Using a USB midi keyboard on Ubuntu

SNDRTJ'S BLOG 2019-10-19 | English

I've been using Ubuntu since 2006, but even after 13 years I *still* dual boot with Windows. There's still a few things that keep me from going Linux-only. Mostly that's games, but there's a few other things that Linux unfortunately does not excel at. One of those things is anything related to music and sound.

While sound in general has dramatically improved over the last 13 years, connecting any musical instrument typically spells trouble.

I have an M-Audio Keystation Mini32 that I like to use for, well, simply playing some notes. On windows, it's pretty much plug-and-play. On Linux, that's not quite the same experience, tho it was less cumbersome than I initially thought it would be.

My system

First some details about my system. I'm using Kubuntu 18.04. That *probably* means this guide will also work on other Ubuntu variants, as well as Ubuntu derivatives such as Linux Mint.

Large parts of this guide *may* also work on other linux distros, but YMMV.

Getting started

While not exactly plug-and-*play*, when connecting the keyboard, the machine does immediatelty detect the keyboard and key presses *are* in fact registered. This even works before installing any packages.

You can check whether your MIDI device was detected with the following command:

```
aconnect -i
```

This should list all connected MIDI devices. On my machine, with my keyboard plugged in the USB port, this returns:

```
client 0: 'System' [type=kernel]
    0 'Timer           '
    1 'Announce        '
client 14: 'Midi Through' [type=kernel]
    0 'Midi Through Port-0'
client 20: 'Keystation Mini 32' [type=kernel,card=1]
    0 'Keystation Mini 32 MIDI 1'
```

So we can see the Keystation Mini32 show up. We can now see the key presses being registered by running:

```
aseqdump -p <client_id_of_the_device>
```

When I press and release a C, this shows:

```
Waiting for data. Press Ctrl+C to end.
Source  Event                  Ch  Data
 20:0   Note on                 0, note 60, velocity 62
 20:0   Note off                0, note 60
```

This is all nice and well, but we don't *hear* anything yet. For that, we need to install some packages.

Installing required packages

To actually hear some notes we need to install a number of packages.

1. qsynth
2. jackd
3. qjackctl

All three live in the default repositories, so can simply be installed with

```
sudo apt install qsynth jackd qjackctl
```

Qsynth

Qsynth is a GUI wrapper around the fluidsynth software synthesizer. This will be the application that actually converts MIDI signals to audible sound. You will need at least one soundfont file to be able to hear anything. Thankfully, the `qsynth` package on Ubuntu comes with the `fluid-sountfont-gm` package which provides a soundfont file.

jackd & qjackctl

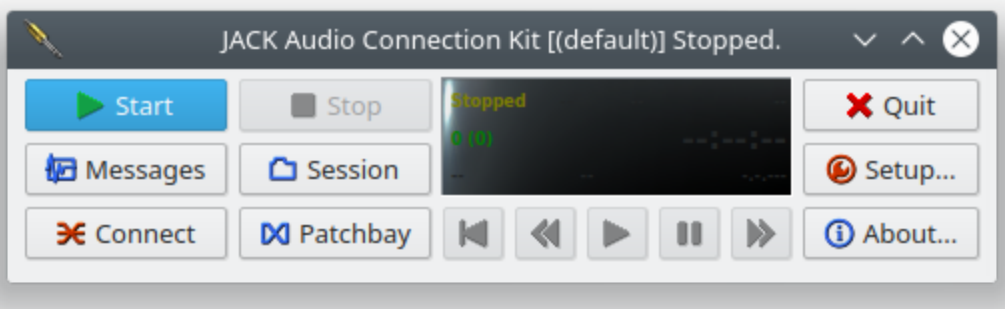
Jackd is where things get hairy. Jack is an 'audio connection kit', that allows applications to connect sound streams to other applications. It comes with a server, `jackd`, that performs all the routing between applications (which then function as jack clients).

It is unfortunately not the most intuitive application. Hence, we need a small GUI, `qjackctl` to configure, start, and stop jackd.

