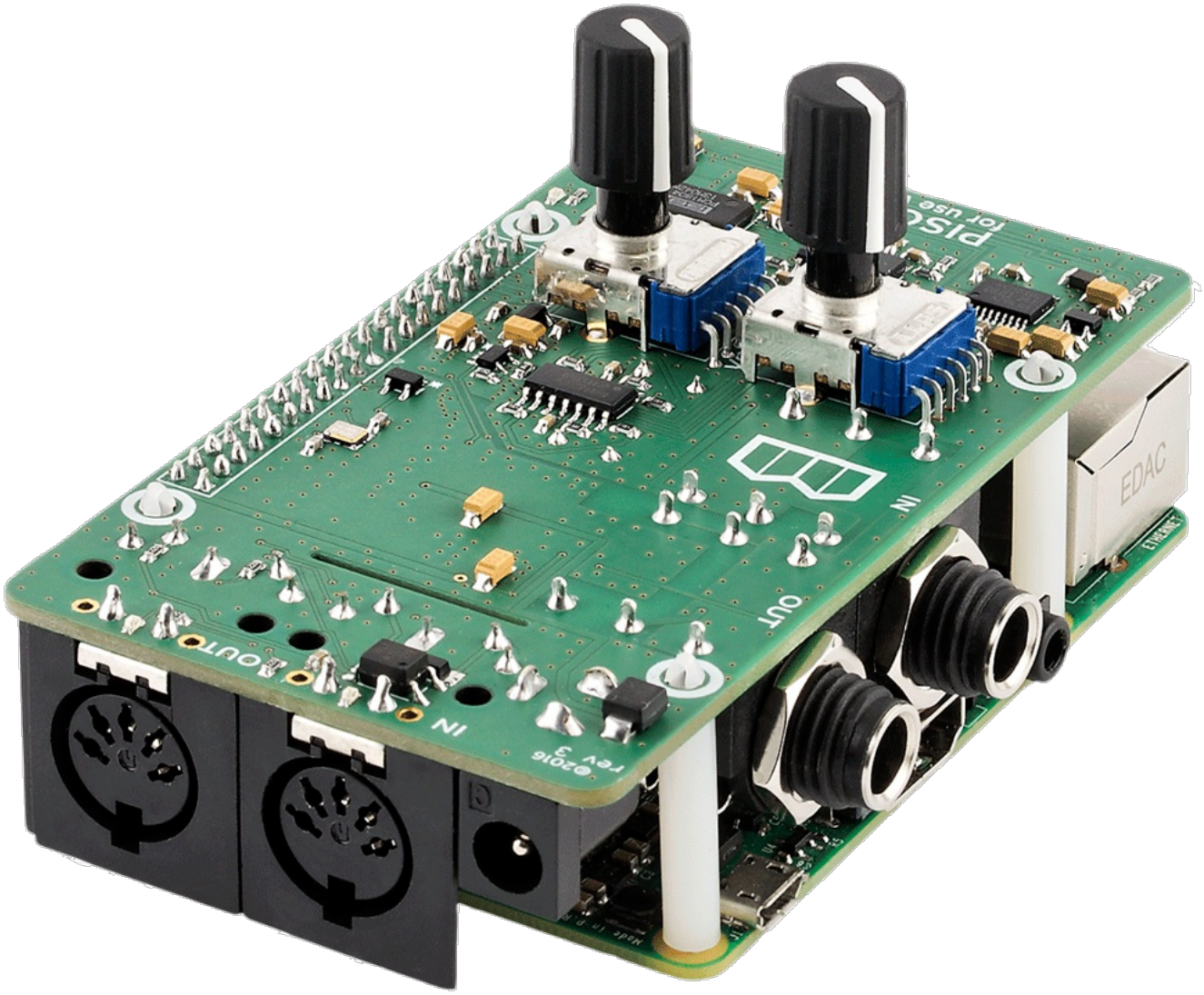
Documentation

Table of contents

Home

Getting Started



pisound is an ultra-low latency high-quality soundcard and MIDI interface specially designed for Raspberry Pi pocket computers. Equipped with 192kHz 24-bit Stereo Input and Output driven by the legendary Burr-Brown chips, DIN-5 MIDI Input and Output ports, user-customizable button and bundled software tools, it has everything you need to bring your audio projects to life in no time.

Hardware Setup

Mount pisound on top of your Raspberry Pi via the 40-pin header and fasten it with the screws provided while the RPi is unpowered, so it appears as in the image at the top.

Driver Setup

If you don't have any Linux OS running on your Raspberry Pi, we suggest starting with Raspbian.

Power your Raspberry Pi up and install the driver by running the below commands in a terminal window:

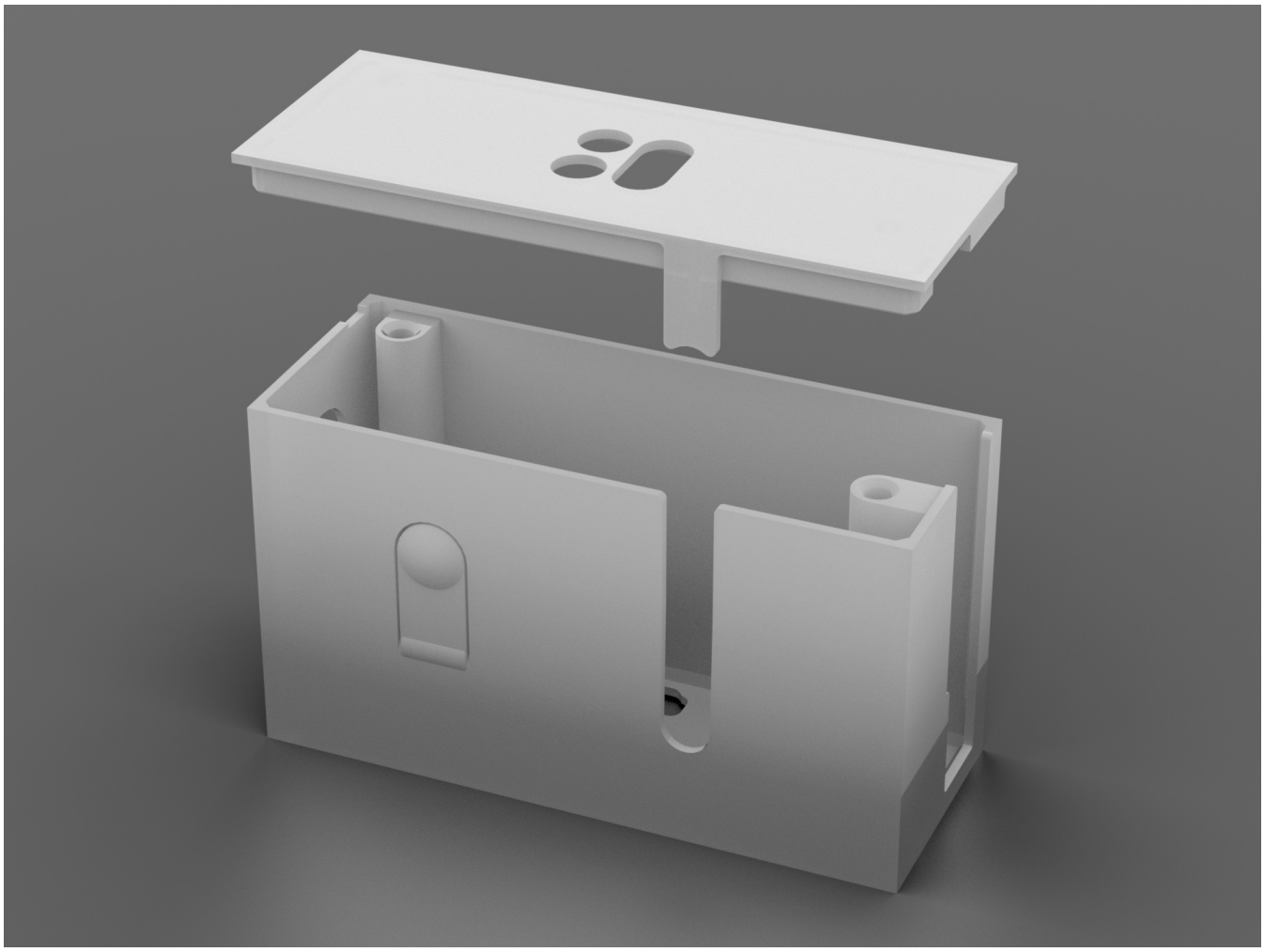
```
wget http://blokas.io/pisound/install-pisound.sh -O install-pisound.sh
chmod +x install-pisound.sh
./install-pisound.sh
```

