Connecting a MIDI keyboard to your Linux system

MAKING MUSIC


Kzenon, Fotolia

A MIDI keyboard is a useful extension to any audio workstation. Learn how to connect a MIDI instrument to your Linux sound studio through a MIDI interface device. **BY JOHN MARTIN UNGAR**

Linux is growing increasingly competitive as an environment for composing and playing electronic music. One important feature of the New Music scene is an electronic keyboard that outputs digital information in MIDI (Musical Instrument Digital Interface) format. The MIDI protocol supports communication between electronic musical instruments and computers. MIDI does not transfer sounds, but sound descriptions, such as "play a C-sharp on a grand piano as a quarter note at 80 beats per minute." This approach makes it easier to encode sound events in a far more compact way than would be possible with real sound data streams. At the same time, you can use any suitable MIDI sequencer and a MIDI keyboard to load melodies directly into your computer, where you can then change the notes and beat, transpose the key, or use different instrument sounds for the arrangement.

To connect a MIDI instrument to your Linux computer, you'll need an interface device that receives input from the instrument through a standard MIDI cable and passes information in and out of the computer with a PC-friendly connection to an ordinary USB port.

Several MIDI-to-USB interface devices are currently available. Unfortunately, like so many other commercial products on the market, these devices are not always ready for Linux in their default state. Luckily, the open source community has responded with firmware updates and other tools necessary for supporting MIDI devices. In this article, I'll describe how to connect a MIDI instrument to your Linux sound studio through a MIDI interface device, the popular M-Audio MIDISPORT 1x1 USB/MIDI interface, which costs about US$ 49.95. For other devices, see the vendor documentation for Linux configuration options.

Interface

Figure 1 shows the front and back of the M-Audio device. A MIDI instrument connects to the front of the device, and the MIDI data is then carried to and from the computer

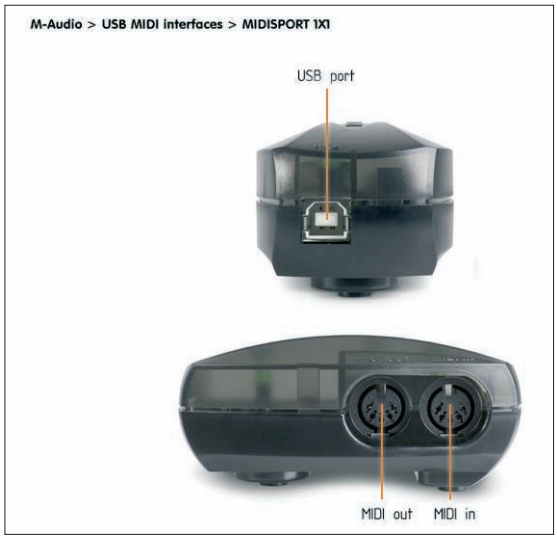

Figure 1: The MIDISPORT 1x1 interface connects to MIDI devices and then passes the information to the computer through the USB port.

through the USB port on the back. The "1x1" in the name means that the device has one MIDI In port and one MIDI Out port. (The MIDI Out connection lets your Linux system output MIDI instructions to an external sound-generating device.) M-Audio makes interface models with up to eight inputs and outputs.

To use the the M-Audio MIDI interface with Linux, you'll need to install new firmware. Debian users will prefer the DEB package [1]; users of Ubuntu 7.04 need a separate version [2]. SUSE and openSUSE, Red Hat/Fedora, and Mandriva can use the RPM packages [3]. Users with other distributions will need version 1.2 of the source code [4].

Listing 1: New Device

```
01 nonumber
02 Bus 002 Device 001: ID
   0000:0000
03 Bus 001 Device 002: ID
   0763:1011 Midiman
04 Bus 001 Device 001: ID
   0000:0000
```

Choosing a Keyboard

To leverage the features offered by modern, virtual instruments, I recommend a keyboard with touch response, as well as additional after-touch that provides modulation options at the push of a button. Also, the keyboard should have a pitch bend wheel to change the key and another modulation wheel to change the sound.

Regardless of your distribution, you will need a utility called fxload to upload the firmware to the MIDI interface; your distribution's package manager should give you an install option.

On Debian and Ubuntu, simply double-click the package after downloading it and follow the instructions of the package manager.

