for the week, which helps identify problems. Each entry is colour-coded by start/stop time and stop mode. If the colours are not in pairs it *may* indicate a problem — but not always, if you are using priorities. Items skipped due to a priority conflict are marked in red; if a higher-priority schedule ends while a lower-priority one should still be playing, the lower-priority schedule starts when the higher one finishes. Hover over a question mark for a summary of that entry.

19.2 Calendar View

New in FPP 10, **Calendar View** shows the schedule as a familiar calendar rather than a week strip — useful for checking a whole season at a glance, or confirming that a holiday entry lands on the right date. Open it from the **Calendar View** button on the Scheduler page, or from the **Preview** drop-down on the Status page.



The Schedule Preview calendar.

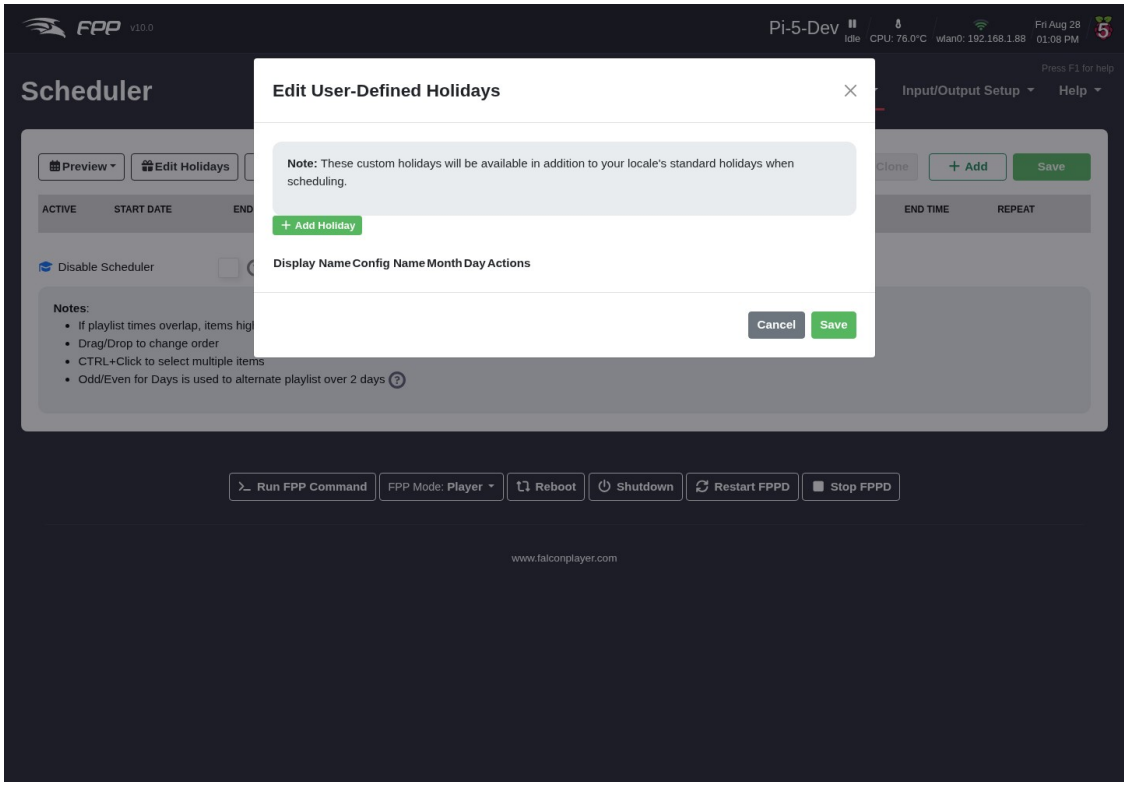
- Switch between **Month**, **Week**, **Day** and **List** views with the buttons at the top right, and move around with the ‹ / › arrows and **Today**.
- Entries are colour-coded by type — **Playlist**, **Sequence**, **Command** — plus **Overridden** (an entry a higher-priority one preempts) and **Disabled**.
- **Show disabled entries** adds inactive entries to the view; **Hide overridden** (on by default) removes the ones that will not actually run, so you see what will really happen.

Note: The calendar expands occurrences from your schedule *rules*, so you can preview any month, however far ahead. FPPD itself only

commits the next **28 days** to its run queue — dates beyond that are shown as planned, not queued.

19.3 Page controls

- **Reload** – reload the saved schedule (discarding unsaved changes).
- **Clear Selection** – deselect any selected rows.
- **Delete** – delete selected rows (not final until you **Save**).
- **Clone** – copy selected rows to the bottom of the list.
- **Edit Holidays** – opens the **Edit User-Defined Holidays** dialog, where you define your own holidays in addition to the standard ones that come from your configured **Locale**. They can then be picked by name in an entry's Start/End Date.



Editing user-defined holidays.

+ Add Holiday adds a row, where you give the holiday a **Display Name** (what you see in the date lists), a **Config Name** (the identifier FPP stores), and the **Month** and **Day** it falls on. **Save** stores them.

- **Save** – must be clicked to store any additions or changes.
- **Add** – create a new entry, then fill in its columns.

19.4 Schedule entry fields

- **Active** – enable or disable the entry (e.g. prepare a future entry without running it yet).
- **Start Date** – when the playlist begins playing; can be far in the past (it plays at the next matching time/day). With a **Locale** configured you can pick **holidays by name**.
- **End Date** – the last day it plays (it still starts on this date); can be far in the future. Holidays by name are supported here too.
- **Day(s)** – any combination of days; common combinations are in the drop-down, or choose **Day Mask** and tick specific days.
- **Start Time** – when to begin, in your configured time format (defaults to 24-hour). Half-hour times are suggested, but you can enter any time. You can also choose **Dawn**, **Sunrise**, **Sunset** or **Dusk** to follow local daylight (requires Latitude/Longitude).
- **Schedule Type** – **Playlist**, a single **Sequence**, or an **FPP Command**.
- **Playlist/Command Args** – the playlist or command to run. (An FPP Command has no end time unless set to repeat — it simply triggers at the start time.)
- **End Time** – when to end; a playing sequence ends according to the **Stop Type**. Sun-based end times are supported. If the schedule runs past midnight, set the end time you want and FPP will understand it is on the next day.
- **Repeat** – three types:
  - **None** – play once.
  - **Immediate** – restart the Main playlist from the beginning each time it finishes, repeating until the end date/time. Useful when a previous playlist might still be finishing (e.g. a Graceful Stop) when this one is due to start.
  - **Timed** – run once every *X* minutes (each iteration includes Lead In, Main and Lead Out). The preview shows multiple instances from the start time, every X minutes, up to the end time.

19.5 Stop Type

Each entry's **Stop Type** controls how a running item ends at the end time — **Graceful** (finish the current item), **Graceful after loop**, or **Hard Stop** (stop immediately).

> **Tip:** A common pattern is a nightly entry from *Sunset* to a fixed time with a **Graceful** stop (so songs are not cut off), plus a **Command** entry a few minutes earlier to switch on power or run a pre-show.

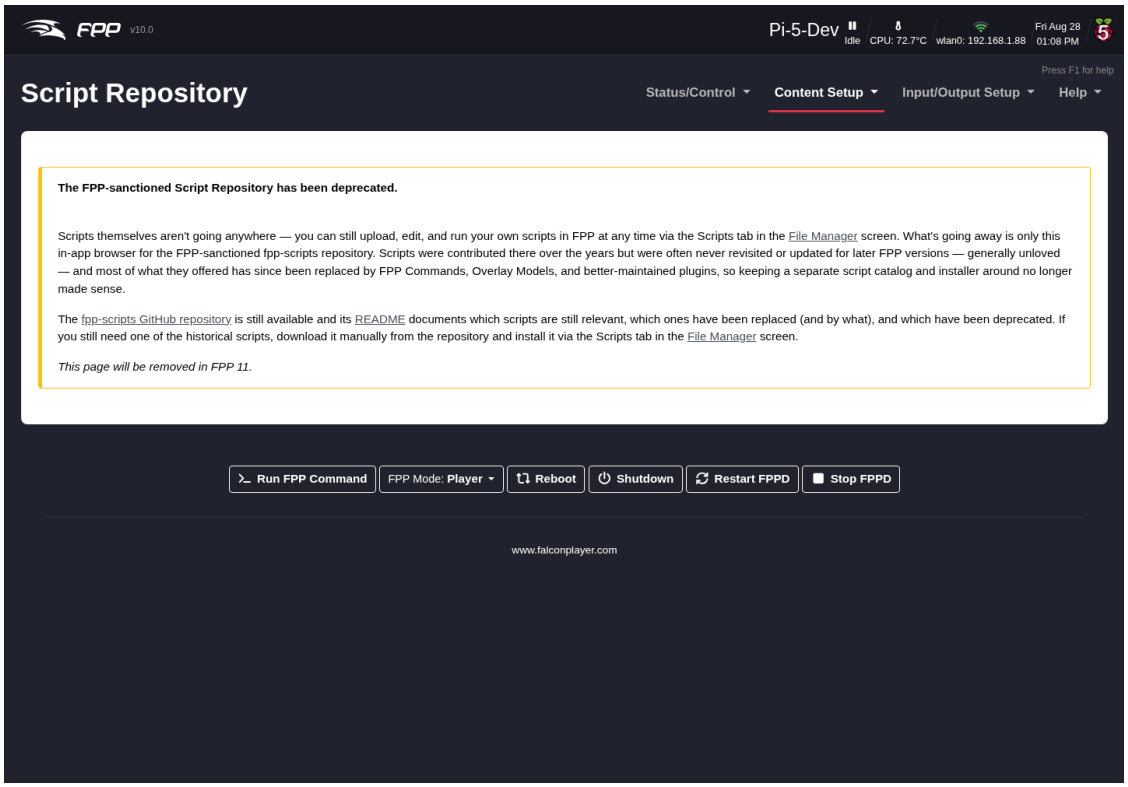
20 Scripts

Scripts are small programs that perform specific functions. They can be used within a playlist, from a Command Preset, the Scheduler or a GPIO input, or as part of a plugin.

You write, upload, edit and run your own scripts from the **Scripts** tab of the **File Manager** (Content Setup → File Manager). That is unaffected by the change described below.

20.1 The Script Repository is deprecated

Content Setup → Script Repository Browser used to list community scripts by category and install them for you. In FPP 10 that in-app browser has been **deprecated**, and the page now shows only a notice explaining the change.



The Script Repository page in FPP 10.

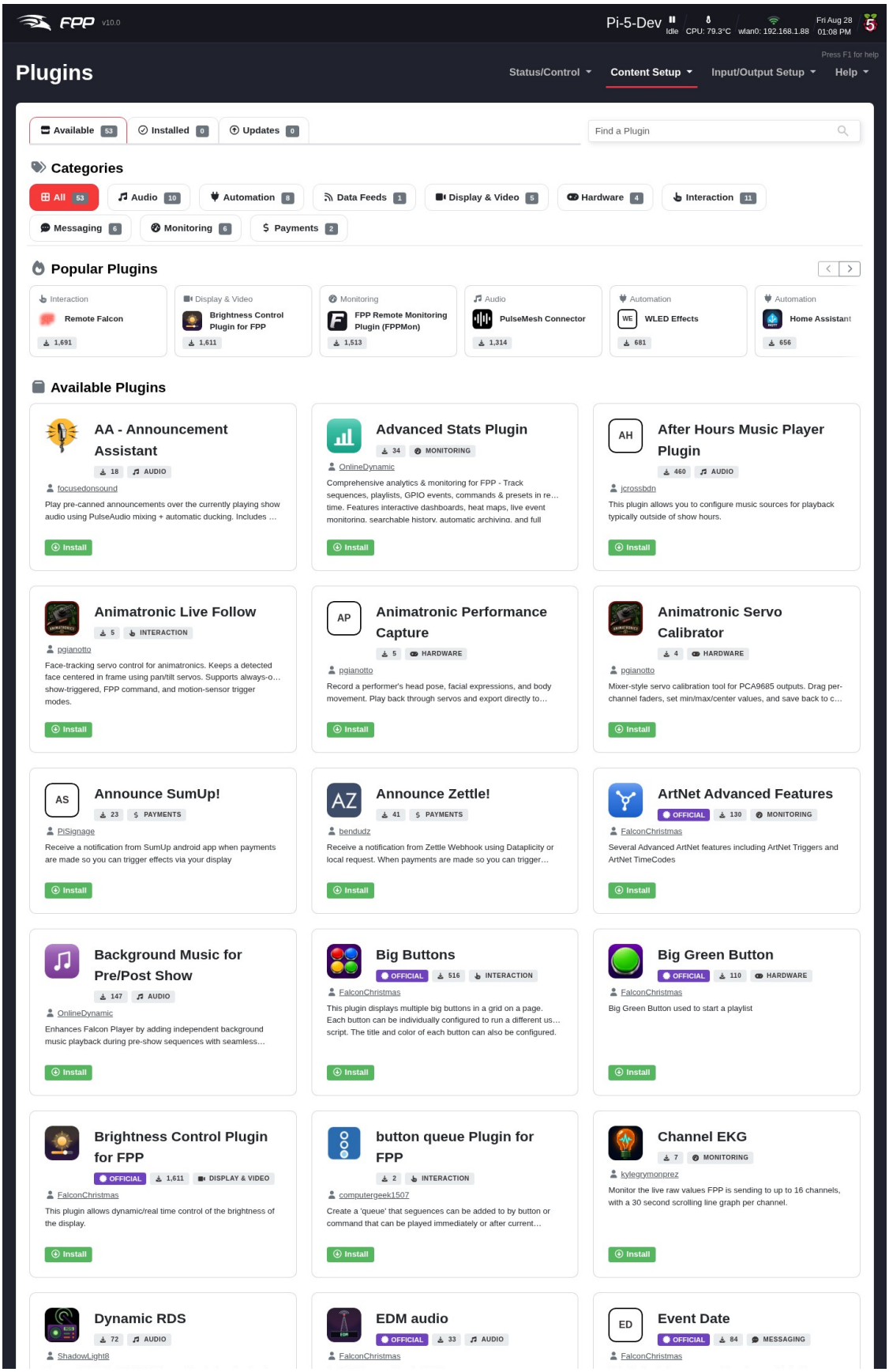
Scripts contributed to the FPP-sanctioned `fpp-scripts` repository over the years were often never revisited or updated for later FPP versions, and most of what they did has since been replaced by **FPP Commands**, **Pixel Overlay Models** and better-maintained **plugins** — so keeping a separate catalogue and installer around no longer made sense.

The fpp-scripts GitHub repository is still available, and its **README** documents which scripts are still relevant, which have been replaced (and by what), and which are deprecated. If you still need one of the historical scripts, download it from the repository by hand and install it via the **Scripts** tab in the File Manager.

> **Note:** The Script Repository page **will be removed in FPP 11**. If any of your workflow depends on installing scripts from it, move to the File Manager route now.
>
> **Note:** Scripts run on the device with full access to it, so only install scripts you trust — the same caution that applies to plugins.

21 Plugins

Plugins are additional components — developed by the FPP developers or by others — that add functionality within FPP, for more complex operations than a script. You can install a plugin from the list, install a third-party plugin, or develop your own. Open **Content Setup → Plugin Manager**.


FPP v10.0
Pi-5-Dev
Idle
CPU: 79.3°C
wlan0: 192.168.1.88
Fri Aug 28
01:08 PM
Press F1 for help
Plugins
Status/Control
Content Setup
Input/Output Setup
Help
Available 53
Installed 0
Updates 0
Find a Plugin
Categories
All 53
Audio 10
Automation 8
Data Feeds 1
Display & Video 5
Hardware 4
Interaction 11
Messaging 6
Monitoring 6
Payments 2
Popular Plugins
Interaction
Remote Falcon
1,691
Display & Video
Brightness Control Plugin for FPP
1,611
Monitoring
FPP Remote Monitoring Plugin (FPPMon)
1,513
Audio
PulseMesh Connector
1,314
Automation
WLED Effects
681
Automation
Home Assistant
656
Available Plugins
AA - Announcement Assistant
18 AUDIO
focusedonsound
Play pre-canned announcements over the currently playing show audio using PulseAudio mixing + automatic ducking. Includes …
Install
Advanced Stats Plugin
34 MONITORING
OnlineDynamic
Comprehensive analytics & monitoring for FPP - Track sequences, playlists, GPIO events, commands & presets in re… time. Features interactive dashboards, heat maps, live event monitoring, searchable history, automatic archiving, and full
Install
After Hours Music Player Plugin
460 AUDIO
jcrossbdn
This plugin allows you to configure music sources for playback typically outside of show hours.
Install
Animatronic Live Follow
5 INTERACTION
pgianotto
Face-tracking servo control for animatronics. Keeps a detected face centered in frame using pan/tilt servos. Supports always-o… show-triggered, FPP command, and motion-sensor trigger modes.
Install
Animatronic Performance Capture
5 HARDWARE
pgianotto
Record a performer's head pose, facial expressions, and body movement. Play back through servos and export directly to…
Install
Animatronic Servo Calibrator
4 HARDWARE
pgianotto
Mixer-style servo calibration tool for PCA9685 outputs. Drag per-channel faders, set min/max/center values, and save back to c…
Install
Announce SumUp!
23 PAYMENTS
PiSignage
Receive a notification from SumUp android app when payments are made so you can trigger effects via your display
Install
Announce Zettle!
41 PAYMENTS
bendudz
Receive a notification from Zettle Webhook using Dataplicity or local request. When payments are made so you can trigger…
Install
ArtNet Advanced Features
OFFICIAL 130 MONITORING
FalconChristmas
Several Advanced ArtNet features including ArtNet Triggers and ArtNet TimeCodes
Install
Background Music for Pre/Post Show
147 AUDIO
OnlineDynamic
Enhances Falcon Player by adding independent background music playback during pre-show sequences with seamless…
Install
Big Buttons
OFFICIAL 516 INTERACTION
FalconChristmas
This plugin displays multiple big buttons in a grid on a page. Each button can be individually configured to run a different us… script. The title and color of each button can also be configured.
Install
Big Green Button
OFFICIAL 110 HARDWARE
FalconChristmas
Big Green Button used to start a playlist
Install
Brightness Control Plugin for FPP
OFFICIAL 1,611 DISPLAY & VIDEO
FalconChristmas
This plugin allows dynamic/real time control of the brightness of the display.
Install
button queue Plugin for FPP
2 INTERACTION
computergeek1507
Create a 'queue' that sequences can be added to by button or command that can be played immediately or after current…
Install
Channel EKG
7 MONITORING
kylegrymonprez
Monitor the live raw values FPP is sending to up to 16 channels, with a 30 second scrolling line graph per channel.
Install
Dynamic RDS
72 AUDIO
ShadowLight8
EDM audio
OFFICIAL 33 AUDIO
FalconChristmas
Event Date
OFFICIAL 84 MESSAGING
FalconChristmas

The Plugins page.

FPP 10 rebuilt this page around a **card grid**: every plugin is a tile showing its icon (or its initials if it has none), name, author, a short description, its category, how many installs it has, and an **Install** / **Uninstall** button.

21.1 Finding a plugin

- **Tabs** – **Available**, **Installed** and **Updates** each carry a live count, so you can see at a glance how many plugins are installed and how many have an update waiting.
- **Search** – the single box at the top right filters as you type. It doubles as the way to add a third-party plugin: paste the URL of a plugin's `pluginInfo.json` and FPP loads its details automatically. (Without a valid `pluginInfo.json`, a plugin will not install. FPP warns if the URL is missing a scheme such as `https://`.)
- **Categories** – pills across the top (**Audio**, **Automation**, **Data Feeds**, **Display & Video**, **Hardware**, **Interaction**, **Messaging**, **Monitoring**, **Payments**, **Other**, plus **All**) filter the grid, each showing how many plugins it holds. The counts follow your search, so you can see where the matches are.
- **Popular Plugins** – a row of the most-installed plugins across the community, as a starting point if you do not know what you are looking for.

21.2 Installing and managing

Click **Install** on a card to install a plugin, and **Check for Updates** to re-query the plugin sources (the button shows a spinner while it works). Where several plugins need attention there are bulk **Update All**, **Reinstall All** and **Uninstall All** actions, and each installed plugin also has its own **Reinstall**. An installed plugin gains an **Open** button that jumps straight to its configuration page.

Plugins marked **OFFICIAL** are maintained as part of the FPP project. A plugin that has not been updated for this release is flagged **NOT UPDATED FOR FPP 10** and its button changes to **Install anyway** — it may work, but it has not been verified against FPP 10. Plugins known to be incompatible are tucked into a collapsed **Incompatible Plugins** section at the bottom of the page rather than cluttering the grid.

> **Note:** In FPP 10 plugins can be loaded and unloaded while FPP is running, so installing or removing one no longer always means restarting FPPD. FPP will tell you when a restart *is* needed. After an FPPOS upgrade, FPP prompts you to reinstall your plugins.
>
> **Note:** Most plugins require some configuration before they work correctly. The plugin author chooses which menu heading the

plugin appears under; its configuration pages appear at the bottom of that menu drop-down (there may be more than one). Refer to the plugin's home/help page for setup, and only install plugins you trust, as they run with full access to the device.

Tip: At the **Developer** UI level each card also shows the plugin's open issue and pull-request counts, which is a quick way to gauge how actively a plugin is maintained.

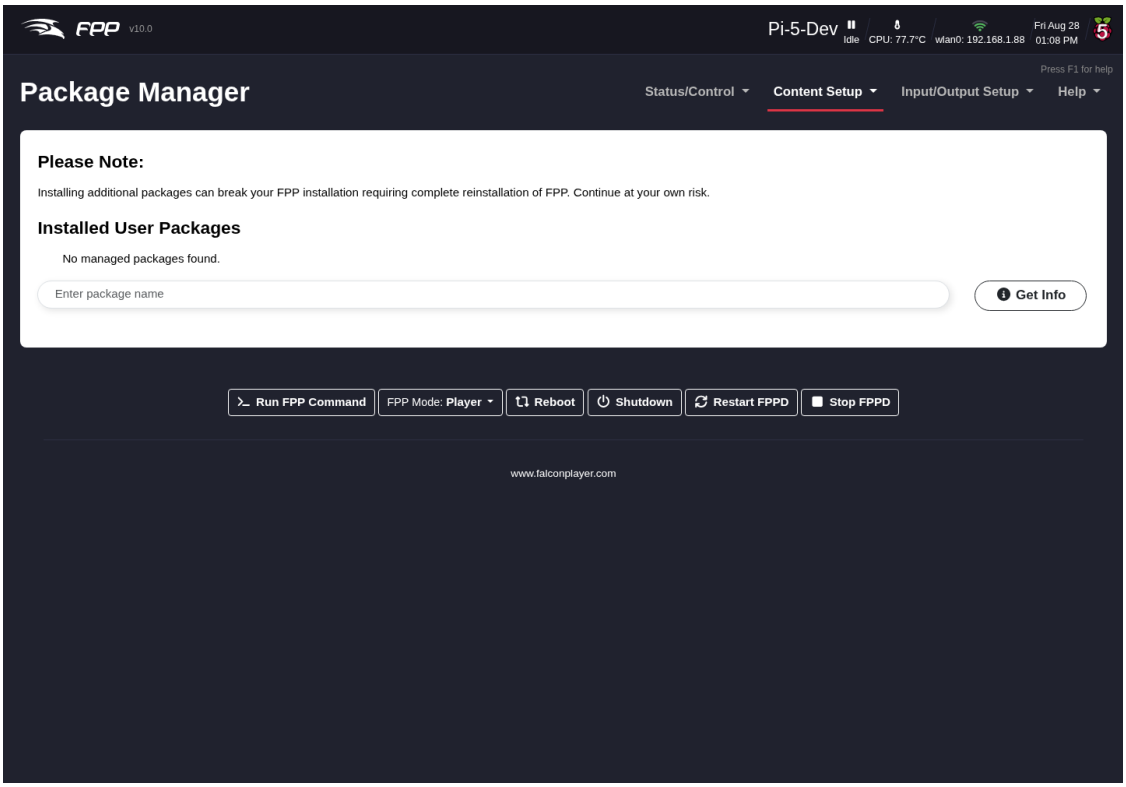
21.3 Plugin logs and health

All plugin activity is written to a single unified **plugin.log**, which makes install and runtime problems much easier to trace — see Help → Troubleshooting Commands. FPP also tracks whether each installed plugin is official, community or unknown, and surfaces that in the **System Health Check**.

Tip: Take an FPP Backup before installing plugins, packages or scripts, so you can roll back if something misbehaves.

22 Packages

The **Packages** page (**Content Setup → Packages**) installs optional operating-system software that some workflows need but that is not part of the base image. This keeps the base image lean while still allowing advanced setups to add what they require.



The Packages page.

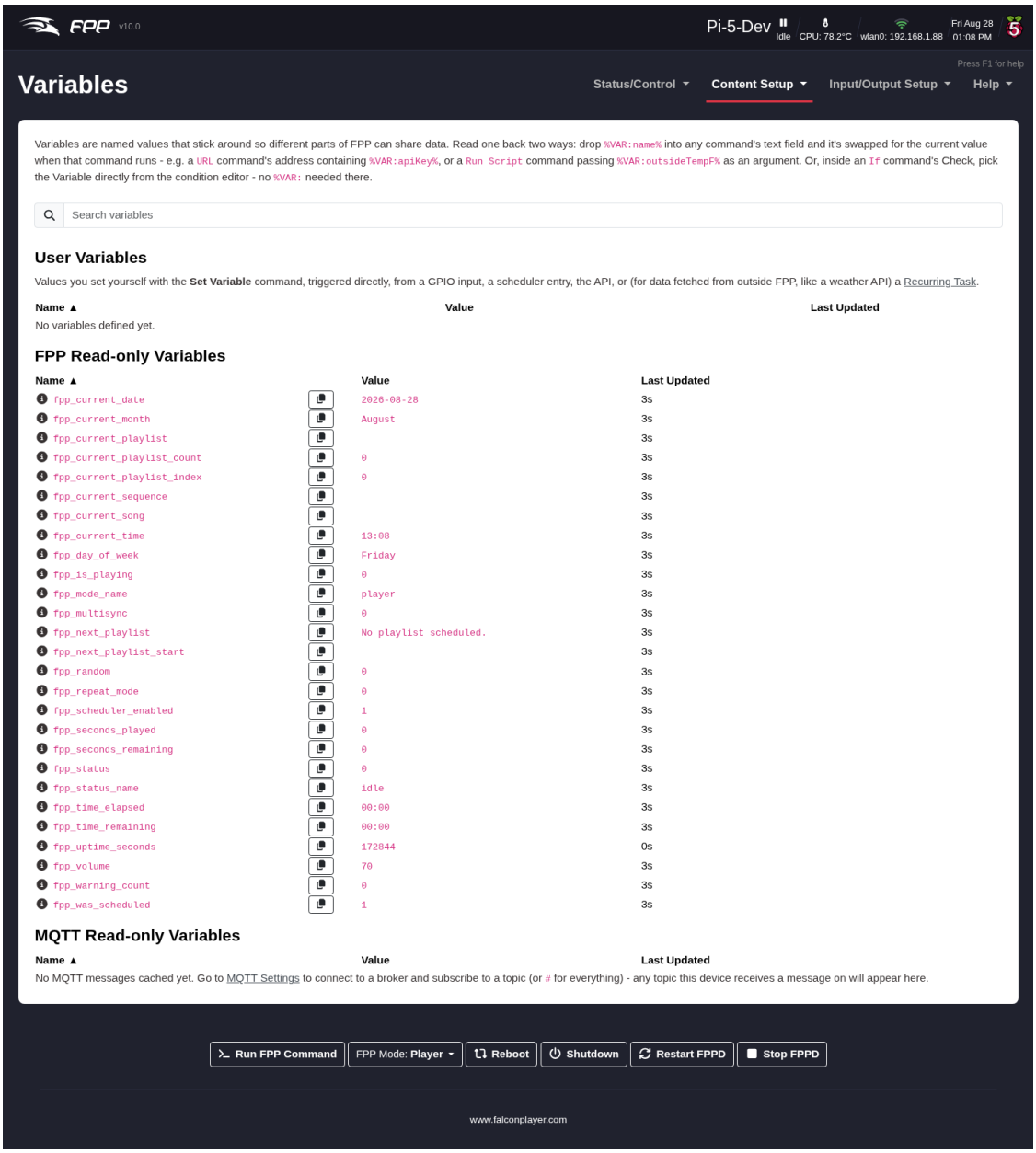
23 Variables

FPP 10 can store named **Variables** and act on them, so a show can remember things between commands — a counter of how many times a button has been pressed, a value fetched from a web service, a mode you set at the start of the night. The related Recurring Tasks page works alongside this one. Both appear once your **UI Level** is **Advanced** or higher (see FPP Settings → UI).

> **Note:** These pages were introduced during FPP 10 and were briefly at the *Experimental* UI level; they are now **Advanced**. If you cannot see them, raise your UI Level, or use the temporary Advanced button on *FPP Settings → UI*.

23.1 The Variables page

Open **Content Setup → Variables**.



The Variables page.

The page is a live view of every variable FPP currently knows about, each listed with its **Name**, **Value** and **Last Updated** time, and a **Search variables** box to filter a long list. It is a *monitor*, not an editor — variables are created and changed by the **Set Variable** command (below), by **Recurring Tasks**, by plugins, and by FPP itself. The page is split into three sections:

- **User Variables** – the ones you set yourself with the *Set Variable* command, whether triggered directly, from a GPIO input, a scheduler

entry, the API, or a Recurring Task. These are the only variables you can write to.

- **FPP Read-only Variables** – values FPP maintains about itself, updated continuously. They include `fpp_status_name`, `fpp_current_playlist`, `fpp_current_playlist_index`, `fpp_current_playlist_count`, `fpp_current_sequence`, `fpp_current_song`, `fpp_seconds_played`, `fpp_seconds_remaining`, `fpp_time_elapsed`, `fpp_time_remaining`, `fpp_volume`, `fpp_is_playing`, `fpp_repeat_mode`, `fpp_next_playlist`, `fpp_next_playlist_start`, `fpp_scheduler_enabled`, `fpp_was_scheduled`, `fpp_mode_name`, `fpp_multisync`, `fpp_current_date`, `fpp_current_month`, `fpp_current_time`, `fpp_day_of_week`, `fpp_uptime_seconds`, `fpp_warning_count`, `fpp_status` and `fpp_random`. Each has an **i** icon explaining it and a copy button that copies its `%VAR:…%` reference to the clipboard.
- **MQTT Read-only Variables** – values from MQTT topics this device has received messages on. Nothing appears until you connect to a broker and subscribe to a topic (use `#` to subscribe to everything) — see FPP Settings → MQTT.