Configuring jackd

After installing the required packages, we need to configure jackd.

Open the jack GUI by typing `qjackctl` in a terminal. This should give you something like this:



In here, we need to go to "Setup" to change some settings.

1. Untick the realtime radio button (more info at the bottom of this page)
2. Change the number of frames per period to 512.
3. Go to the "advanced" tab and tick the "soft mode".

Stay in the the "advanced" tab, while opening a terminal. By default, jackd is assuming your output device lives on card 2, with subdevice 0. This isn't always the case, and in my case, actually changes at each reboot. Unfortunately this means I may have to re-configure jackd at each reboot.

To check what your current output device is, run `aplay -l` in a terminal. This returns something like:

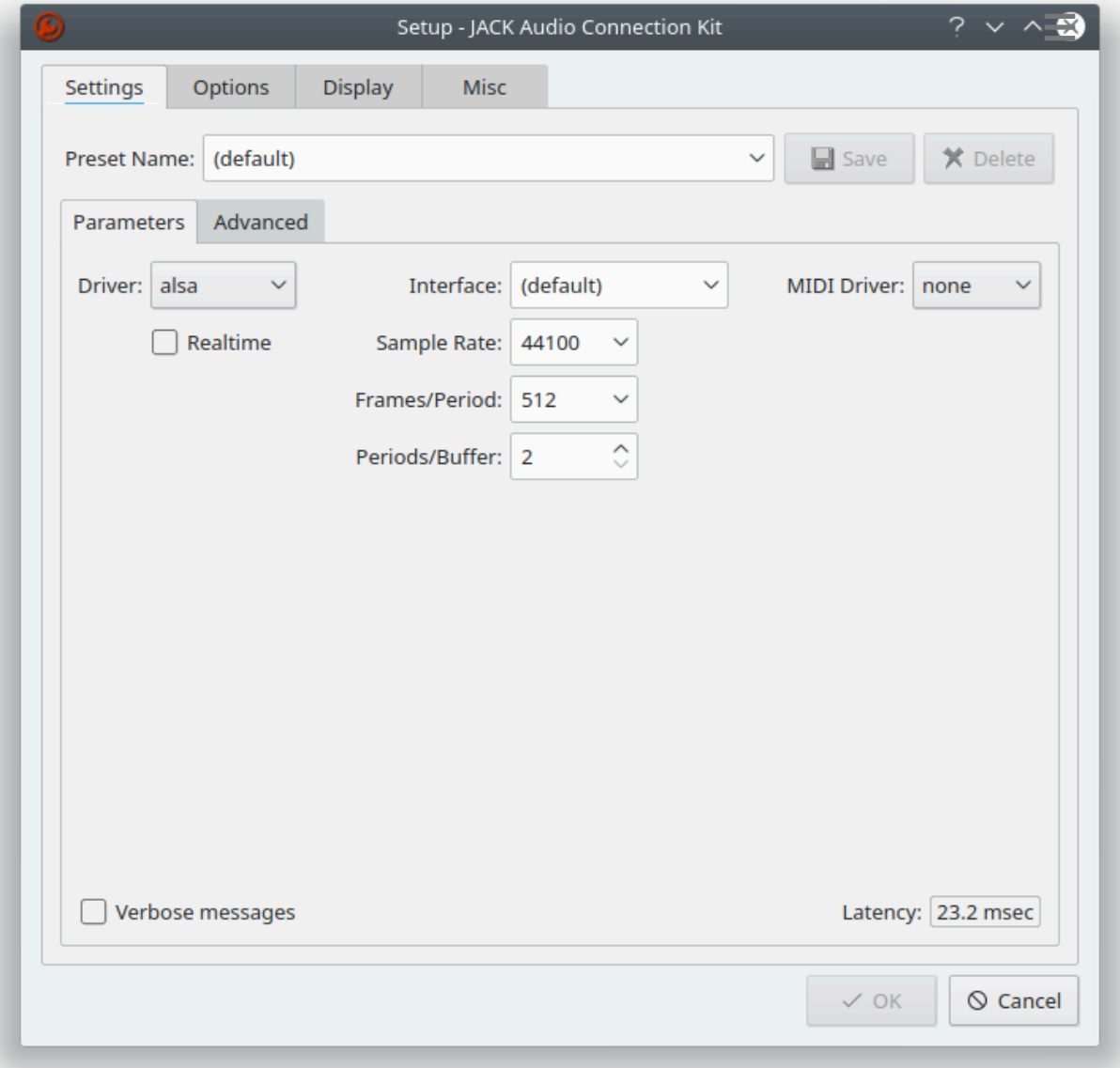
```
**** List of PLAYBACK Hardware Devices ****
card 0: Generic [HD-Audio Generic], device 3: HDMI 0 [HDMI 0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: Generic [HD-Audio Generic], device 7: HDMI 1 [HDMI 1]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: Generic [HD-Audio Generic], device 8: HDMI 2 [HDMI 2]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: Generic [HD-Audio Generic], device 9: HDMI 3 [HDMI 3]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Generic_1 [HD-Audio Generic], device 0: ALC892 Analog [ALC892 Analog]
```

