
Raspberry Pi Audio Receiver

A simple, light weight audio receiver with Bluetooth (A2DP), AirPlay 2, and Spotify Connect.

Features

Devices like phones, tablets and computers can play audio via this receiver.

Requirements

- A USB Bluetooth dongle (the internal Raspberry Pi Bluetooth chipset turned out as not suited for audio playback and causes all kinds of strange connectivity problems)
- Raspberry Pi OS 12 Lite
- Internal audio, HDMI, USB or I2S Audio adapter (tested with Adafruit USB Audio Adapter, pHAT DAC, and HifiBerry DAC+)

Again: do not try to use the internal Bluetooth chip, this will only bring you many hours of frustration.

Installation

The installation script asks whether to install each component.

```
1 wget https://raw.githubusercontent.com/nicokaiser/rpi-audio-receiver/main/install.sh
2 bash install.sh
```

Note: the installation process is not reversible, there is no uninstall. The script is meant to be run on a clean device that is not used for anything else.

Basic setup

Lets you choose the hostname and the visible device name (“pretty hostname”) which is displayed as Bluetooth name, in AirPlay clients and in Spotify.

Bluetooth

Sets up Bluetooth, adds a simple agent that accepts every connection, and enables audio playback through ALSA. A udev script is installed that disables discoverability while connected.

AirPlay 2

Installs Shairport Sync AirPlay 2 Audio Receiver.

Spotify Connect

Installs Rasptotify, an open source Spotify client for Raspberry Pi.

Additional steps

Enable HiFiBerry device

When using a HiFiBerry or similar I2C device, a device tree overlay needs to be enabled in `/boot/firmware/config.txt` (replace `dacplus` with the overlay that fits your hardware):

```
1 ...
2 dtoverlay=hifiberry-dacplus
```

To enable the software volume mixer, `/etc/asound.conf` needs to be created:

```
1 defaults.pcm.card 0
2 defaults.ctl.card 0
3
4 pcm.hifiberry {
5     type hw
6     card 0
7     device 0
8 }
9 pcm.dmixer {
10     type dmix
11     ipc_key 1024
12     ipc_perm 0666
13     slave.pcm "hifiberry"
14     slave {
15         period_time 0
16         period_size 1024
17         buffer_size 8192
18         rate 44100
19         format S32_LE
20     }
21     bindings {
22         0 0
23         1 1
24     }
25 }
26 ctl.dmixer {
```

```
27  type hw
28  card 0
29  }
30  pcm.softvol {
31  type softvol
32  slave.pcm "dmixer"
33  control {
34  name "Softvol"
35  card 0
36  }
37  min_dB -90.2
38  max_dB 0.0
39  }
40  pcm.!default {
41  type plug
42  slave.pcm "softvol"
43  }
```

Read-only mode

To avoid SD card corruption when powering off, you can boot Raspberry Pi OS in read-only mode. This can be achieved using the `raspi-config` script (in the “Performance” section).

Disable Wi-Fi power management

Disabling Wi-Fi power management might resolve some connection issues:

```
1 sudo nmcli connection modify preconfigured wifi.powersave 2
```

Disable internal Bluetooth and Audio

When an external audio device (HDMI, USB, I2S) is used, the internal audio can be disabled in `/boot/firmware/config.txt` (replace `hifiberry-dacplus` with the overlay which fits your installation):

```
1 ...
2 dtoverlay=disable-bt
3 dtparam=audio=off
4 dtoverlay=vc4-kms-v3d,noaudio
5 dtoverlay=hifiberry-dacplus
```

Add Bluetooth devices

The device should be visible for new Bluetooth connections, but in some cases you might need to pair them manually:

```
1 sudo bluetoothctl
2 power on
3 agent on
4 # Now search for available bluetooth devices from your device
5 # Once paired note down the MAC address of your device
6 trust 00:00:00:00:00:00 # Put device MAC address here so after reboot
   it can automatically re-connect again
```

Disable Wi-Fi when Bluetooth is connected

Disclaimer: You really might want to use a Bluetooth USB dongle. The internal Bluetooth module of the Raspberry Pi (all versions) is very limited and using it for audio transmission will most certainly lead to problems and unexpected behaviour, see [raspberrypi/linux/#1402](#).

If you really want to use the internal Bluetooth module, you almost certainly need to disable Wi-Fi.