For openSUSE, download the ezusbmidi package to your machine and launch YaST to install.

Workaround

If any problems occur, you can try a workaround. First, load the firmware onto your system and unpack the archive. Then, pop up a console, change to the new directory created at the last step (*midisport-firmware-1.2*), and enter the following commands:

```
nonumber
$ ./configure
$ make
$ sudo su
# make install
```

This gives you the firmware, and you will have a udev rule to match in */etc/udev/Rules.d*, rule 42, which will be called something like *42-midisport-firmware.rules*.

The corresponding firmware scripts are located in */usr/local/share/usb/maudio* and are identifiable by the file extension *.ihx*.

Make sure that none of these scripts is 0 bytes. If so, the installation script has caused an error and you will need to manually copy the file from the *midisport-firmware-1.2* folder to the firmware directory */usr/local/share/usb/maudio*.

The device is attached, but you still can't use it. To do so, you first must load the firmware to the MIDI box via USB. Previously, this task was handled by the hotplug daemon, but modern distributions use the Hardware Abstraction Layer (HAL) and the udev rules, which are located below */etc/udev/Rules.d*. The rules for your MIDI box are still missing, so let's create them.

With HAL and udev, the system takes about 60 seconds to find the new device and activate it with the use of the udev rule. Debian-based systems do this reliably, but you might need to give openSUSE 10.2 a helping hand. The *fxload* command tells udev to upload the firmware to the device. If this does not happen automatically, you can perform the steps manually. To do so, pop up a console and load the firmware as follows:

Figure 2: The main window in the Jack control program, QJackCtl.

```
$ fxload -D /dev/bus/usb/USB-Bus
/Device -v -s /usr/local/share/u
sb/maudio/MidiSportLoader.ihx -I
 /usr/local/share/usb/maudio/Mid
iSport1x1.ihx
```

Replace *USB-Bus* with the numeric identifier that *lsusb* shows you; in this example, this is *001*. In the same vein, use the device number that is shown for *Device – 002*, in this case. Then, open the KDE

Listing 2: Checking for Required Modules

```
01 nonumber
02 snd_usb_audio 88736 0
03 snd_usb_lib 21888 1 snd_usb_
   audio
04 snd_pcm 89096 6 snd_usb_
   audio,snd_hda_intel,
    snd_hda_codec,snd_pcm_oss
05 snd_rawmidi 31392
    2 snd_usb_lib,snd_seq_midi
06 snd_hwdep 15240 1 snd_usb_
   audio
07 snd 65256 17 snd_usb_
   audio,snd_hda_intel,
    snd_seq_oss,snd_hda_
   codec,snd_pcm_oss,
    snd_mixer_oss,snd_seq,snd_
   pcm,
    snd_timer,snd_rawmidi,snd_
   seq_device,
    snd_hwdep
08 usbhid 45088 0
```

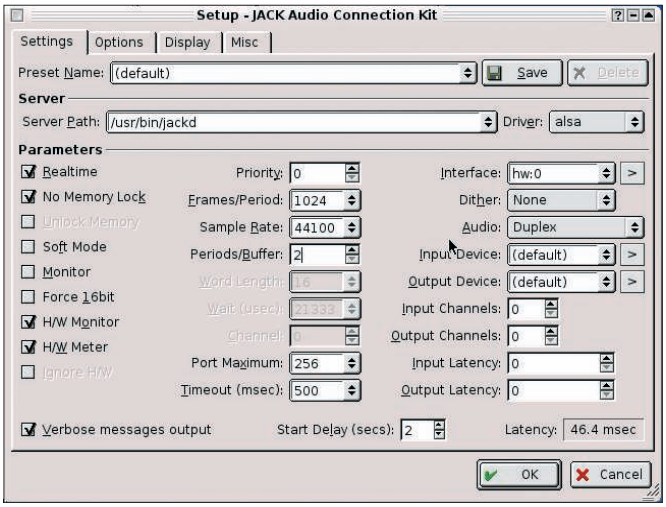

Figure 3: The Jack setup menu with critical settings.

Desktop Settings program and change to *Sound and Multimedia | Sound system | Hardware*. Now you will see your device or at least be able to enable it in *Select MIDI device*. Note that Debian and Ubuntu do not have their *usb/maudio* directories below */usr/local/share*, but below */usr/share* instead.

