python-lpd8 Documentation

Release 1.0.0

Lukas Jackowski

May 08, 2018

Contents:

python-lpd8 is a pythonic abstraction for the Akai LPD8 midi controller. It allows you to attach callback functions to each pad and knob, making integration into your application extremely easy.

This is a minimal working example:

```
from lpd8mido import LPD8DeviceMido

from time import sleep

def exampleCallback(programNum: int, padNum: int, knobNum: int, value: int, noteon:␣
↪int, noteoff: int, cc: int, pc: int):
    print("CB program: %s pad: %s knob: %s value: %s" % (programNum, padNum, knobNum,␣
↪value))

lpd8 = LPD8DeviceMido()
for i in range(8):
    lpd8.addPadCB(0, i, exampleCallback)
    lpd8.addKnobCB(0, i, exampleCallback)

while(True):
    lpd8.tick()
    sleep(0.01)
```

While you've activated the first program and are in PAD-mode you should get messages that looks like this every time you hit a pad or turn a knob:

```
CB program: 0 pad: 0 knob: None value: 19
CB program: 0 pad: 1 knob: None value: 4
CB program: 0 pad: 2 knob: None value: 20
CB program: 0 pad: 3 knob: None value: 7
CB program: 0 pad: None knob: 5 value: 37
CB program: 0 pad: None knob: 5 value: 38
CB program: 0 pad: None knob: 5 value: 39
CB program: 0 pad: None knob: 5 value: 40
```

CHAPTER 1

Quickstart

Python-lpd8 works by handling all the midi communication with the lpd8 device for you. All you have to do is decide which callback you want attached to which pad or knob. This quickstart will show you how to use python-lpd8 with mido, a MIDI library. The mido interface is currently the only one that is currently available.

1.1 Opening device

First we need to create a device instance:

```
from lpd8mido import LPD8DeviceMido

lpd8 = LPD8DeviceMido()
```

Python-lpd8 will fetch the first LPD8 device that isn't yet occupied. If it can't find a free device, it throws an exception.

1.2 Attaching Callbacks

Callbacks are called with the following signature:

```
def callback(programNum: int,
             padNum: int,
             knobNum: int,
             value: int,
             noteon: int,
             noteoff: int,
             cc: int,
             pc: int) -> None:
   pass
```

programNum, *padNum* and *knobNum* refer to the indices of the program and pad/knob of the pad/knob that triggered the callback.

Note: In python-lpd8 all indices are 0-indexed, while LPD8 is 1-indexed. For example: Program 1 has the index 0, pad 7 has the index 6!

value refers to either the velocity with which pads were hit or the value a knob has been turned to.

The additional parameters shall not concern us for now.

A simple callback function might look like this:

```
def exampleCallback(programNum: int, padNum: int, knobNum: int, value: int, noteon:␣
↪int, noteoff: int, cc: int, pc: int):
    print("CB program: %s pad: %s knob: %s value: %s" % (programNum, padNum, knobNum,␣
↪value))
```

It does nothing more than to print the program index, pad/knob index and value it receives. To have this callback called everytime we press or release pad 1, we do the following:

```
lpd8.addPadCB(0, 0, exampleCallback)
#lpd8.addKnobCB(0, 0, exampleCallback) # For knobs
```

If you press and release pad 1 you will see the following printed:

```
CB program: 0 pad: 0 knob: None value: 63
CB program: 0 pad: 0 knob: None value: 127
```

CHAPTER 2

LPD8

The Akai LPD8 is a midi controller with 8 velocity-sensitive pads and 8 knobs.

2.1 Programs

The LPD8 memory bank can store 4 different programs. A program consists out of a configuration of notes and controls associated with each pad and knob.

2.2 Pads

The pads can operate in three different modes: Pad (Notes), CC (Control Change) and Prog Chng (Program Change). Depending on the mode you are using, different midi events are emitted when you press and release buttons. They also differ in how the velocity is handled.

Action	Pad	CC	Prog Chng
Pad Down	note_on	control_change	program_change
Pad Release	note_off	control_change	N/A
Velocity Down	0-127	0-127	N/A
Velocity Release	127	0	N/A

All pads can be put into toggle mode. When in toggle mode, you have to press the pad a second time to trigger the release event.

When in Pad mode you can turn on and off the lights of each pad by sending a note_on or note_off midi message with the corresponding note. The state of the light will be overwritten on the next button press, so you need to resend note_on messages after a button press if you want the button to be permanently lit.

2.3 Knobs

Each knob emits a control_change message when it is turned. The value ranges from 0 to 127. It is possible to change upper and lower value limit, but this comes with a reduction in resolution and is thus not suggested.

CHAPTER 3

Control Ambiguities

To prevent ambiguities, python-lpd8 checks that each pad and knob has a unique note/control/program associated with it. If an ambiguity is detected, you are left with a choice: Shall python-lpd8 throw an exception or shall it attemt to fix the problem?

By default python-lpd8 will throw an exception letting you know that there is a problem. If you want it to be fixed, call the constructor like this:

```
lpd8 = LPD8Device(solveAmbiguity=True)
```

python-lpd8 will then try to fix any ambiguity by increasing the note/control/program until it is unique across the entire device. Additionally all pad toggles will be set to off and all knob ranges are set to 0-127.

Warning: All programs on the device will be (partially) overwritten by this! Backup your programs before enabling this setting.

CHAPTER 4

Callbacks

Python-lpd8 works with callbacks to let you know that a pad has been pressed or a knob has been turned. Information about which program's pad/knob has been actuated is passed to the callback.

4.1 Signature

Any callback function must have a signature compatible to the following:

`callback` (*programNum: int*, *padNum: int*, *knobNum: int*, *value: int*, *noteon: int*, *noteoff: int*, *cc: int*, *pc: int*)

Parameters

- **`programNum`** – Index of program pad/knob belongs to
- **`padNum`** – Index of pad that triggered call
- **`knobNum`** – Index of knob that triggered call
- **`value`** – Velocity for pads, value for knobs
- **`noteon`** – midi note of noteon message
- **`noteoff`** – midi note of noteoff message
- **`cc`** – midi control of control change message
- **`pc`** – midi program of program change message

The signature allows you to attach the same callback to all pads and knobs in all three modes. There are four possible calling schemes depending on the modes used:

4.1.1 Pad in PAD mode

In pad mode you will get one call on pad press, and another one on pad release. You can differentiate between the two by checking if noteon/noteoff is None. They will always be mutually exclusive.

Param	Value press	Value release
programNum	0-3	0-3
padNum	0-7	0-7
knobNum	None	None
value	1-127	0
noteon	0-127	None
noteoff	None	0-127
cc	None	None
pc	None	None

4.1.2 Pad in CC mode

In cc mode you will get one call on pad press, and another one on pad release. The only way to differentiate between the two is to check if value is 0.

Param	Value press	Value release
programNum	0-3	0-3
padNum	0-7	0-7
knobNum	None	None
value	1-127	0
noteon	None	None
noteoff	None	None
cc	0-127	0-127
pc	None	None

4.1.3 Pad in PROG CHNG mode

In program change mode you will only get a call when the button is pressed. You also get no velocity data.

Param	Value press
programNum	0-3
padNum	0-7
knobNum	None
value	None
noteon	None
noteoff	None
cc	None
pc	0-127

4.1.4 Knob

The knob behaves the same way in all three modes. You get the position of the knob as the value.

Param	Value turn
programNum	0-3
padNum	None
knobNum	0-7
value	0-127
noteon	None
noteoff	None
cc	0-127
pc	None

Index