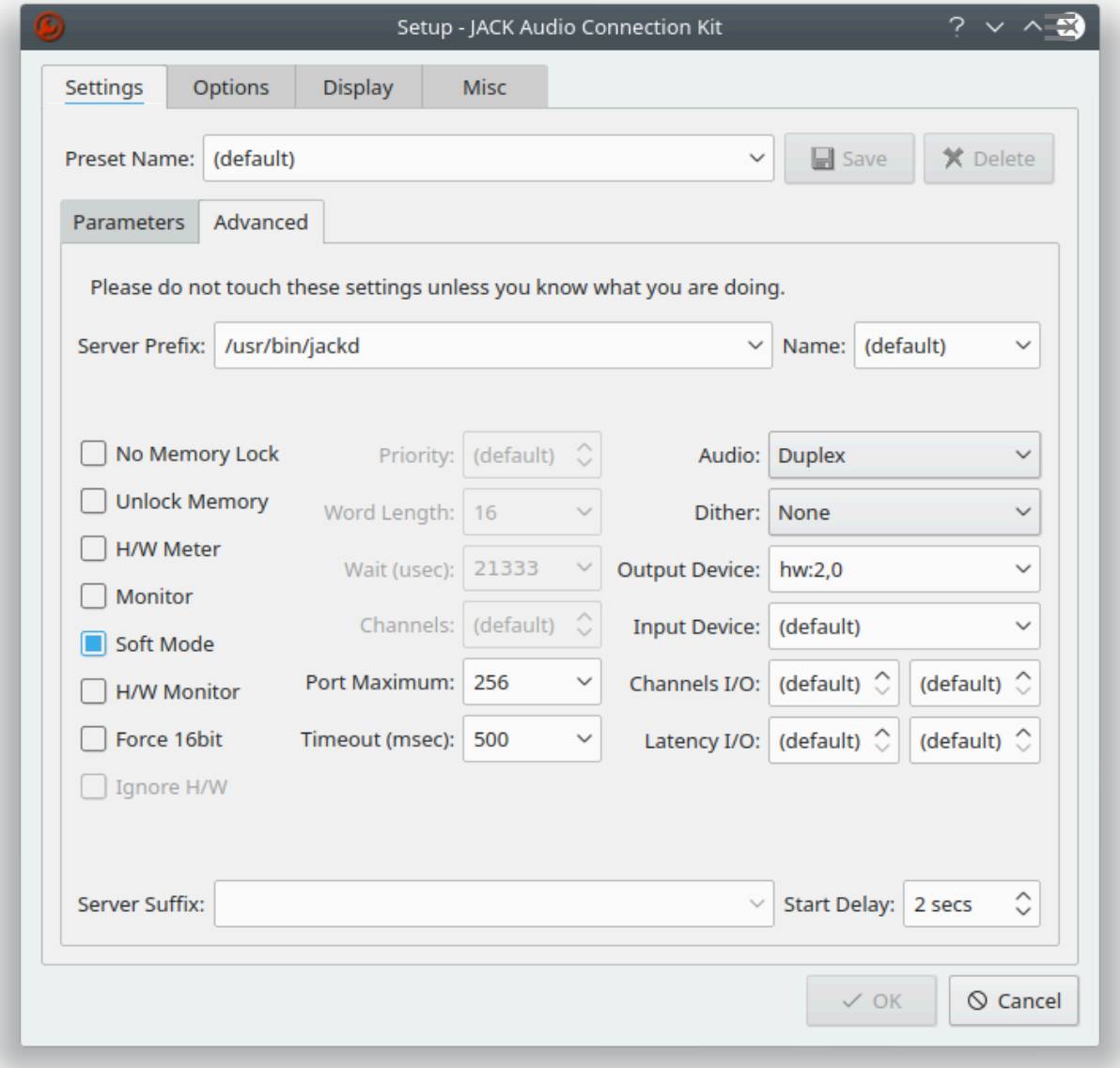
```
    Subdevices: 0/1
    Subdevice #0: subdevice #0
```