23.1.1 Using a variable

Anywhere a command takes a text field, you can substitute a variable by writing `%VAR:name%` — for example a *Text Overlay* command with the message `Now playing: %VAR:fpp_current_song%`. In an **If** command you pick the variable directly from the *Check* list rather than typing it.

23.2 The Set Variable command

Set Variable is an FPP Command (category *Events*, Advanced UI level), so it can be used anywhere commands are — a Command Preset, a playlist entry, a GPIO input, the scheduler, or the API. Its arguments are:

- **Variable Name** – the name the value is stored and looked up under.
- **Set To** – how the value is produced:
  - **Value** – store the text or number you type, as-is.
  - **Increment** – add **Amount** to the variable's current value each time the command runs (use a negative amount to count down). Handy for counters.
  - **Random** – pick a whole number between **Minimum** and **Maximum**.
  - **Expression** – calculate the value from a formula. The formula must start with `=` and may reference other variables by name — for example `=2+3*4`, or `=temp*1.8+32` to convert a temperature to Fahrenheit, or `=fpp_volume+10`.

- **Persist** – if ticked, the value is written to disk and reloaded when FPP starts, so it survives a restart or reboot. Unticked (the default), the variable lives only in memory and is unset again after every restart.

23.3 The If command

If (category *Events*, Advanced UI level) branches on one or more conditions, letting a single trigger do different things depending on the state of the show.

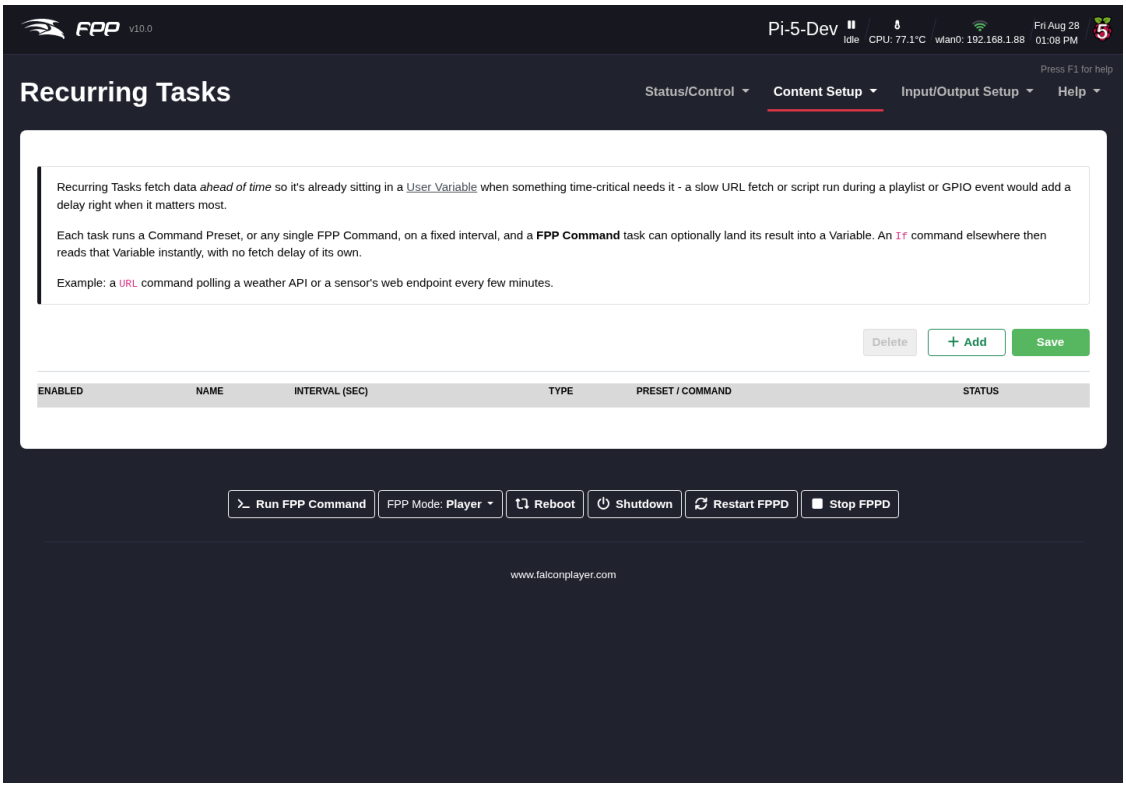
- **Check** – what to test. Build one condition, or add more and choose whether **ALL** or **ANY** of them must be true. The AND/OR choice only appears once there are two or more conditions, so the simple case stays simple.
- **Then Run** – the commands to run when the check is **True**, plus an **Order** toggle: **Sequential** waits for each command to finish before starting the next; **Parallel** fires them all at once without waiting.
- **Otherwise Run** – the commands to run when the check is **False**, with the same **Order** toggle. Leave it empty to do nothing.

Tip: Combine the two — a *Set Variable* with **Increment** to count button presses, and an *If* that checks the counter and plays a different playlist on every third press.

24 Recurring Tasks

A recurring task runs a Command Preset, or any single FPP Command, over and over on a fixed interval — no schedule entry or playlist required. This page appears once your **UI Level** is **Advanced** or higher (see FPP Settings → UI), alongside the related Variables page.

Open **Content Setup → Recurring Tasks**.



The Recurring Tasks page.

The point is to fetch data **ahead of time**, so it is already sitting in a variable when something time-critical needs it. A slow URL fetch or script run performed in the middle of a playlist or from a GPIO event would add a delay at exactly the wrong moment; a recurring task does that work in the background, and an **If** command elsewhere reads the resulting variable instantly. The classic example is a `URL` command polling a weather API or a sensor's web endpoint every few minutes.

Each row in the table has:

- **Enabled** – run this task, or leave it defined but idle.
- **Name** – a label for the task.
- **Interval (sec)** – how often to run it.
- **Type** – run a saved **Command Preset**, or an **FPP Command** chosen directly.
- **Preset / Command** – which preset or command to run, and its arguments.
- **Status** – the outcome of the last run, so you can see at a glance whether the task is working.

Use **Add** to create a row, **Delete** to remove selected rows, and **Save** to store your changes — as elsewhere in FPP, nothing takes effect until you save.

24.1 Capturing the result into a variable

An **FPP Command** task can store what its command returns into a variable, using the **Result Variable / Filter** fields:

- **Result Variable** – the variable name to store the result under, and a **Persist** option that behaves exactly as it does for *Set Variable* (saved to disk and reloaded at startup, versus memory-only).
- **Filter** – how to extract the value you want from the raw result:
  - **None (use raw result)** – store the whole response.
  - **JSON Field** – pull a single named field out of a JSON response.
  - **Between Markers** – keep the text between two marker strings, which is a simple way to scrape a value out of plain text or HTML. Leave the first marker blank to start at the beginning, or the second blank to keep everything to the end.
  - **Regex (advanced)** – extract the value with a regular expression.

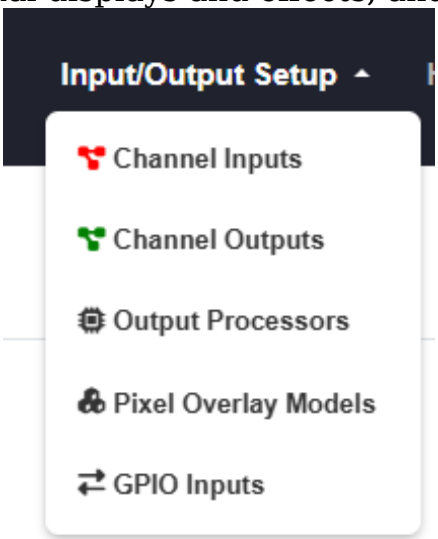
Tip: A short interval on a task that calls out to the internet will hammer the remote service and can slow FPP down. Poll no faster than you actually need — for most data (weather, sunset, a thermostat) minutes are plenty, not seconds.

Note: Recurring tasks run continuously while FPPD is running, whether or not a show is playing. If a task's command depends on a playlist being active, guard it with an **If** command that checks `fpp_is_playing`.

25 Input/Output Setup Section

The **Input/Output Setup** menu configures how show data gets in and out of this FPP device: incoming **Channel Inputs** (bridging E1.31/ArtNet/DDP/DMX from another controller), outgoing **Channel Outputs** to your lights, any **Output Processors** applied along the way,

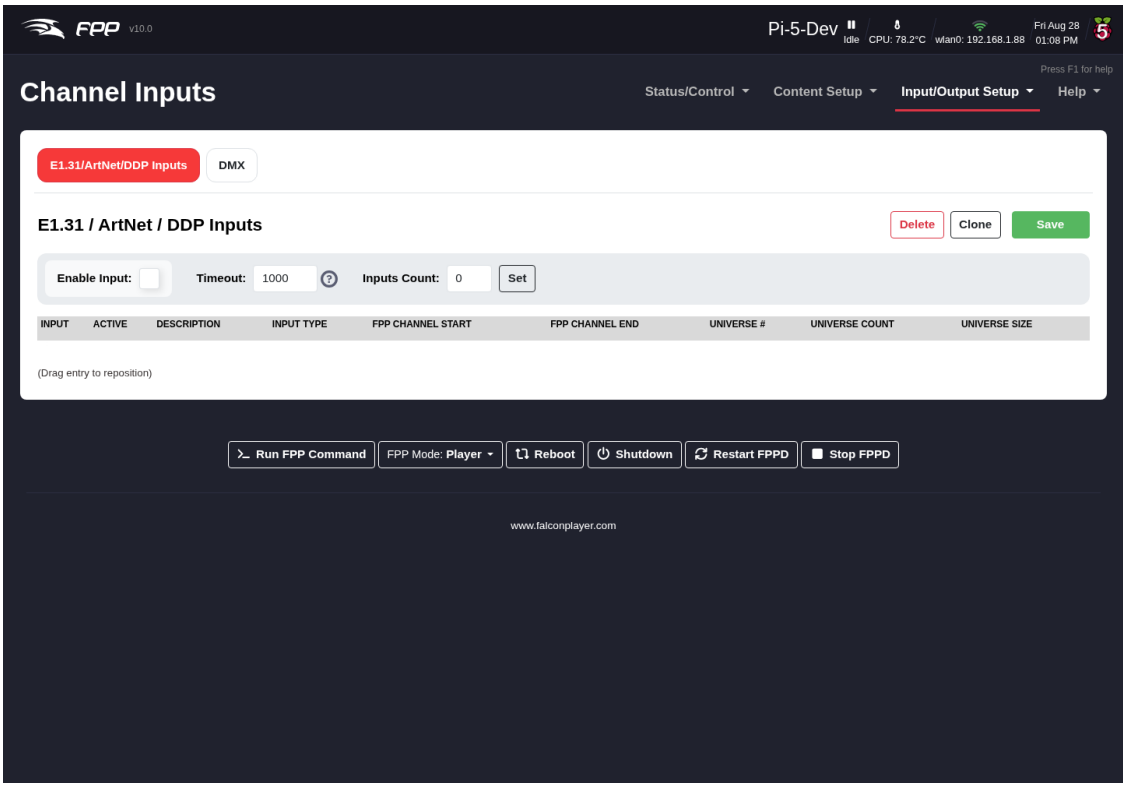
Pixel Overlay Models for virtual displays and effects, and **GPIO Inputs** for



physical buttons and sensors.

26 Channel Inputs

The **Input/Output Setup** section defines the channel/universe configuration of the controllers connected to FPP, and other input/output settings. The **Channel Inputs** page identifies incoming data this FPP should “listen to” — turning FPP into a **bridge**. Open **Input/Output Setup → Channel Inputs**.



The Channel Inputs page.

Configuring inputs lets you control your lights/props directly from **xLights**, **xSchedule** or other software, and is useful for testing from xLights. If you will not send pixel data from an outside device, you do not need to configure inputs.

The page has two tabs: **E1.31/ArtNet/DDP Inputs** for network input, and **DMX** for receiving DMX over a serial adapter.

To pass incoming E1.31/ArtNet/DDP data through to a connected controller or other device, enable the input (and, for E1.31/ArtNet, configure it). Any events or output processes triggered by channel data are still processed.

> **Note:** For **DDP** you do **not** need to configure inputs — just enable the input, and FPP receives and recognises DDP packets automatically. Do not enter E1.31/ArtNet universes that other FPPs are already using.

26.1 E1.31/ArtNet/DDP Inputs

26.1.1 Settings

- **Enable Input** – allow FPP to process incoming channel data.
- **Timeout** – seconds after which, if no E1.31/DDP/ArtNet signal is received, a blanking signal is sent out.

- **Inputs Count** – the number of input lines to configure; press **Set** to populate the table.

For each line:

- **Input** – a reference number for the line.
- **Active** – make the line's universes active (untick to disable).
- **Description** – identify the controller these universes are assigned to.
- **Input Type** – the type being received; **E1.31 Unicast** is usually best.
- **FPP Channel Start** – the start channel for these universes (typically 1 for a single controller; otherwise determined from your xLights setup).
- **FPP Channel End** – calculated automatically.
- **Universe #** – the first universe number (1–63999; need not start at 1).
- **Universe Count** – how many universes to assign. Universes do not map directly to ports; define them per controller and only as many as you need.
- **Universe Size** – channels per universe; commonly **512** or **510** (both work equally well). Use the same numbering and sizes everywhere for this device (xLights, the controller, etc.).

Your universes, FPP start channels and sizes must match your sequencer, show player and controller.

> **Important:** If you change the FPP Start Channel, Universe #, Universe Count or Universe Size after configuring, adjust the other universe lines too — FPP does **not** auto-correct them.

26.1.2 Adding E1.31/ArtNet inputs

Normally enter **1** for Inputs Count, click **Set**, and fill in the fields. The **Clone** button copies a selected line's settings (Universe #, Count, Size, Type) to lines below it, incrementing the universe number on each (you specify how many to clone; there must be enough lines below). **Delete** removes a selected line.

> **Important:** Click **Save** after any change, then **Restart FPPD** when prompted for it to take effect.
>
> **Tip:** For testing from xLights you only need to *enable* the inputs, since xLights sends DDP, which needs no configuration. Bridge mode is independent of playing local sequences — a common workflow is to bridge during sequencing, then switch to **Player** mode to run the show from uploaded FSEQ files.

26.2 DMX

The **DMX** tab receives DMX from a serial adapter rather than over the network — for example, to drive FPP's outputs from a lighting desk or another DMX source.

FPP lists each serial device it detects, one row per device:

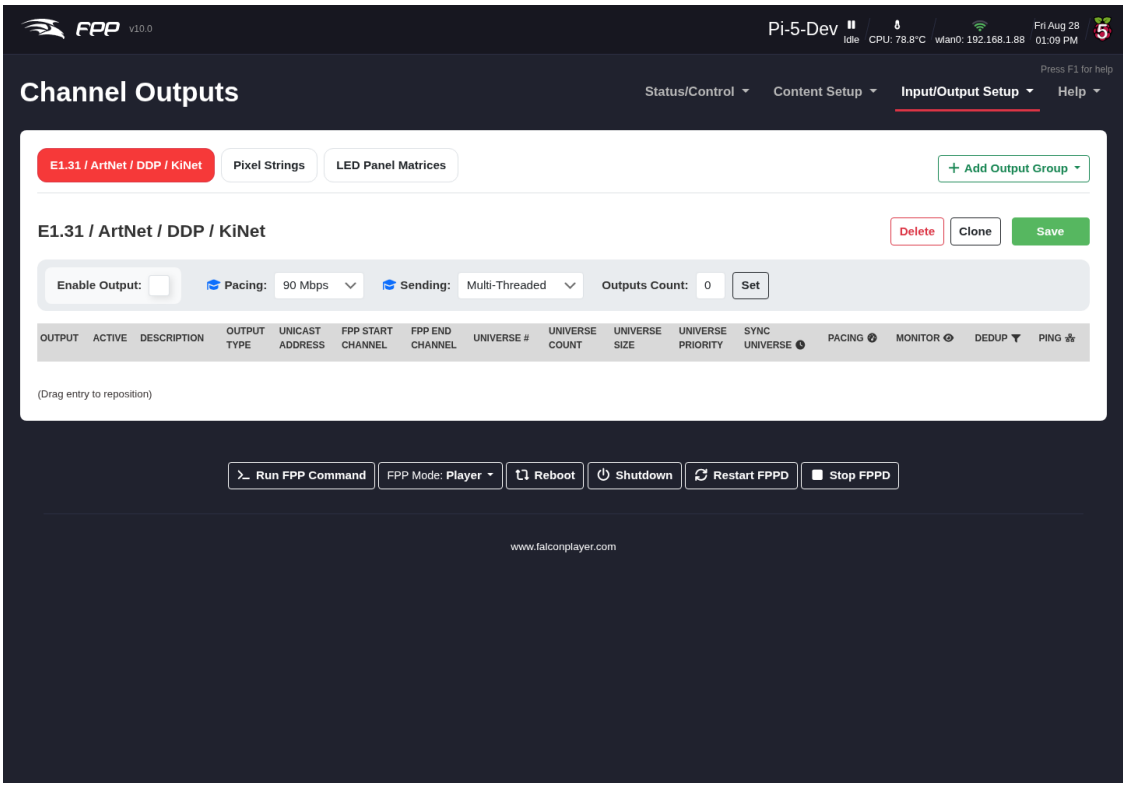
- **Enable** – receive DMX on this serial port.
- **Serial Port** – the detected adapter (for example a USB-to-DMX dongle).
- **Start Channel** – the FPP channel the first received DMX channel maps to.
- **Num Channels** – how many channels to read from the DMX stream (a full DMX universe is 512).

Click **Save** when done, then **Restart FPPD** when prompted.

> **Note:** Serial ports only appear here when FPP detects the adapter, so plug it in before loading the page. If the row you expect is missing, check the adapter is recognised under *Help → Troubleshooting Commands → USB*.

27 Channel Outputs

The **Channel Outputs** page configures how FPP outputs channel data to the controllers, hats and capes connected to it. Open **Input/Output Setup → Channel Outputs**. The available tabs depend on your SBC and any attached cape.



Channel Outputs — the E1.31 / ArtNet / DDP / KiNet tab.

Set up your outputs to match the controller connected to FPP and the settings in your sequencing software. The output types are:

- **E1.31 / ArtNet / DDP / KiNet** – for controllers connected over Ethernet (or a switch). **DDP** is preferred where supported (e.g. Falcon and KulpLights controllers with recent firmware, ESP PixelStick). Output over Wi-Fi is possible but not recommended.
- **Pixel Strings** (shown per cape, e.g. *PiHat Pixel Strings*, *K8-B*) – WS281x pixels driven directly by a Pi hat or BeagleBone cape, or from the GPIO pins. FPP uses the attached cape's EEPROM to build the correct page; without a programmed EEPROM the section is blank and you must install a **Virtual EEPROM** (below). On a Raspberry Pi, FPP 10 drives these pins with **DPI** on every model.
- **LED Panel Matrices** – P10/P5 panels via a BeagleBone Octoscroller-type cape, a Pi matrix hat, or a ColorLight card.
- **PWM** – PWM outputs (servos, single-colour dimming) on a cape that provides them and has a signed EEPROM.

Every other output type — **DMX / Serial**, **GPIO**, **Virtuals**, **SPI**, **PWM** (via PCA9685) and **Control Signal** — is added on demand with the **+ Add Output Group** button, described under *Additional output groups* at the end of this chapter.

> **Changed in v10:** The Channel Outputs screen was redesigned. The old single **Other** tab, which held DMX, serial, control and virtual types together, has been **removed**. Those outputs are now organised into categories that you add as tabs with **+ Add Output Group**, so the page shows only the output types you actually use.

27.1 E1.31 / ArtNet / DDP / KiNet

> **Note:** You only need to enable/configure these outputs if this device sends pixel data to **external** devices over the network. It does not apply to locally attached hats/capes or serial DMX ports.

Your universes, FPP start channels and sizes must match your sequencer and controller. Using **FPP Connect** in xLights (the **UDP** option under Tools) to upload the configuration is the recommended method, as it avoids typing errors.

> **Warning:** If you do **not** intend to send E1.31/ArtNet/DDP data but select the UDP option in FPP Connect, it will configure **and activate** these outputs, which can cause lag/stutter and unexpected results.

- **Enable Output** – enable network output.
- **Sending** – the send strategy *(Advanced)*:
  - **Multi-Threaded Blocking** *(default)* – multiple threads; send a packet and wait for acknowledgement before the next. Uses FPP's multi-threading for better performance.
  - **Single-Threaded Blocking** – one thread, wait for acknowledgement.
  - **Multi-Threaded Non-Blocking** – multiple threads, send the next packet as soon as it is ready.
  - **Single-Threaded Non-Blocking** – one thread, send as soon as ready.
- **Outputs Count** – the number of output rows, typically one **per controller** (even when a controller uses several universes). Click **Set** to create the rows.
- **Set / Save / Clone / Delete** – create the rows, save, copy a row to those below it, or delete a row.

For each output row:

- **Active** – transmit this line's universes (activate only the outputs you actually need).
- **Description** – identify the controller.
- **Output Type** – how the data is sent: **DDP** (recommended where supported), **E1.31 Multicast**, **E1.31 Unicast** (more efficient — prefer Unicast for E1.31), or **ArtNet**. **DDP** is normally *Raw Channel*

numbers; **DDP-One Based** makes each controller start at channel 1 (you must then configure those devices to match).

- **Unicast Address** – for Unicast or DDP, the target device's IP.
- **FPP Start Channel** – the absolute channel configured in your sequencer for this range.
- **FPP End Channel** – calculated, to help verify your entry.
- **Universe #, Universe Count, Universe Size** – the starting universe, how many universes on this line (multiple per line is recommended), and channels per universe (commonly 512 or 510 — keep it consistent across your show).
- **Universe Priority** – priority for the E1.31 packets when more than one source targets a device.

27.2 Pixel Strings

The Pixel Strings tab (named for the detected cape) configures WS281x pixels wired to the hat/cape or GPIO. Common controls:

- **Enable (Cape Type)** – enable the pixel-string output (untick to disable without losing the configuration).
- **Cape Config** – the cape type from the EEPROM (some, like the K16, offer expansion-board and serial options).
- **Testing** – output test patterns (stays active until turned off): **Port Number** (white pixels at the start of each string indicate its port), **Pixel Count by Port**, **Pixel Count by String**, and **Red/Green/Blue/White Fade**.
- **Pixel Timing** – selects the timing group the ports run at. Changing it re-offers each port's protocol list; a port set to a protocol the new group does not contain moves to that group's first protocol.
- **Clone String** – copy a string's settings to others, advancing the start channel.

Per port:

- **Port** – the hat's output port; click **+** to add a **virtual string** to a port (for different daisy-chained models needing individual adjustment).
- **Description** – a label for the port.
- **Start Channel** – matches the start channel in your sequencer (highlighted orange if there may be an error — hover for details).
- **Pixel Count** – pixels on the port (red if it exceeds the port's capacity).
- **Press F2 to auto set** – fills the next row's start channel for contiguous ports.
- **Group Count** – group pixels that always display identically.
- **End Channel** – the ending channel (calculated).
- **Direction** – **Reverse** feeds data as if from the end of the string.

- **Protocol** – the pixel chipset the port drives. FPP 10 added **per-port protocols**, so one cape can drive different pixel types on different ports. Besides the usual **ws2811** / **ws2801**, the list covers the **TM18xx** family (tm1803, tm1804, tm1809, tm1812, tm1814, tm1814a), the **UCS** family (ucs1903, ucs1904, ucs1912, ucs2903, ucs2904, and the 16-bit ucs7604, ucs8903, ucs8904), **GS8206** / **GS8208**, and **SK6812** / **SK6812RGBW**. The column is hidden on capes where every port offers only one protocol, so you will not see it on a ws281x-only cape.
- **Color Order** – match your pixels' colour order.
- **Start Nulls / End Nulls** – number of null nodes used to boost transmission distance at each end.
- **Zig Zag** – for props like a mega-tree where one string feeds several strands; enter how many times the string changes direction. Do **not** use this if you set Strands/String in your sequencer.
- **Brightness** – lower brightness can look better on dense props and reduces power draw.
- **Gamma** – correction for the non-linear way we perceive brightness, and to match pixels from different vendors.

27.2.1 Configuring the Virtual EEPROM

From FPP 6 onward, the advanced pixel output protocols require an EEPROM. If your hat/cape has none, configure a **Virtual EEPROM**, choosing the type for your output. Examples:

- **PiHat** – two ports (two GPIO pins), driven via DPIPixels so on-board audio keeps working; limited to 50 pixels per port without a licence.
- **DPIPixels-24** – up to 24 ports without disabling on-board audio; 50 pixels per port without a licence.
- **rPi-28D / rPi-MFC** – Hanson Electronics boards.
- **F16-B / F32-B / F4-B / F8-B / F8-Bv2 / F8-PB** – Falcon/Kulp DIY boards.

Note: FPP 10 removed the old `rpi_ws281x` driver on Raspberry Pi and drives those pins with **DPIPixels** instead. DPI uses the same GPIO pins, works on every supported Pi model, and does not conflict with the on-board audio — so the variants that used to disable on-board audio are gone, and RPIWS281X capes are no longer offered in the UI.

Existing configurations migrate themselves, with no action needed from you: FPP remaps the output type when the config loads, rewrites `co-pixelStrings.json` on disk at boot, and writes the migrated version back out the next time you save the page. A third-party cape with a physically burned RPIWS281X EEPROM is used as-is (the pin format is the same); any pin DPI cannot drive is

skipped with a warning rather than failing the whole cape. The **rPi-28D** is a special case — its third output is ws2801 over SPI, which DPI cannot drive, so it maps to the 4-output variant and that protocol is rewritten to ws2811. - **RGB-123 / PB-16 / PocketScroller / Spixel** – various boards (Spixel drives 16 strings of APA102/LPD6803/LPD8806 directly from the Pi GPIO).

Some types offer additional board-specific options — get the correct EEPROM and board type from your vendor. If a Virtual EEPROM needs a licence for its advanced features, a blue banner explains this; the **Cape Info** link opens the Cape Info page with more detail and a link to obtain the licence (see Pixel Port Licensing).

> **Screenshots pending — cape hardware required.** Full captures of the Pixel Strings tab need the relevant cape fitted so the ports are shown; these will be added from a cape-enabled system.

27.3 LED Panel Matrices

Configures LED panels (P10/P5 are most common), driven by a ColorLight card or a connected hat/cape. One FPP device can control multiple drivers; the practical limit depends on matrix size and the SBC's single-core speed, so test for performance.

> **Note:** For any output beyond a hat/cape you need a **dedicated Ethernet port for each ColorLight receiver**.

Click **Add Panel Matrix** and choose **Hat/Cape** or **ColorLight**. There are three settings screens — BeagleBone Hat/Cape, ColorLight, and Pi Hat/Cape. Common settings:

> **Note:** FPP 10 allows **several panel matrices to share one cape**, so a cape-driven setup is no longer limited to a single matrix the way it was in earlier releases. Multiple ColorLight panels were already supported.

Each matrix you add gets its own sub-tab — **Panel Matrix 1**, **Panel Matrix 2** and so on — so every matrix keeps its own layout, start channel and settings. Work through them one tab at a time; the settings below apply per matrix.

- **Enable LED Panels** – enable panel output.
- **Interface** – for ColorLight, the dedicated Ethernet port for that receiver.
- **Matrix Name** – names each matrix (shown on its Panel Tab).
- **Panel Layout (WxH)** – number of panels wide × high.
- **Single Panel Size (WxH)** – pixel size and scan rate of each panel (P10 = 32×16, P5 = 64×32).

- **Model Start Corner** – typically **Top Left** for xLights, **Bottom Left** for Vixen (match your sequencer).
- **Panel Gamma / Brightness** – gamma correction and overall brightness.
- **Panel Interleave** – for panels using non-standard data transmission.
- **Color Depth** – number of colours; reduce to minimise flicker on large sets (Hat/Cape only; ColorLight uses LEDVision for this).
- **Panel Row Address Type / LED Panel Type** – for panels with different row addressing or specialty panels (Pi Hat/Cape only).
- **Start Channel / Channel Count** – the panel array's absolute start channel and total channels.

Screenshots pending — cape hardware required.

27.4 Additional output groups (Add Output Group)

The tabs above (E1.31/ArtNet/DDP/KiNet, the cape's Pixel Strings, PWM and LED Panels) appear automatically based on your hardware. Every *other* output type lives in an **output group** that you add yourself: click **+ Add Output Group** at the right of the tab strip and pick a category. That category then becomes a new tab, where you add and configure individual outputs.

The categories, and the output types in each, are:

Category (tab)	Output types
DMX / Serial	DMX-Open, DMX-Pro, Generic Serial, uDMX, Pixelnet-Lynx, Pixelnet-Open, Renard, LOR, LOR Enhanced, Generic UDP
GPIO	GPIO, GPIO-595, PCF8574, MCP23017
Virtuals	HTTP Virtual Display, HTTP Virtual Display 3D, Virtual Display, Virtual Matrix
SPI	Generic SPI, SPI-nRF24L01, SPI ws2801, MAX7219 Matrix
PWM	PCA9685
Control Signal	MQTT Output, Control Channel, USB Relay

A category disappears from the **+ Add Output Group** menu once you have added it, and the menu reads *All groups already added* when every category is on screen.