Connecting the Keyboard

Attaching any keyboard to the 1x1 interface is possible; any old organ with a MIDI port will do, but you can just as easily hitch up to a modern keyboard or synthesizer from your live equipment. For my experiments, I used a fairly ancient Yamaha PSR 400 and was happy enough with it.

Like the MIDI interface, the keyboard must have at least one MIDI In and one MIDI Out port. Connect the two with the appropriate connectors of the MIDI interface device. (Note that the signal going *out* of the keyboard is going *in* to the interface, and the signal going *out* of the interface device is going *in* to the keyboard.) Using the USB lead, connect the box to your computer.

Now the system should show the new device. Popping up a console and entering *lsusb* lets you check this. If everything is wired up correctly, the output should be similar to Listing 1.

Meet Jack

After your PC has identified the interface as a USB device, you can start using the keyboard. To do so, you will need the Jack [5] sound server.

Jack is a low-latency system that is ideal for a trouble-free sound system. Latency refers to the wait time between triggering and processing a signal. In music and sound generation, too much latency will cause clicks and dropouts, and in recording, it will cause an unacceptable lack of synchronization in signal processing. The kernel is responsible for these delays on Linux; however, a solution has been available since the introduction of kernel version 2.6, and version 2.6.10 even introduced a working low-latency variant. Ubuntu Studio [6] and JAD [7] pre-install the kernel, but you can build your own low-latency kernel for any other distribution.

The Jack sound server cuts into the Advanced Linux Sound Architecture (ALSA) at a low level to handle high-bandwidth audio streams at maximum speed. Whether the sound streams come from the analog converter on your sound card, an editor, effects, or virtual instruments makes no difference. Jack lets you send data both to ALSA's *pcm_out* and any other program that can generate a Jack-in port. This gives users the framework for an extremely flexible and extensible modular software system. Because Jack latches into the system at a low level, you need root privileges to launch the system, which also applies to any application that uses Jack.

Installing Jack

For basic MIDI studio operations, you only need the Jack daemon and QJackCtl, the Qt-based GUI. Both packages are provided by any standard distribution and can be installed easily with YaST or Synaptic.

Immediately after the install, Jack is ready for use. Before you launch Jack for the first time, check to see whether the kernel has all the required modules. To do so, pop up a console and enter *lsmod | grep usb*.

The output should look something like Listing 2; otherwise, you will need to load the USB sound module before you continue. To do so, enter the command *modprobe snd_usb_audio*. Then type *lsmod | grep usb* again. Having the *snd_seq* module is important. If this module does not appear in the list, type *modprobe snd_seq* to load it.

Starting Jack

To start the Jack front end, click *Applications | Multimedia | Music* in your KDE menu and select *QJackCtl*. The spartan main window provided by the front end appears (Figure 2). If you click *Start* immediately, you will probably see an error message because the default settings are not normally correct. Thus, it makes more sense to select the *Setup* menu (Figure 3).

To begin, enable the parameters *Realtime*, *No Memory Lock*, *H/W Monitor*, and *H/W Meter*, as well as *Verbose messages output*. In the *Interface* drop-down box, select the *hw:0* option and, above that, accept the settings as *Driver: alsa*. Pressing *OK* saves the settings; now quit Jack and restart it.

Next, you can move on to wiring the system. In the multimedia section, launch the *ZynAddSubFX* synthesizer. If the synthesizer is not installed yet, use your package manager to install it. Then, in the Jack control window, open *Patchbay* (Figure 4).

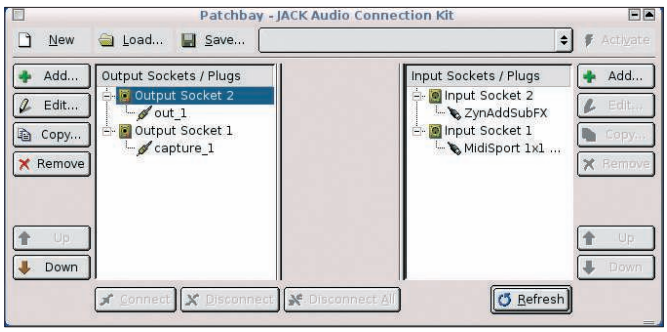

Figure 4: QJackCtl Patchbay is where the connected audio applications are patched.