Which device you are looking for depends on your situation. If you have connected your PC to a TV, it's going to be one of the HDMI devices. If you used the headphone jack, it's going to be the Analog device. In any case, the format is going to be `hw:{card},{subdevice}`.

E.g, with headphones connected, in this case it's going to be `hw:0,2`. Fill in this string in the "advanced" tab under "Output device".

Your setup should now look like:


Setup - JACK Audio Connection Kit
Settings
Options
Display
Misc
Preset Name:
(default)
Save
Delete
Parameters
Advanced
Driver:
alsa
Interface:
(default)
MIDI Driver:
none
Realtime
Sample Rate:
44100
Frames/Period:
512
Periods/Buffer:
2
Verbose messages
Latency:
23.2 msec
OK
Cancel



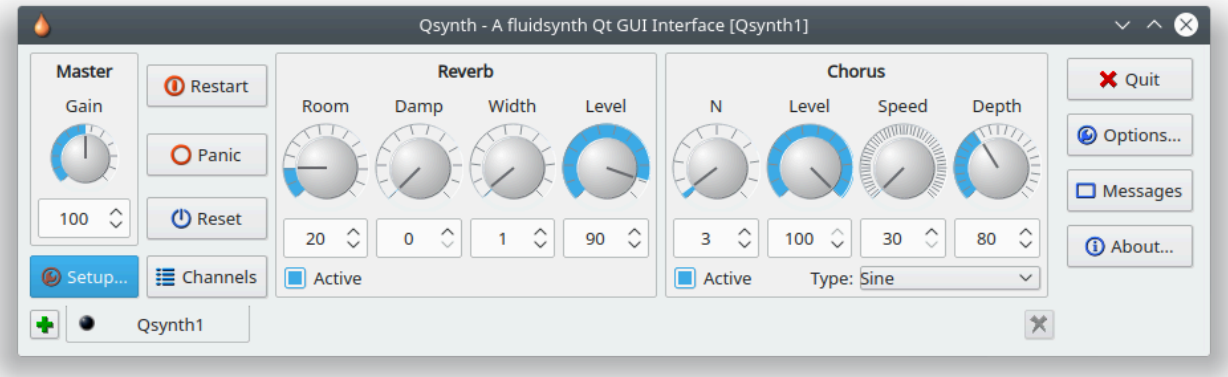
Getting ready to play some notes

You can now close the Setup screen and start jackd. You may hear an audible thump.