After driver installation process is complete, reboot your Raspberry Pi. That's it. Now you can choose pisound as your main sound card through native Volume widget and use it with any Linux Audio/MIDI software. Done! Thank You!

If you hit any issues during driver setup, please provide us feedback.

Print Your Own Case

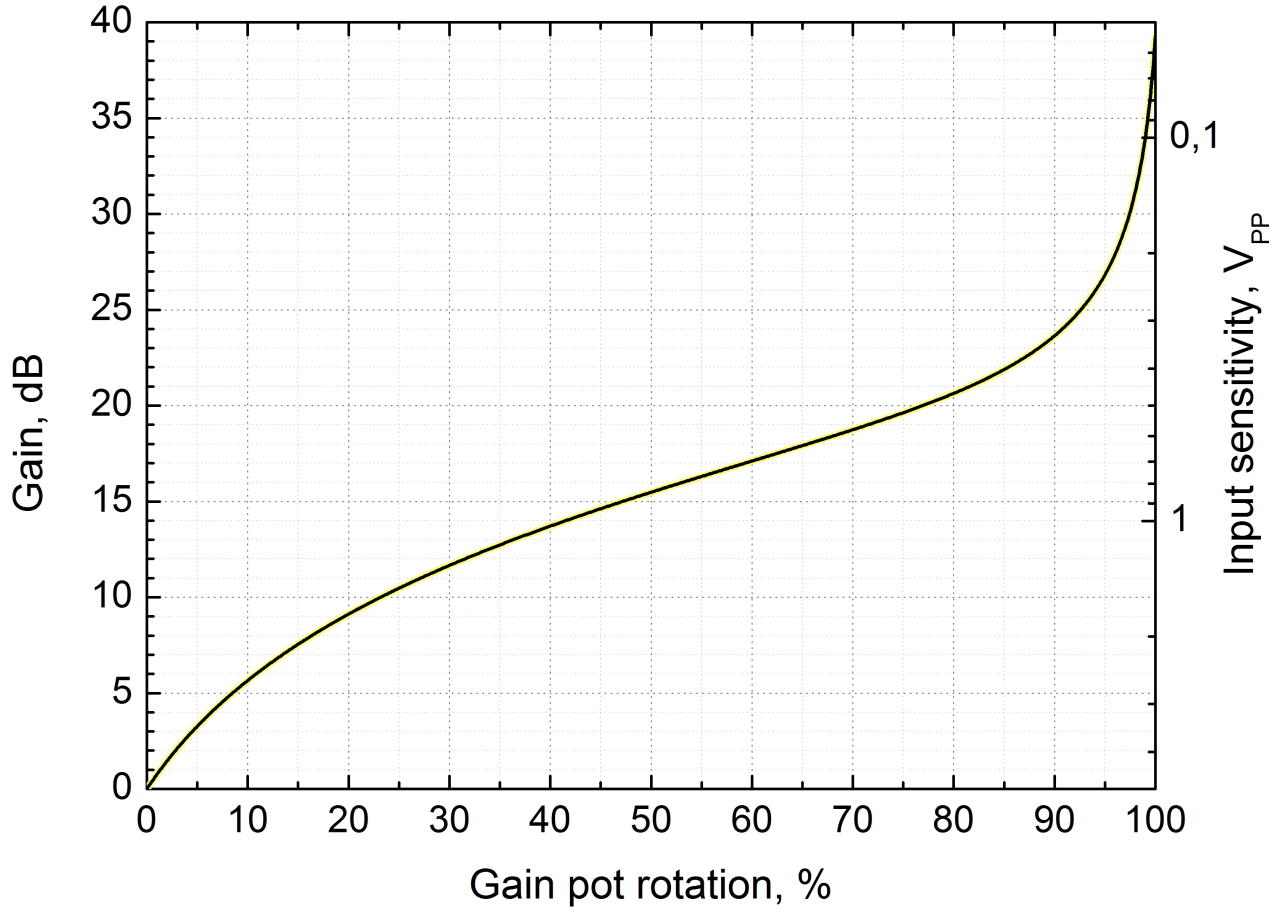
To take the setup process one step further, you can 3D-print your own case. All necessary files can be found here.



Audio

Audio Input

There is one unbalanced stereo input accessible via 1/4" (6.35mm) jack slot on pisound. One stereo channel can also be used as two unbalanced mono channels. Audio inputs are AC coupled via metalized polypropylene capacitors to a gain stage built using OPA4134 op-amps. Input resistance is 100kOhm for each channel. The gain can be adjusted simultaneously for both the left and the right channels from 0dB to +40dB with an on-board potentiometer. The maximum audio signal level before clipping is 5Vpp (at 0dB gain). The range of the gain adjustment can be divided into two sections. The first section occurs at rotation between 0% and 80% and it is used to precisely adjust for the high-level signals (line out, headphone amp out, etc...). The tight section at the maximum rotation of the gain pot acts as a +20dB switch for the low-level signals (guitar, microphone, etc.). When the signal clipping occurs in any channel, the red LED lights up and fades out after last clipped sample. Audio to digital conversion is carried out by PCM1804 converter. An on-board clock oscillator delivers a clock signal to the ADC, which divides it according to the selected sample rate. ADC acts as the master of I2S line. pisound supports three sample rates: 48kHz, 96kHz and 192kHz. A filter at the input stage of PCM1804 ensures good anti-aliasing.