To enable input, click *Add* to the right of *Input Sockets/Plugs*. In the *Type* dialog that appears, select *MIDI* and then select the MIDI interface below *Client*. Clicking *Add Plug* adds the MIDI interface to the list of avail-

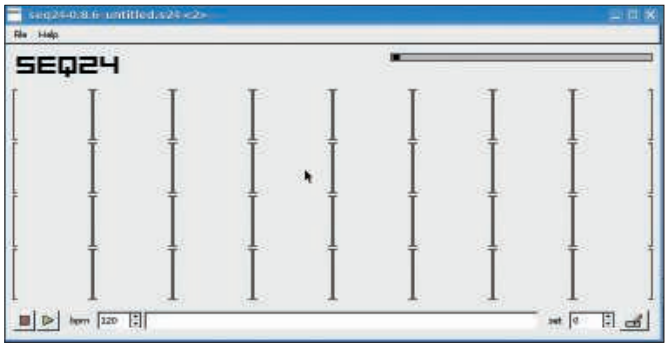

Figure 5: The spartan welcome window of the Seq24 sequencer gives users space to create new patterns.

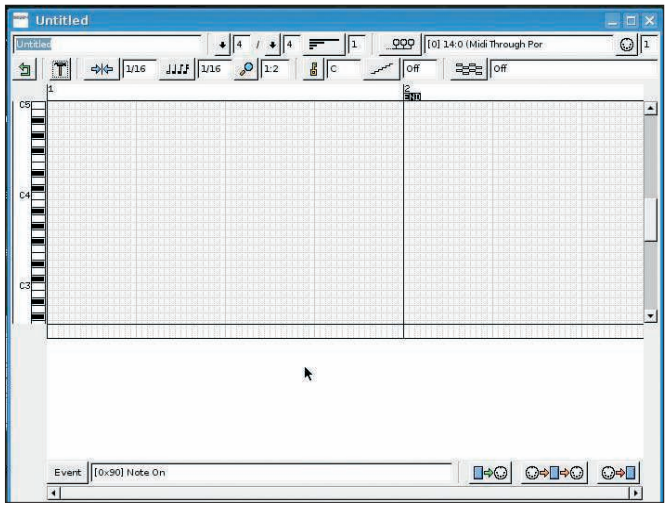

Figure 6: Create the individual components of your song in the Seq24 Pattern Editor.

able plugs. Pressing *OK* confirms your selection; then, quit the dialog. Now your keyboard is available as an input device. In the same way, you can enable output on the other side of the Patchbay window with *ZynAddSubFX* as your MIDI input.

Connecting the two plugs is the last step. To do so, select the *Input Socket* and the *Output Socket* and click *Connect* then *Activate*. Clicking the close control (x) button closes the Patchbay window.

Now you have wired up your keyboard and synthesizer with Jack and you can be creative – load a sound into ZynAddSubFX and play!

If the sound is scratchy, has drop-outs, or is very late, change the values for *Sample Rate*, *Frames/Period*, and *Periods/Buffers* in the Jack setup pane; note that this could affect your sound quality.

Working with the Sequencer

To make MIDI music, you need what is known as a sequencer. The sequencer records the melodies you play on your keyboard, helps you edit them, and outputs the results via the attached instruments. The choice of sequencers on Linux is not particularly opulent, especially considering that the most popular open source sequencers – Rosegarden and Muse – cause considerable trouble without a low-latency kernel. On the other hand, this complication makes the choice easier – I went for the simple, all-round sequencer Seq24 for these tests. To set up the sequencer, use your distribution's package manager.

If the Jack front end is not running, then launch QJackCtl before you launch the Seq24 sequencer at the console by entering */usr/bin/seq24*. At first, you only get an empty main window (Figure 5) filled with rows and columns. Right-clicking a free field and selecting *New* to open what is known as the Pattern Editor (Figure 6) lets you create sequences and loops.

Creating and editing the individual sound bits of your composition is done

MIDI Tools

MIDI channels are numbered 0 through 15 or 1 through 16 depending on the convention the vendor follows. Assuming you have 16 instruments, or at least that many digital sound generators and effects capable of receiving MIDI signals on different channels, you can send a sound stream to an instrument on any of these channels. Normally you can freely assign sound generators to any MIDI channel. General MIDI [8] restricts this freedom for the sake of standardizing instruments. For example, the drum track is always MIDI channel 10.