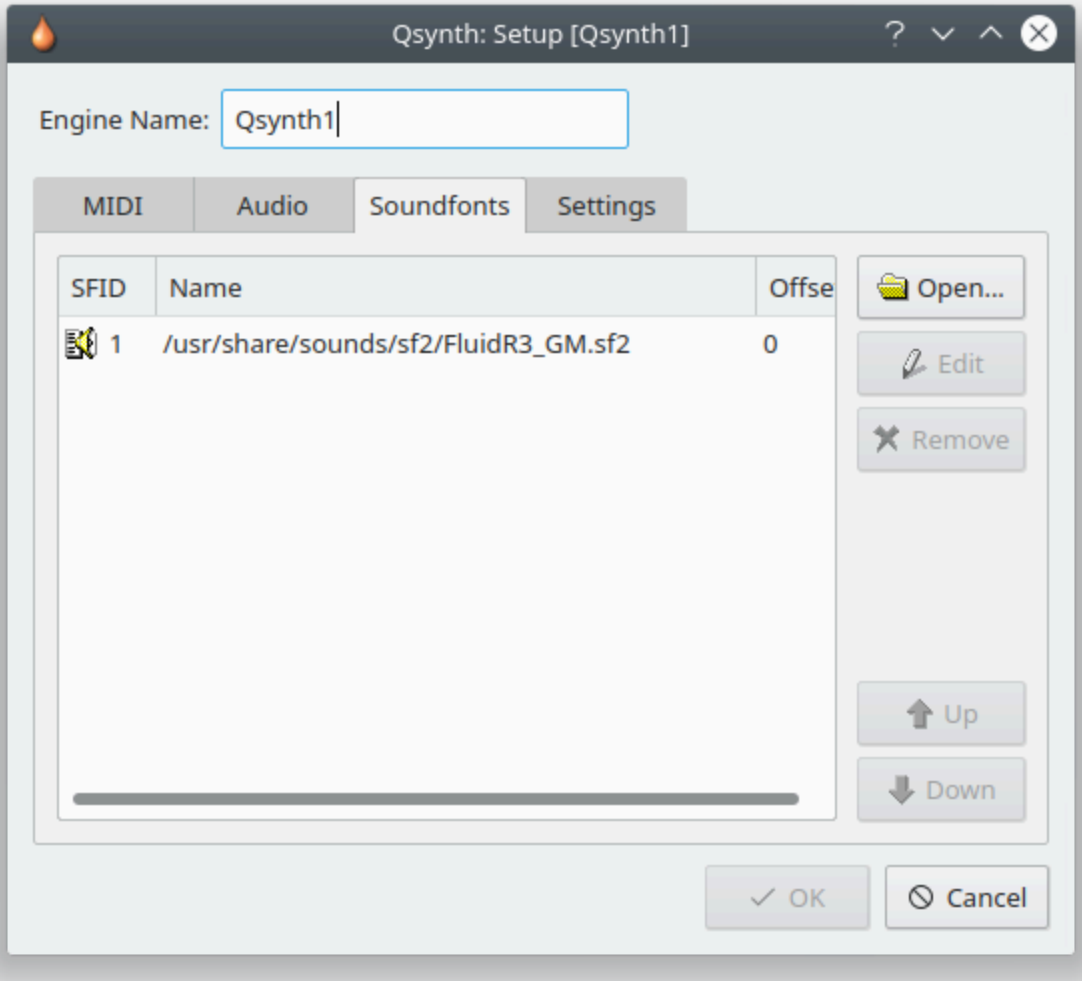
If, at this stage, you get an error stating `Cannot write socket fd = {x} err = broken pipe`, that means your setup uses the wrong output device. It may be a bit of trial and error figuring out which output device you need.

Once jackd is started, you can finally launch qsynth.