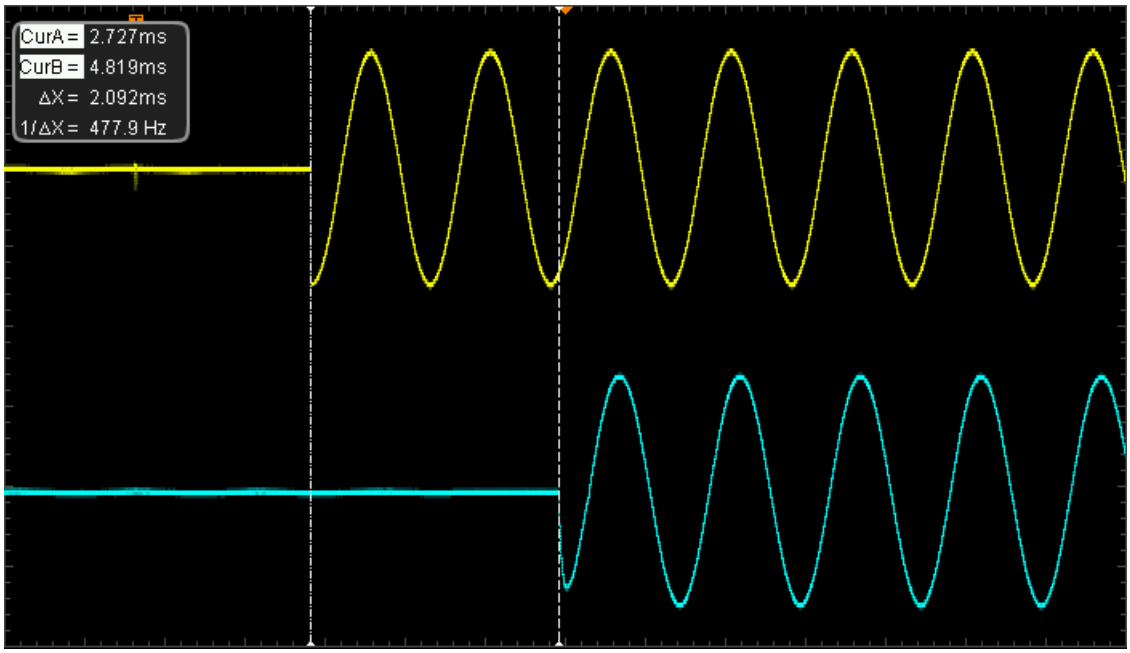
Gain and input sensitivity versus position of **GAIN** pot

Audio Output

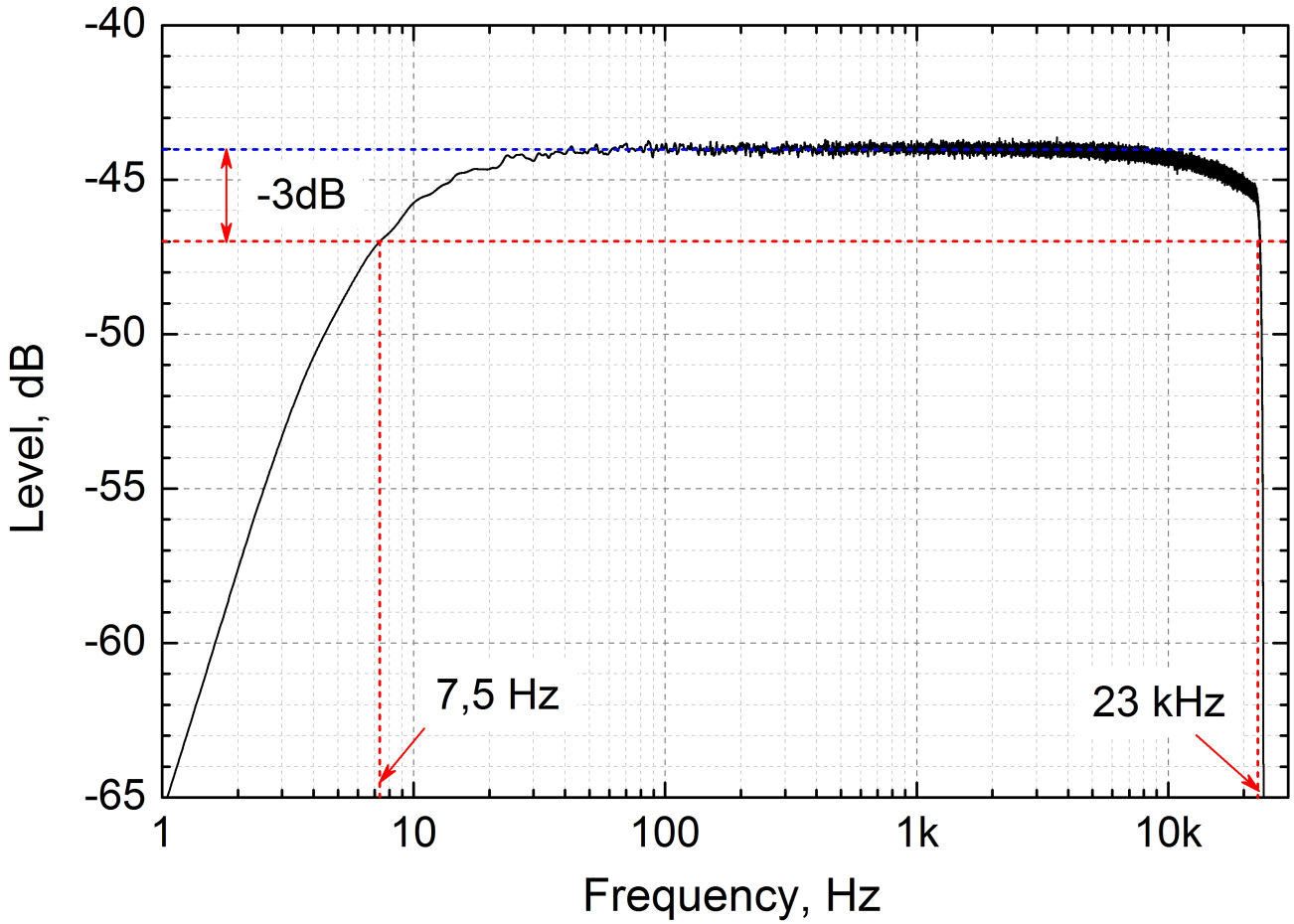
Audio output is DC coupled and can be accessed via the female 1/4" (6.35mm) stereo jack connector. Output volume level can be adjusted with on-board potentiometer. Maximum output level is 2.1Vrms when driving 1kOhm load. Digital to audio conversion is done in PCM5102A converter which has both Signal-to-Noise ratio and dynamic range of 112dB. The DAC acts as a slave on I2S line and the sampling rate is dictated by the ADC. An intelligent muting system is used to prevent pops and clicks when disconnecting power supply.