The open source software synthesizers ZynAddSubFX and Hydrogen are equally suitable for interpreting incoming MIDI signals. A third useful tool is the software wavetable synthesizer TiMidity++ [9], which converts MIDI files to sound that is then output via Jack and ALSA; your distribution should include the instrument. Also, you might want to download the Freepats package [10] to get your wavetable synthesizer playing.

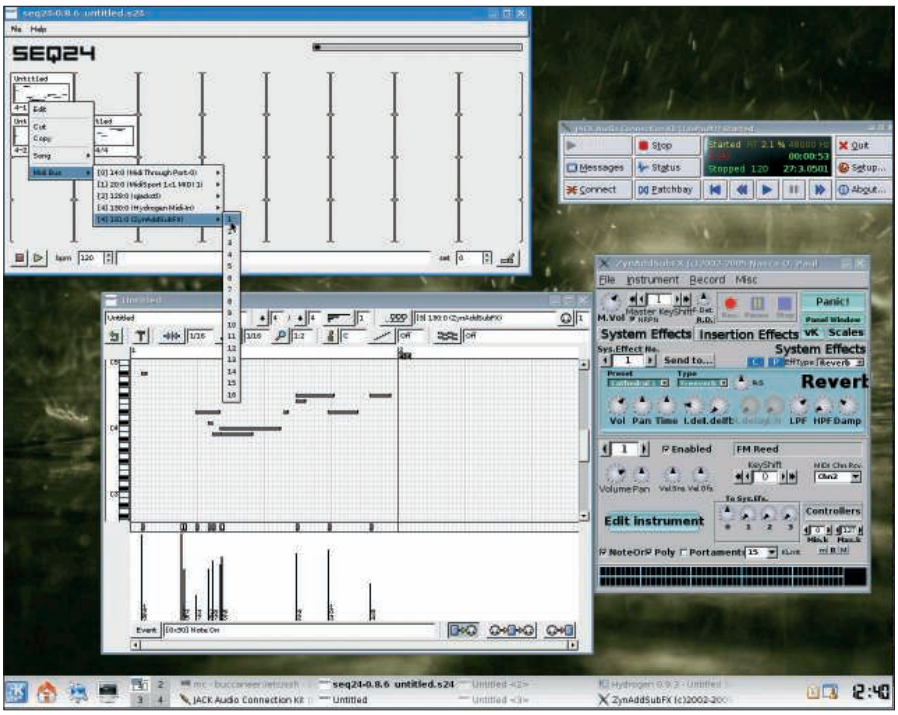

Figure 7: Seq24 lets you play your pattern via a connected synthesizer – ZynAddSubFX in this case.

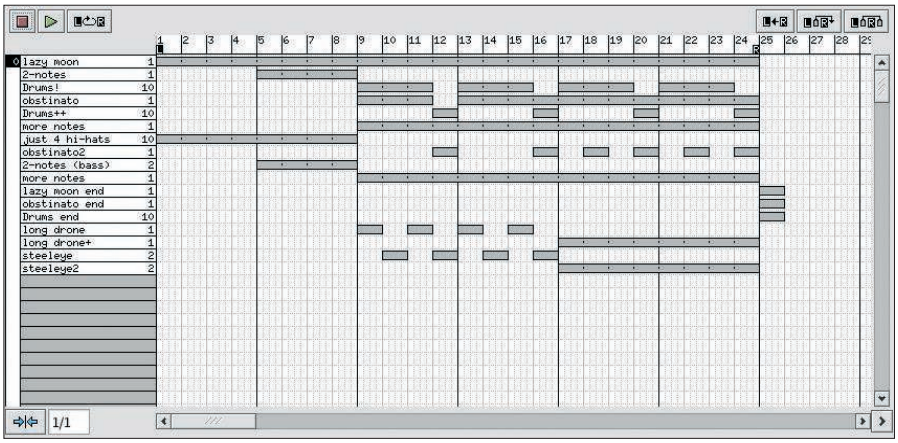

Figure 8: An example of an arrangement with Seq24.

in the Pattern Editor. Type a name for the pattern in the first field at the top left – for example, *Intro*, *Bridge*, or *Refrain*. Then, use the two arrows to set the number of beats to the bar, or simply use the default, *4/4*. The next field to the right lets you define the length of the pattern in bars.