Modify `/usr/local/bin/bluetooth-udev` and remove the comments (#) around the `ifconfig` calls:

```
1 ...
2 if [ "$action" = "add" ]; then
3     ...
4     # disconnect wifi to prevent dropouts
5     ifconfig wlan0 down &
6 fi
7
8 if [ "$action" = "remove" ]; then
9     # reenale wifi
10    ifconfig wlan0 up &
11    ...
12 fi
```

Bluetooth A2DP volume

To enable A2DP volume control, add the `--plugin=a2dp` parameter to the `bluetoothd` command line. This helps setting the volume via Bluetooth, but does not work on all setups.

```
1 # Enable A2DP volume control
2 mkdir -p /etc/systemd/system/bluetooth.service.d
3 cat <<'EOF' > /etc/systemd/system/bluetooth.service.d/override.conf
```

```
4 [Service]
5 ExecStart=
6 ExecStart=/usr/libexec/bluetooth/bluetoothd --plugin=a2dp
7 EOF
```

Bluetooth Fast Connectable

Using the `FastConnectable` flag may lead to faster Bluetooth connections, but may also lead to poor sound quality. You can try and see if it works for you. See #70

Add the flag to the `General` section in `/etc/bluetooth/main.conf`:

```
1 [General]
2 ...
3 FastConnectable = true
4 ...
```

Bluetooth pairing with PIN

To enable pairing with a PIN code instead of Simple Secure Pairing mode, the following steps are required:

1. Change `sspmode 1` to `sspmode 0` in `/etc/systemd/system/bt-agent@.service`
2. Add `--pin /etc/bluetooth/pin.conf` to the `ExecStart` line in `/etc/systemd/system/bt-agent@.service`
3. Add a file `/etc/bluetooth/pin.conf` which contains PIN code for the devices:

`AA:BB:CC:DD:EE:FF 1234` (replace `AA:BB:CC:DD:EE:FF` with your Bluetooth devices's Mac address or `*` to use PIN 1234 for all devices)

Notes

- **This does not work with iOS devices.** iOS refuses to pair with devices that do not support SSP. So `sspmode` needs to be on
- Even with `sspmode=1` and a PIN file, iOS would not connect at all
- macOS allows pairing with devices with `sspmode=0` (and a PIN configured on the Pi). When `sspmode=1`, macOS decides upon a PIN and either the Pi has a fixed PIN list (which does not match => connection refused), or requires keyboard input on the Pi (which is not desired for a headless device).

So you need to try yourself if this works with your setup.

Limitations

- Only one Bluetooth device can be connected at a time, otherwise interruptions may occur.
- The device is always open, new clients can connect at any time without authentication.
- To permanently save paired devices when using read-only mode, the Raspberry has to be switched to read-write mode until all devices have been paired once.
- You might want to use a Bluetooth USB dongle or have the script disable Wi-Fi while connected (see [bluetooth-udev](#)), as the BCM43438 (Raspberry Pi 3, Zero W) has severe problems with both switched on, see [raspberrypi/linux/#1402](#).
- The Pi Zero may not be powerful enough to play 192 kHz audio, you may want to change the values in [/etc/asound.conf](#) accordingly.

Disclaimer

These scripts are tested and work on a current Raspberry Pi OS setup on Raspberry Pi. Depending on your setup (board, configuration, sound module, Bluetooth adapter) and your preferences, you might need to adjust the scripts. They are held as simple as possible and can be used as a starting point for additional adjustments.

Upgrading

This project does not really support upgrading to newer versions of this script. It is meant to be adjusted to your needs and run on a clean Raspberry Pi OS install. When something goes wrong, the easiest way is to just wipe the SD card and start over. Since apart from Bluetooth pairing information all parts are stateless, this should be ok.

Updating the system using [apt-get upgrade](#) should work however.

Contributing

Package and configuration choices are quite opinionated but as close to the Debian defaults as possible. Customizations can be made by modifying the scripts, but the installer should stay as simple as possible, with as few choices as possible. That said, pull requests and suggestions are of course always welcome. However I might decide not to merge changes that add too much complexity.

Related projects

There are many forks and similar projects that are optimized for more specific requirements.

-
- [Arcaria197/rpi-audio-receiver](#) - a fork that uses Raspbian 10 (legacy) and runs on Raspberry Pi Zero W hardware
 - [HiFiBerryOS](#) - a more sophisticated approach on this, using an entirely custom (buildroot) ecosystem

References

- [Shairport Sync](#): AirPlay 2 audio player
- [Raspotify](#): A Spotify Connect client that mostly Just Works™

License

MIT