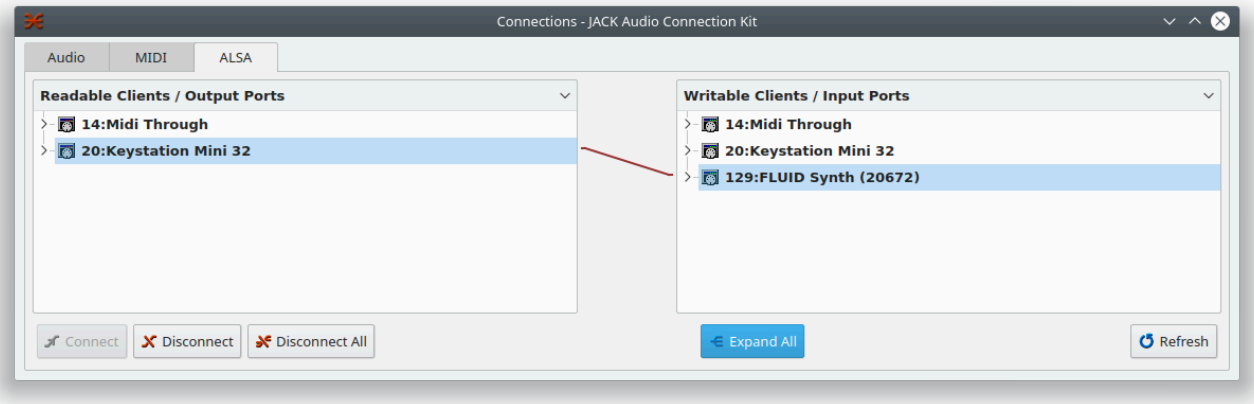
A window grossly looking like the following should open.



Go to Setup, and load the soundfont file in the Soundfonts tab. If qsynth was installed through apt, you'll have a soundfont file in `/usr/shares/sounds/sf2`



Now, we are *almost* done configuring things. We now need to go back to qjackctl. Now, hit connect. Go to the Alsa tab - strangly *not* the Midi tab. You'll see your midi device listed on the left side, and qsynth listed on the right side. You'll want to connect both. Select both, then hit the connect button, and a line connecting your midi device to qsynth should show up.



Now, finally, you should be able to hear you play a nice piano sound on your midi keyboard! Wooohoo, playing with a midi keyboard on Linux!

But wait, now I can't hear anything else?!

Yes, now you are able to use your midi keyboard, but that'll be the *only* sound you'll now hear. We've broken audio. Oops.

This is due to jackd not working nicely with pulseadio. Pulseadio is the sound server that practically all other applications use. While jackd is active, applications using pulseadio will not produce any audio.