What each category is for:

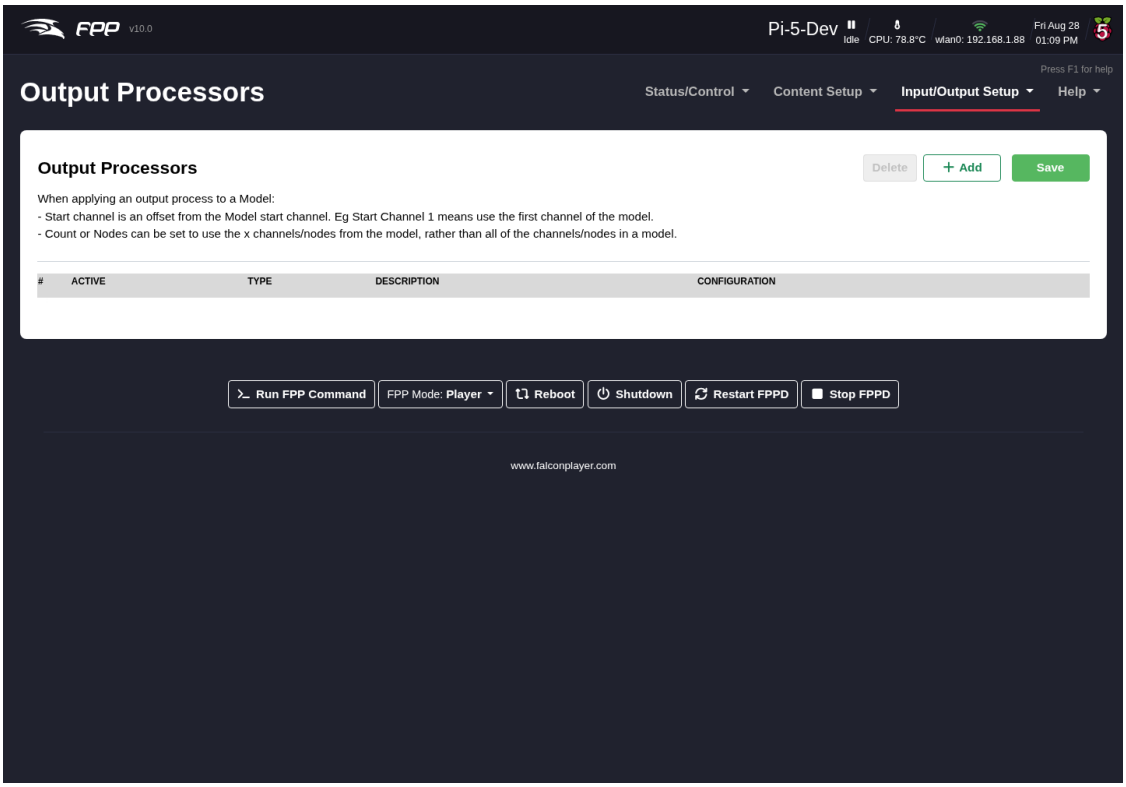
- **DMX / Serial** – DMX, Pixelnet, Renard and LOR over a USB/serial adapter. Select the serial device and protocol and map the channel range to it. *Generic UDP* sends raw channel data to an arbitrary UDP destination.
- **GPIO** – drive GPIO pins, or an I/O expander (PCF8574, MCP23017) or shift register (GPIO-595), from channel data — typically for relays.
- **Virtuals** – on-screen outputs rather than physical ones. Enabling **HTTP Virtual Display 3D** here activates the browser-based 3D preview, and **HTTP Virtual Display** the 2D one — see the 3D Virtual Display chapter. *Virtual Matrix* renders channel data to a framebuffer/display.
- **SPI** – devices on the SPI bus, including ws2801 pixels and MAX7219 matrices.
- **PWM** – PCA9685 PWM controllers, for servos and single-colour dimming. (A cape with its own PWM hardware gets a dedicated PWM tab instead, provided the cape's EEPROM is signed.)
- **Control Signal** – outputs that signal rather than light: publish channel values over **MQTT**, drive a **USB Relay**, or use a **Control Channel** to trigger FPP behaviour from channel data.

After any change, click **Save** and, when prompted, **Restart FPPD**.

> **Note:** Adding an output group only creates the tab. You still add individual outputs inside it, and each output needs its own start channel and channel count.

28 Output Processors

Output Processors modify the outgoing channel data — useful for things like moving a prop after the `.fseq` has been saved to FPP, or adjusting brightness when a controller does not support dimming. You can apply a processor to an absolute start channel or to one of your Pixel Overlay Models. Open **Input/Output Setup → Output Processors**.



The Output Processors page.

> **Note:** Output Processors are only available at the **Advanced** UI Level or higher. They apply to all content (sequences, effects, bridge data) and are evaluated in order.

The available processors are:

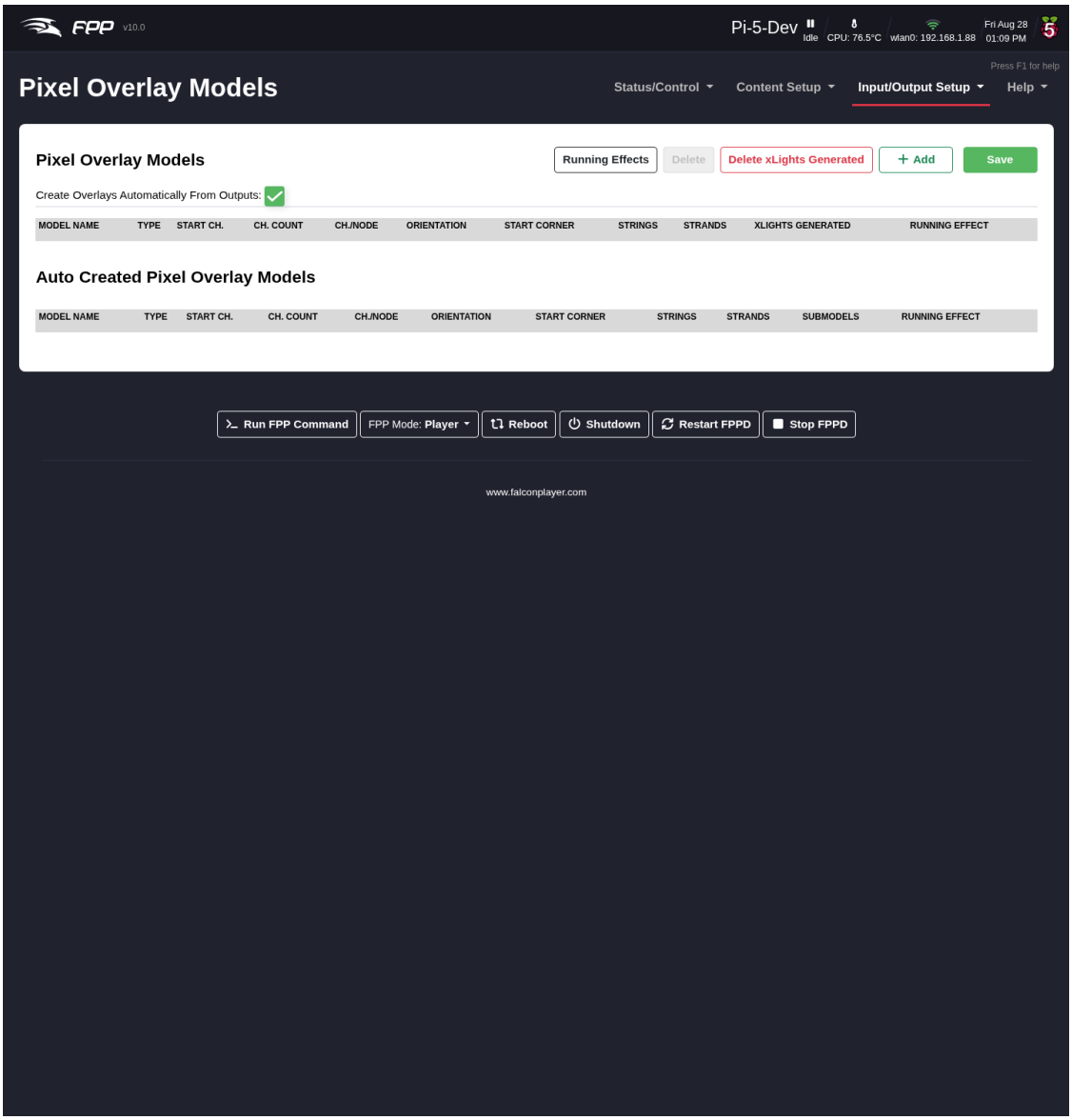
- **Remap** – copy or move a block of channels, e.g. when a prop is moved to a different port, or a “dumb” string is replaced with a pixel string.
  - **Description** – note the reason for the remap.
  - **Source Channel** – the first channel to remap.
  - **Destination** – the first channel to remap the data to.
  - **Count** – the number of channels to remap.
  - **Loops** – how many times to repeat the remap (e.g. to replace a dumb string with a pixel string, set Count to 3 and Loops to the number of pixels).
  - **Reverse** – reverse the mapping (by channel or by pixel), e.g. when a prop is wired in reverse.
- **Brightness** – adjust brightness or gamma on a channel range (useful when the controller cannot).
- **Hold Value** – when a channel value is 0, output the last non-zero value instead. Useful for servos or moving heads that you do not want returning to 0 between sequences.

- **Set Value** – set a fixed value for a channel range.
- **Reorder Colors** – change the colour order (e.g. when replacing part of a string with pixels of a different order).
- **Clamp Value** – set a maximum output value.
- **Scale Value** – scale values up or down (useful for servo control).
- **Three to Four** – convert 3-channel RGB data to 4-channel RGBW, for when your sequencing is RGB but some strings are RGBW.
  - **Output Color Order** – how the four output channels are ordered: **RGBW** or **WRGB**. Match your pixels.
  - **White Algorithm** – how the white channel is derived: **No White** leaves it at zero; **R=G=B->W** lights white only where the RGB values are equal (so greys and whites use the white element, and colours are untouched); **Advanced** extracts the white component from every colour, which gives a cleaner result on pixels with a good white element.
- **Override Zero** – force any zero RGB values to a remapped value.
- **Fold** – similar to a zig-zag.

Tip: Because processors apply to *all* content, they are ideal for display-wide corrections such as a global brightness cap. Order matters — apply a colour-order fix before a brightness limit, for example.

29 Pixel Overlay Models

Pixel Overlay Models let you manually manipulate a particular model in real time — via plugins, scripts, the API or MQTT — before its data is sent to the controllers, while the rest of your display keeps playing sequenced data. They are typically used for matrices or mega-trees (or other dense props), but work for any model you want to take manual control of, and are handy for individually testing models through FPP. Open **Input/Output Setup → Pixel Overlay Models**.



The Pixel Overlay Models page.

A model must match the settings in your sequencing software and the string ports in your controller.

29.1 Creating models

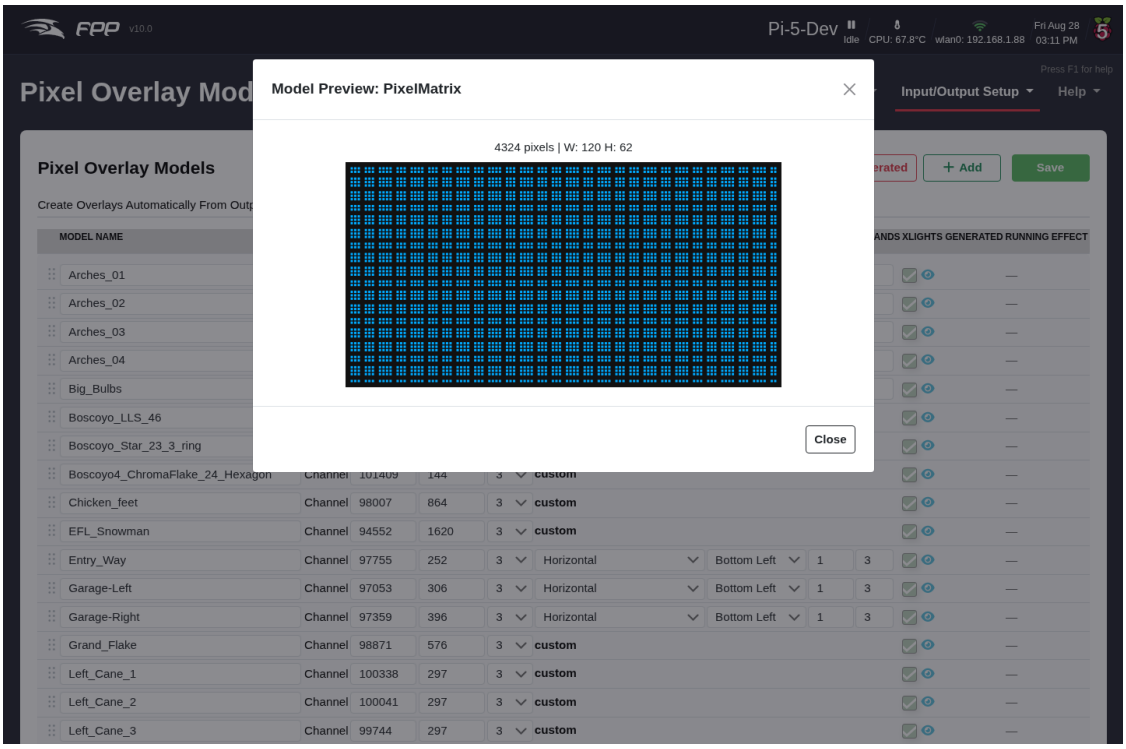
- **Create Overlays Automatically from Outputs** *(recommended)* – FPP builds the models from your configured outputs.
- **Export from xLights** (FPP Connect → **Models**) – uploads models, but for *all* models in your show (usually more than you want), and they can become incorrect if you change xLights and forget to re-upload.
- There is an option to **delete all xLights-generated models**.

29.2 Manual settings

- **Model Name** – used to reference the model elsewhere (plugins, scripts, etc.).
- **Type:**
  - **Channel** – typically for models made of pixel strings.
  - **Frame Buffer** – for a Virtual Matrix / Virtual Display, playlist *image* entries, and non-accelerated video output on Raspberry Pis.
  - **Sub Model** – to address only a portion of a model.
- **Start Ch.** – must match the start channel in your sequencer and string ports.
- **Ch. Count** – the number of channels (not pixels) the model uses.
- **Ch./Node** – channels per node (typically 3).
- **Orientation** – as configured in your sequencer.
- **Start Corner** – usually **Top Left** for xLights, **Bottom Left** for Vixen (P10/P5 panels); otherwise match your sequencer.
- **Strings** – typically the number of rows for P10/P5 panels; otherwise match your sequencer.
- **Strands** – 1 for P10/P5 panels; otherwise match your sequencer.

29.3 Previewing a model

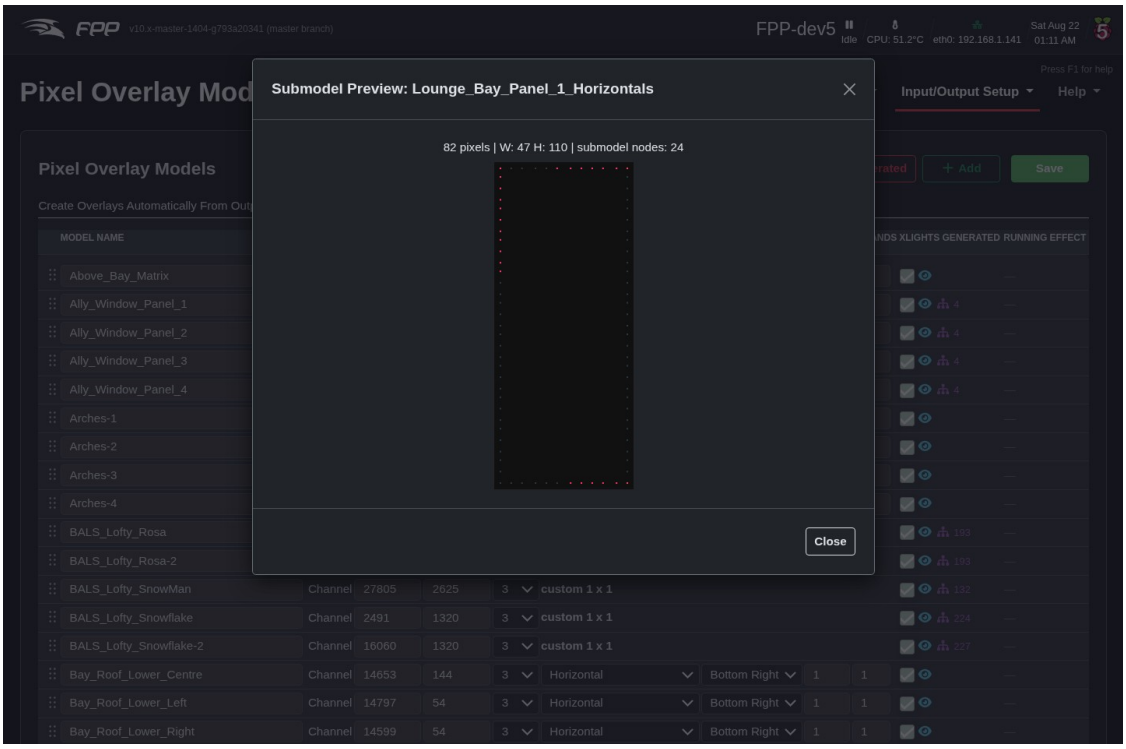
New in FPP 10, each model has a **preview** (the eye icon on its row) that opens a diagram of the model's actual pixel layout, drawn from the display map uploaded by xLights FPP Connect. This lets you confirm a model's geometry — its shape, dimensions and pixel arrangement — without sending any data to the lights.



A Pixel Overlay Model preview, showing the model's pixel layout.

29.4 xLights Submodels

Also new in FPP 10, FPP now understands xLights **submodels** — named regions within a model, such as individual window panes, the horizontal versus vertical runs of a prop, or sections of a mega-tree. When you upload from xLights FPP Connect, submodels appear **under their parent model**: a model with submodels shows a sitemap icon that expands a list of them, each with its own preview.



A submodel preview (a named region of its parent model).

Crucially, a submodel is addressable **by name** through *every* overlay command — **Overlay Model Effect**, **Fill**, **Clear**, **State** and **Text** — and through the API, MQTT and plugins, exactly like a top-level model. So you can run an effect on just the “Horizontals” of a window, or scroll text across one section of a prop, without sequencing it.

> **Note:** Submodels are read from the digested data that xLights FPP Connect uploads (`config/xlights-submodels.json`). They are loaded on demand — systems that never use a submodel pay no memory or startup cost. If your submodels do not appear, re-run FPP Connect after changing them in xLights.

29.5 Model Groups

FPP 10 also supports xLights **model groups** — collections of models and submodels that xLights treats as one arrangement. When present, the page shows a **Model Groups** section listing them. A group is addressable by name through the same overlay commands as models and submodels, but it renders across the whole group as a single canvas laid out by **real-world position**: an effect sweeping across the group sweeps across it in *physical space* (the familiar xLights “moving across the group” look). This makes it easy to run one coherent effect across a symmetrical arrangement of props.

> **Note:** Model groups render as RGB (an RGBW member’s white channel is not driven), and FPP provides a single sensible spatial

layout rather than every xLights buffer style. Like submodels, groups are loaded on demand from the data uploaded by FPP Connect (`config/xlights-modelgroups.json`).

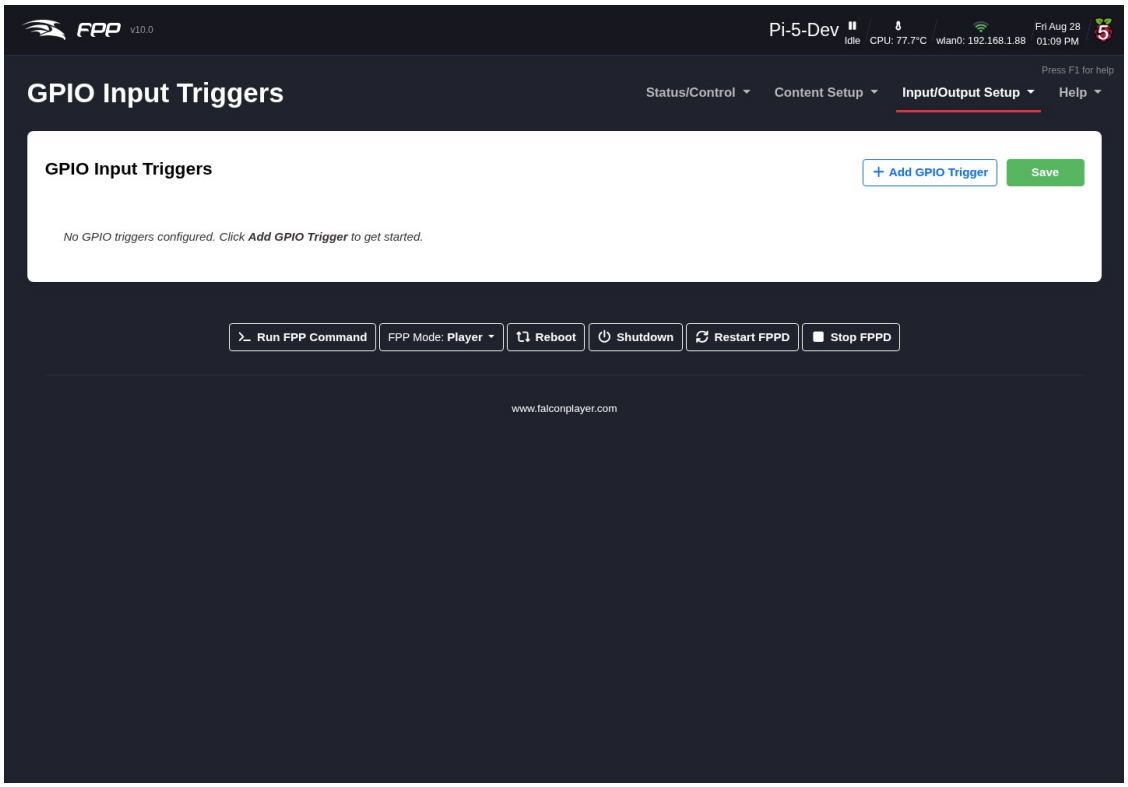
29.6 Uses

- Displaying real-time dynamic text on a matrix or mega-tree.
- Displaying the current time or a Christmas countdown on a matrix.
- Turning individual channels on/off (a Tune-To sign, inflatables, etc.) without sequencing them in every file.

Tip: The **Matrix Tools** plugin (via the Plugin Manager) uses the overlay feature to display and scroll dynamic text on a model from a web interface — you can even draw on your matrix in real time with your mouse.

30 GPIO Inputs

GPIO Inputs trigger internal FPP events from an external input — a button, a motion sensor, a switch. Each input connects to a pin on the FPP's GPIO header or to an add-on I/O board such as the PiFace. Open **Input/Output Setup → GPIO Inputs**.



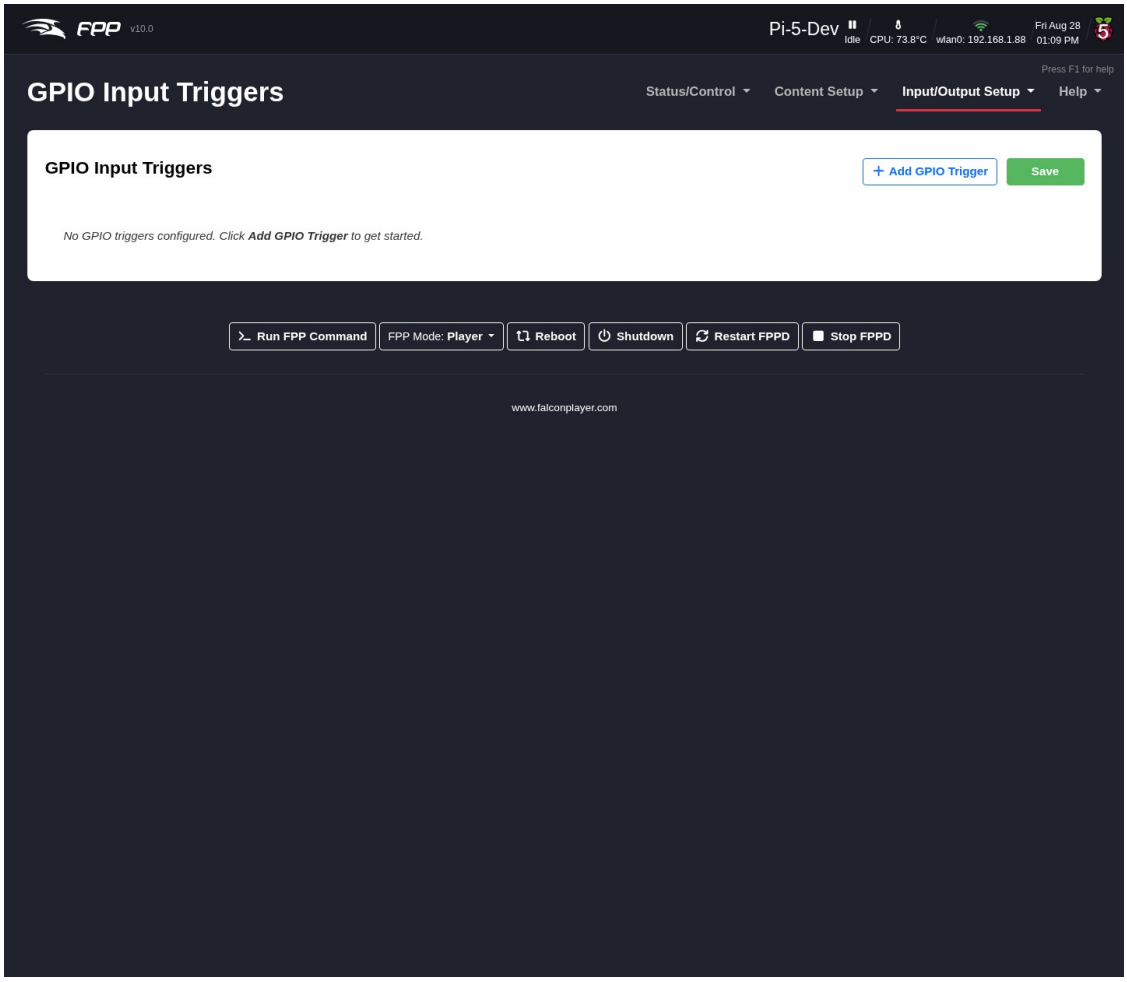
The GPIO Input Triggers page.

Each configured pin is shown as a card listing, at a glance, whether it is enabled, which pin it uses, its description, its pull and debounce settings, and the commands attached to its rising and falling edges. Use **+ Add GPIO Trigger** to create one, **Edit** to change one, the bin icon to remove one, and **Save** to store your changes.

> **Note:** As elsewhere in FPP, nothing takes effect until you click **Save** on the page itself — closing the edit dialog with **Apply** only updates the card.
>
> **Note:** If you select **None/External** pull, make sure your circuit establishes a definite high or low state — a floating pin can cause false triggers. Pi GPIO pins are 3.3 V and are **not** 5 V tolerant.

30.1 The Configure GPIO Trigger dialog

Adding or editing a trigger opens a dialog where all of a pin’s configuration lives. It is organised into three panels.



Configuring a GPIO trigger.

30.1.1 Pin & General Settings

- **GPIO Pin** – the pin this trigger uses. Pins already assigned to another trigger are not offered, so you cannot configure the same pin twice.
- **Enabled** – whether this trigger is active. Leave it unticked to keep a configuration without it firing.
- **Description** – what the input is for, e.g. *Start button* or *Emergency stop*. Up to 128 characters.
- **Pull Up / Down** – the resting state of the pin:
  - **None / External Pull** – relies on a pull resistor in your own circuit.
  - **Pull Up** – the pin reads HIGH at rest, so a button wired to ground produces a **falling** edge when pressed.
  - **Pull Down** – the pin reads LOW at rest, so a button wired to 3.3 V produces a **rising** edge when pressed.

30.1.2 Trigger Commands

Each pin can run commands on up to three events. Every list uses the same **Add Command** button, which opens the **FPP Command Editor** (see Command Presets), and commands can be reordered or deleted once added.

- **Rising Edge Commands** – run when the input goes HIGH (typically the button being pressed, with a pull-down).
- **Falling Edge Commands** – run when the input goes LOW (typically the button being released).
- **Long-Press / Hold Commands** – optional. Set a **Hold Time (ms)** and the commands here fire once the input has been held for that long. **0 disables** the hold behaviour; the maximum is 30000 ms.

  Note: When a hold command fires for a press, the **falling** commands for that same press are **suppressed** — so a short press and a long press can do genuinely different things without the short-press action also running.

30.1.3 Options

- **Debounce** – filters mechanical switch bounce.
  - **Debounce Time (ms)** – signals within this window are ignored. The default is 100 ms; raise it for a particularly “dirty” switch (10–60000 ms).
  - **Debounce On** – **Both edges** debounces press and release; **Rising only** debounces the press and lets the release fire immediately; **Falling only** does the reverse.
- **Re-enable GPIO After Trigger** – optional suppression that stops a button being re-triggered too quickly. While suppressed, further presses are ignored.

  - **Always enabled (no suppression)** – no delay; immediate re-triggering is allowed.
  - **Re-enable after a fixed delay** – lock the input for a set **Delay (ms)** after each trigger (100 ms to 1 hour).
  - **Re-enable when player becomes idle** – keep the input locked until FPP finishes playback. If no playback starts within 2 seconds, the input re-enables automatically, so a mis-fire cannot lock the button out.
- **Illuminated Button LED** – optional; drives a GPIO **output** for a lit button, so the button itself can show what the show is doing.
  - **LED Output Pin** – which GPIO output the LED is wired to, or **None**.
  - **Idle LED Mode** – what the LED does at rest: **Off** (dark), **On** (lit, showing the input is ready), or **Pulsing** at the **Pulse half-period (ms)** you set (500 gives a 1 Hz blink). While the input is suppressed after a trigger, the LED is held off whatever this is set to.
  - **Trigger Mode** – what the LED does when the button fires: **None** (idle mode only), **Follow input** (lit while held), **Flash N times** (150 ms on, 150 ms off, then back to idle), or **Stay on for N ms** then back to idle.

Click **Apply** to accept the dialog, then **Save** on the page.

30.2 Typical uses

- A **button** to start or stop a sequence or the show.
- A **motion sensor** to activate a special sequence.
- A **switch** to trigger another external device.
- A **lit button** whose LED pulses while idle and stays on while the show runs, using the *Illuminated Button LED* options above.
- A **single button doing two jobs** — a short press starts the show, a long press (via Hold Commands) shuts the device down.