Audio Latency and Other Parameters

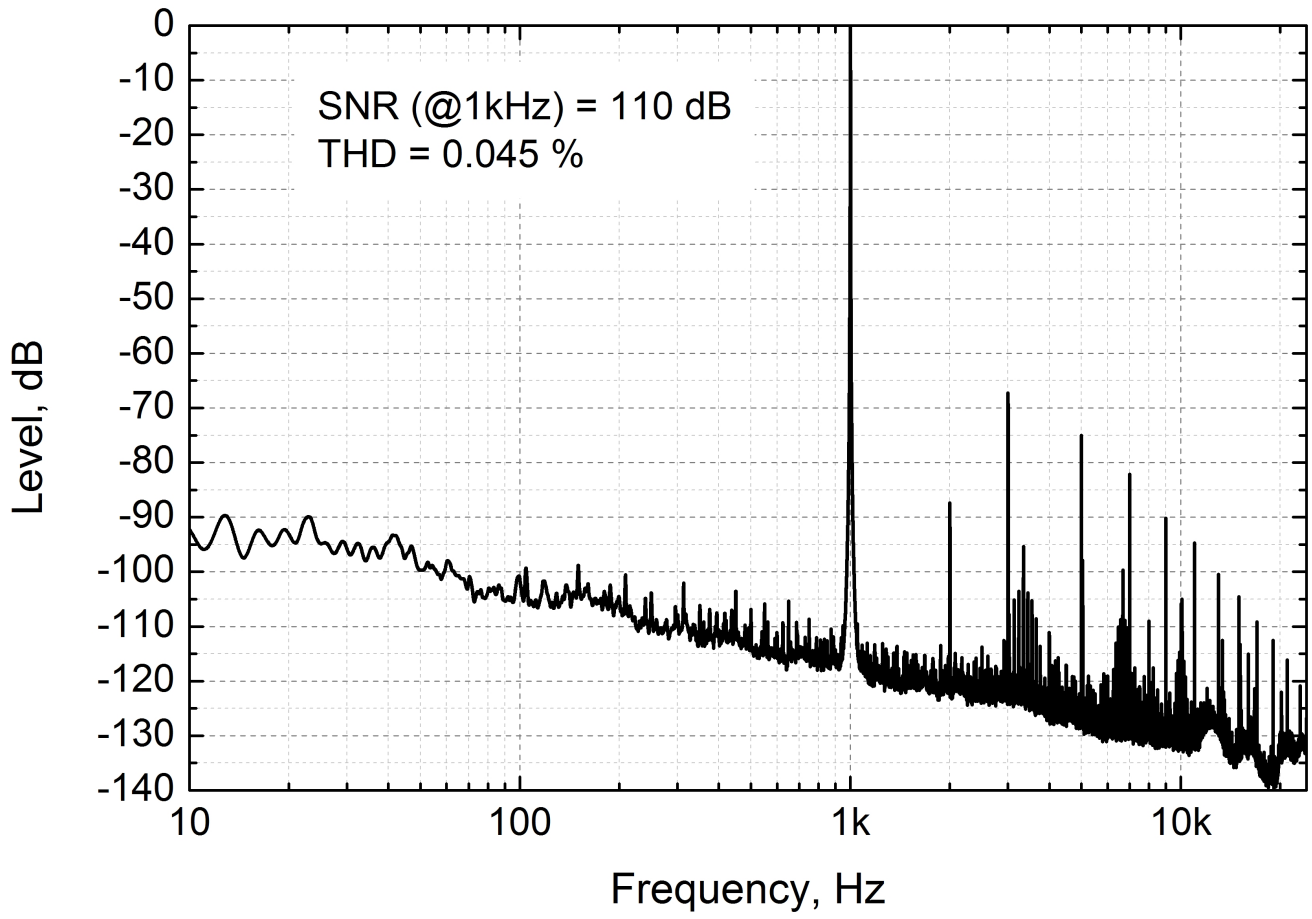
In digital audio equipment it takes time for the signal at input to be processed and delivered at output. This time is called audio latency. There's three parts to it. The first is the time required for the ADC to do the conversion and to send digital data to the processing unit. The second step is to process data and prepare it for the transfer to the DAC. And the final part is getting the data to the DAC and converting it to the analog signal. The most time consuming is the second part. Parts one and three often require no more than 1 ms depending on architecture of ADC and DAC, sample rate and in-built digital filters.



An oscillogram showing audio latency of 2.092 ms. pisound and Raspberry Pi 2 working at sample rate of 192 kHz



A spectrum of looped white noise signal showing the bandwidth of the pisound



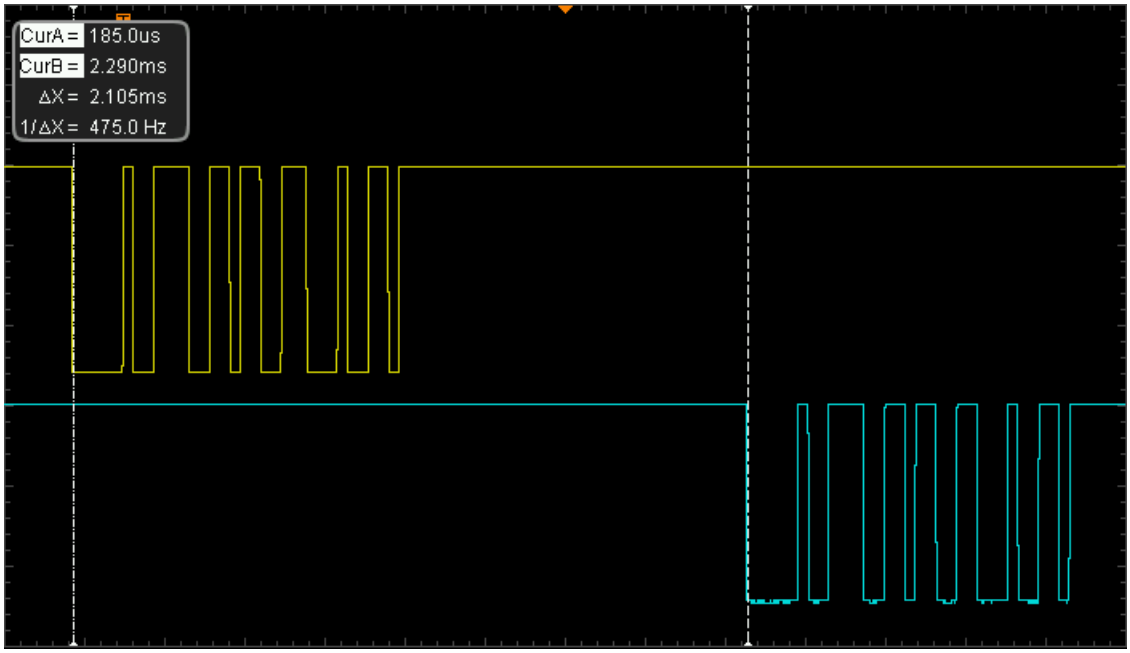
A spectrum of looped 1 kHz sine signal showing the THD of the pisound

MIDI

A standard MIDI interface is available via two female DIN-5 connectors. Unlike usual MIDI solutions for Raspberry Pi, MIDI on pisound is implemented using high speed SPI and a dedicated microcontroller for translating SPI data to serial MIDI byte streams and it's readily recognized in audio software as an ALSA MIDI device. The loopback latency of MIDI was measured to be 2.105ms. In addition, there are MIDI activity LEDs for both Input and Output indicating the flow of MIDI events.

In addition, pisound lets you take advantage of WiFi-MIDI. When WiFi Hotspot mode is enabled via triple clicking The Button, touchosc2midi daemon gets launched. It translates OSC messages to MIDI events so you can control audio software running on Raspberry Pi from your smartphone or tablet.

And of course you can use USB-MIDI devices as usual by connecting them to Raspberry Pi USB ports.



An oscillogram showing signal delay between MIDI input (yellow) and MIDI output (cyan) of 2.105 ms when echoing

Software

Drivers

The support software for pisound consists of two pieces - the Linux kernel module and user-space pisound-btn daemon. The kernel module implements the soundcard as an ALSA Input / Output / Raw MIDI device.

The pisound button daemon is a user space program which implements monitoring of The Button on the board by registering a GPIO interrupt handler. Therefore it takes minimal CPU resources, but is still able to react to button pushes just at the moment it was interacted with. Read more on The Button functionality below.

You can find the source code for The Button here and kernel module here.