When you are done, click the slightly cryptic button with the three MIDI connectors and select an instrument to output the MIDI events. Using the MIDI icon on the far right, select the MIDI channel that you will be using to output the signals. In the next row, set the resolution for the pattern. The default, *1/16*, tells the sequencer to drag any notes that are not located exactly on a sixteenth of a bar to the sixteenth automatically to assure precise values. Also, you can set the key and adjust the *major* and *minor* setting here.

To tap in a sequence, choose one of two options. The first is to use the matrix in the Pattern Editor to add tones. First, press the right mouse button to change the cursor to a pen, and press the left mouse button to insert events into the matrix. Then you can hold down the left mouse button on any event to move it or select the event and press the Delete key to delete it.

The more intuitive approach is to use the attached keyboard as your input device. To do so, close the Pattern Editor, click *File* | *Options* in the main window, and open the *MIDI Input* tab.

Select a MIDI interface or instrument and change back to the *Jack Sync* tab. Enabling the *Jack Transport* and *Jack Master* settings tells the sequencer to use Jack to organize the data streams and Seq24 as the master program for Jack audio control. The *Master Conditional* button only passes control to the sequencer if there is no other master.

The *Live Mode* and *Song Mode* options let you specify whether you will manipulate the pattern while you are inputting it or would like to play a complete arrangement that you have created with the Song Editor, without the ability to change it. Pressing *OK* confirms your selection.

In the bottom right-hand side of the Pattern Editor, you will find a small icon with the function *Records incoming midi-data*. In the main Seq24 window, enable the button and click the green triangle. This somewhat roundabout approach lets you record MIDI events from the keyboard. To stop recording, click the red square to the left.

Playing Back Sequences

Selecting ZynAddSubFX in the Pattern Editor's *Select Output Bus* menu lets you hear what you have played. The synthesizer must be enabled with the sound program loaded for this to work (Figure 7). Again, in the bottom right-hand side of the Pattern Editor, click the left button with the function *Sequence dumps data to midi-bus*. This tells the sequencer to send the recorded MIDI data to the attached sound generator so you can hear the sounds.

If the application fails to produce sound output, check your Jack Control patch field: Is the synthesizer audio output connected to the ALSA system? If not, change the setting.

Modify Signals

Because MIDI signals are simple control signals and not audio events, you can easily modify the signals any time later – for example, to change the key.

To modify signals, first click on one of the bars representing the key and duration of the note. Once the bar has turned orange, you can move it or press the Delete key to delete. This lets you create and modify the individual patterns in your song.

Create a Song

To put your patterns together and create a song, launch the Song Editor by clicking the pencil icon in the bottom right-hand side of the main Seq24 window. In the left column of the Song Editor, you will see an overview of existing patterns, and the timeline for your song is at the top in bars (Figure 8).

Holding the right mouse button down while also pressing the left mouse button places a pattern on the timeline and removes the pattern from the list.

Holding down the left mouse button moves a pattern in the timeline. Clicking the third icon from the left in the icon bar outputs your composition as an infinite loop.

Making music with your Linux PC really can be this simple – so have fun composing! ■

INFO

[1] Debian package for MIDISPORT: *http://packages.debian.org/unstable/sound/midisport-firmwaremc/*

[2] Ubuntu package for MIDISPORT: *http://packages.ubuntu.com/feisty/misc/midisport-firmware*

[3] RPM package for MIDISPORT: *http://rpm.pbone.net/index.php3/stat/26/dist/7/size/71557/name/ezusbmidi-2002_11_17-1.src.rpm*

[4] MIDISPORT firmware: *http://usb-midi-fw.sourceforge.net/*

[5] Information on Jack: *http://www.jacklab.org/*

[6] Ubuntu Studio: *http://ubuntustudio.org*

[7] Jacklab Audio Distribution: *http://distrowatch.com/table.php?distribution=jacklab*

[8] General MIDI (Wikipedia): *http://en.wikipedia.org/wiki/General_MIDI*

[9] TiMidity++: *http://freshmeat.net/projects/timidity/*

[10] Freepats packages: *http://freepats.opensrc.org*