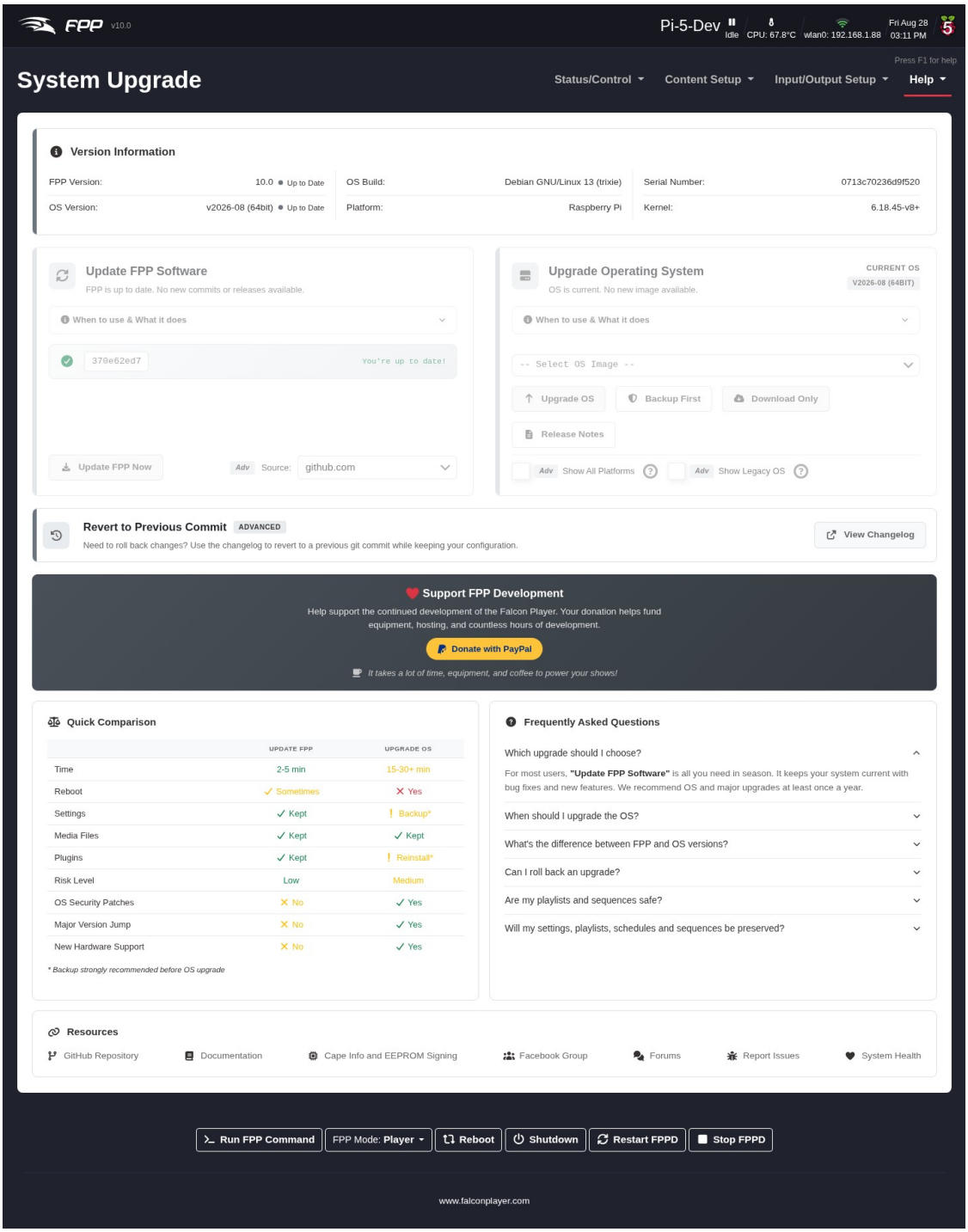
Note: GPIO inputs require real board hardware and correct wiring. When in doubt about voltage levels, use an appropriate interface or opto-isolator.

31 Help and Troubleshooting

The **Help** menu gathers FPP’s information, upgrade, support and diagnostic tools.

31.1 System Upgrade (About)

The **About** page (**Help → System Upgrade**) shows the current FPP version and statistics about the running system, and is where you perform a manual update.



The System Upgrade / About page.

31.1.1 Version Info

- **FPP Version** – the current FPP version.
- **Platform** – the SBC platform of this device.
- **FPP OS Build** – the current OS build. The OS should match the requirement for the FPP version (see the release notes or the *Upgrade OS* drop-down).
- **OS Version** – the SBC base OS version; some capabilities need the OS upgraded to match the build.
- **Hardware Serial Number / Kernel Version** – device identifiers.
- **System Boot Time / fppd Uptime** – when the device booted and how long the daemon has run (restarting fppd resets this).
- **Local Git Version** – the installed version (with a **ChangeLog** link and an update indicator).
- **Remote Git Version** – the latest available version; *Unknown* means no internet (often a network/DNS problem). A **Preview Changes** link shows what an update provides.

31.1.2 Upgrade FPP

A minor update (same branch) is indicated on the Local Git Version and by an icon in the header; click **Upgrade FPP** to install it. A major upgrade requires an OS update. Options include:

- **FPP Upgrade Source** – upgrade from GitHub or from another of your FPP devices (useful when some devices have no internet); the chosen source is also used for upgrades launched from the MultiSync page. *(Advanced.)*
- **Upgrade OS** – select an FPPOS version to download (or download and upgrade). With internet access the list includes the currently supported FPPOS files.
- **Show All Platforms** – view/download files for other platforms, e.g. to act as an upgrade source for Pi *and* BB devices. *(Advanced.)*
- **Show Legacy OS's** – show older, deprecated versions.
- **Preserve /opt/fpp** – when already on master and newer than the fppos image, upgrade the OS without downgrading FPP from master. *(Developer.)*

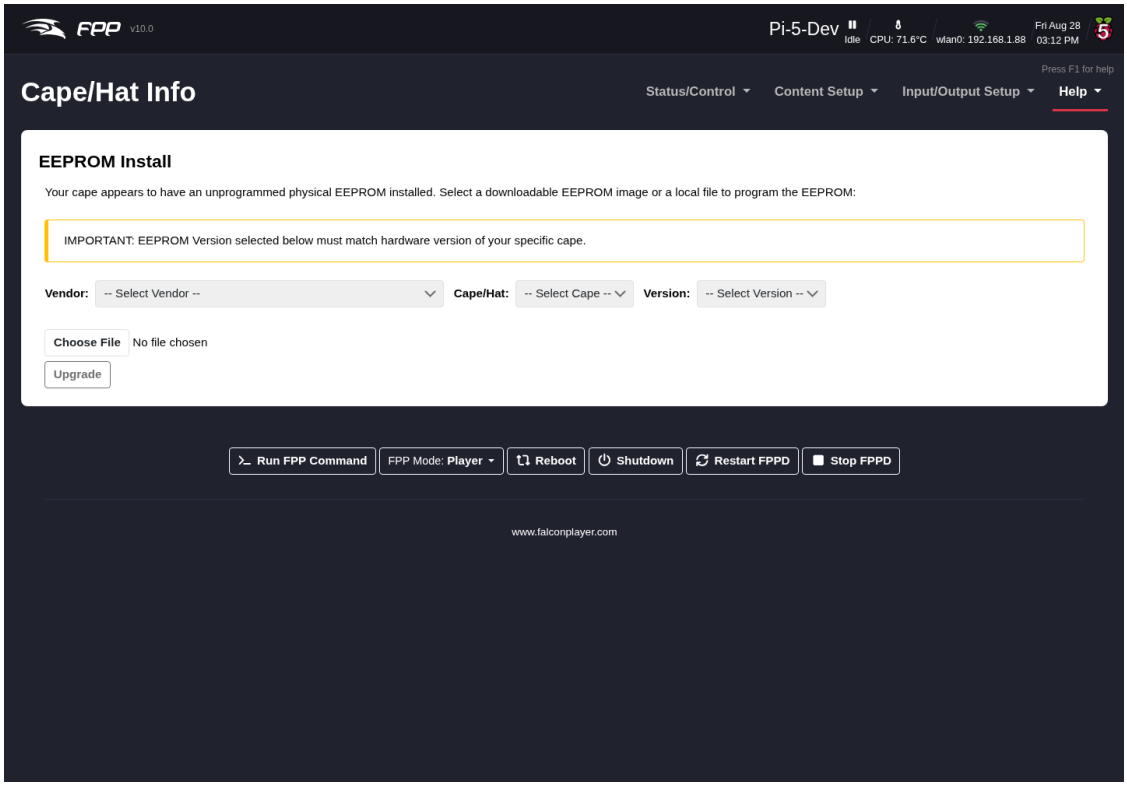
Upgrade methods: from GitHub (updates to the newest version of the current branch), **from another FPP** (matches another device's FPP version, not the OS; Advanced), or **from FPPOS** (an in-place upgrade of both FPP *and* OS without re-imaging — download the appropriate `.fppos` and upload it; requires being on at least 5.5-24 first). Major branch/OS changes otherwise require a re-image.

> **Warning:** Always take an **FPP Backup** before upgrading.

Below Version Info the page also shows **System Utilization**, **Player Stats** and **Disk Utilization** for the device.

31.2 Cape Info

If a cape/hat is installed, **Help → Cape Info** shows information about it and lets you upgrade or sign the EEPROM.



The Cape Info page.

The page has four tabs:

- **About** – **Name**, **Version**, **Serial Number**, **Designer**, **Licensed Outputs** (and licence status), **Output Driver**, and **Vendor Name/URL/E-mail**.
- **EEPROM Signature** – sign your EEPROM once you have an **Order number** and **License Key** for the pixel-string outputs (see Pixel Port Licensing). This is also where **Off-Line Signing** is done, for a device with no internet access (see Pixel Port Licensing → Off-Line Signing).
- **Voucher Redemption** – redeem a voucher from your vendor or shop.falconplayer.com to sign your EEPROM.
- **EEPROM Upgrade** – upgrade the cape's firmware/EEPROM from a file, or restore it from a previous backup. An option lets you **reset the**

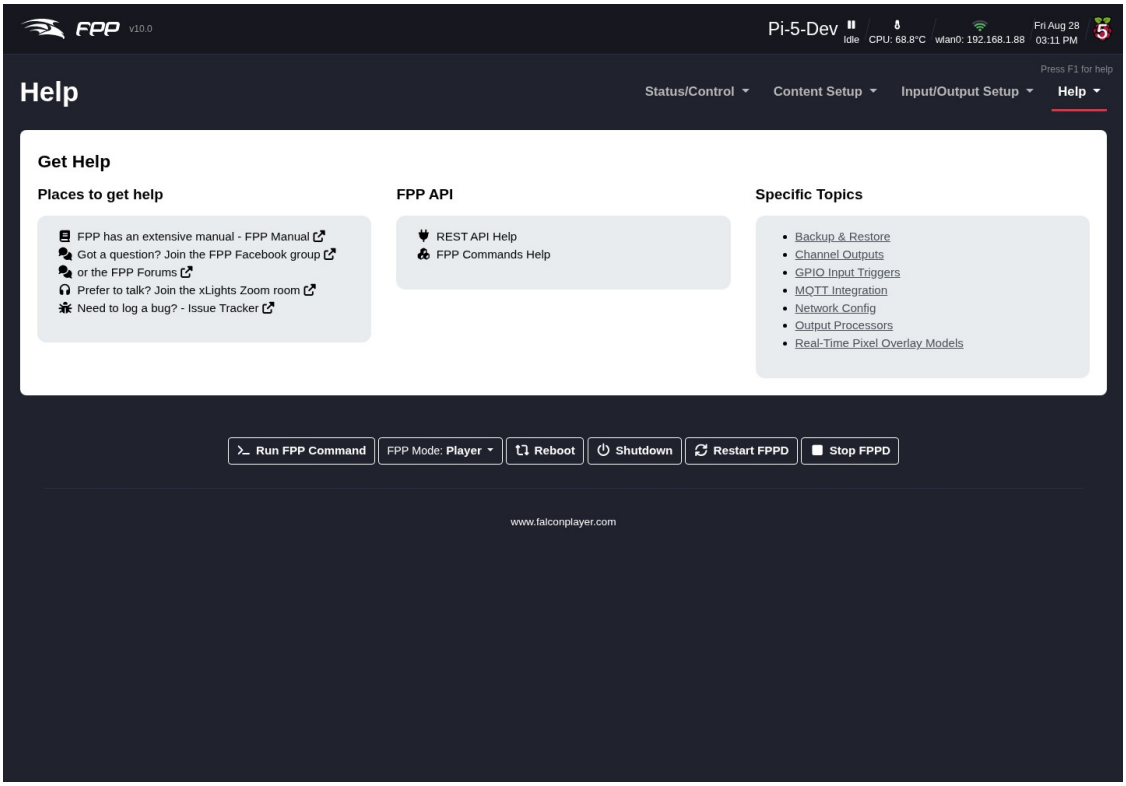
configuration to defaults as part of the restore. FPP streams the progress and keeps the dialog open until the flash finishes.

Warning: Do not power off or reboot the device while a cape firmware flash is running — an interrupted flash can leave the EEPROM unusable.

Note: This page only appears when a cape or hat is detected, so it is absent on a bare device. The fields shown vary with what the cape's EEPROM reports.

31.3 Get Help

Help → Get Help provides support resources and API references.



The Get Help page.

Places to get help:

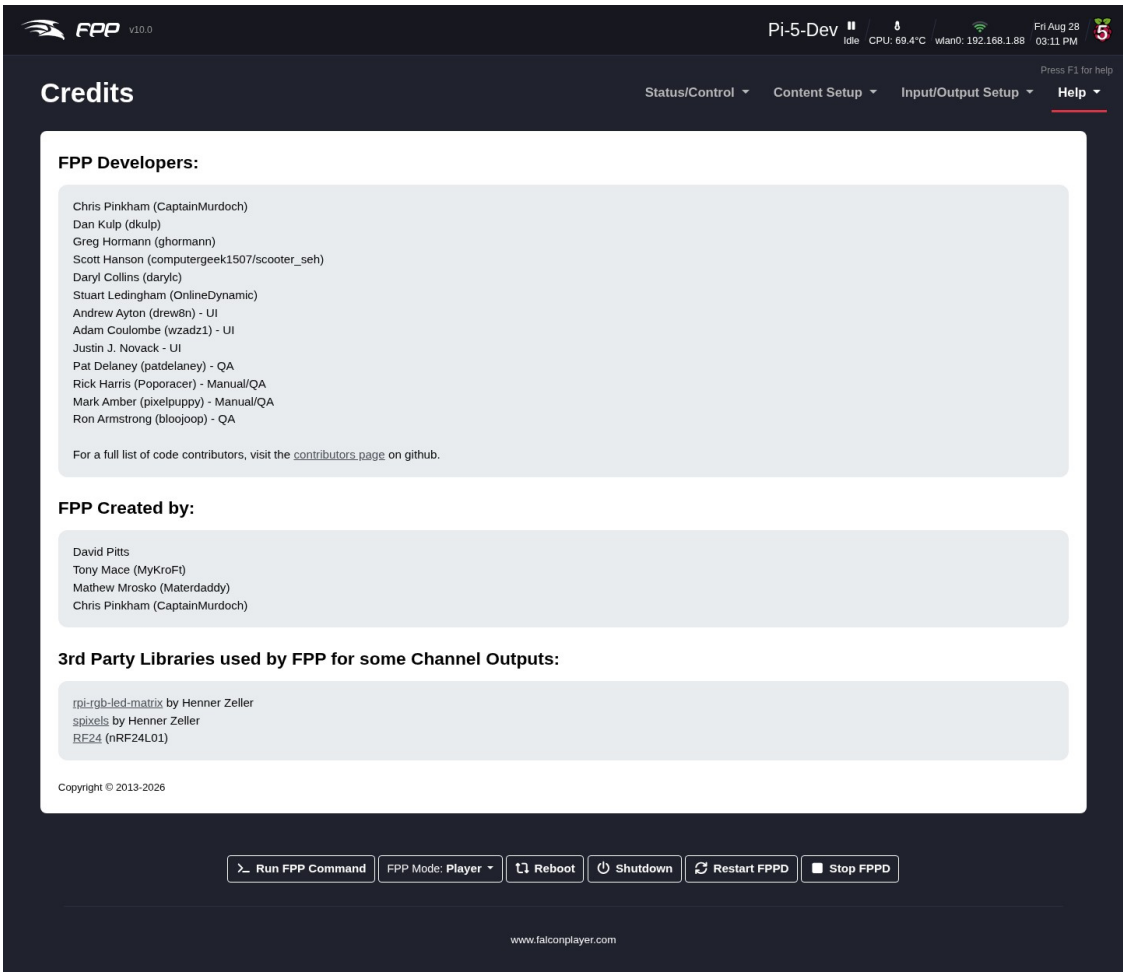
- **FPP Manual** – the current manual (offline copy at `https://falconchristmas.github.io/FPP_Manual.pdf`).
- **FPP Facebook Group** and the **Falcon Christmas Forums** – community help.
- **xLights Zoom Room** – often the fastest way to get any lighting question answered (not just xLights).

FPP API:

- **REST API Help** – a list of endpoints with a test facility, for plugin and software developers; click an endpoint to run it and see your device's output.
- **FPP Commands Help** – three sections: **Command Tester**, a **Command List** (all commands with typical arguments), and **MQTT Instructions**.

31.4 Credits and Donate

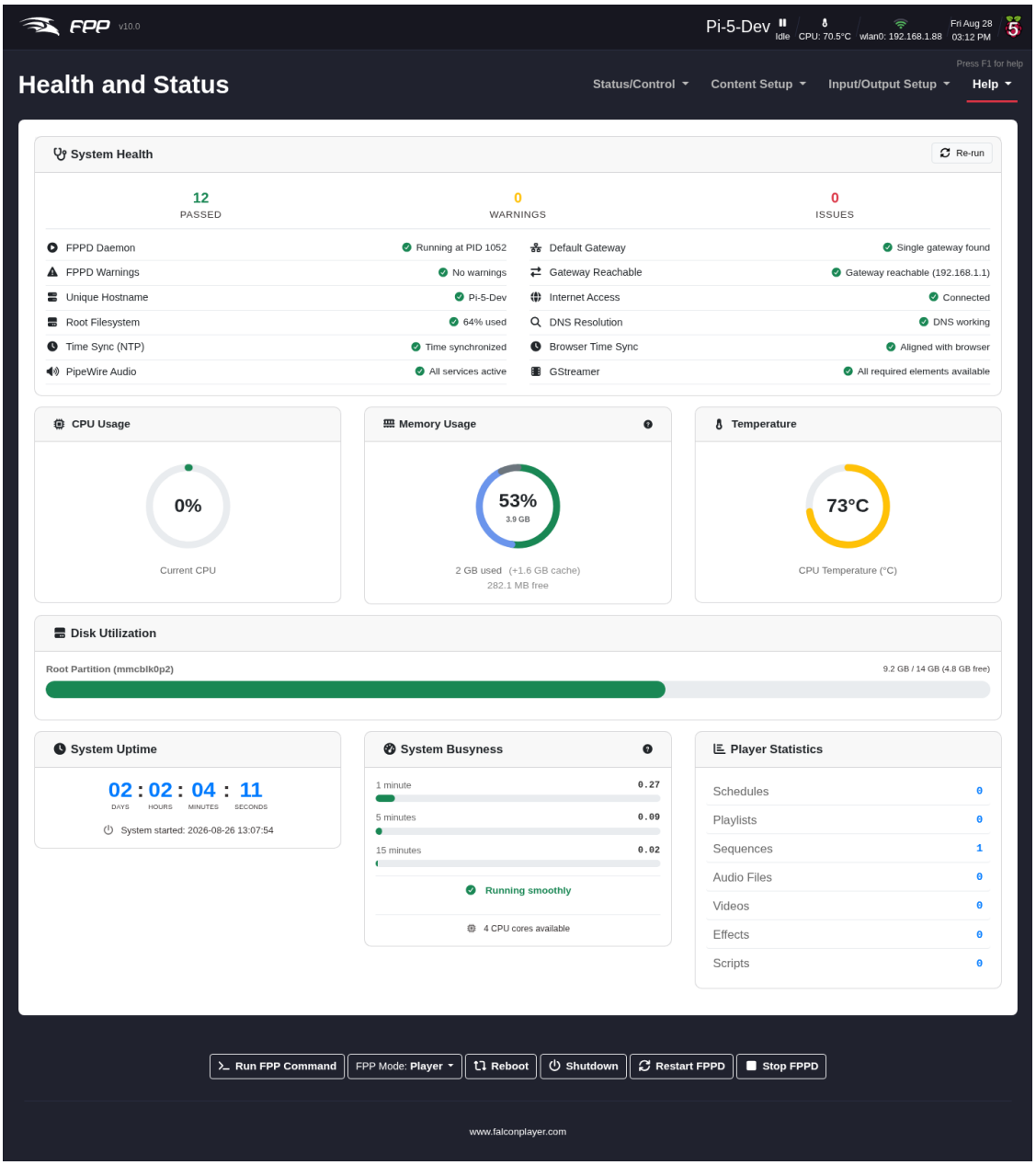
Help → Credits lists the people and projects behind FPP. If FPP has been useful, **Donate to FPP** links to support the developers.



The Credits page.

31.5 System Health Check

The **System Health Check** (**Help → System Health Check**) is a consolidated dashboard, new in FPP 10, that checks the device and surfaces anything wrong.



The System Health Check (Health and Status) page.

A **System Health** panel runs checks and summarises them as **Passed / Warnings / Issues**, including: **FPPD Daemon** and **FPPD Warnings**; **Unique Hostname** and **Root Filesystem** usage; **Time Sync (NTP)** and **Browser Time Sync**; **PipeWire Audio** and **GStreamer**; **Scheduler**; and network checks (**Default Gateway**, **Gateway Reachable**, **Internet Access**, **DNS Resolution**). Use **Re-run** to check again. Live panels below show **CPU Usage**, **Memory Usage**, **Temperature**, **Fan Monitoring** (on devices with a controllable fan — see *FPP Settings → System*), **Disk Utilization**, **System Uptime**, **System Busyness** and **Player Statistics**.

> **Tip:** This is the first page to open when something is not working — a red **Issue** or amber **Warning** usually points straight at the cause.

31.6 Troubleshooting Commands

Help → Troubleshooting Commands runs a set of read-only system commands and shows their output on one page — logs, process/service status, network information and configuration dumps — so you can inspect the device (and copy the output into a support request) without a shell.

Troubleshooting Commands

Download Support Bundle (Logs / Config / Troubleshooting)

Networking | Disk | Date / Time | Memory / CPU | USB | Audio | Media Backend | Midi | Video | OS, Kernel, and SD image | i2c | Processes | Boot | Git | GPIO | PHP | RPI Utils | Webserver

Set of tools for network troubleshooting

Interfaces

Command Description:
Command: ifconfig -a

```
eth0: flags=4099  mtu 1500
        ether d8:3a:dd:ac:b6:3f  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 105

lo: flags=73  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 785039  bytes 367473302 (350.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 785039  bytes 367473302 (350.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163  mtu 1500
        inet 192.168.1.88  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::da3a:ddff:feac:b641  prefixlen 64  scopeid 0x20
        ether d8:3a:dd:ac:b6:41  txqueuelen 1000  (Ethernet)
        RX packets 1410717  bytes 795721308 (758.8 MiB)
        RX errors 0  dropped 40  overruns 0  frame 0
        TX packets 980100  bytes 556952020 (531.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Back to top

Wired

Command Description:
Command: for interface in $(ifconfig | grep -oP "^\w+(?=:)" | egrep "eth|enx"); do echo " "; ethtool "$interface"; done;

```
Settings for eth0:
        Supported ports: [ TP    MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Supported pause frame use: Transmit-only
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Advertised pause frame use: Transmit-only
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Speed: Unknown!
        Duplex: Unknown! (255)
        Auto-negotiation: on
        master-slave cfg: preferred slave
        master-slave status: unknown
        Port: Twisted Pair
        PHYAD: 1
        Transceiver: external
        MDI-X: Unknown
        Supports Wake-on: ag
        Wake-on: d
        Link detected: no
```

Wireless

Command Description:
Command: iw dev; cat /proc/net/wireless