Installing

To install the user-space button daemon and enable pisound, run the below commands in a terminal.

```
wget http://blokas.io/pisound/install-pisound.sh -O install-pisound.sh
chmod +x install-pisound.sh
./install-pisound.sh
```

The above installs a button daemon named 'pisound-btn' and its scripts for button actions. If there were existing scripts, they are first backed up to `/usr/local/etc/pisound/backups/<current date>`, so if you had overridden the default functions, you will have to restore your scripts manually.

The install-pisound.sh script does the following steps:

Verifying It Works

Once the system boots up, run a terminal and run:

```
amidi -l
arecord -l
aplay -l
```

You should see output similar to:

```
pi@raspberrypi:~ $ amidi -l
Dir Device    Name
IO  hw:1,0    pisound MIDI PS-10JAD4Q
pi@raspberrypi:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: pisound [pisound], device 0: PS-10JAD4Q snd-soc-dummy-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
pi@raspberrypi:~ $ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: pisound [pisound], device 0: PS-10JAD4Q snd-soc-dummy-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Feedback

In case you're having difficulties with getting pisound's driver to run, contact us and the community here: http://community.blokas.io/, provide the exact error and the last command you've executed.

Compatible Software

pisound is compatible with virtually all Linux distributions and software as it comes with an ALSA audio and MIDI driver integrated in mainline Raspbian Linux kernel (ver. 4.4.27+).

Please add or let us know if you have pisound working on a distribution that is not in the list yet!

- Raspbian
- arch linux
- Ubuntu Mate

Software we tested which required no special changes, in alphabetic order:

- Audacity
- Carla / LV2
- darkice and icecast
- Jack
- Supercollider
- Pure Data

Software which required tweaks or workarounds, in alphabetic order:

- Sonic Pi
- Volumio

Sonic Pi workaround guide

Sonic Pi has a special case for Raspberry Pi during startup. By default it kills the existing jackd server, and starts one configured to use the built in Raspberry Pi audio with hardcoded parameters. That makes Sonic Pi unusable with other audio cards, unless the below workaround is applied, so Sonic Pi would behave in the same way as if it's run on any other linux device.

Do this once (might need to be done again in case any software updates touch it):

- Edit `/opt/sonic-pi/app/server/sonicpi/scsynthexternal.rb` as root user using your favorite editor (such as vim, nano, geany, or ...).
- Locate the following code snippet:

```
case os
when :raspberry
  boot_server_raspberry_pi
when :linux
  boot_server_linux
when :osx
  boot_server_osx
when :windows
  boot_server_windows
end
true
```

- Change it to:

```
case os
when :raspberry
  #boot_server_raspberry_pi
  boot_server_linux
when :linux
  boot_server_linux
when :osx
  boot_server_osx
when :windows
  boot_server_windows
end
true
```

Do this every time you want to start Sonic Pi:

- Start Jack server configured to use pisound.
- Start Sonic Pi, it will connect to the Jack server you had just launched.

The Button

The Button is a customizable button on the pisound board. There's 4 possible interactions with it:

1. Press, hold and release after 1 or more seconds passed. Executes `/usr/local/etc/pisound/hold.sh`
2. Click once. Executes `/usr/local/etc/pisound/single_click.sh`
3. Double click. Executes `/usr/local/etc/pisound/double_click.sh`
4. Triple click. Executes `/usr/local/etc/pisound/triple_click.sh`

Every **down** and **up** movement will also execute `/usr/local/etc/pisound/down.sh` and `/usr/local/etc/pisound/up.sh` respectively.

You may modify these scripts to do whatever you wish.

Note: The pisound-btn daemon process must be running in order for the button to work. The 'make install' automatically adds it to the Desktop autostart.

Default Hold Action

By default, holding the button down for more than 1 second and releasing will cause the system to shutdown cleanly. The MIDI activity LEDs will blink once to confirm that the hold action was triggered. It is safe to unplug the power supply once RPi's LEDs stop flashing on and off.

Default Single Click Action

By default, clicking the button once runs a script that scans the attached media storage devices for '**main.pd**', or, if not found in external media, scans `/usr/local/etc/pisound-patches/`, kills all existing instances of Pure Data, then starts a new instance opening the file, enables audio and connects all the detected MIDI inputs and outputs to the Pure Data's virtual MIDI ports. ALSA engine of Pure Data is used.

The script will blink the MIDI Activity LEDs once just after it reacted to the click, and again 2 times after it succeeded launching the patch, or one long duration blink if an error occurred or no patch was found.

If you don't have Pure Data installed, chances are you can install it by running:

```
sudo apt-get install puredata
```

Default Double Click Action

Double clicking will stop all Pure Data instances and unmount all attached external media, so it can be safely removed.

Default Triple Click Action (RPi3 or using SoftAP capable WiFi USB adapter)

Triple-clicking will reconfigure the WiFi of RPi3 board or an external SoftAP capable USB WiFi adapter to behave as an Access Point (a.k.a. Wireless Router), as well as start 'touchosc2midi' monitor which will be ready to listen and forward MyOsc data as MIDI to other software such as puredata.

By default, the AP will appear as '**pisound**', and the default password is '**blokaslabs**' (without quotes). You can change the name and password by modifying `/usr/local/etc/pisound/hostapd.conf` Keep in mind that this file gets backed up and overwritten after reinstalling or updating the pisound-btn software.

The default IP address of RPi3 is 172.24.1.1 which you can use for ssh, VNC or wireless OSC / MIDI data! That means that you can easily interact with your system using just your laptop or phone, no more wires apart from power supply is needed! And it gets better, if the LAN cable is connected to RPi, it will share the Internet with the connected devices.

To access RPi using ssh and/or VNC, make sure they're enabled in `raspi-config`. To enable, run in terminal:

```
sudo raspi-config
```

Go to **Interfacing Options** and make sure **ssh** and **VNC** are enabled. If you don't see these options, go to **Advanced Options** and do **Update**.

Triple-clicking again will revert to regular WiFi behavior.

Specifications

The shield itself conforms to Raspberry Pi's Hardware Attached on Top (HAT) specifications and connects to Pi via the 40-pin header. The shield is slightly bigger in length (56x100 mm) than RPi itself. It has two female DIN-5 connectors for MIDI in/out and two 1/4" (6.35mm) stereo jack connectors for stereo audio in/out. There are two pots for gain and volume control, a programmable button and MIDI activity and input clip LEDs.

Audio

Parameter	Conditions	Value
Connectors type	-	1/4" (6.35mm) stereo jacks
Sampling frequency (Fs)	-	48kHz, 96kHz, 192kHz
Input/Output resolution	-	24bit
Input/Output SNR@1kHz	G = 0 dB	110dB
Input impedance	-	100kOhm ll 2pF
Input gain (G)	-	0dB to +40dB
Input clip LED	-	Yes
Input clip voltage	G = 0 dB	5V (peak to peak)
Full scale output	Load impedance > 1 kOhm	0V to 2.1V (RMS)
Loopback bandwidth (-3 dB)	G = 0 dB, Fs = 48 kHz	7.5Hz - 23kHz
Loopback THD@1kHz	G = 0 dB, Fs = 48 kHz	< 0.045%
Loopback latency	Fs = 192 kHz, RPi2, buffer size = 128 frames	2.092ms
Phantom power	-	None

MIDI

Parameter	Value
Connectors type	DIN-5 sockets
MIDI loopback latency	2.105ms
Activity LEDs	Input & Output

Other

Parameter	Value
Current Draw	< 300mA @ 5.1VDC
Dimensions	56mm x 100mm
Weight	67g

Power Supply

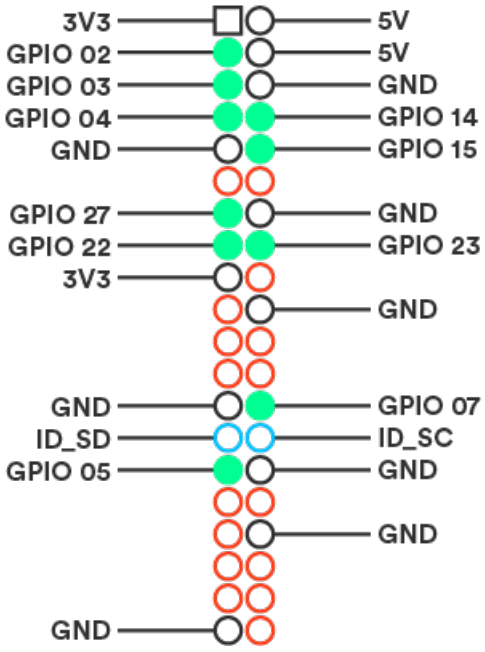
There are two versions of pisound regarding power supply:

- **9V version (beta):** 7.2V - 12.6V, 18W minimum. 5.5x2.1 mm coaxial power jack connector. The inner connector is connected to the positive terminal, and the sleeve is connected to the ground. The power adapter connected to pisound supplies the RPi board too, so RPi does not need to have its USB supply port connected. The pisound itself has a power consumption of about 1.8W. A 9VDC power supply capable of delivering at least 2 Amps of current is recommended for this version.
- **5.1V version (latest):** pisound has no power connection and requires no additional power supply. It powers up from RPi power supply via pins on RPi header. pisound consumes no more than 300mA at 5.1VDC. When using this version of pisound, we recommend to use the official 5.1VDC RPi power supply.

Supported Raspberry Pi Models

Compatible models	
Raspberry Pi 1 Model A+	
Raspberry Pi 1 Model B+	
Raspberry Pi 2	
Raspberry Pi 2 version 1.2	
Raspberry Pi 3	
Raspberry Pi Zero version 1.2	
Raspberry Pi Zero version 1.3	

Pinout of Pi Header



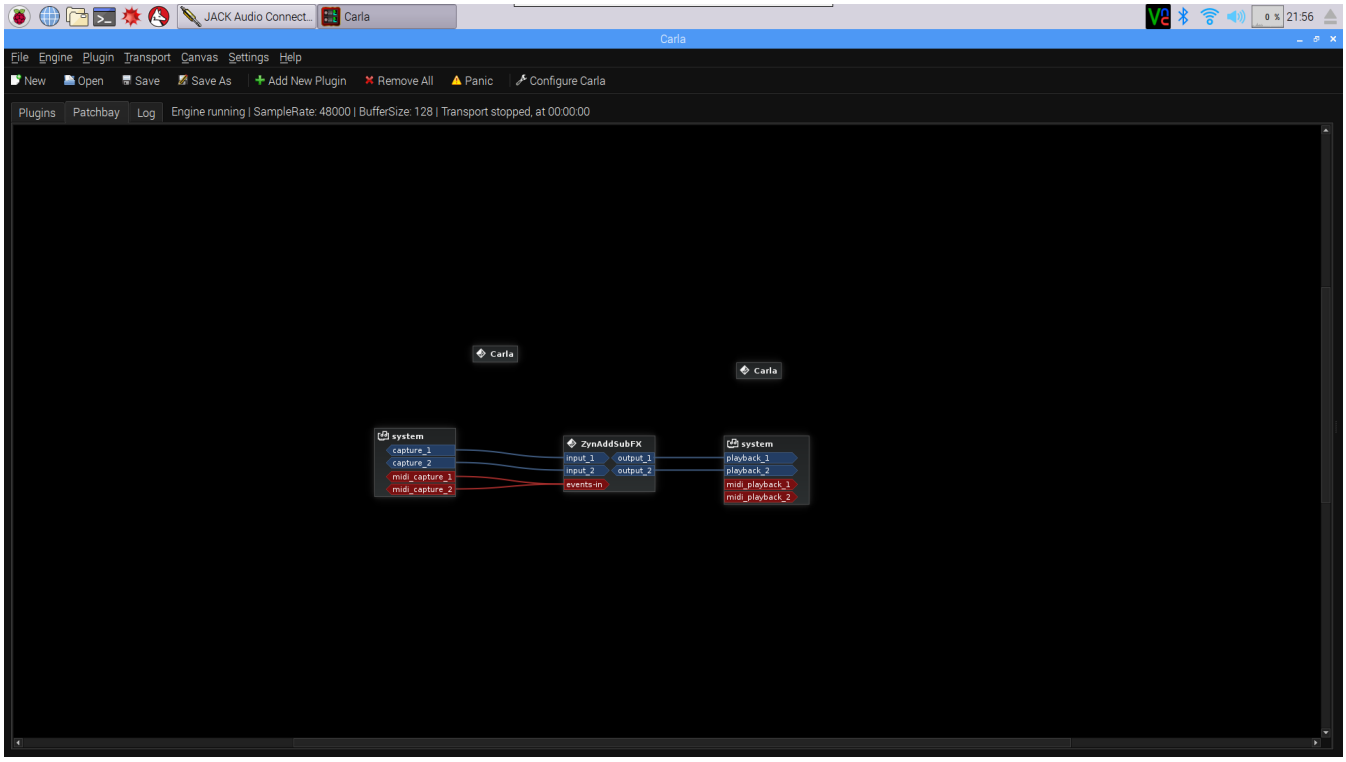
- Black - Power supply pins.
- Red - Pins used by pisound.
- Green - Pins available for your use.
- Blue - Pins reserved for Raspberry Pi hats use.

Projects

Projects using pisound

pisound and LV2 plugins

On http://rpi.autostatic.com/ repository you can find some cool software packages for audio. One of those is **Carla** - a host for LV2 plugins.



So let's give it a go.

First we have to add the repository to apt-get:

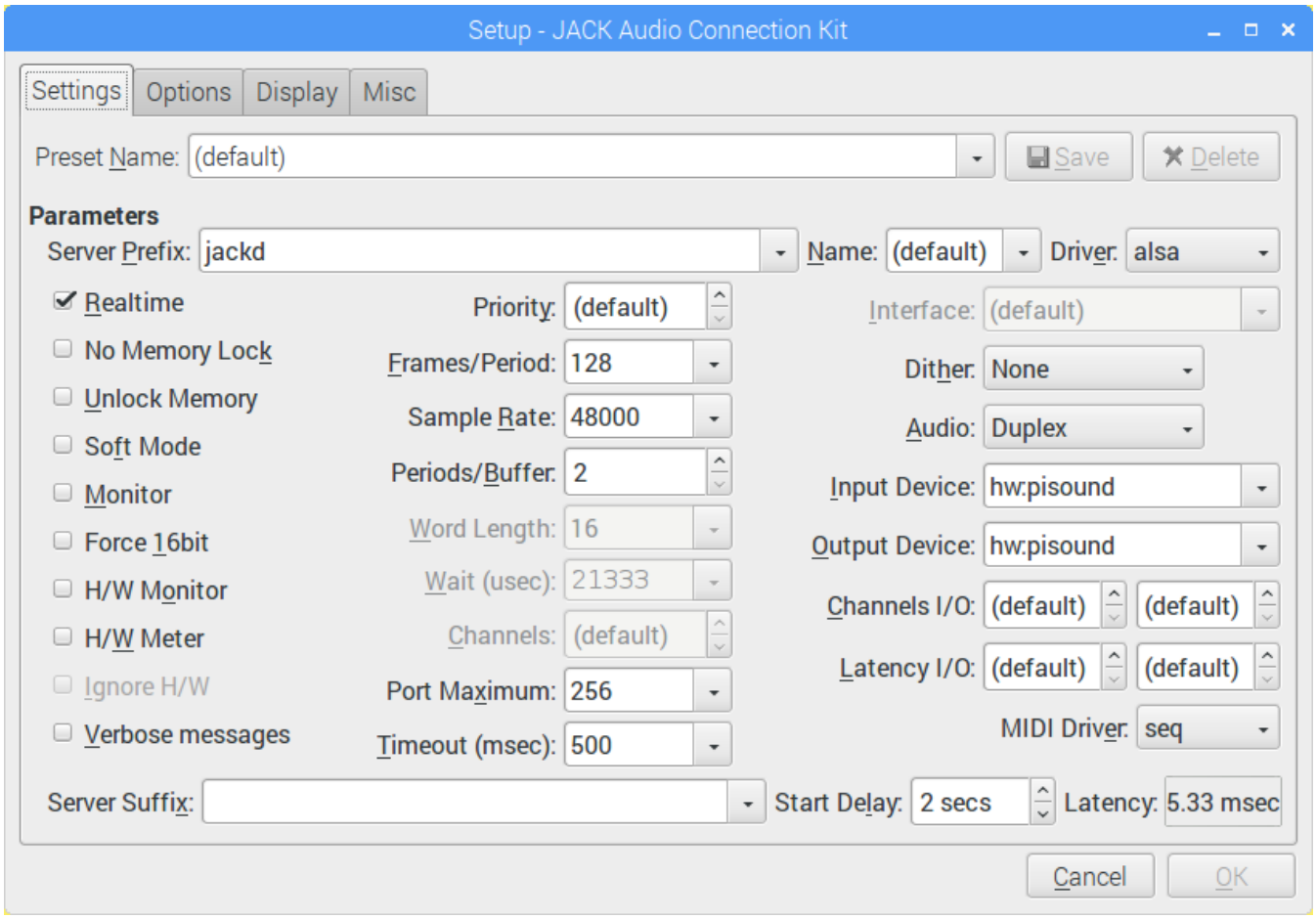
```
wget -q -O - http://rpi.autostatic.com/autostatic.gpg.key | sudo apt-key add -
sudo wget -q -O /etc/apt/sources.list.d/autostatic-audio-raspbian.list http://rpi.autostatic.com/autostatic-audio-raspbian.list
sudo apt-get update
```

After that's done, let's install **Carla** as well as **jackd** server and **qjackctl** for configuring it:

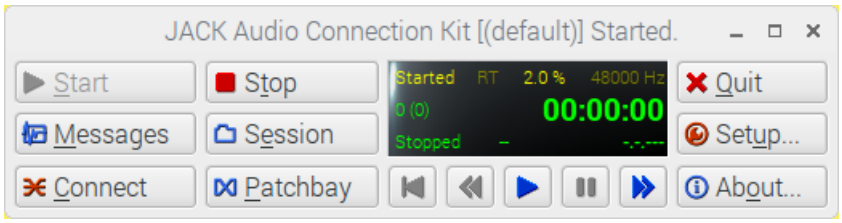
```
sudo apt-get install carla jackd qjackctl
```

Then start **qjackctl**, you can do that by hitting Ctrl+F2 and typing in **qjackctl** and pressing enter.

Configure it in Setup... like so:



And start the jack server by pressing Start:



Now run Carla (Alt+F2, carla, enter)

Click Configure Carla and make sure that Audio Driver is set to JACK. It will connect to the already started Jack server. For process mode I've used Single Client, but you can use any option you like.

Then you can add plugins in Plugins section, and patch them up in Patch Bay section.

The top screenshot shows **ZSynAddSubFX** connected to pisound's Input and Output, as well as MIDI data routed to the synth plugin.

There's a couple of plugins readily available to play around within Carla, and you should be able to find more LV2 plugins built for Raspberry Pi to experiment with on the Internet.

Internet Radio Station

Using **icecast** and **darkice** you can easily setup your own radio station and broadcast whatever is connected to the pisound input!

First let's get the required software installed:

```
sudo apt-get install icecast2 darkice
```

Then we have to configure the darkice, feel free to customize to your liking.

Save the below contents to `/etc/darkice.cfg`

```
# this section describes general aspects of the live streaming session
[general]
duration      = 0               # duration of encoding, in seconds. 0 means forever
bufferSecs    = 2               # size of internal slip buffer, in seconds
reconnect     = yes             # reconnect to the server(s) if disconnected

# this section describes the audio input that will be streamed
[input]
device        = hw:1,0          # Alsa soundcard device for the audio input
sampleRate    = 48000           # sample rate in Hz. try 48000, 96000 or 192000
bitsPerSample = 16              # bits per sample. try 16
channel       = 2               # channels. 1 = mono, 2 = stereo.
                                # Only stereo mode is supported by pisound.

# this section describes a streaming connection to an IceCast2 server
# there may be up to 8 of these sections, named [icecast2-0] ... [icecast2-7]
# these can be mixed with [icecast-x] and [shoutcast-x] sections
[icecast2-0]
bitrateMode   = cbr             # variable bit rate
bitrate       = 128
format        = mp3             # format of the stream: mp3
quality       = 0.8             # quality of the stream sent to the server
server        = localhost       # host name of the server
port          = 8000            # port of the IceCast2 server, usually 8000
password      = hackme          # source password to the IceCast2 server
mountPoint    = pisound         # mount point of this stream on the IceCast2 server
name          = pisound         # name of the stream
description   = DarkIce on pisound # description of the stream
url           = http://localhost # URL related to the stream
genre         = my genre        # genre of the stream
public        = no              # advertise this stream?
#localDumpFile = recording.mp3   # Record also to a file
```

And finally, execute 'darkice' in a terminal or using Alt+F2:

```
darkice
```

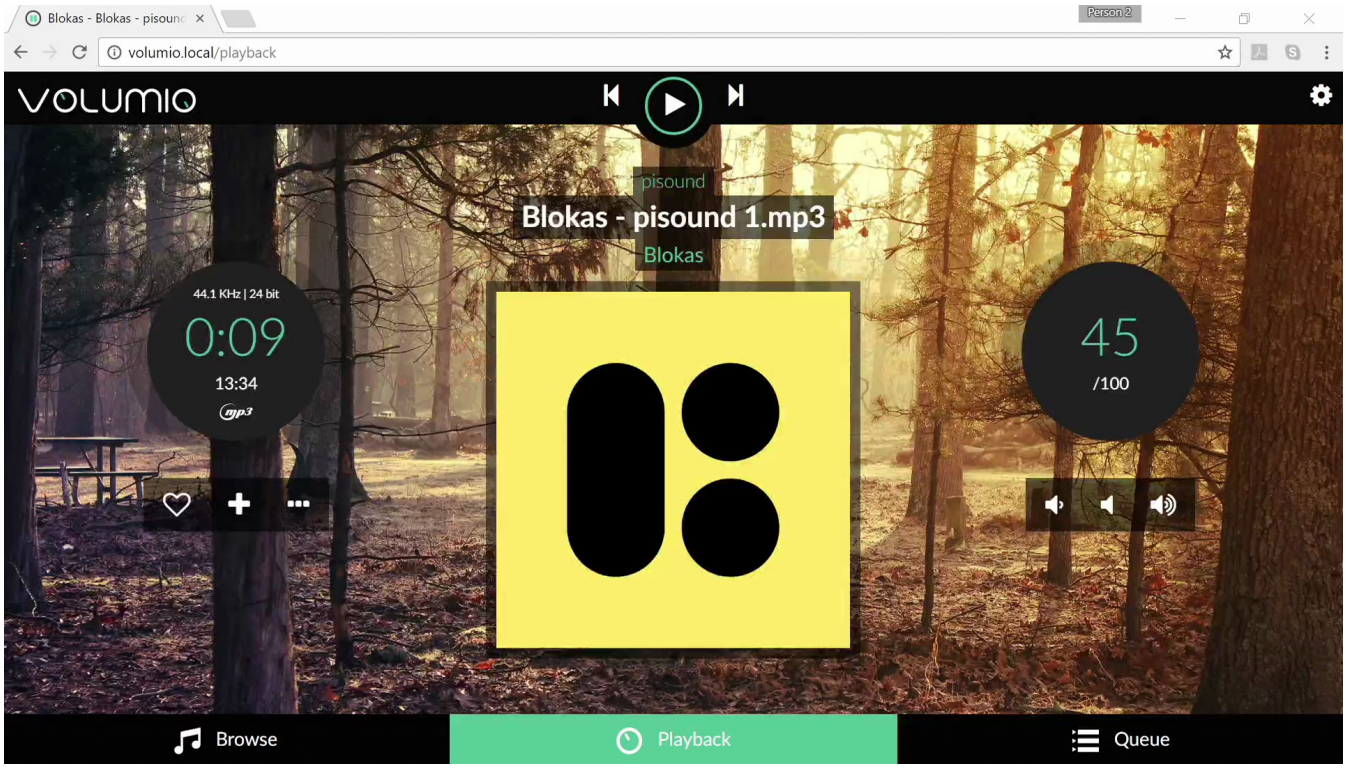
Then from other devices you can connect to `http://raspberrypi_ip:8000/` to see generic information about the station and `http://raspberry_pi:8000/pisound` to listen. (Replace raspberrypi_ip in the URLs using the IP of the Raspberry Pi)

For your station to be reachable outside of your local network, you need to have an externally accessible IP address provided by your ISP and you need to configure port forwarding on your router to forward requests on some port to port 8000 on Raspberry Pi. However, this is out of scope for this guide, there should be plenty of info around on how to set that up.

Network enabled Hi-Fi player

You can use pisound with Volumio! Though they're still running on an older kernel which was before the pisound driver was integrated, so some manual tweaks are still necessary. Our changes to integrate pisound are still pending on them updating their kernel, you can express your interest here: https://github.com/volumio/Volumio2/pull/955

Eventually none of this manual setup is going to be necessary, as soon as Volumio makes a release with an updated kernel.



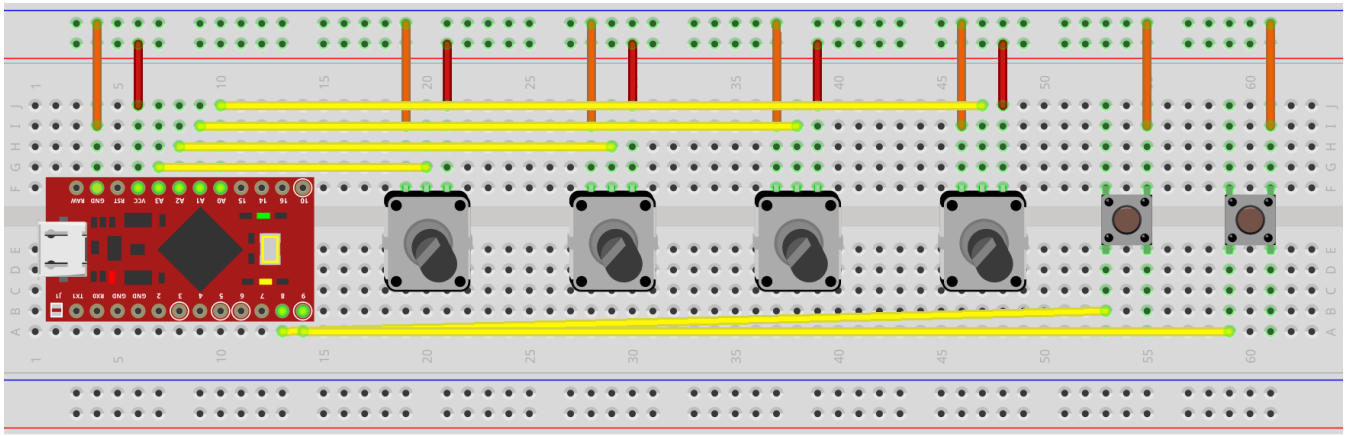
Steps to get started:

1. Download latest Volumio for Raspberry Pi: https://volumio.org/get-started.
2. Flash the image to an SD card, as usual.
3. Boot it.
4. SSH to volumio@volumio.local (or use the IP directly after @), password is volumio.
   - It may take a while after booting until SSH is ready to receive connections.
5. Download kernel source code using:
   - `volumio kernelsource`
6. The current volumio kernelsource seems to fail before adjusting the kernel version in the sources, but we can fix it up ourselves:
   - `sudo nano /usr/src/rpi-linux/include/config/kernel.release`
   - Make sure there's a '+' at the end of the version, so it looks like so: `4.4.9-v7+`
   - Press ctrl+x, y and enter to save and exit.
7. Download the pisound driver code:
   - `git clone https://github.com/BlokasLabs/pisound`
8. Build and install it:
   - `cd pisound/pisound-module`
   - `sudo make install`
9. Add pisound to Volumio's DAC selection:
   - `sudo nano /volumio/app/plugins/system_controller/i2s_dacs/dacs.json`
   - Add a line somewhere in the middle like so:
     `{"id":"pisound","name":"pisound","overlay":"pisound","alsanum":"1","mixer":"","modules":"","script":"","needsreboot":"no"},`
   - Note that a comma after this line is necessary, unless it's the last entry in the scope (before }), if there's a syntax error in this file, Volumio won't be able to show playback option in its web interface.
   - Save and exit using ctrl+x, y and enter.
10. Open http://volumio.local on your PC, go to Settings -> Playback Options.

11. Enable I2S DAC.
12. Select 'pisound'
13. Click save. As it saves the configuration and starts the card, the MIDI activity LEDs should flash briefly.
14. Change Mixer Type to 'Software'.
15. Done! Thank you! Now you can enjoy using pisound as a network media player!

pisound with DIY MIDI controller

If you want to add more controls to your pisound audio projects, you can make your own MIDI controller using a couple of potentiometers, push buttons and any Arduino compatible board that uses ATmega32U4 microcontroller. Alternatively, it should be straightforward to adapt the example code to use DIN-5 MIDI ports, as the usbmidi API interface is compatible with Serial API.



Go here for full instructions.