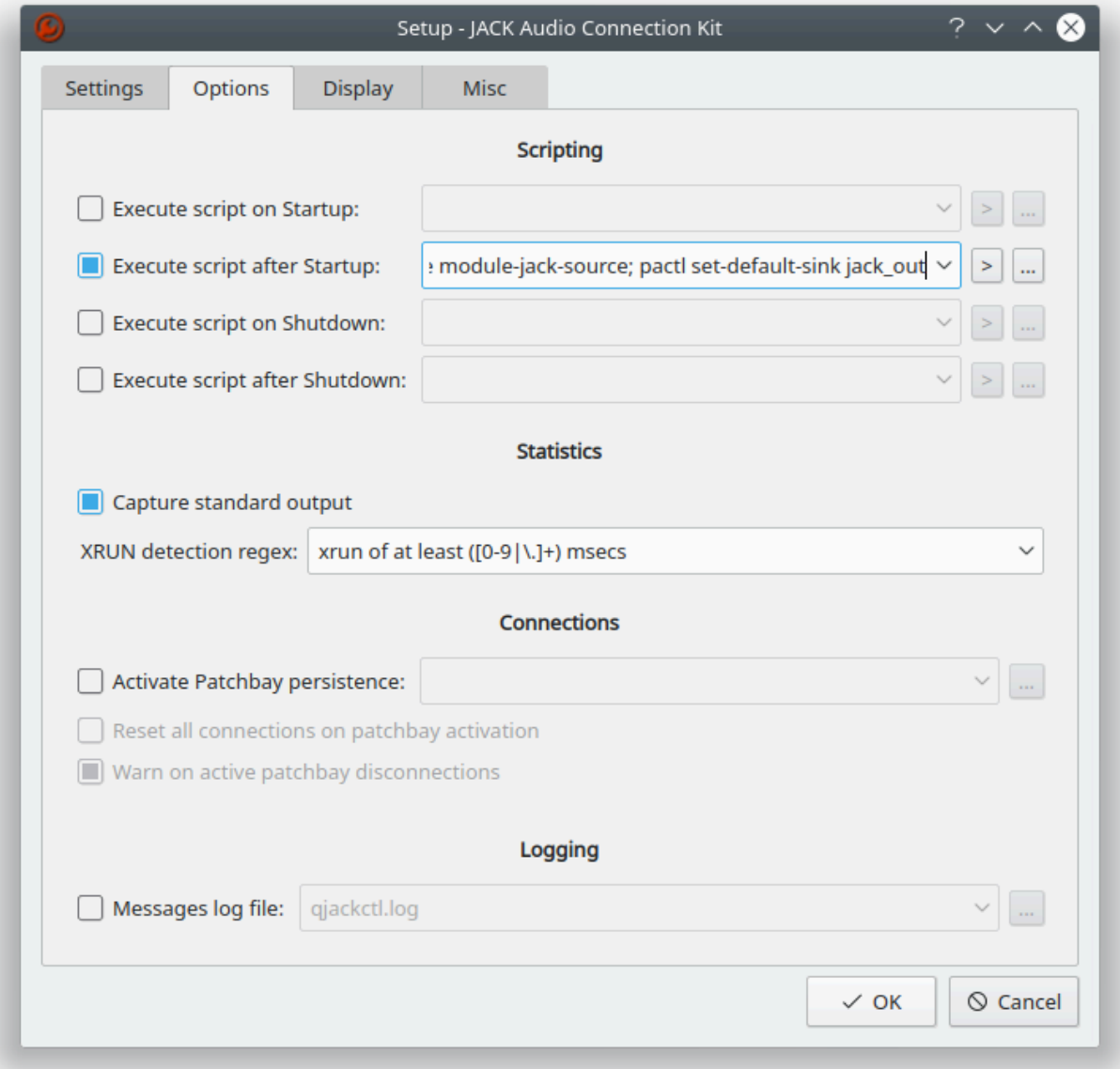
There is a way to fix this. We need to install yet another package.

```
sudo apt install pulseaudio-module-jack
```

This will install a pulseaudio module that allows one to tell pulseaudio to send its output streams to jackd.

Once installed, go back to qjackctl, and re-open the setup screen. Now go to the Options tab, and enter the following under the Execute script after Startup radio button:

```
pactl load-module module-jack-sink channels=2; pactl load-module module-jack-sourc
```



This will, just after startup of jackd, tell pulseaudio to send its output to jack.

Restart the jack server, reconnect your keyboard to qsynth, and lastly run `pulseaudio -k`. This will restart pulseadio.

Now you should have regular sound (you may have to reload YouTube videos), while simultaneously using the midi keyboard :-).

Conclusion

So it is possible to use a USB midi keyboard on Linux without horribly breaking the sound system, but some things could definitely be a bit smoother.

For one, I don't understand why `pulseaudio-module-jack` doesn't come as at least a recommended package when installing `jackd`. It seems to me that running jackd *and* pulseaudio at the same time on a modern Linux desktop is a very logical thing to do. It would be even greater if jackd would come preconfigured with pulseaudio working out of the box.

Anyway, I am happy with getting my midi keyboard to work. One reason less to keep dual booting!

Addendum: latency and real-time

We unticked the 'realtime' radio button in our jackd setup. That makes it more probably that jackd boots up, but may not be optimal.

While playing instruments, we usually want our latency to be as low as possible. To get the lowest latency possible, you should grant realtime priority to jack (or actually, you should use a realtime *kernel* as well). Doing so, however, means you *have* to configure your user to be able to grant realtime priority.

To do so, you need to create the `/etc/security/limits.d/99-realtime.conf` file with the following contents:

```
@realtime   -  rtprio     99
@realtime   -  memlock    unlimited
```

Add your user to the `realtime` group, and start a new session (log out and back in).

If realtime priorities are misconfigured, jackd will *not* start.