```
phy#0
        Unnamed/non-netdev interface
                wdev 0x2
                addr da:3a:dd:ac:b6:41
                type P2P-device
                txpower 31.00 dBm
        Interface wlan0
                ifindex 3
                wdev 0x1
                addr d8:3a:dd:ac:b6:41
                ssid HarrisChristmas
                type managed
                channel 10 (2457 MHz), width: 20 MHz, center1: 2457 MHz
                txpower 31.00 dBm
Inter-| sta-|   Quality        |   Discarded packets               | Missed | WE
```

The Troubleshooting Commands page.

The output is grouped into tabs by subject, so you can go straight to the area you are investigating:

Tab	What it shows
Networking	Interfaces, addresses, routes, DNS and connectivity
Disk	Mounted file systems, free space and partition layout
Date / Time	System clock, time zone and time-sync (chrony) status
Memory / CPU	Memory use, load and processor information
USB	Connected USB devices — the place to confirm a dongle or adapter was detected
Audio	Sound cards, ALSA/PipeWire devices and current audio routing
Media Backend	The media backend's state (PipeWire/GStreamer)
Midi	Detected MIDI devices
Video	Video outputs, displays and modes
OS, Kernel, and SD image	OS release, kernel version and the image FPP was installed from
i2c	Devices on the I2C bus — useful for OLED displays and capes
Processes	Running processes, including whether `fppd` is up
Boot	Boot configuration and boot-time messages
Git	The FPP source checkout's branch and status
GPIO	GPIO pin state and configuration
PHP	The PHP environment behind the web UI
RPI Utils	Raspberry Pi specific tools — throttling, voltage and firmware
Webserver	Apache configuration and error logs

> **Note:** Every command here is **read-only** — opening a tab inspects the device, it does not change anything.

31.6.1 Download Support Bundle

Rather than copying individual command output by hand, use **Download Support Bundle (Logs / Config / Troubleshooting)**. It packages the device's logs, its configuration, the troubleshooting command output and the **System Health Check** results into a single `.zip` you can attach to a forum post or bug report. FPP shows progress while the zip is generated, since gathering everything takes a few moments.

> **Tip:** Attaching a support bundle is the single most useful thing you can do when asking for help — it saves a long back-and-forth about versions, settings and log contents.
>
> **Note:** A support bundle contains your configuration and logs, which may include your host name, network details, Wi-Fi SSID, file names and email settings. Look through it before posting it somewhere public.

31.7 SSH Shell

Help → SSH Shell opens a browser-based shell to the device for advanced users.

31.8 General troubleshooting tips

- **Raise log levels** for the relevant subsystem on FPP Settings → Logging, reproduce the problem, then read the logs from *File Manager → Logs*.
- **No output?** Check Channel Outputs are enabled and saved, that FPPD was restarted after changes, and use Display Testing to isolate wiring from configuration.
- **Audio/video issues?** Check the **PipeWire Audio** and **GStreamer** health checks and the Audio/Video settings.
- **Sync problems?** Confirm **Send MultiSync Packets** on the player, matching sequences on remotes, and a stable (ideally wired) network.
- **Cannot reach FPP?** Most often a network/DNS problem — check the address, host name, and the Network settings.

32 Appendix B: Protocols, Ports and the API

32.1 Output and input protocols

Protocol	Transport	Typical use	In/Out
E1.31 (sACN)	UDP multicast/unicast	Ethernet pixel/DMX controllers	Out & In
DDP	UDP unicast	Efficient Ethernet pixel control	Out & In

Protocol	Transport	Typical use	In/Out
		(e.g. Falcon, WLED)	
ArtNet	UDP	Stage/DMX ecosystems	Out & In
KiNet	UDP	Color Kinetics gear	Out
DMX	Serial (USB)	DMX fixtures via USB dongle	Out
Pixelnet	Serial	Legacy pixel control	Out
Renard	Serial	Legacy DIY controllers	Out

Direct **WS281x pixel strings** and **LED matrix panels** are driven through a cape/hat rather than a network protocol (see *Capes* appendix).

32.2 Common network ports

Port	Protocol	Purpose
80	HTTP	FPP web UI and REST API
5568	E1.31 (sACN)	Streaming ACN lighting data
6454	ArtNet	ArtNet lighting data
4048	DDP	Distributed Display Protocol
32320 / 32328	FPP MultiSync	Player→remote sync and discovery
1883	MQTT	Broker connection (configurable)
5004	AES67 (RTP)	Uncompressed audio-over-IP (default, per instance)
5005	Opus RTP	Compressed audio streaming (default, per instance)
9875	SAP	AES67 stream announcement/discovery
319 / 320	PTP (IEEE 1588)	AES67 clock synchronisation

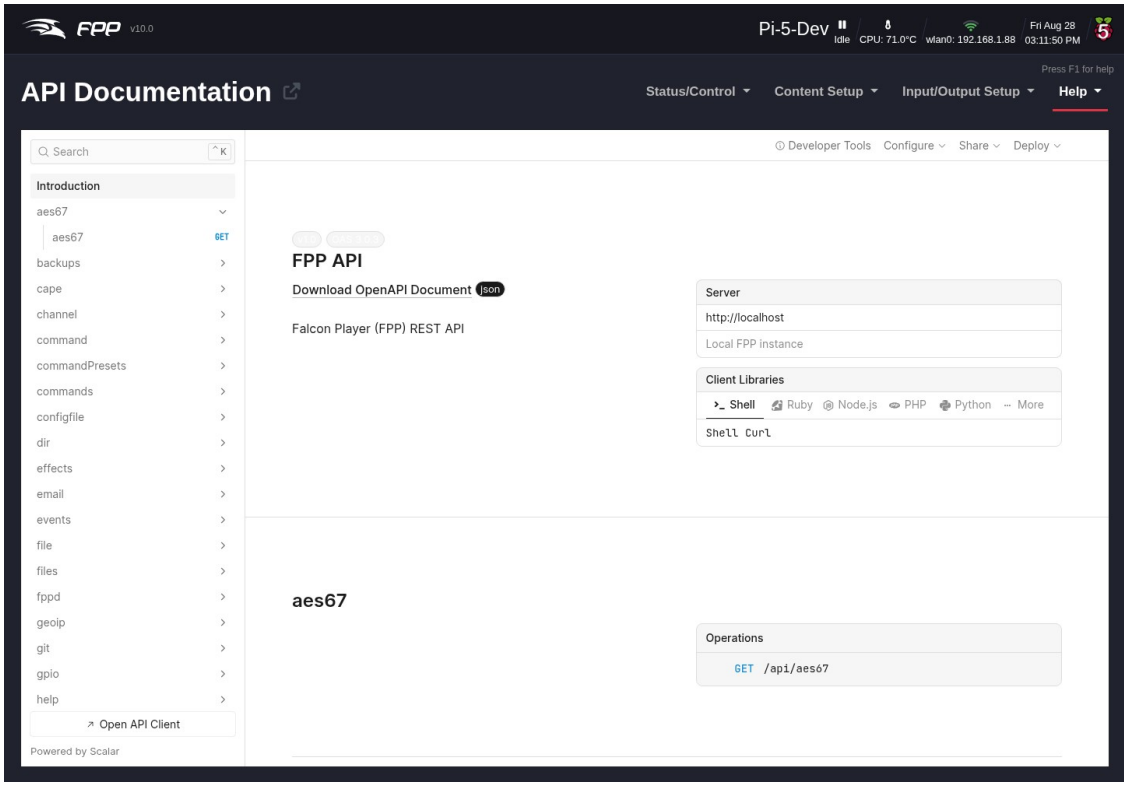
(The audio streaming ports are per instance and are set on the AES67 and Opus RTP pages reached from FPP Settings → Audio/Video; PTP's domain and clock role are set there too.)

32.3 The FPP API

FPP exposes a **REST/HTTP API** on port 80 for status and control — starting and stopping playlists, reading status, setting channel data, controlling overlay models, and much more. The same actions are also available through **Commands**, **MQTT** and the **Scheduler**, but the API lets you drive FPP directly from scripts, home-automation platforms, or your own applications.

32.3.1 The interactive API explorer

New in FPP 10 is a built-in, interactive **API explorer**. Open it from **Help → REST API**, or browse directly to `http://<fpp-host>/api/`.



The interactive API explorer (Help → REST API).

The explorer is a live, browsable reference for every endpoint on *this* device:

- Endpoints are grouped into sections by area — **player**, **playlist**, **playlists**, **command**, **commandPresets**, **channel**, **models**, **overlays**, **media**, **files**, **network**, **pipewire**, **gpio**, **backups**, **plugin**, and more.
- Selecting an endpoint shows its HTTP method and path, its parameters, and the request and response **schemas** with example values.

- A built-in client lets you **send a request to the device and see the response inline** — a quick way to discover exactly what a call returns before you wire it into a script.

Note: The FPP API has **no authentication**. Requests you send from the explorer (or anywhere else) act on the device immediately, so take care with `POST`/`PUT`/`DELETE` calls — especially against a player that is running a live show.

Tip: The raw specification is served at `http://<fpp-host>/api/openapi.json` (OpenAPI 3.0.3). You can import that URL into tools such as Postman or Insomnia, or feed it to a client-code generator. Endpoints added by installed **plugins** are merged into the live spec automatically, under `/plugin/<name>/`.

32.4 MQTT

When configured (FPP Settings → MQTT), FPP connects to an MQTT broker and both publishes status and subscribes to command topics under a configurable prefix, making it easy to integrate with home-automation platforms such as Home Assistant or Node-RED. See the FPP MQTT documentation for the full topic list.

32.5 Further resources

- **FPP GitHub:** `https://github.com/FalconChristmas/fpp`
- **FPP releases/downloads:** `https://github.com/FalconChristmas/fpp/releases`
- **xLights** (sequencing): `https://xlights.org`
- **Community support:** the FPP/xLights forums and Facebook groups.

33 Glossary

Absolute Channel – A numbering system that uniquely identifies every channel, simply 1 to the last channel needed. Displays can have very large channel counts, which can be hard to manage.

BBB – BeagleBone Black; commonly used as shorthand for the whole BeagleBone series of single-board computers.

Broadcast – Sending the same network data to every device on the network.

btrfs – A Linux filesystem with advanced features including compression, at the cost of more CPU usage.

Channel – An identifier for a component in a lighting display — commonly a pixel’s colour/brightness, but also DMX channels for other devices.

DHCP – Dynamic Host Configuration Protocol; automatically assigns IP addresses to devices. Addresses are not permanent. Requires a DHCP server (most routers have one).

DNS – Domain Name System; lets you use friendly host names instead of IP addresses. Requires a DNS server configured in your network settings (most routers provide one).

E1.31 – A network protocol for transmitting DMX data; the most common protocol in the animated-lighting hobby.

Effect – A small sequence, usually for one model, used to overwrite the data a sequence is playing; triggered by an event or manually.

eMMC – On-board flash memory on the BeagleBone Black/Green; the FPP OS can be stored on it.

eth0 – The wired network interface of the FPP.

Event – A sequence or script run when a trigger fires, or triggered manually from the FPP interface.

ext4 – The standard Linux filesystem.

FPP – Falcon Player; a widely used player and operating system for animated-lighting displays.

FSEQ file – The standard raw data file format that tells controllers how to illuminate each light or DMX channel.

Gateway – The IP address a device sends data to when it does not know how to route it — usually your router, or an FPP bridging two subnets.

Git – A distributed version-control system; the core of GitHub.

Host Name – A friendly name for a device, used instead of its IP address.

IP Address – A numerical device address: four parts (0–255) separated by dots.

Multicast – Sending the same data only to devices that requested it.

MultiSync – FPP's mechanism to discover relevant FPP instances in a Player/Remote configuration and keep them synchronised.

Netmask – A mask defining a subnet's size; typically `255.255.255.0` on home networks.

Network – A group of devices connected to share data.

NTP – Network Time Protocol; keeps system clocks synchronised to accurate time.

P10/P5 panel – Display panels (typically 6″×12″) whose pixel spacing matches the type (P5 = 5 mm apart). Combined into larger matrices, often used for “Tune To” signs.

Pixel – An individually addressable LED whose colour is independent of the other pixels in the string.

Player/Remote – An FPP method to synchronise several FPP devices via small sync signals — useful for large or widespread displays where Ethernet cabling is impractical.

Playlist – An ordered list of items FPP plays to control lights and props.

Plugin – A component that adds functionality to FPP.

Port – The physical connection point on a controller for a pixel string.

Raspberry Pi – A single-board computer used to play sequences or act as a controller interface.

Real Time Clock (RTC) – A component with an accurate timing crystal that keeps time when there is no network.

SBC – Single-Board Computer; the Raspberry Pi and BeagleBone series are the most common in this hobby.

Script – A small program that performs a specific function in FPP.

SSH – Secure Shell; a secure command-line way to access a device (for advanced users).

Subnet – A portion of a larger network, usually the first three segments of an IP address (e.g. `192.168.0.x`, where `192.168.0` is the subnet).

Tethering – Connecting two devices directly (via Ethernet, USB or Wi-Fi) with no router or switch.

UI – User Interface; how you interact with a device or program. FPP’s UI is web-based.

Unicast – Sending data individually to each target device; usually more efficient than Multicast or Broadcast unless the same data goes to many devices.

Universe/Channel notation – A numbering system that groups channels into user-defined **universes** (up to 512 channels each), often easier to manage than absolute numbering. Universes can have gaps and need not start at 1.

uSD card – A micro-SD card; in FPP it usually holds the operating system and related files.

wlan0 – The Wi-Fi (wireless) network interface of the FPP.

WPA Pre-Shared Key – The password for a Wi-Fi network.

WPA SSID – The technical name for a wireless network.

34 Pixel Port Licensing

Two pixel-output protocols need a licence to use their advanced features: **DPIPixels** and **BBBStrings48**. No other use of FPP requires a licence. The advanced features are **more than 50 pixels per port** and support for **remote receivers**.

> **Note:** If you are not using these protocols, or do not need their advanced functionality, you do **not** need a licence. There is a licensing FAQ at `https://shop.falconplayer.com/faqs/`.

34.1 Do I need a licence?

34.1.1 Raspberry Pi based controllers

- Using a Pi hat for **up to 3 ports** (or pixels straight on the GPIO pins) with the **PiHat** protocol — **no licence**. (This uses the Pi's PWM, so on-board audio is unavailable; use a USB audio adapter such as the SoundBlaster Play 3 — a very common setup.)
- Using a hat that drives **more than 3 ports**, or wanting on-board audio with a 2-port hat — **licence required**, *unless* you use **50 pixels or fewer per port**.

34.1.2 BeagleBone based controllers

- **50 pixels or fewer per port** — no licence (but Differential Remotes cannot be used without one).
- **More than 50 pixels on any port**, or using remote receivers — **licence required**.

34.2 Getting a licence

The vendor you bought your controller from may have supplied a **voucher** — if so, see *Redeeming a Voucher* below. Otherwise decide how many ports you need, go to `https://shop.falconplayer.com/`, choose the key for that number of ports, and add it to your cart (the *How To Guides* at the top of the shop are helpful; click a key for more detail).

> **Note:** For a **DIY board**, choose the appropriate key and enter the relevant **coupon code** at checkout to get the licence for free.

34.2.1 Purchasing a licence

Add the licence to your cart and pay with PayPal/Venmo, or **Proceed to checkout** for other methods. On completion you receive an **Order Number** and **License Key** on the confirmation page and by email. You can then sign your EEPROM as below.

34.3 Applying a licence key

There are two ways to apply a licence:

- **On-Line Signing** – easiest; use it if your FPP has internet access.
- **Off-Line Signing** – a few more steps, for when the FPP network has no internet (temporarily connecting the FPP to the internet just to sign may be easier).

34.3.1 On-Line Signing

1. Open **Input/Output Setup → Channel Outputs → Pixel Strings**. If no cape is detected, install the correct **Virtual EEPROM** first (see *Channel Outputs → Configuring the Virtual EEPROM*).
2. Click the **Cape Info** link in the blue banner.
3. On the Cape Info page, open the **EEPROM Signature** tab.
4. Enter your **Order Number** and **License Key** and click **Sign EEPROM**.
5. When it shows *Signing Complete. Please Reboot*, click **Close**, then **Reboot** and confirm.

Once signed, the Pixel Strings tab shows the cape type as its name and the blue banner disappears.

34.3.2 Off-Line Signing

1. As above, open **Channel Outputs → Pixel Strings** (installing the Virtual EEPROM if needed) and click the **Cape Info** link.
2. Open the **Offline Signing** tab, enter your **Order Number** and **License Key**, and click **Download Offline Signing Packet**. The downloaded file's name includes the device's host name — so keep host names unique to match the right file to the right device.
3. Move that file to a computer with internet access (physically move the computer, or copy via USB), go to `https://shop.falconplayer.com/offline-signing/`, enter the same Order Number and License Key, select the signing file, and click **Sign**. A **signed** EEPROM file is downloaded.
4. Bring the signed file back to the computer connected to the FPP, return to **Cape Info → Offline Signing**, choose the signed file (its name contains *signed* and the host name), and click **Upload Signed Packet**.

After a few seconds it shows your EEPROM signature details.

34.4 Redeeming a voucher

A vendor may supply a voucher when you buy a controller (this does **not** apply to Falcon or KulpLights controllers, which are pre-signed). There are two ways to redeem it:

- **FPP Redemption** *(easiest, needs internet on the FPP)* – get the licence and sign the EEPROM directly from the FPP interface: go to **Help → Cape Info**, open the **Voucher Redemption** tab, fill in the details, and click **Redeem Voucher**.
- **shop.falcon Redemption** – log in at `https://shop.falconplayer.com/`, redeem the voucher there to get your key, then apply the key on your FPP device as in *Applying a Licence Key*.

  Screenshots pending — cape hardware required. The signing screens appear on the Cape Info and Pixel Strings pages, which need a cape/Virtual EEPROM present; these will be captured on a cape-enabled system.

35 Advanced Options

This chapter covers material beyond the basic setup and configuration. For networking topics — subnetting, static routes, and choosing a show network layout — see the Networking chapters.

35.1 Projector Control

FPP can control a projector (for example to power it on before the show and off afterwards) using scheduled **Commands**, **scripts**, GPIO, or network/serial control depending on the projector. Combine a scheduled *Command* entry a few minutes before the show with the appropriate projector control method.

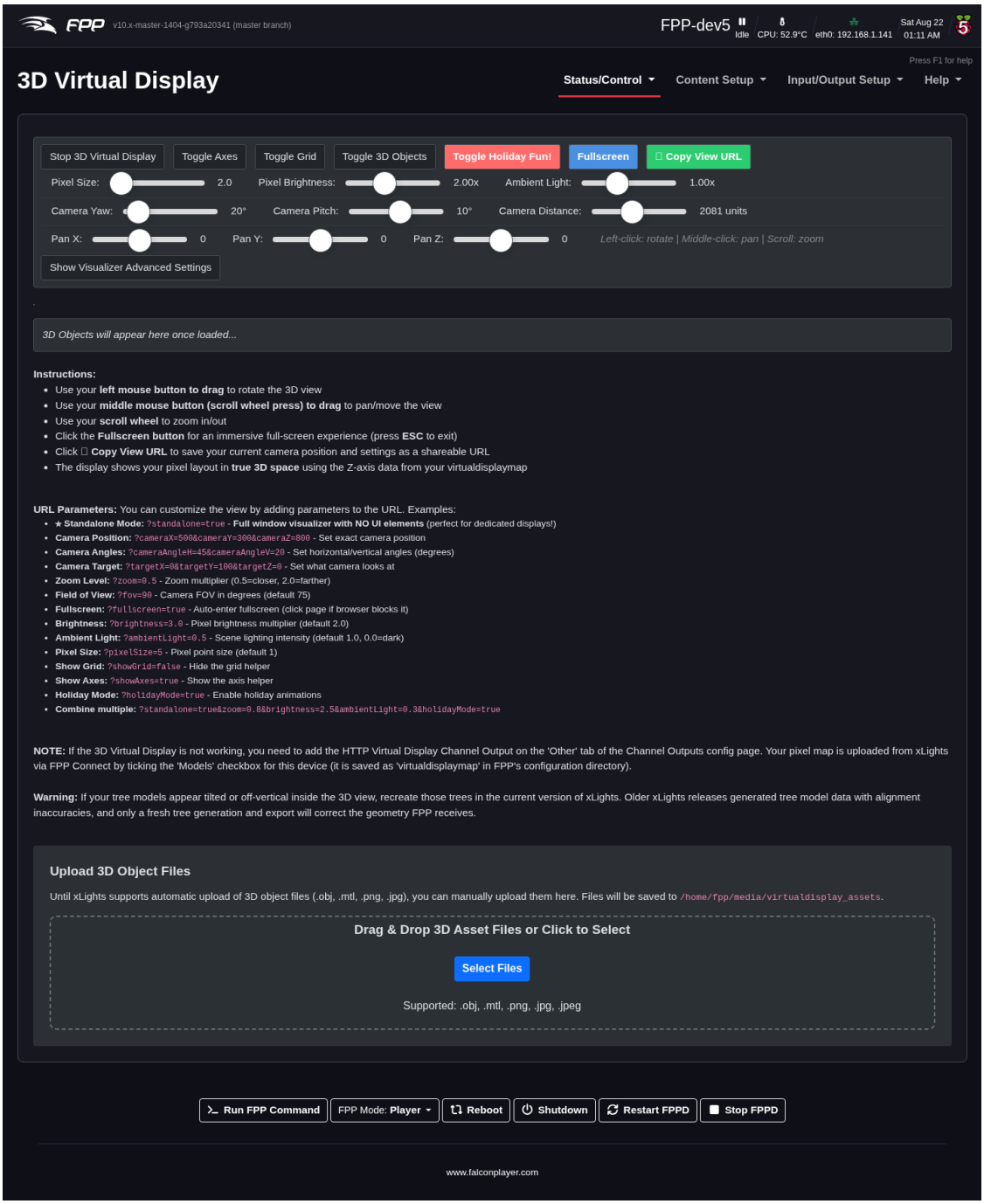
35.2 Plugin Development

Plugins extend FPP with new pages, commands, playlist entry types and outputs. The **Plugin Manager** includes a **Template Plugin** to help authors with the required structure; a plugin provides a `pluginInfo.json` describing itself and hooks into the relevant FPP menus. Refer to the FPP developer documentation in the repository for the plugin API and packaging details.

> **Note:** This chapter summarises the most-referenced advanced topics. The FPP forums, the xLights Zoom Room, and the developer documentation in the FPP GitHub repository are the best resources for deeper or edge-case configurations.

36 3D Virtual Display

New in FPP 10, the **3D Virtual Display** renders your entire show as a live, true-3D model in the browser. It reads the geometry your props actually have — including their depth (the Z axis) — and streams the sequenced pixel colours to it in real time, so you can preview a running sequence from any angle without any lights connected. It is ideal for testing at your desk, for demonstrating a show, or for driving a dedicated on-screen display.



The 3D Virtual Display showing a running sequence in true 3D space.

36.12D vs. 3D virtual display

FPP has long offered a **2D Virtual Display**, which groups pixels by their on-screen X/Y position — pixels that sit at the same place on a flat plane share a colour, so depth is lost. The new 3D display treats **every pixel independently** using the Z-axis data from your layout, so pixels at the same X/Y but different depth (for example the front and back faces of a

mega-tree) light up separately. The result is a faithful three-dimensional preview you can orbit around.

Both displays are driven from the same source: the **virtualdisplaymap** that xLights uploads to FPP.

36.1.1 Using the 2D Virtual Display

The 2D display is enabled the same way as the 3D one, but with a different output type. Open **Input/Output Setup → Channel Outputs → Virtuals**, add an **HTTP Virtual Display** output, set **Enabled** to Yes, then **Save** and **Restart FPPD**. A **2D Virtual Display** entry then appears in the **Status/Control** menu; it is hidden whenever the output is disabled.

The 2D view shows your display as a flat plane of pixels and needs no WebGL, so it is the lighter option on an older browser or a low-powered device, and it is perfectly adequate for confirming that a sequence is playing and that the right props are lighting. Use the 3D display when depth matters.

> **Note:** You can enable both outputs at once — each adds its own menu entry — but each one costs the device a little CPU while a browser is watching it. Turn off whichever you are not using before running a show.

36.2 Requirements

The 3D display needs two things in place before its menu link appears:

1. **A layout map from xLights.** In xLights, open **FPP Connect**, tick the **Models** checkbox for this FPP device, and upload. This saves a `virtualdisplaymap` into FPP's configuration directory describing the 3D position of every pixel.
2. **The HTTP Virtual Display 3D output enabled** (see below). The channel count is detected automatically from the uploaded map, so you do not set it by hand.

36.3 Enabling the output

Open **Input/Output Setup → Channel Outputs** and select the **Virtuals** tab. Add an **HTTP Virtual Display 3D** output and enable it:

- **Enabled** – Yes.
- **Channel Count** – shown as **Auto**; it is taken from the virtualdisplaymap.
- **Update Interval (ms)** – how often the display is refreshed. The default is **25 ms** (≈ 40 fps); lower is smoother, higher is lighter on the device.

The output uses a fixed network port (32329) internally; you do not need to configure it. Click **Save** and, when prompted, **Restart FPPD**.

Once the output is enabled, a **3D Virtual Display** entry appears in the **Status/Control** menu (alongside **2D Virtual Display**, which shows only when the 2D output is enabled).

36.4 Navigating the view

The display uses standard orbit controls:

- **Left mouse button – drag** to rotate the view around your display.
- **Middle mouse button (scroll-wheel press) – drag** to pan/move the view.
- **Scroll wheel** to zoom in and out.
- **Fullscreen button** for an immersive full-screen view; press **ESC** to exit.
- 📋 **Copy View URL** captures your current camera position and settings as a shareable link — paste it to reopen the display at exactly that angle and zoom.

Tip: If your tree models appear tilted or off-vertical in the 3D view, recreate those trees in the current version of xLights and re-upload. Older xLights releases generated tree geometry with alignment inaccuracies that only a fresh export will correct.

36.5 URL parameters

Because the view is a normal web page, you can preset it with query parameters — handy for a kiosk or a saved bookmark. Append them to the page URL (combine several with `&`):

Parameter	Effect
`standalone=true`	**Full-window visualiser with no FPP UI** — perfect for a dedicated display.
`cameraX`, `cameraY`, `cameraZ`	Set the exact camera position.
`cameraAngleH`, `cameraAngleV`	Set the horizontal/vertical camera angles (degrees).
`targetX`, `targetY`, `targetZ`	Set the point the camera looks at.
`zoom`	Zoom multiplier (`0.5` = closer, `2.0` = farther).
`fov`	Camera field of view in degrees (default 75).
`fullscreen=true`	Enter fullscreen automatically

Parameter	Effect
	(click the page if the browser blocks it).
`brightness`	Pixel brightness multiplier (default 2.0).
`ambientLight`	Scene lighting intensity (default 1.0; `0.0` = dark).
`pixelSize`	Point size of each pixel (default 1).
`showGrid=false`	Hide the ground grid.
`showAxes=true`	Show the X/Y/Z axis helper.
`holidayMode=true`	Enable holiday animations.

For example, a dedicated display might use:

```
virtualdisplaywrapper3d.php?
standalone=true&zoom=0.8&brightness=2.5&ambientLight=0.3
```

36.6 Uploading 3D object files

The page includes an **Upload 3D Object Files** area for adding scenery or prop meshes to the scene — for example a model of your house or a static decoration. Drag and drop (or select) `.obj`, `.mtl`, `.png`, `.jpg`/`.jpeg` files; they are saved to `/home/fpp/media/virtualdisplay_assets` on the device.

> **Note:** Automatic upload of 3D object files from xLights is not yet available, so these assets are added manually here. The pixel layout itself always comes from the xLights **Models** upload described above.

36.7 Troubleshooting

- **The menu link is missing.** Enable the **HTTP Virtual Display 3D** output on the **Virtuals** tab and restart FPPD; the link appears only when the output is on.
- **Nothing lights up / the display is empty.** Confirm you uploaded **Models** from xLights FPP Connect (this creates the `virtualdisplaymap`) and that a sequence is playing. Do a hard refresh (Ctrl+Shift+R) after enabling the output.
- **Pixels look grouped or flat.** That is the 2D display; make sure you opened **3D Virtual Display**, not 2D.

37 Networking Overview

For your devices to communicate, the network must be configured correctly. You need to know your **home router's IP address** and **subnet** — usually `192.168.0.1` or `192.168.1.1` (check a label on the router, or run `ipconfig` on Windows / `ifconfig` on Mac to see the default gateway).

An IP address is four groups of numbers (0–255) separated by dots, e.g. `192.168.0.1`. On most home networks the first three groups are the **subnet** (here `192.168.0`) and the last group is the **host** (here `1`). Devices can communicate **directly only with other devices in the same subnet**; reaching a different subnet requires telling the systems how to route between them (see Static Routes and Proxy Settings).

Some key rules that recur throughout this manual:

- Give show controllers **static** addresses (or router reservations) so they do not change.
- If a device uses **two interfaces**, put them on **different subnets** and give a **gateway to only one** (normally the home-facing interface).
- All FPP/controller **host names must be unique**.

37.1 Choosing a show network layout

There are four common "show network" arrangements, each covered in its own chapter. Each has trade-offs, and you can combine them as needed. When numbering devices, it is suggested to use the higher end of the address range to avoid clashing with DHCP-assigned addresses.

- **Standalone** – a single FPP device drives the whole display directly (via a cape/hat or a locally attached controller) with no show network. Simplest; suited to small displays.
- **Home Network** – FPP and the controllers sit on your existing home network. Easy to reach for configuration and updates, and good for testing, but show traffic shares the home network.
- **Separate Network** – the controllers (and often the FPP's second interface) live on their own isolated network, keeping heavy pixel traffic off the home network.
- **Player/Remote (MultiSync)** – one FPP **player** synchronises any number of FPP **remotes** with small sync packets, so each remote drives a nearby section of the display. Ideal for large or widespread displays where long cable runs are impractical. Each remote needs a copy of the sequences; the player holds the playlists and schedules. See the MultiSync chapter for the full page reference, including how to choose between Multicast, Broadcast and Unicast sync.

37.2 Static Routes

When you need a computer to reach a controller on a different subnet without FPP proxying, you can add a **static route** on the computer (or router) telling it to reach the controller's subnet via the FPP device's address. Note that static routes on Macs are not persistent across reboots — FPP's **Proxy Settings** are usually a better solution (see Proxy Settings).

See Configuring a Static Route for step-by-step instructions for both the router and computer methods.

38 Standalone

A single FPP device drives the whole display directly — through a cape/hat or a locally attached controller — with **no separate show network** at all. This is the simplest network layout, and the one most small displays use.

- No second FPP or remote controller to keep in sync; the device that plays the show is the only device on the network.
- FPP can create its own Wi-Fi access point (**tethering**, see Network → Tethering) so you can reach the device's web UI to configure it in the field, even with no router nearby.
- Because there is no show-only network segment, there is nothing extra to troubleshoot beyond the device's own connection to your home network (or none at all, if you only ever configure it over tethering).

  Tip: Standalone is also a good layout to fall back to while troubleshooting — if a MultiSync or separate-network setup is misbehaving, confirm the same show plays correctly with the controllers wired directly to a single FPP first.

See Networking → Overview for how this compares with the other layouts.

39 Home Network

FPP and its controllers sit on your **existing home network**, alongside your computers, phones and router — no separate hardware or extra configuration beyond the Network page itself.

- Easiest layout to reach for configuration, software updates, and using xLights' **FPP Connect** to upload shows.
- Good for **testing** a display before it moves to its final location, since every device is already reachable from your computer.
- Show traffic (E1.31/ArtNet/DDP, MultiSync) shares bandwidth with everything else on the home network — usually not a problem for

small-to-medium displays, but worth watching if the network is busy or the display is large.

- Give each show controller a **static address** (or a reservation in your router), so it does not change and break saved configuration — see Networking → Overview.

Tip: Many builders start here — it is the fastest way to get a display running, and a fine permanent choice for a small or wired setup — and move to a Separate Network only once the show network's traffic or size calls for it.

40 Separate Network

The controllers — and often the FPP device's second network interface — live on their **own isolated network**, keeping heavy pixel traffic (E1.31/ArtNet/DDP) off your home network entirely.

- FPP typically **bridges** the two networks: one interface faces your home network (for configuration/updates), the other faces the show network (the controllers). Keep them on **different subnets**, and give a gateway only to the home-facing interface — see Networking → Overview.
- FPP can run a **DHCP server** on the show-network interface so controllers get addresses automatically — see the **DHCP Server** option under Network → Interface Settings.
- Reach a controller's own web UI from your computer through FPP's Proxy Settings rather than adding static routes — the simplest way to configure a controller that only exists on the isolated show network. A manual static route on the computer or router also works but, notably, is not persistent across reboots on a Mac.
- All FPP/controller **host names must still be unique**, even though the two networks are otherwise isolated from each other.

This layout suits larger or permanent displays where pixel traffic could otherwise saturate a home network, at the cost of a little more setup than Home Network.

41 The PipeWire Audio & Video Pipeline

FPP 10 replaces the simple audio/video handling of the 9.x series with a flexible pipeline built on **PipeWire** (the audio/video graph), **WirePlumber** (which links the graph together) and **GStreamer** (which decodes media and handles network streaming). This lets FPP route one audio or video source to **multiple simultaneous outputs**, each with its own volume, delay,

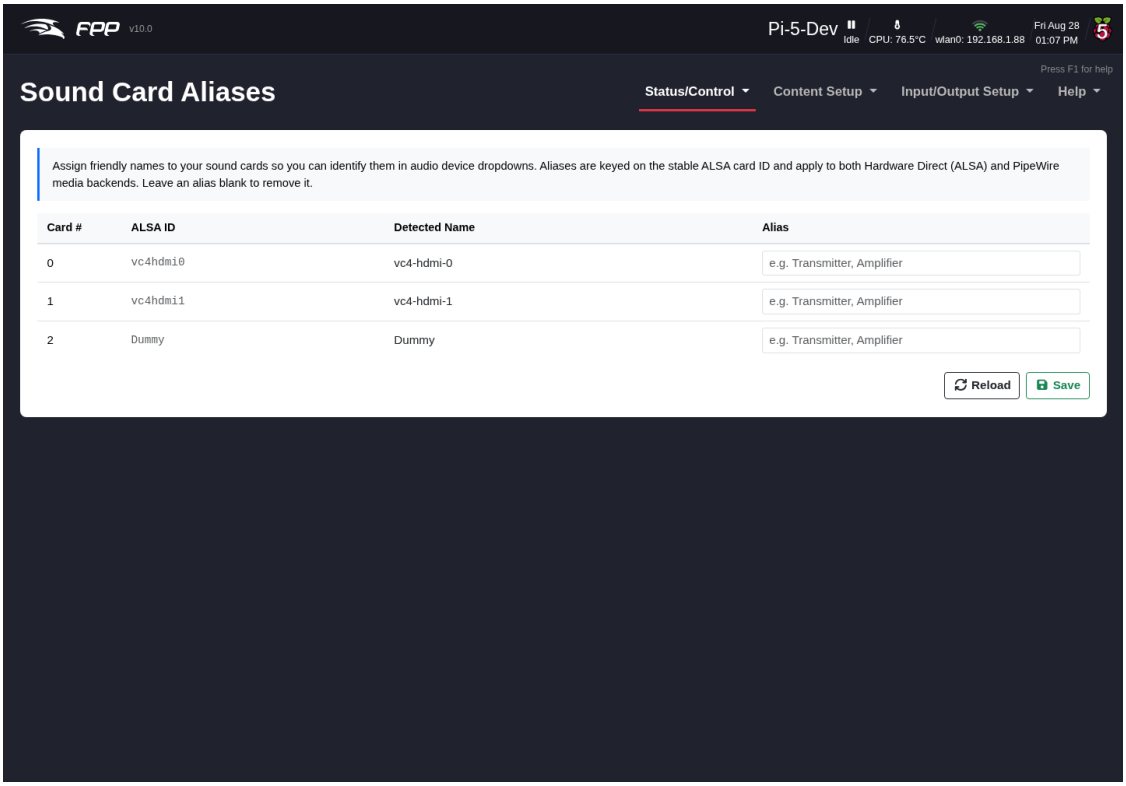
equalisation and format — and to send and receive audio and video over the network.

Everything in this chapter is reached from the buttons on **FPP Settings → Audio/Video** (with **Media Backend** set to **PipeWire (Advanced)**). Each page runs live against the PipeWire graph; most have **Save** and **Save & Apply** buttons (Apply regenerates the pipeline and re-links it).

> **Key concepts.** An **output group** is a “combine sink” that fans one signal out to several destinations. An **input group / mix bus** collects one or more sources and routes them to output groups. The **routing matrix** connects inputs to outputs. These apply to both **audio** and **video**.

41.1 Sound Card Aliases

Audio/Video → Configure Sound Card Aliases.

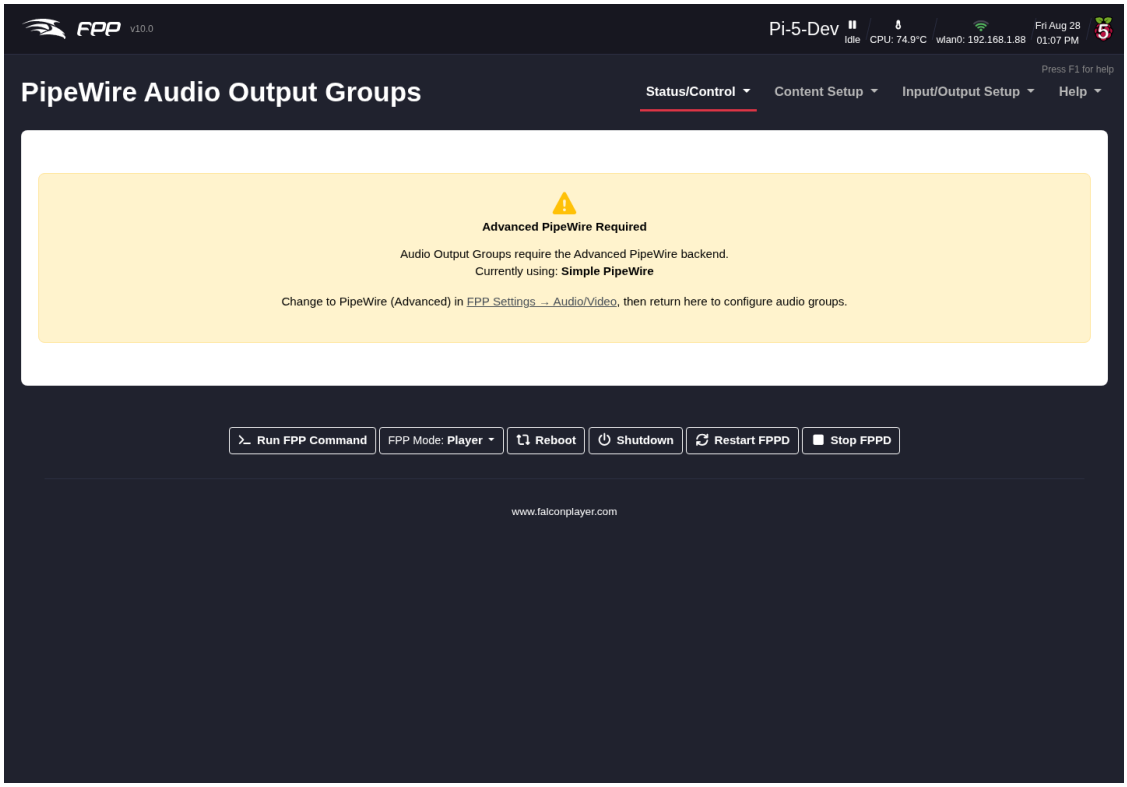


Sound Card Aliases.

Audio devices have long, cryptic system names (for example `usb-Creative_Technology_Ltd_Sound_Blaster_Play__3...`). This page lets you give each card a short, friendly **alias** (e.g. “Stu Blaster”) that is then used throughout the audio pages, making groups much easier to configure.

41.2 Audio Output Groups

Audio/Video → Configure Output Audio Groups.



PipeWire Audio Output Groups, with per-card channel mapping, delay and parametric EQ.

An **audio output group** is a virtual sink that plays the same audio through several sound cards at once. Click **Add Group**, name it, and add the sound cards that belong to it. For the group you set:

- **# of Channels Group Accepts** – e.g. *2ch (Stereo)* or *8ch (7.1)*.
- **Latency Compensation** – align outputs that have different latencies.
- A group **volume**.

For each **sound card** in the group:

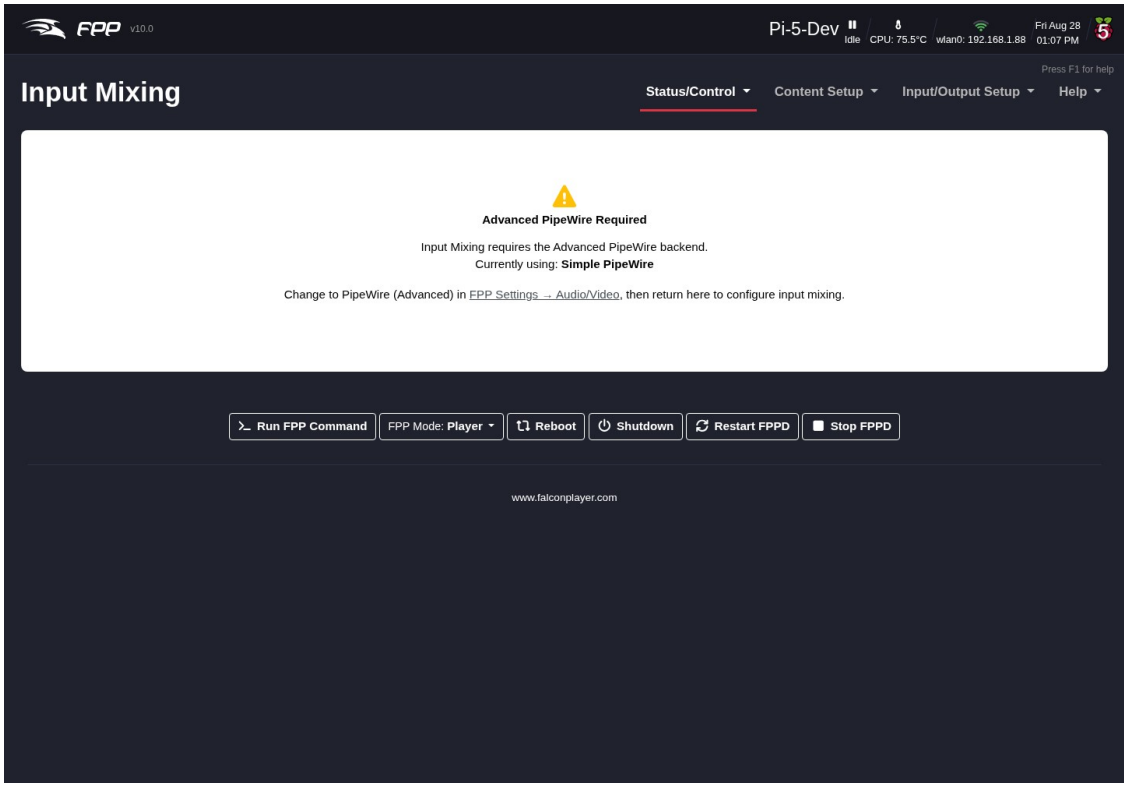
- **Sound Card** – chosen by its alias.
- **Card Channels** – how many channels the card provides.
- **Channel Mapping** – map each group channel to a card channel (e.g. `FL → FL`, `FR → FR`, `LFE → LFE`), so you can send, say, only the left channel to one card.
- **Volume**, **Delay (ms)** – per-card level and delay (delay is very useful to time-align distant speakers or compensate for network/receiver latency).

- **Rate / Period** – sample-rate and buffer period (usually **Auto**).
- **Parametric EQ** – enable per-card EQ and **+ Band** to add bands, each with a **Type** (Low Shelf, Peaking, High Shelf), **Frequency (Hz)**, **Gain (dB)** and **Q**.

Add Sound Card adds another member; **Sync Calibration** helps measure and set the per-card delays.

41.3 Input Mixing (Mix Buses)

Audio/Video → Configure Input Mixing.

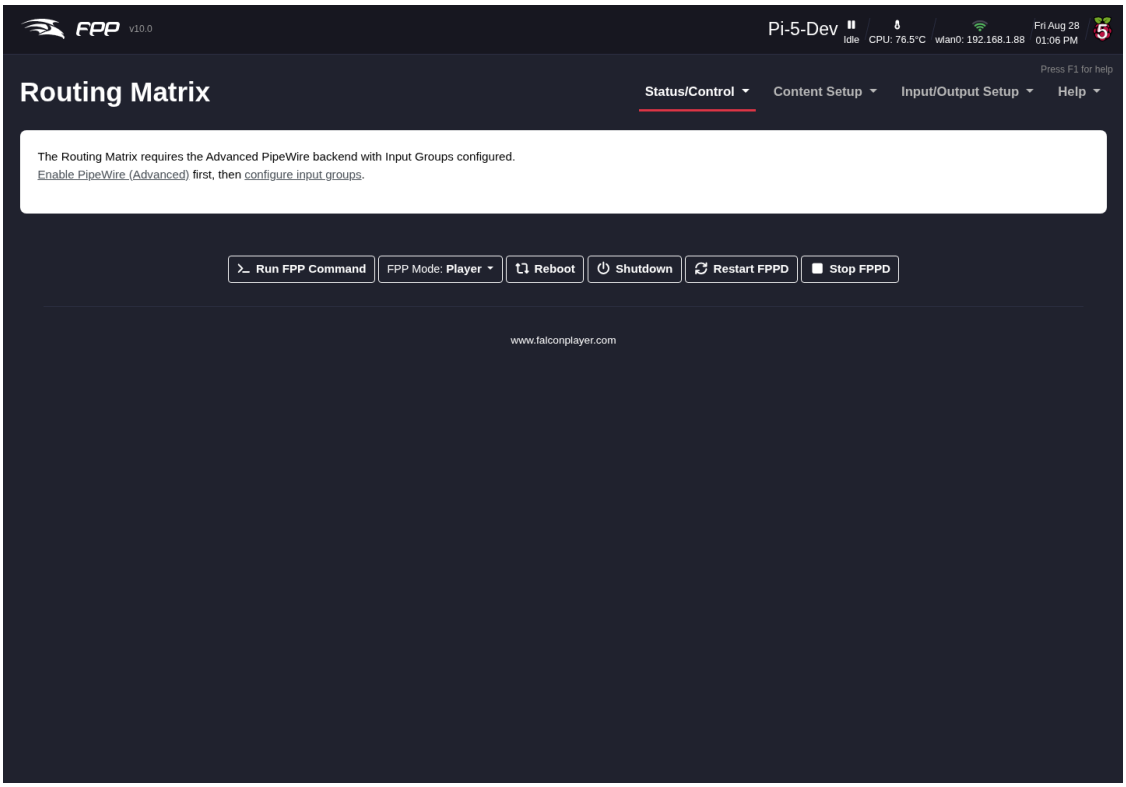


PipeWire Input Mixing — input groups and their sources.

An **input group** (mix bus) gathers one or more audio **sources** and routes them to output groups. For each input group you set its name, whether it is **Enabled**, and its channel count, then add sources. Each source has a **Type** (e.g. *fppd Stream*), a **Source** (e.g. *FPP Media Stream 1*), a **Name**, a **Volume** and a **Mute** control. Tick the **Route to Output Groups** boxes to send the mixed result to the chosen output groups (or open the **Routing Matrix** for a grid view).

41.4 Routing Matrix

Audio/Video → Open Routing Matrix.



The Routing Matrix — audio input×output grid with per-path volume, input-group EQ, and video routing.

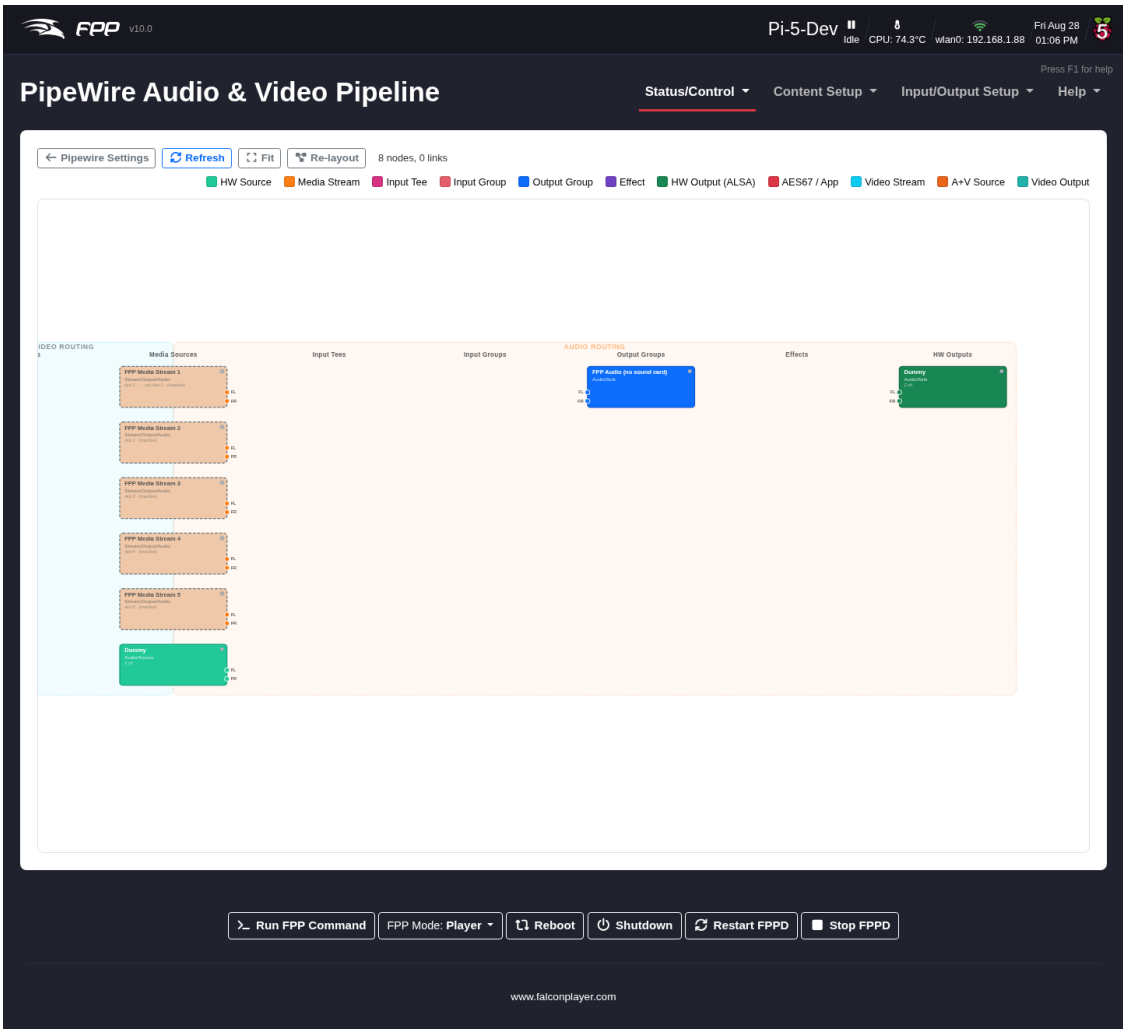
The **Routing Matrix** is the single place to connect everything:

- **Audio Routing** – a grid of **Input Groups** (rows) against **Output Groups** (columns). Tick a cell to route that input to that output, with a **per-path volume** slider for each connection.
- **Input Group Effects (EQ)** – enable EQ on an input group and add bands.
- **Video Routing** – a grid of **Video Sources** against **Video Output Groups**; select which source feeds each video group.
- **Routing Presets** – save a complete routing configuration (audio *and* video assignments) by name and reload it later.

Click **Save & Apply** to activate the routing.

41.5 Pipeline Graph

Audio/Video → Visualise Current Pipeline.

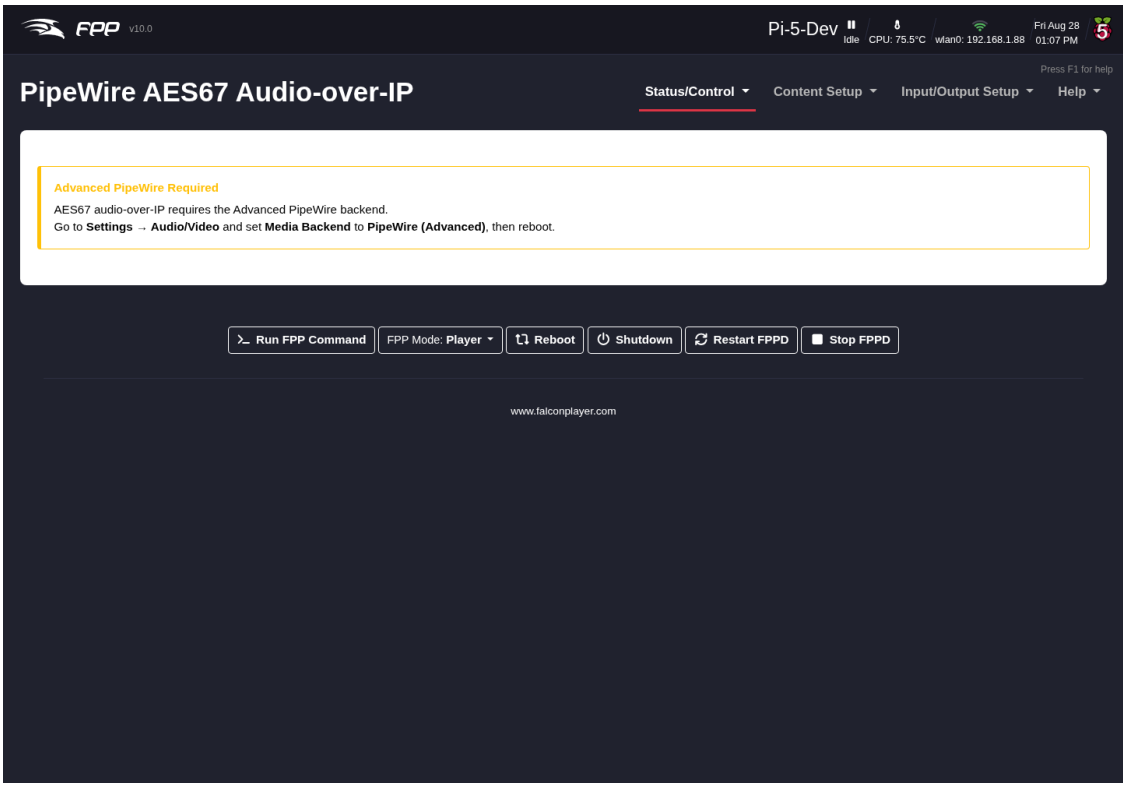


The live PipeWire pipeline graph.

This page draws the **live PipeWire graph** — the media streams, combine sinks, filter chains (delay/EQ), sound cards and video nodes, and the links between them. Producers and consumers are colour-coded. It is a valuable troubleshooting view for confirming that audio and video are flowing where you expect.

41.6 AES67 Audio-over-IP

Audio/Video → Configure AES67 Instances.



AES67 Audio-over-IP configuration: the global PTP clock settings above a send and a receive instance.

AES67 streams uncompressed, PTP-synchronised audio over the network as multicast RTP — the professional standard used by many consoles, DSPs and powered speakers. Each **AES67 instance** you define appears in the PipeWire graph as a virtual sink (Send) or a virtual source (Receive), so you can add it to an audio output group and per-card **delay and EQ apply to it too**. FPP announces streams via SAP and derives RTP timestamps from a PTP clock, so compliant receivers discover and lock to them automatically.

Click **Add AES67 Instance** to create one, and **Save & Apply** to rebuild the pipeline. The page requires **Media Backend** to be set to **PipeWire (Advanced)**; otherwise it shows a notice instead of the settings.

41.6.1 Stream status

The status line at the top of the page reports PipeWire's state, **how many of the configured streams are actually running**, the PTP clock state, and how many AES67 streams have been **discovered** on the network. It refreshes every ten seconds, so it is worth watching for the first minute after FPPD starts — PTP moves from *listening* to master or follower as the clock election settles.

41.6.2 PTP clock synchronisation

PTP (IEEE 1588 Precision Time Protocol) provides sample-accurate clock synchronisation between AES67 devices. These settings apply to the whole device, not to a single instance:

- **Enable PTP** – run the PTP clock. Enable it when you need tight sync between several FPP instances or with professional AES67 gear.
- **Network Interface** – which interface the PTP clock runs on. Choose the wired Ethernet interface; **(Default)** uses the system's primary route.
- **Domain** – the PTP domain number, **0–127**. Every device that must share a clock has to be on the same domain. AES67 gear normally uses domain **0**, so leave this alone unless your console or DSP is set otherwise.
- **Clock Role** – who provides the clock:
  - **Auto (prefer other devices)** – the default. FPP joins the election at a low priority: it becomes the clock when it is the only device on the domain, but yields to a console or DSP that wants the role.
  - **Follower only (never master)** – FPP never becomes the clock.
  - **Prefer master** – FPP tries to win the election. Use this only if FPP is meant to be the master clock for the network.

Once PTP is running the status line shows either **PTP master (this device)**, or **PTP synced to** the grandmaster's clock ID together with the current offset (shown in ns, µs or ms as it settles). If it has not locked yet you get **PTP not synced** plus the port state, which is normal for the first few seconds.

> **Tip:** If FPP keeps announcing itself as the clock when you have a proper master on the network, check that both are on the same **Domain** and set FPP's **Clock Role** to *Follower only*.
>
> **Note:** FPP marks its own traffic for QoS automatically — RTP audio as **AF41** (DSCP 34) and PTP messages as **EF** (DSCP 46), the codepoints AES67 recommends. There is nothing to configure. If the installed PTP software is too old to accept DSCP settings, FPP retries without them and notes it in the log.

41.6.3 Per-instance settings

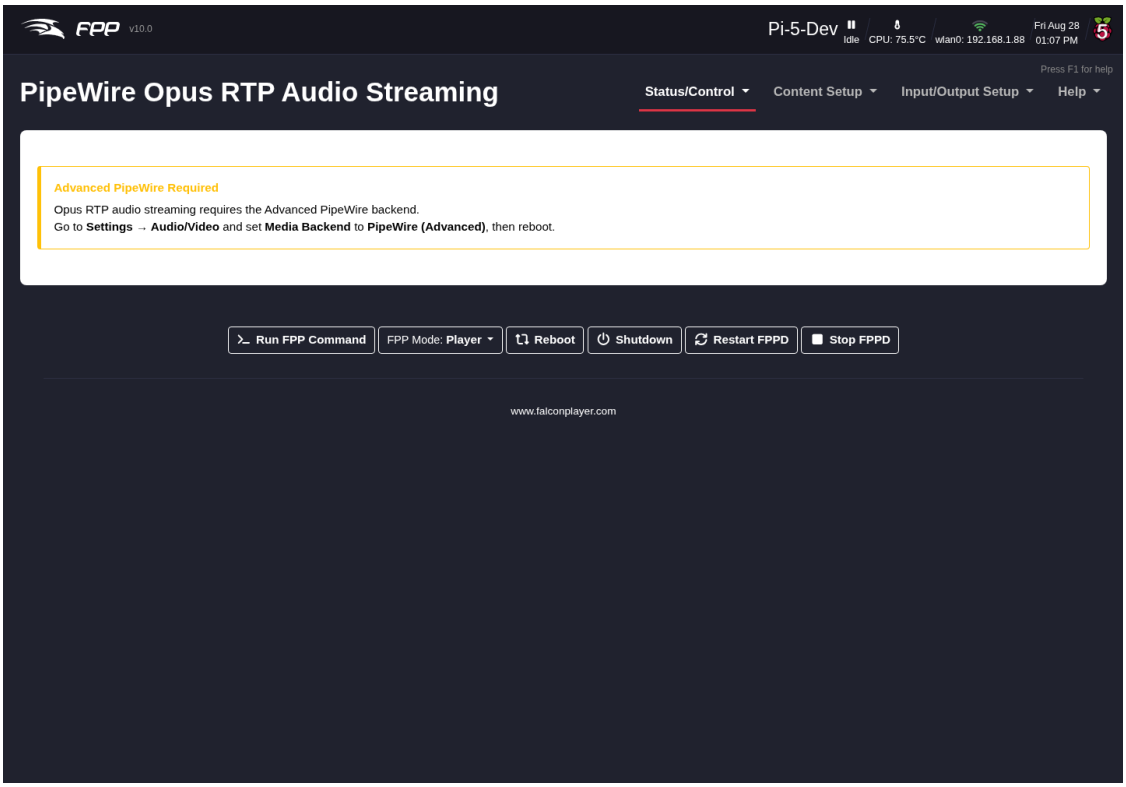
Each instance has a checkbox to enable it, an editable name, a badge showing the PipeWire node it creates, and a delete button. Its settings are:

- **Stream Mode** – **Send (Transmit)**, **Receive**, or **Both (Send & Receive)**. Send creates a virtual sink; Receive creates a virtual source.

- **Multicast IP Address** – the multicast group, from the AES67 `239.69.x.x` range. Give each instance its own address to avoid conflicts.
- **RTP Port** – the UDP port for RTP traffic; the AES67 default is **5004**. Use different ports if several instances share one multicast address.
- **Audio Channels** – channels in this stream (standard AES67 allows up to 8; most setups use 2).
- **Network Interface** – the interface used for the multicast traffic. Wired Ethernet gives the best results.
- **Packet Time (ptime)** – the packetisation interval: **1 ms** (lower latency, more CPU) or **4 ms** (the default, more widely compatible). It must match at both ends.
- **Session Name** – the name other devices see in the SAP announcement.
- **Network Latency** – the target buffer for *receive* streams, in milliseconds (AES67's minimum is 1 ms; 1–20 ms is typical). Lower means less delay but more risk of dropouts on a busy network.
- **SAP Discovery** – announce sent streams, and auto-create receive streams from announcements heard on the network.

41.7 Opus RTP Audio Streaming

Audio/Video → Configure Opus RTP Instances.



Opus RTP audio streaming configuration.

Opus RTP streams **compressed** (Opus-encoded) audio over the network — far lower bandwidth than AES67 and tolerant of packet loss, which makes it the right choice for WiFi links and for remote players or listeners. Like AES67, each instance appears in the PipeWire graph and can be used in the audio routing.

Use **unicast** (the receiver's own IP address) over WiFi: most access points send multicast at the lowest data rate with no retransmission. On wired networks both unicast and multicast (`239.x.x.x`) work well, and multicast lets one sender feed several receivers.

The status line next to the PipeWire indicator shows **how many of the configured streams are running**, along with the error text for any pipeline that failed to start; it refreshes every ten seconds. Previously the page reported only PipeWire's own state, so a stream that never started looked the same as a healthy one.

Each instance has:

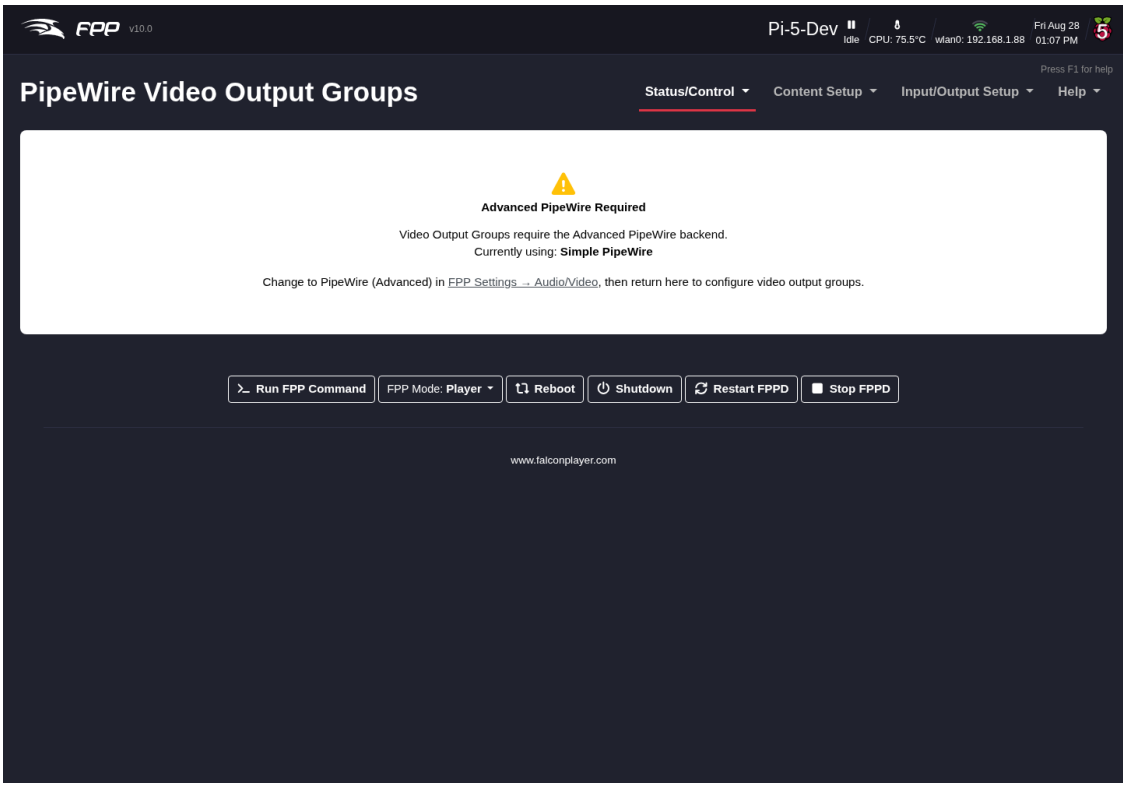
- **Stream Mode** – **Send (Transmit)**, **Receive** or **Both**.
- **Destination IP** – the receiver's IP for unicast, or a shared multicast group address for wired multicast.
- **RTP Port** – the UDP port; the default is **5005**.

- **Audio Channels** – Opus handles mono and stereo natively; higher counts use multiple Opus streams.
- **Network Interface** – the interface to send or receive on (e.g. `wlan0` with unicast, `eth0` for either).
- **Bitrate** – 32 kbps to 320 kbps; **128 kbps** is the default and is ample for stereo music, while 64–96 kbps copes where bandwidth is tight.
- **Jitter Buffer** – receive-side buffering in milliseconds: around 50 ms for wired unicast, 100–150 ms for WiFi unicast, more again if you must use WiFi multicast. Higher values add delay but ride out network timing variations.
- **Forward Error Correction** – Opus in-band FEC, which adds redundancy so lost packets can be reconstructed. Recommended on WiFi.
- **Discontinuous Transmission (DTX)** – send fewer packets during silence to save bandwidth, at the cost of slight artefacts when audio resumes.
- **Expected Packet Loss** – tunes the FEC encoder for the loss rate you expect; 5% is typical for WiFi, 0–1% for wired.

Note: Opus RTP audio is marked **AF41** (DSCP 34) automatically, so switches and access points that honour QoS give it priority over bulk traffic.

41.8 Video Output Groups

Audio/Video → Configure Video Output Groups.



PipeWire Video Output Groups.

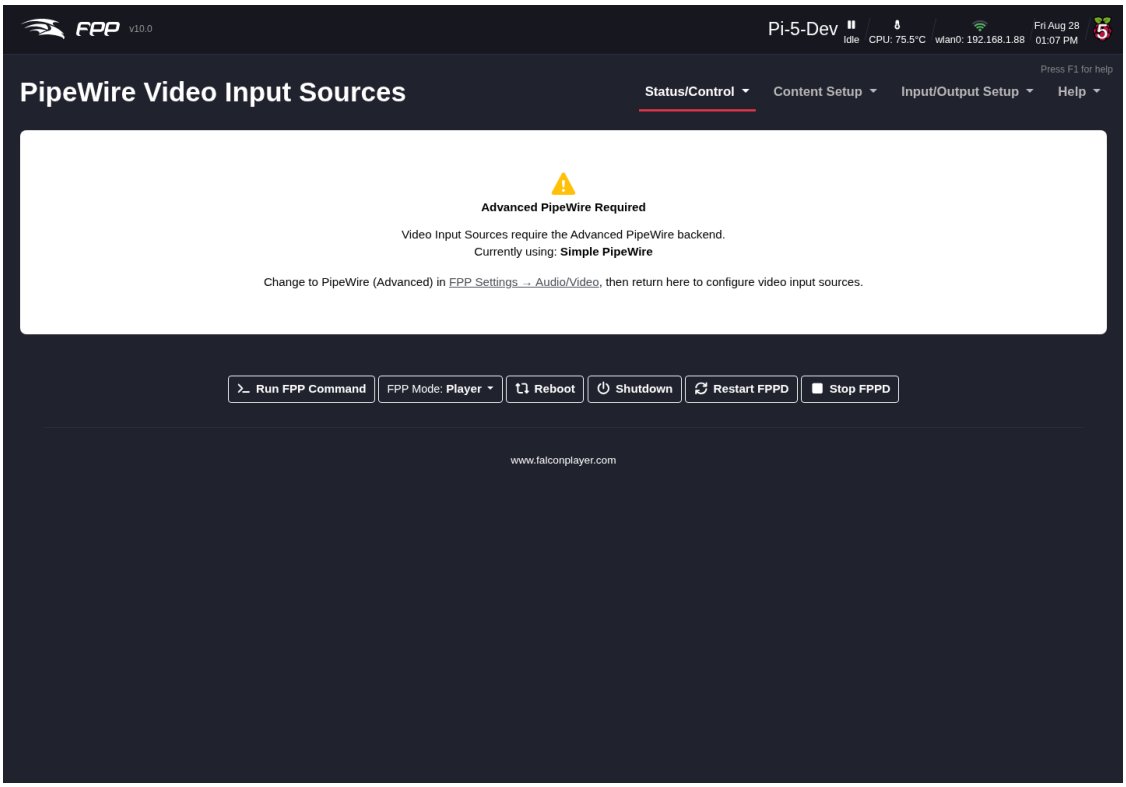
A **video output group** fans a single video stream out to several destinations at once. (The primary HDMI display is always driven directly by GStreamer for zero-latency output; these groups handle *additional* outputs routed through the PipeWire graph.) For each group set a name, the **Video Source** (Media Playback, or a persistent input source), optionally which **Stream Slots** (1–5) it uses, then add outputs. Each output has:

- **Output Type** – **HDMI Display**, a **Pixel Overlay** model, or a **network (RTP)** stream.
- **Destination** – the specific HDMI connector, overlay model, or network address.
- **Options** – scaling such as **Fit**.

This is how you drive a second HDMI screen, mirror video onto an LED matrix (via an overlay model), and stream video over the network — all from one playing video.

41.9 Video Input Sources

Audio/Video → Configure Video Input Sources.



PipeWire Video Input Sources.

Video input sources are persistent video producers that video output groups can route from (they survive consumers connecting and disconnecting). Supported types include:

- **Test Pattern** (`videotestsrc`) – e.g. SMPTE bars at a chosen size/frame rate.
- **USB Camera** (`v4l2src`) – a connected camera device (`/dev/video0`, …).
- **IP Camera** (`rtspsrc`) – an RTSP stream from a network camera.

Define a source here, then select it as the **Video Source** of a video output group to send a live camera or test pattern to HDMI, an overlay model, or the network.

41.10 A note on remote synchronisation

When FPP runs as a **remote**, the GStreamer pipeline continuously fine-adjusts its playback rate to converge on the player's position, keeping audio and video frame-accurate over long shows. This is automatic; the *remoteIgnoreSync* setting disables it if ever needed. See the MultiSync chapter for the player/remote setup.

> **Note:** This chapter covers the configuration UI. The full technical architecture (services, WirePlumber linking, GStreamer pipelines,

diagnostics) is documented in the FPP repository under `docs/FPP_Audio_Architecture.md`, `docs/PipeWire_Video_Routing.md` and `docs/GStreamer_PipeWire_Clock_and_Sync.md`.

42 Resources

This section collects reference material that sits outside FPP's own menus: where to get help from the community, background on networking concepts used throughout the manual, and step-by-step configuration recipes that are useful across several pages.

- **Help** – community forums, video resources, and live help.
- **Configuring a Static Route** – reaching a controller on a different subnet from your router or computer.
- **Creating a Proxy Host** – reaching a proxied controller through an FPP device, without a static route.
- **FPP Connect** – uploading sequences, media and settings from xLights to your FPP devices.
- **GPIO Button Input** – wiring a physical button or switch to a GPIO pin.
- **Networking Explained** – a plain-language look at IP addresses, subnets and gateways, and how Universe/Channel addressing maps onto ports.
- **Troubleshooting** – common Status page warnings and installation problems, with their likely causes and fixes.

43 Help

There are several resources available for help. Some of the more popular ones:

falconchristmas.com/forum – This forum is very active and you can usually get a response fairly quickly.

auschristmaslighting.com/forums – This forum is also very active and you can usually get a response fairly quickly. You don't have to be from Australia to post here.

auschristmaslighting.com – The parent site for the AusChristmasLighting forums, with several resources — one that is a must-have is the *AUSManual101.pdf*. You need to register on the site to download it.

www.xLights.org – Links to several resources including a forum. The information is geared towards xLights users, but it is not exclusively about xLights.

videos.xLights.org – Several videos covering all aspects of the animated-lighting hobby.

Canispater Christmas YouTube channel – Many very informational videos.

xEssentials Zoom Room – A live video conference with xEssentials training classes every Wednesday at 8:00 PM Eastern time, but there is usually someone there most of the time — a good way to get instant help. You will need to install the Zoom software the first time you visit.

Tip: FPP's own **Help → Get Help** page (see Help and Troubleshooting) links to the manual, the FPP Facebook Group, the Falcon Christmas Forums, and the xLights Zoom Room directly from the web UI.

44 Configuring a Static Route

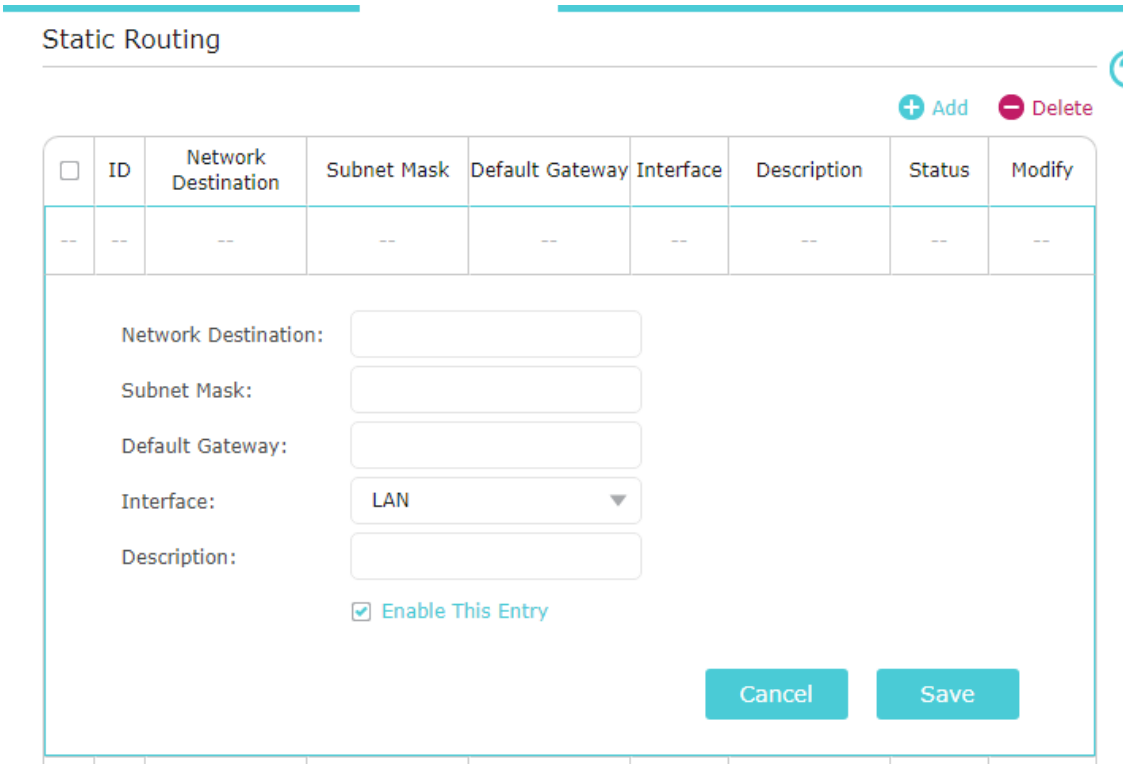
If your network configuration uses more than one subnet, you will probably need to configure a **static route** so that a device on one subnet knows how to reach a controller on another. There are three common methods: a static route in the **router**, a static route on a **computer**, or a **Proxy Host**.

The examples below assume your home/show router has an address of `192.168.0.1`, your FPP connects to the home/show network via Wi-Fi and to the controller via Ethernet, and the controller needs an IP on a different subnet than the home/show network — here `192.168.101.2`. The FPP needs an address on the wlan0 interface in the same network as the router (`192.168.0.101`) and an address on the eth0 interface in the same subnet as the controller (`192.168.101.1`).

> **Note:** This mirrors the general rules in Networking Overview — a device with two interfaces should have them on different subnets, with a gateway on only one (normally the home-facing interface).

44.1 Static Routing in router

Not all routers support static routing, but most do, and this is the **preferred method** — a route added in the router is available to every device on your network, not just one computer. Because there are so many router manufacturers and interfaces, the exact steps vary; look for an advanced section called *Routing*, *Advanced Routing* or *Static Routing*. The fields will be similar to this:



A router's Static Routing configuration page.

- **Network Destination** – the *subnet* of your controller — in this example `192.168.101.0` (note the last number is `0`, **not** the controller's own address).
- **Subnet Mask** – `255.255.255.0`.
- **Default Gateway** – the address of the connected FPP's wlan0 interface — in this example `192.168.0.101`.
- **Interface** – this option may not be available; use **LAN** if it is.
- **Description** – optional, if your router offers it.

44.2 Static Routing in Computer

You can also add a static route directly on a computer. This only gives *that* computer access to the controller — add the route on every computer that needs it. Windows and Mac use different commands.

- **Windows** – open a Command Prompt **as Administrator** and, based on the example above, enter:

  ```
  route -p add 192.168.101.0 mask 255.255.255.0 192.168.0.101
  ```

- **Mac** – open a Terminal window and, based on the example above, enter:

  ```
  sudo route add 192.168.101.0/24 192.168.0.101
  ```

> **Note:** The `route add` command on a Mac is **not persistent** — if the computer is turned off or rebooted, the route must be added again. FPP's Creating a Proxy Host is usually a better solution on a Mac for this reason.

44.3 Creating a Proxy Host

If you cannot add a static route in your router, or you are on a Mac where routes are not persistent, use an FPP device as a **Proxy Host** instead — see Creating a Proxy Host for the full walkthrough, or Proxy Settings for the page reference.

45 Creating a Proxy Host

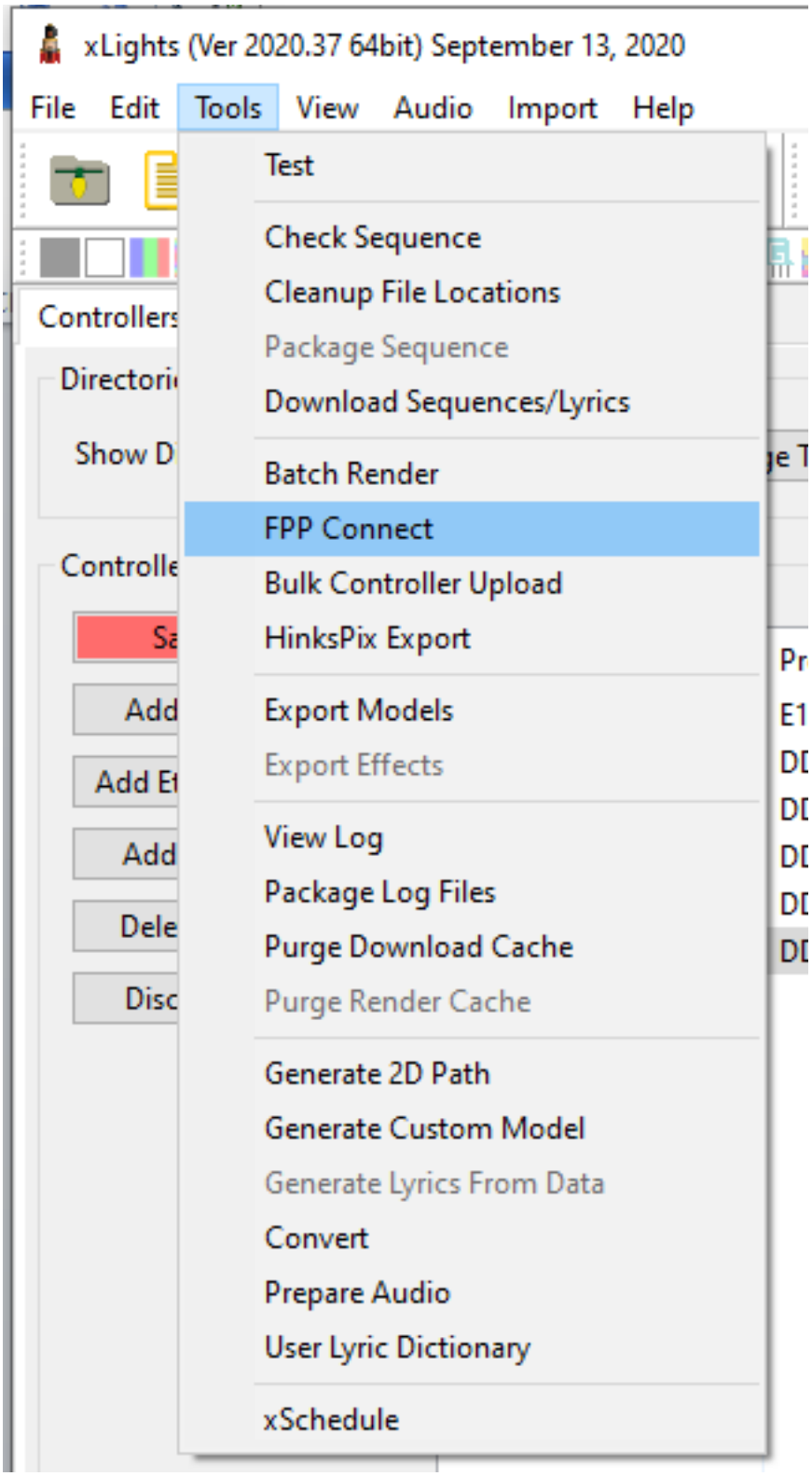
You can use an FPP device to route network traffic between subnets by setting it up as a **Proxy Host**, instead of configuring a static route. This is especially useful when you cannot add a static route to your router and are using a Mac (since Macs do not support persistent routes) — though a Proxy Host works just as well with Windows computers.

Open **Status/Control → Proxy Settings** and enter the IP address of the controller(s) attached to the FPP device. To reach a proxied controller's web UI afterwards, click its link on the Proxied Hosts list, or browse to the FPP device's IP followed by `/proxy/` and the controller's address — for example `192.168.1.101/proxy/192.168.101.2`.

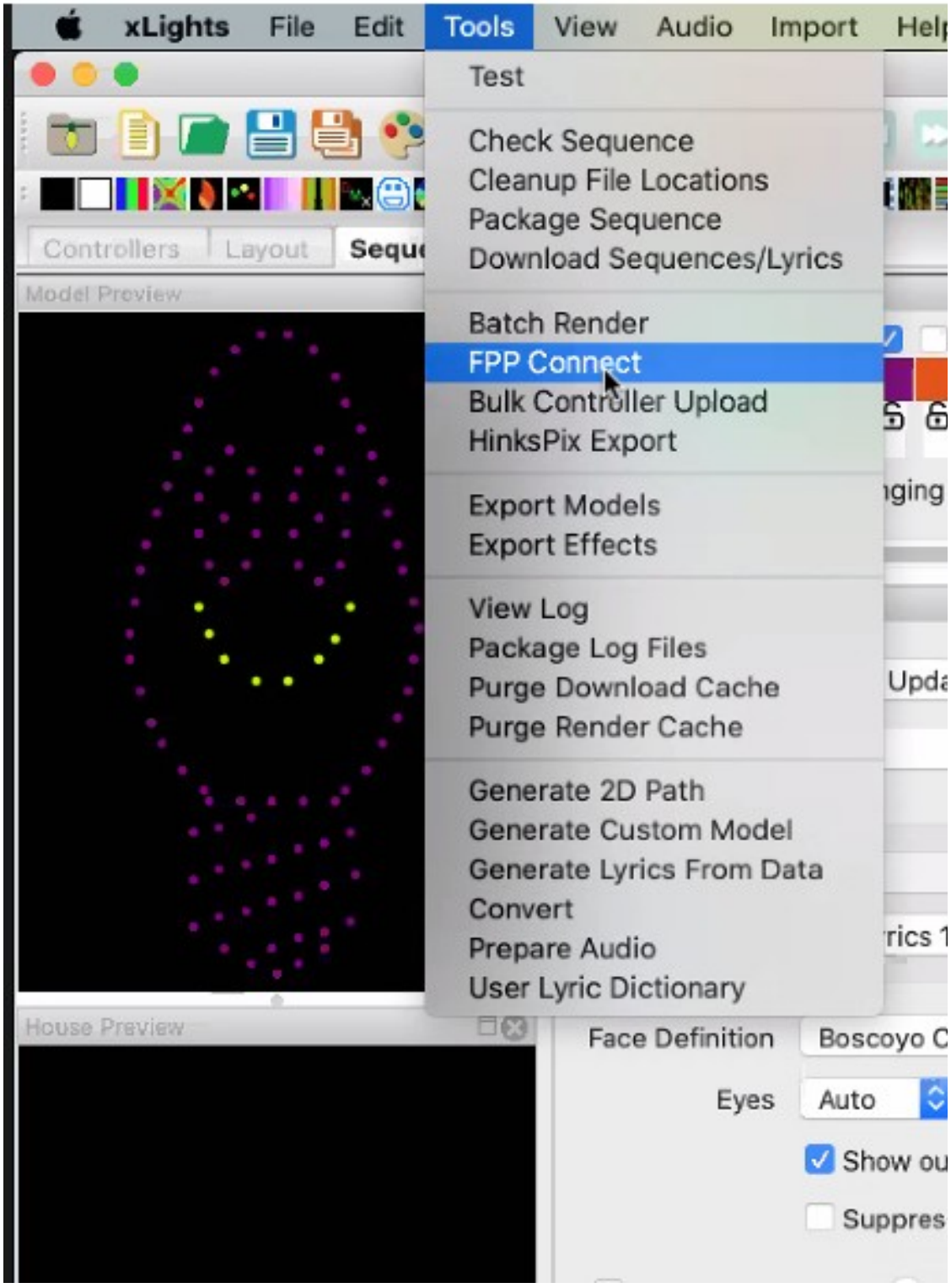
See Proxy Settings for the full page reference, including how to configure proxies directly from xLights via **FPP Connect**.

46 FPP Connect

FPP Connect, found under xLights' **Tools** menu, is the best and most efficient way to upload your sequences and media files to all of the FPP devices that need them. It can also configure most of the settings in FPP for your port outputs. You can upload to one, all, or selected FPP devices in a single step, which greatly simplifies keeping show data up to date.

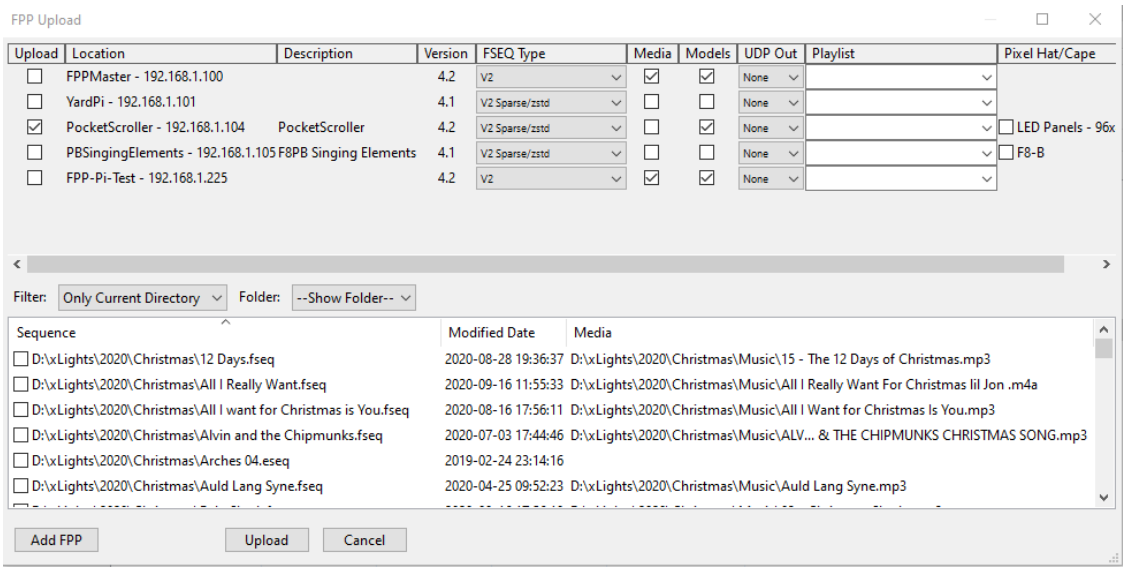


FPP Connect in xLights' Tools menu (Windows).



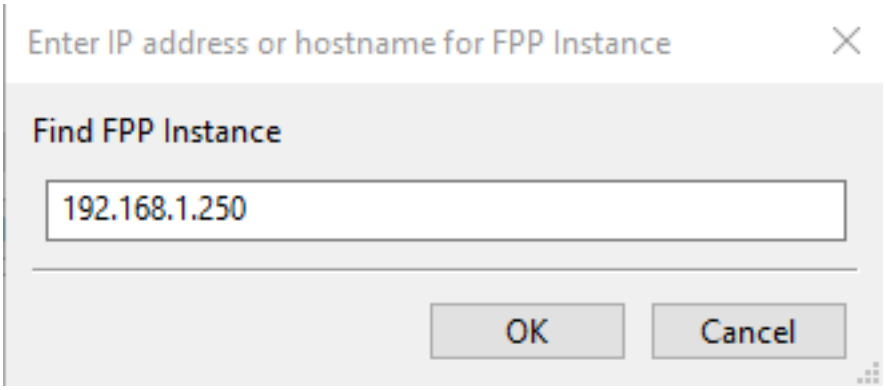
FPP Connect in xLights' Tools menu (Mac).

Once opened, FPP Connect discovers all of the FPP devices on your network.



The FPP Connect upload list, showing discovered FPP devices.

If one of your FPP devices does not show up in the list, add it manually with **Add FPP** and enter its address.



The Add FPP Instance dialog.

> **Note:** A device often fails to be discovered because another device shares the same Host Name (e.g. both left at the default FPP), or because of a network configuration issue. All FPP host names must be unique.

46.1 Options

Each row is one discovered FPP device, with columns controlling what happens to it:

- **Upload** – tick which FPP devices to act on. Your choices in the other columns apply to every device ticked here.

- **Location** – the device's Host Name and IP address (informational only).
- **Description** – the description you entered in FPP, for additional context (informational only).
- **Version** – the FPP version the device is running (informational only).
- **FSEQ Type** – the format of the FSEQ file to upload:
  - **V1** – an older, considerably larger format; typically only used by old FPP versions.
  - **V2** – a compressed format, more efficient than V1, containing all of the sequence's channel data.
  - **V2 Sparse/zstd** – a compressed format customised to the device, containing only the channels that device needs — saves a lot of space.
  - **V2 Sparse/zstd Uncompressed** – the same per-device customisation as above, but left uncompressed for controllers that cannot handle compression (uncommon).
- **Media** – when ticked, uploads the media file (.mp3/.mp4) associated with the sequence, as set in the Media column of the file list below, to that device.
- **Models** – when ticked, uploads your show layout's model data (channels, positions, configuration, etc.) as Pixel Overlay Models on that device. Useful for testing controllers by model, and required for a Virtual Display or HTTP Virtual Display.
- **UDP Out** – configures and enables the E1.31/DDP outputs:
  - **None** – for devices that do not need to output E1.31/DDP data, such as cape/hat controllers or an FPP device running in Player mode.
  - **All** – uploads and enables every E1.31/DDP output channel configured in xLights. A main player outputting to a switch connected to all your controllers typically needs this. You may need to delete or deactivate channels a given device does not need.
  - **Proxied** – uploads and enables only the E1.31/DDP channels for the controller attached to this FPP device, provided the FPP device is configured as a Proxy Host for that controller.
- **Playlist** – to add the uploaded sequences to an existing playlist on that device, select it here.

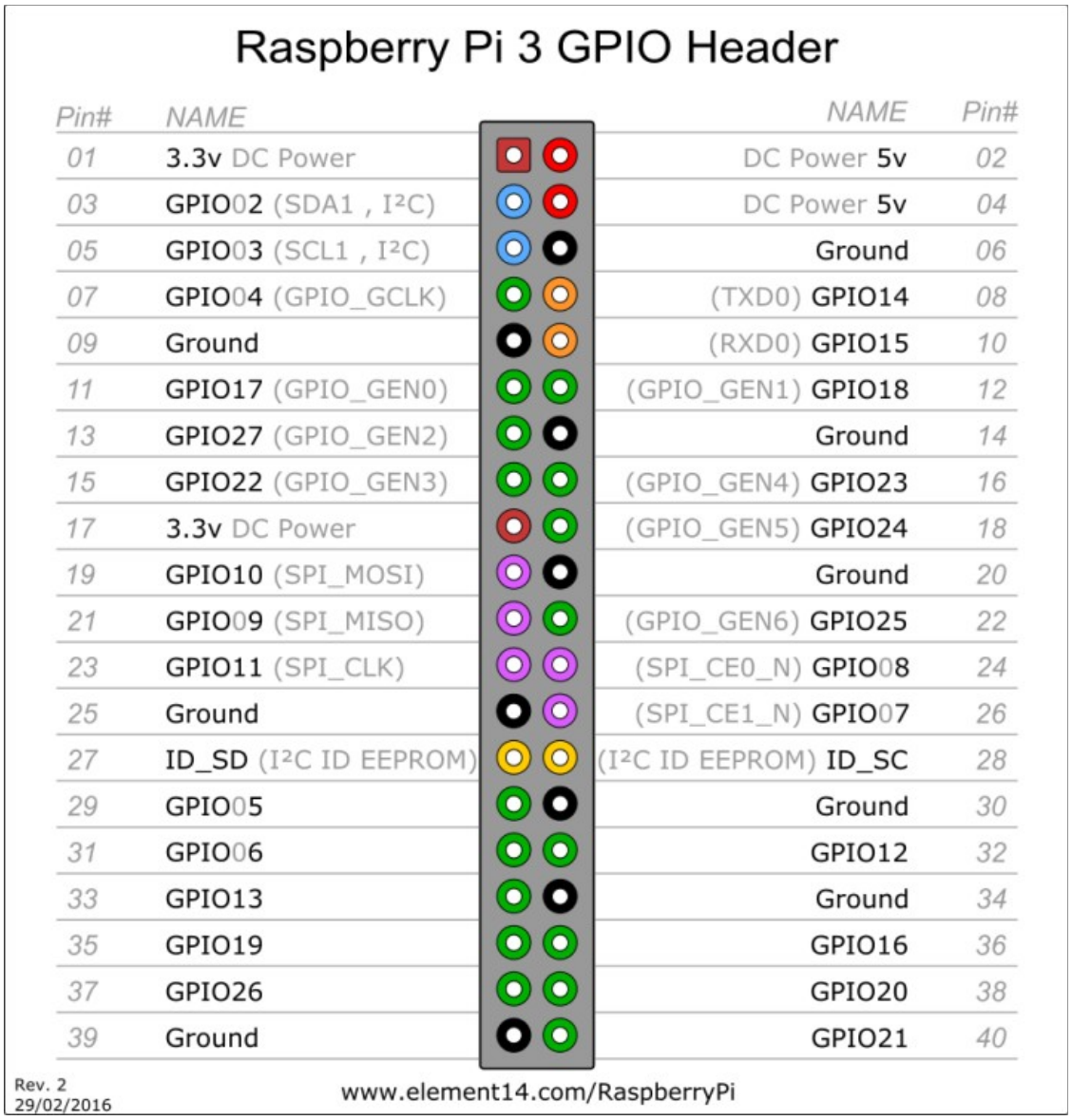
  Note: This always **adds** the sequence(s) to the playlist — if a sequence is already in the playlist, it will end up there twice.

- **Pixel Hat/Cape** – shown only when the FPP device has a hat/cape configured for local pixel ports; when ticked, uploads the pixel port configuration to it.

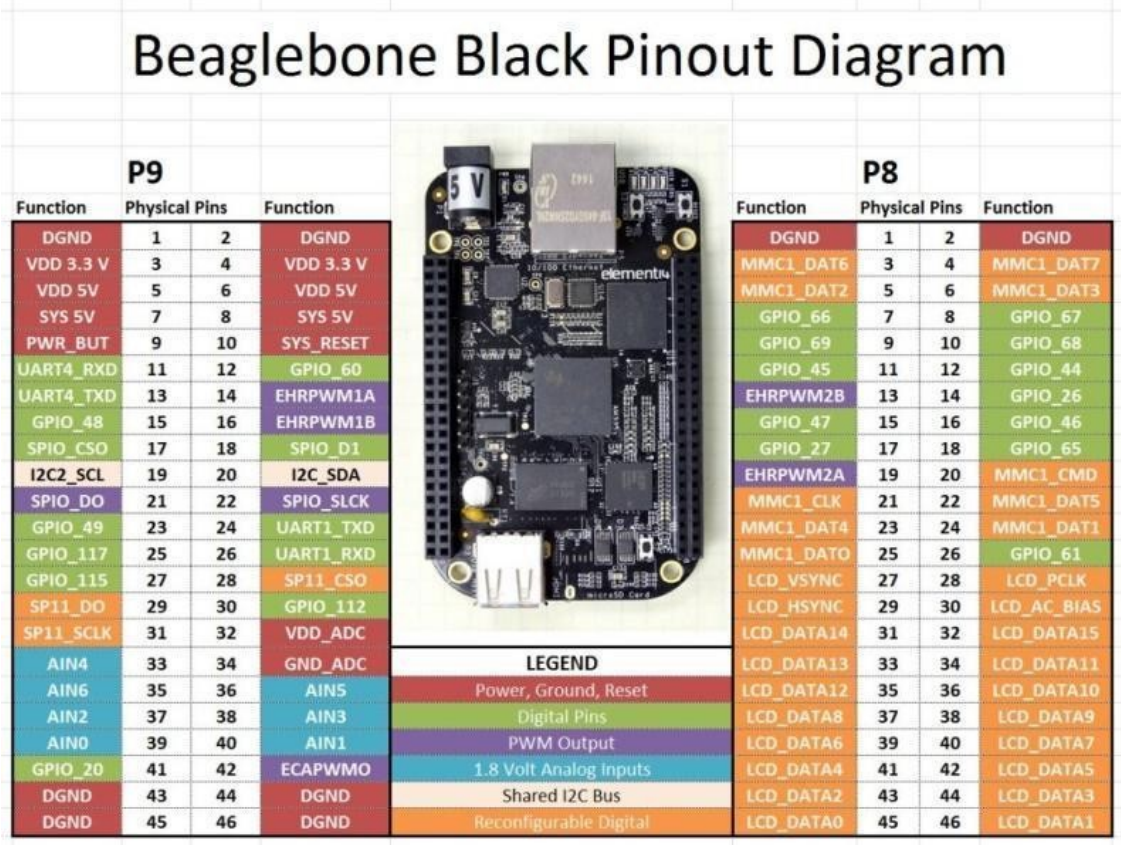
47 GPIO Button Input

You can wire a physical button, switch or sensor to a GPIO pin to trigger an FPP event — see GPIO Inputs for how to configure the trigger itself once it's wired. This chapter covers the wiring side: identifying the right pins and choosing between an external or internal pull resistor.

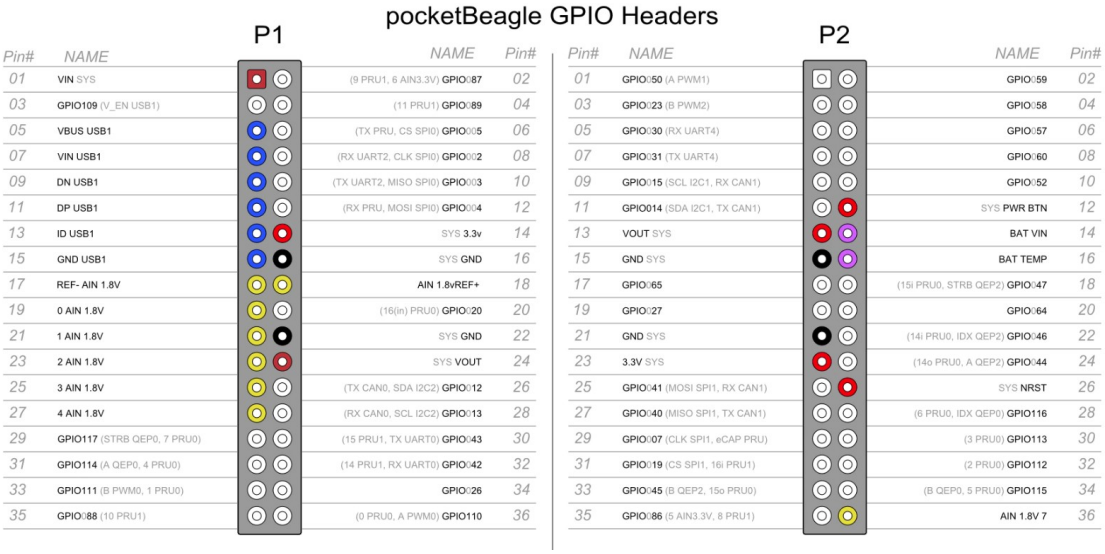
Before making any connections, confirm the GPIO pinout of the SBC you are using, and always wire with the power disconnected.



Raspberry Pi 3 GPIO header pinout.



BeagleBone Black pinout diagram.



pocketBeagle GPIO header pinout.

47.1 Pull-up and pull-down resistors

A GPIO pin left unconnected is in a **floating** state, and must be “forced” high (a positive voltage) or low (ground) so it has a definite resting value.

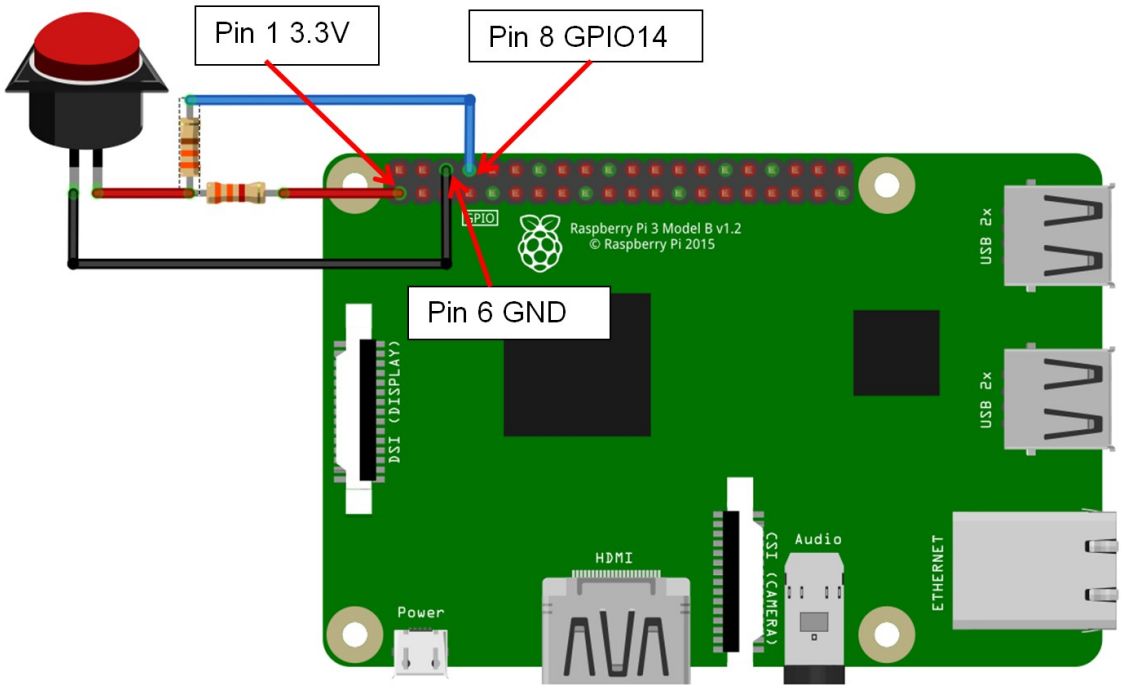
This is done by connecting a resistor from the pin to either 3.3 V (pulling it high) or to ground (pulling it low) — either with a resistor wired into your own circuit (**external**), or using the pull resistor built into the SBC's GPIO controller and enabled in the FPP GPIO Inputs page (**internal**).

> **Note:** Configure only one of the two — do not enable an internal pull resistor **and** wire an external one on the same pin.

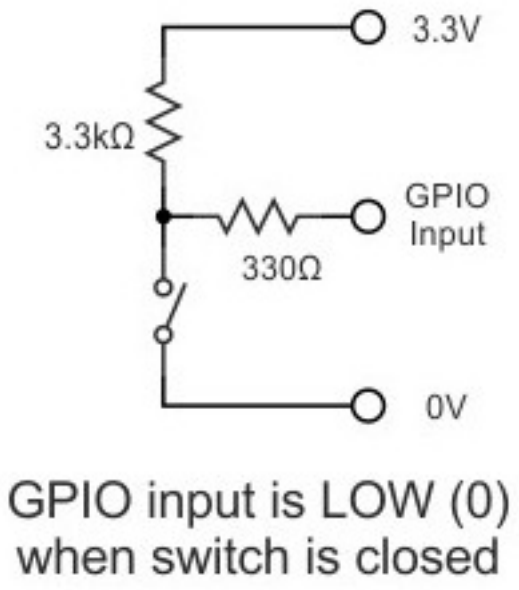
The examples below use GPIO14 on a Raspberry Pi 3. If you use the internal pull-up/down resistor, you can drop the external 3.3 kΩ resistor and the connections it makes.

47.1.1 External resistor, pulled high

With the resistor wired to 3.3 V, the pin rests **high**; pressing the button pulls it to ground, producing a **falling** trigger.



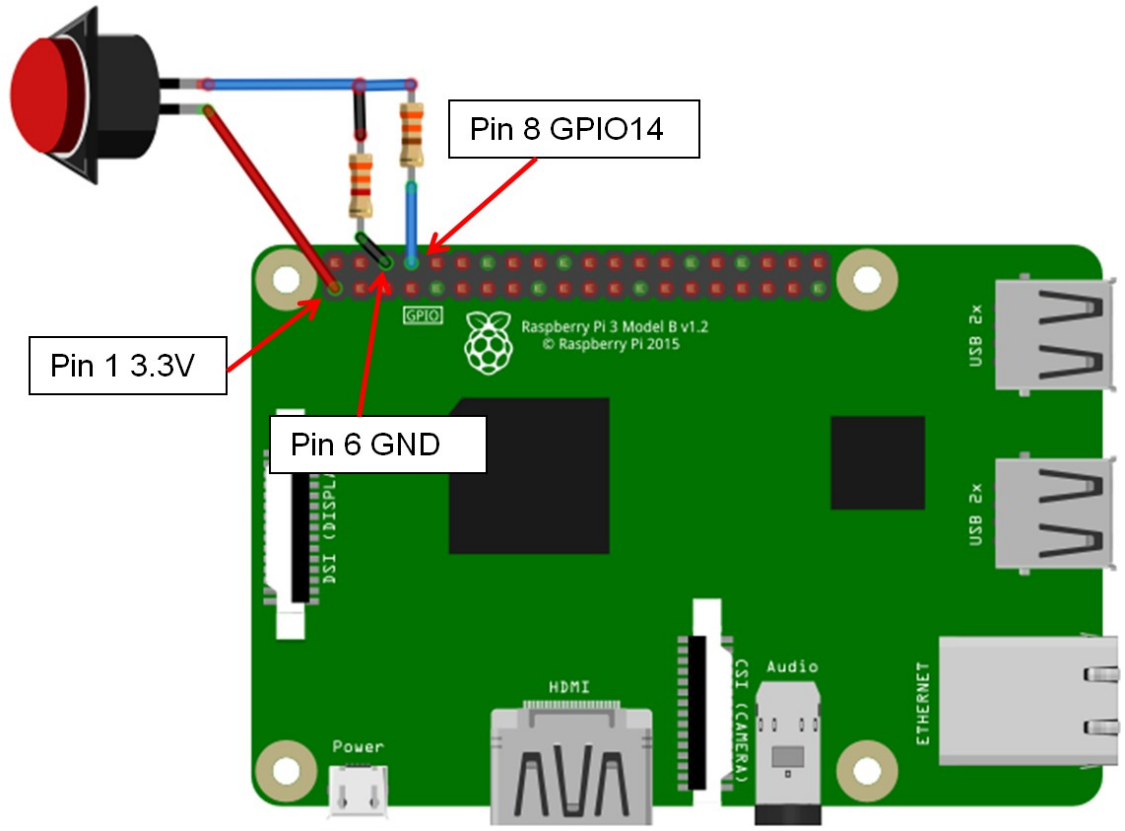
Wiring an external resistor pulling GPIO14 high.



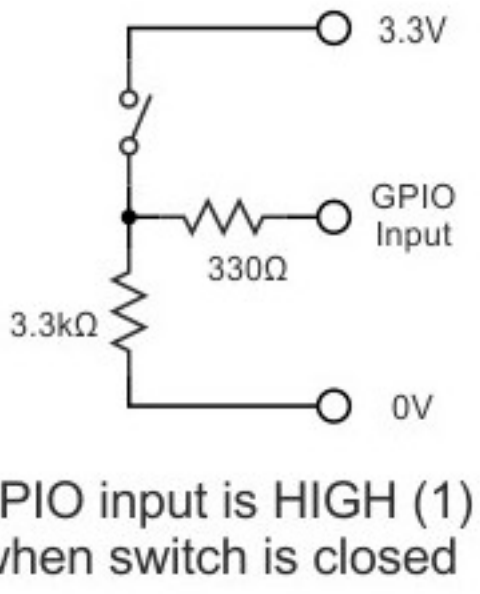
Schematic: external resistor pulling the GPIO pin high.

47.1.2 External resistor, pulled low

With the resistor wired to ground, the pin rests **low**; pressing the button connects it to 3.3 V, producing a **rising** trigger.



Wiring an external resistor pulling GPIO14 low.

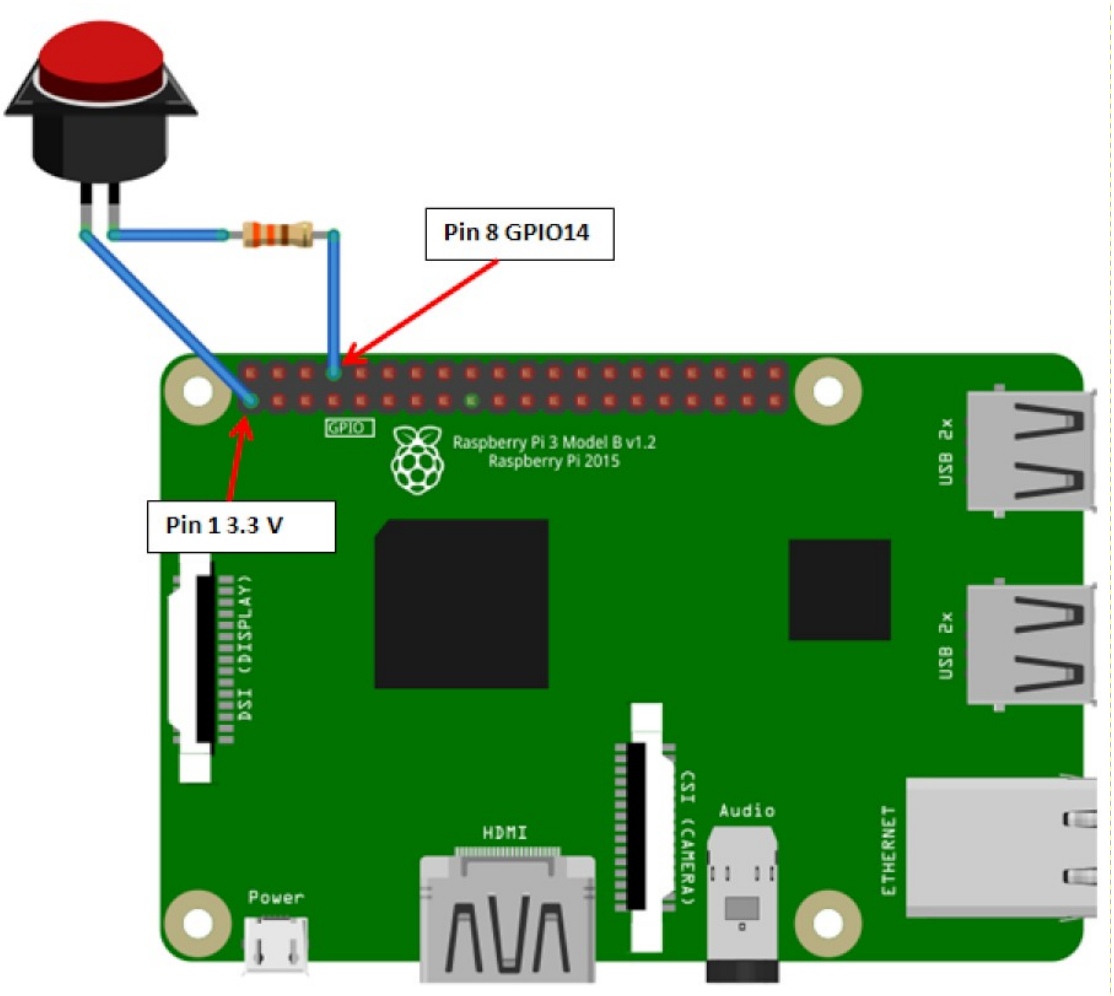


GPIO input is HIGH (1)
when switch is closed

Schematic: external resistor pulling the GPIO pin low.

47.1.3 Internal resistor, pulled low

No external resistor is needed — enable the **Pull Down** option for this pin on the GPIO Inputs page instead. The pin rests low, and pressing the button (wired to 3.3 V) produces a **rising** trigger.

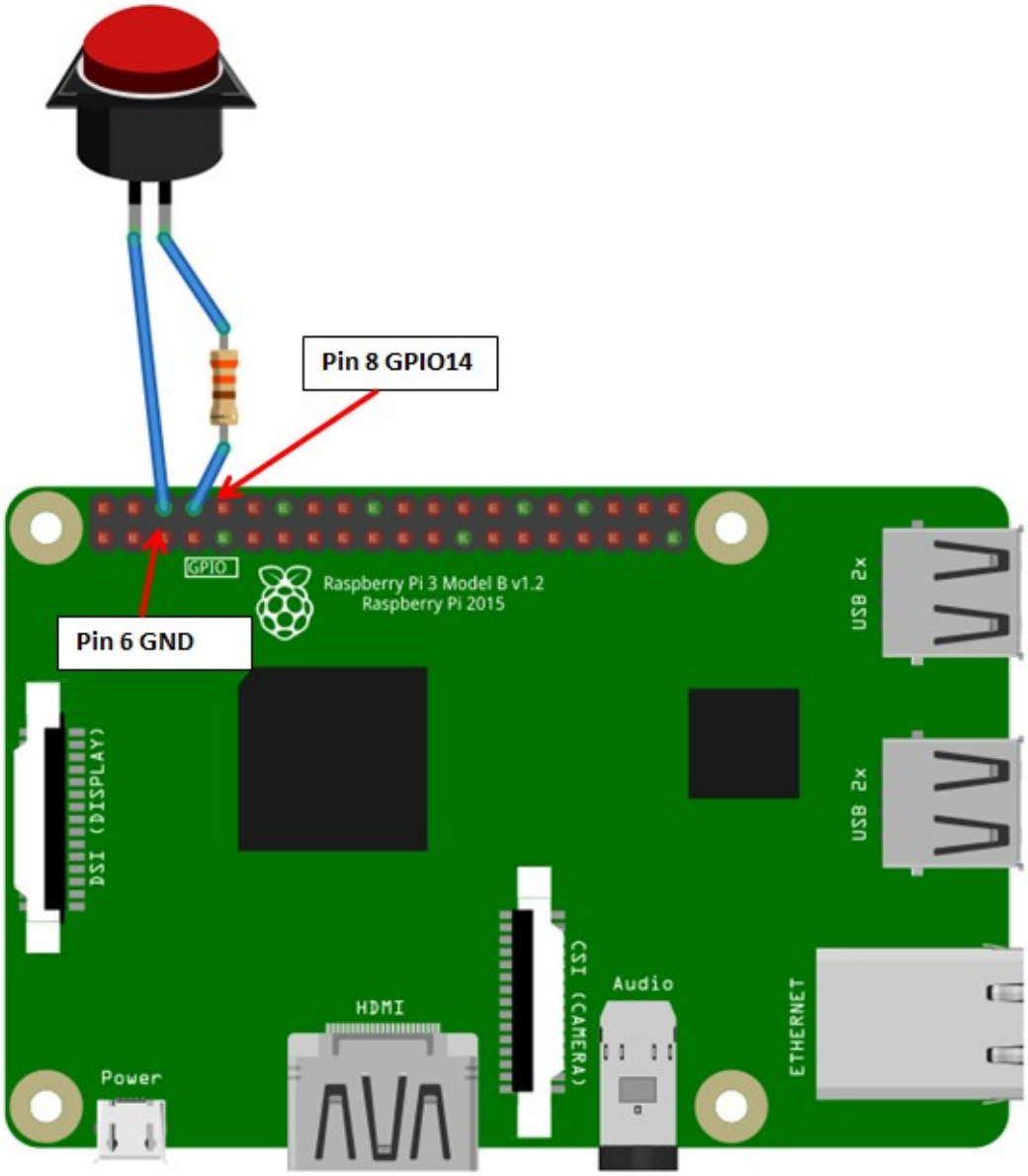


Wiring for an internal pull-down resistor on GPIO14.

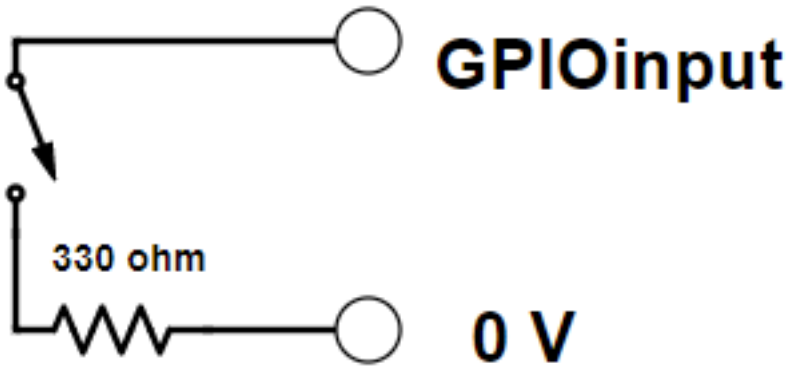
Schematic: internal resistor pulling the GPIO pin low.

47.1.4 Internal resistor, pulled high

Enable the **Pull Up** option for this pin on the GPIO Inputs page. The pin rests high, and pressing the button (wired to ground) produces a **falling** trigger.



Wiring for an internal pull-up resistor on GPIO14.



Schematic: internal resistor pulling the GPIO pin high.

47.2 Switch types and trigger direction

There are two basic types of switches:

- **Normally Open (NO)** – lets electricity flow only while it is pressed.
- **Normally Closed (NC)** – lets electricity flow until it is pressed, then stops the flow.

And the GPIO trigger itself fires on one of two edges:

- **Rising trigger** – the voltage on the pin transitions from ground to a positive voltage (roughly 1.3 V or higher).
- **Falling trigger** – the voltage transitions from a positive voltage back towards ground.

Which edge corresponds to a press depends on the combination of: whether the pin is pulled high or low, whether the switch is Normally Open or Normally Closed, and whether you want the action to happen on press or release — you can configure commands for both edges if you want different actions for each (see GPIO Inputs → Trigger Commands).

For example, with the pin pulled **high** and a **Normally Open** button: pressing the button triggers a **Falling** event, and releasing it triggers a **Rising** event.

48 Networking Explained

Networking trips up a lot of people getting started in animated holiday lighting. This chapter is a plain-language, ground-up explanation of IP addressing and of Universe/Channel addressing — companions to the more

practical Networking Overview and Channel Outputs chapters, if you want to understand *why* the rules there work the way they do.

48.1 Network Overview

A network is a group of devices identified by an **IP address**. Think of an IP address as a telephone number: for devices to communicate, every device needs a unique "number". IP addresses are written as four groups of numbers (0–255) separated by dots, e.g. `192.168.110.23`. The first three groups are like an area code and are called the **subnet** (`192.168.110`); the last group is like the local number and identifies the **device** (`23`). Only devices that share a subnet — the same "area code" — can communicate **directly**, the way you can dial a local number without an area code.

If a device needs to reach another device on a *different* subnet, that's like calling a different area code: extra routing information is needed. When a device gets a request for an address it doesn't recognise as local, it sends the request to its **Gateway** — similar to an old-fashioned telephone operator. A Gateway is any device on the same subnet that is also able to route traffic onward; it is usually your home router, or an FPP device sitting between a controller and the show network. A Gateway only knows how to route addresses in its own small "phone book" — anything it doesn't recognise, it forwards to *its own* configured Gateway. Sometimes a Gateway needs explicit extra instructions to route traffic correctly; this is a **Static Route** (see Configuring a Static Route).

You can add a Static Route on a single computer, so that when it's told to reach an address covered by the route, it sends the data to that route's Gateway — but only *that* computer gains access, unless you repeat the setup on every computer on the network. Most routers let you add Static Routes instead, which then apply to every device on the local network. Static Routes added in Windows can be made persistent; on a Mac they are not, and need to be re-added after every reboot.

48.2 Universes, Channels and Ports, oh my!

> **Note:** xLights and FPP features like Auto Size, Auto Layout Models, the Visualizer and FPP Connect exist specifically so you don't *need* to understand channel addressing to configure your controllers and FPP devices — this section is background for when you want to understand what's happening underneath.

A **port** is simply where you plug in a string of pixels. Large lighting networks can have tens of thousands of channels, and every device — including you — needs a consistent way to know where every pixel is: which model it belongs to, which controller is driving it, and which port(s) carry its data.

Animated lighting borrowed its addressing scheme from DMX: **Universes** and **Channels**. A Universe can hold any number of Channels up to a maximum of 512 — you decide how many channels each of your Universes actually uses. You *can* use **Absolute Addressing**, numbering every channel from 1 upward with no Universe grouping at all, but that quickly becomes hard to manage on a large display; breaking channels into Universes makes it easier.

An analogy: think of your pixels as passengers on a train. Each **model** in your show is a family of passengers. Every passenger orders a drink — that's the color/brightness data sent to that pixel. The **sequencing software** (the travel agent) doesn't know how big the train cars will be, so it just lists every passenger and their drink order, front to back. The **controller** (the Terminal) knows how big each car actually is — up to 512 seats, matching a Universe — and hands each **port** (an attendant) the drinks for its section of passengers, in order, along with where to start serving.

For example, a show needs 1,578 channels, split across cars (Universes) of up to 512 each — the Terminal assigns 4 cars. Attendant (port) 1 serves 490 drinks starting at car 1, seat 1. Attendant 2 serves 560 drinks starting at car 1, seat 491 — finishing the rest of car 1, all of car 2, and the first 26 seats of car 3. Attendant 3 serves 520 drinks starting at car 3, seat 27. Attendant 4 serves the remaining 8 drinks starting at car 4, seat 35.

A few things follow from this:

- There is no fixed relationship between a controller's ports and Universe/ Channel addressing — a port just needs to know which Universe/channel to start at and how many pixels' worth of data to read, then stop.
- Universes can be numbered arbitrarily and don't need to start at 1 or run sequentially — a car can be numbered 100, the next 215, and so on, as long as every port knows which car and seat to start at.
- Universes can be any size up to 512, and don't all need to be the same size.
- A port can supply data for part of a Universe, or span more than one Universe.
- A port doesn't need to start at the beginning of a Universe or of a model — though for your own sanity, it's often worth doing so anyway.

49 Troubleshooting

This chapter collects two kinds of troubleshooting information from the FPP community: warnings you may see on the **Status** page during normal operation, and problems that come up while installing FPP for the first time.

For the built-in diagnostic tools (System Health Check, Troubleshooting Commands, support bundles), see Help and Troubleshooting.

49.1 Status page warnings

The Status page surfaces warnings when FPP detects an abnormal condition. Some of the messages you may see, and what they usually mean:

- **Multiple Frame Skips During Playback – Likely Slow Network.** Something on your network is slowing the data down — usually caused by configuring E1.31/DDP outputs on devices that don't actually need them.
- **Could not resolve Host Name "some name here" – disabling output.** Your DNS server isn't configured correctly. The configured DNS server address needs to point at a device that can actually process DNS requests — most computers and SBCs don't run one, but a home router usually does.
- **Repeated frames taking more than 20ms to send to Colorlite.** Usually a slow network connection to the Colorlight receiver card.
- **Could not create output type "some output type here". Check Logs for details.**
- **Could not initialize output type "some output type here". Check Logs for details.**
- **FSEQ Data Block not available – Likely slow storage.** Usually caused by storing sequences on a USB drive, or a uSD card that isn't rated Class 10 or higher.
- **Could not ping DDP/E1.31 Channel Data Target "some IP address".** Usually means the controller at that address is powered off, or isn't configured correctly, while FPP is set to output E1.31/DDP data to it.
- **Received DDP/E1.31 data from "some IP address".** Usually means FPP or xLights is configured to send E1.31/DDP data to an FPP device that shouldn't normally receive it, such as one running in Player or Remote mode.
- **Sequence file /home/fpp/media/sequences/(some sequence here) does not exist.** Appears on a Remote that received a sync command to start a sequence it doesn't have a copy of.

49.2 Common installation problems

Symptom	Possible causes	Remedy
Can't access the FPP device during **USB Tether** installation.	1. The FPP device has no power.2. (Pi Zero only) The USB cable is in the wrong port.3. The USB cable is	1. Confirm the FPP device is connected via USB and its power indicator is lit.2. The Pi Zero has two USB

Symptom	Possible causes	Remedy
	faulty.4. Wrong IP address for USB tethering.5. The image on the uSD card is corrupt.6. The uSD card itself is faulty.7. The FPP device doesn’t support USB tethering.	ports — one is power-only. Use the one closer to the center of the board, labeled USB (not PWR).3. Some USB cables are charge-only; use one known to support power *and* data.4. You can’t use the device’s Host Name over USB tethering — use the tethering IP address instead: `192.168.7.2` on Windows, `192.168.6.2` on Mac/Linux.5. Format and re-image the uSD card — see Installing the FPP Software.6. Format and re-image a known-good uSD card — see Installing the FPP Software.7. USB tethering is only supported by Raspberry Pi Zero and BeagleBone devices.
Can’t access the FPP device during **Network Connection** installation.	1. The FPP device has no power.2. The Ethernet cable is faulty.3. The Ethernet cable isn’t fully seated in the RJ45 jacks.4. The Ethernet cable is in the wrong port on the router.5. The RJ45 connector’s pins are damaged.6. You’re trying to access the FPP device directly with a keyboard and monitor.7. You’re using the wrong Host	1. Check the power connection and confirm the power LED is lit.2. Test the Ethernet cable, or substitute a known-good one.3. Check that the cable is fully inserted at both ends.4. Make sure the cable is in a LAN port, not the WAN port.5. Visually inspect the pins for damage.6. FPP has no local display — you access its web UI through a browser

Symptom	Possible causes	Remedy
	Name.8. DNS isn't resolving local names.9. The FPP device doesn't have a good network connection.10. The image on the uSD card is corrupt.11. The uSD card itself is faulty.	(e.g. Google Chrome) from another device, not a keyboard/monitor attached directly to it.7. The correct Host Name for a fresh install is `http://fpp/` or `http://fpp.local/`. If you have other FPP devices already on the network, Host Name access may not work — see the next item.8. Some routers won't resolve local host names like `http://fpp`, so you'll need the FPP device's IP address instead. Log into your router's admin page (often on a sticker on the router, or look up the default login for your brand), find the connected-devices/DHCP table, and look for a device named FPP to get its IP. If you can't access your router, use xScanner in xLights or an IP scanning utility.9. Check your computer's available networks for an entry named FPP — if one exists, the FPP device couldn't join your network; check all cables. Also confirm you aren't on a `192.168.7.x` subnet (the USB-tether subnet).10. Format and re-image the uSD card — see Installing the

Symptom	Possible causes	Remedy
		FPP Software.11. Format and re-image a known-good uSD card — see Installing the FPP Software.
Can’t access the FPP device during **Wi-Fi Tether** installation.	1. The FPP device has no power.2. The FPP device isn’t broadcasting the FPP network.3. The FPP network is broadcasting, but you can’t connect to it.4. The image on the uSD card is corrupt.5. The uSD card itself is faulty.	1. Check the power connection and confirm the power LED is lit.2. Make sure the FPP device isn’t already connected via Ethernet or USB to anything — it won’t broadcast its own Wi-Fi network while tethered another way. Confirm the device actually has a Wi-Fi adapter, and that the adapter supports Wi-Fi tethering (most USB Wi-Fi adapters don’t; the on-board adapters on Raspberry Pi and BeagleBone wireless models do).3. Your Wi-Fi adapter likely doesn’t support Wi-Fi tethering — see above.4. Format and re-image the uSD card — see Installing the FPP Software.5. Format and re-image a known-good uSD card — see Installing the FPP Software.

Tip: Most of these installation problems trace back to one of two things — a faulty uSD card/cable, or the wrong address for the connection method you’re using. Re-imaging a known-good card and double-checking the address (`192.168.7.2`, `192.168.6.2`, `http://fpp.local/`) resolves the majority of them.