MIDI in Python

Connecting to and controlling digital instruments with Python

Benjamin Vigny-Pau

What is MIDI?

- Musical Instrument Digital Interface
- Standard communication protocol
- “Instructions” for instruments (usually digital or synthesized)

MIDI Messages

- String of binary bytes
  - Status/Header byte (1_______: 128 - 255)
  - Data bytes (0_______: 0 - 127)
- Many different types of messages
  - Note on
  - Note off
  - Aftertouch
  - …
- Each message type is a command with its own parameters

Note on/off - Note number - Velocity

Status	Parameter 1	Parameter 2

Note Breakdown

python-rtmidi

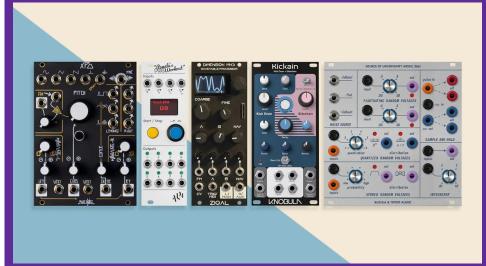
- Wrapper for RtMidi (C++ class)
- Opens communication with a MIDI device
- Allows you to send commands to control the device
- [Link]
- [Github]

What device?

- Synthesizer (keyboard or module)
- Plugin/effect
- Digital instrument

ValhallaSupermassive
Version 1.0.0
www.valhalladsp.com

MIX 50.0 %
WIDTH 100.0 %
DELAY 300.0 ms
WARP 0.0 %
Msec
CLEAR
FEEDBACK 50.0 %
DENSITY 0.0 %
MOD Rate 0.50 Hz
Depth 50.0 %
EQ High 20000 Hz
Low 10 Hz
MODE: Gemini
PRESET: Default

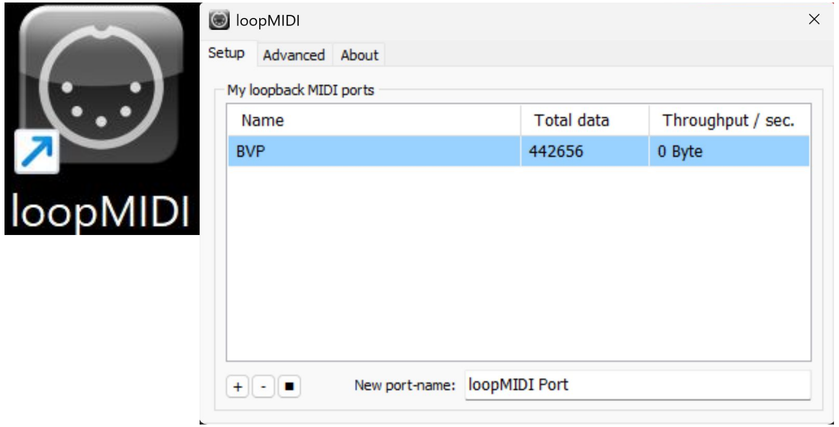


DecentSampler
Options
BROWSE
Subfrost Granular 001
VOLUME 0.00 dB
TUNE 0.00 st
FILE...
Venus Theory
subfrost
Version 1 Free Edition
Amp 1
Luxe

How to Send Messages

MIDI cable:

Virtual MIDI cable



loopMIDI

Setup | Advanced | About

My loopback MIDI ports

Name	Total data	Throughput / sec.
BVP	442656	0 Byte

New port-name: loopMIDI Port

Link

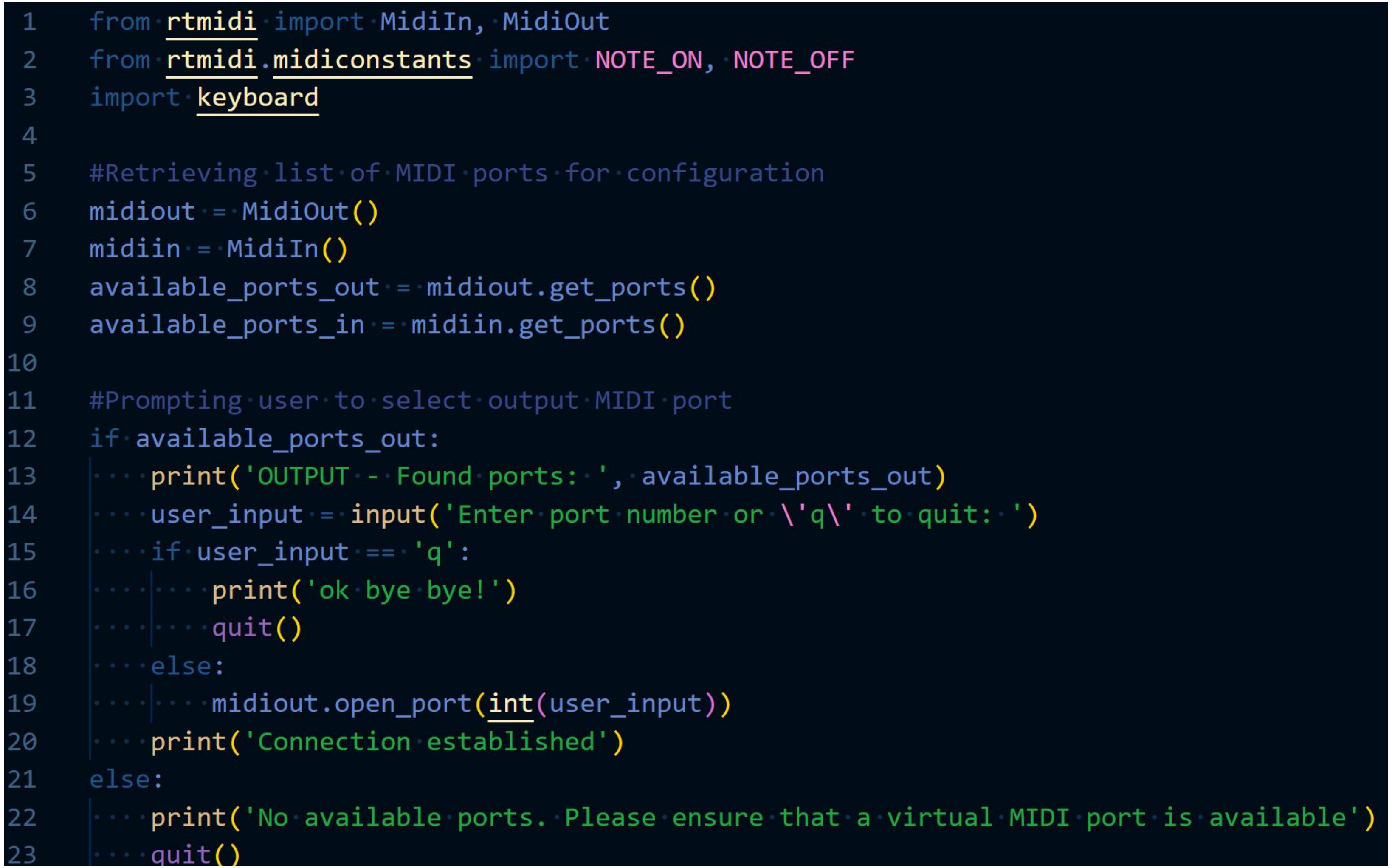
Some Projects

It's all about MAPPINGS

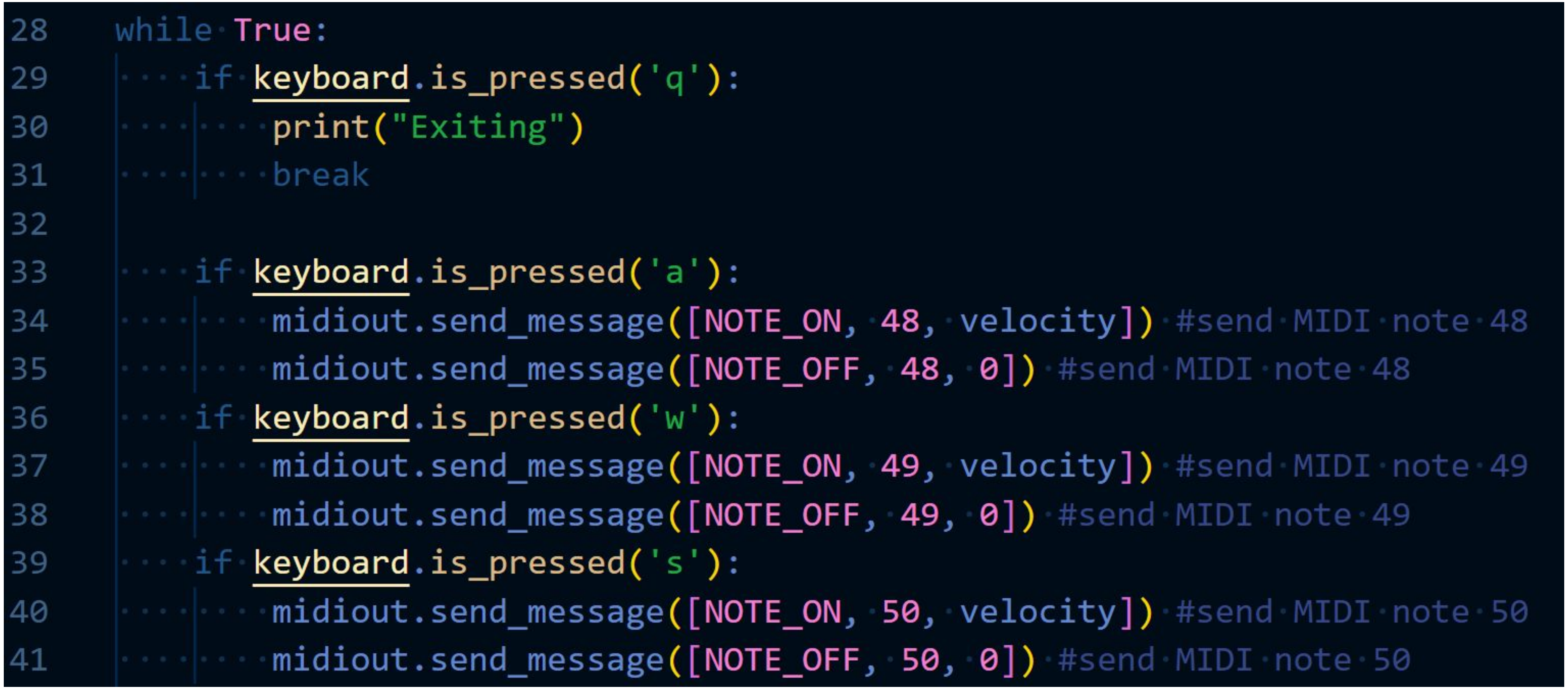
- Camera Remote Trigger
  - Analyze video feed
  - Map event to note or effect

- T-Stick
  - Tube equipped with many sensors
  - Each sensor maps to a parameter
  - Uses Max MSP (runs C++)

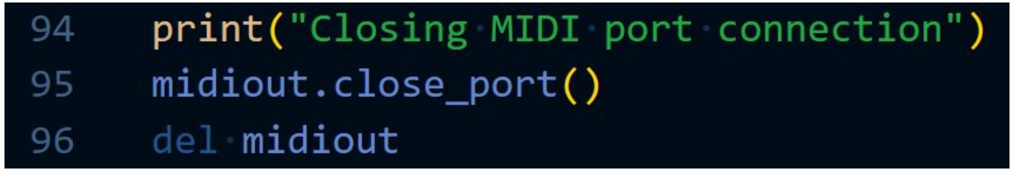
Link



```python
from rtmidi import MidiIn, MidiOut
from rtmidi.midiconstants import NOTE_ON, NOTE_OFF
import keyboard

#Retrieving list of MIDI ports for configuration
midiout = MidiOut()
midiin = MidiIn()
available_ports_out = midiout.get_ports()
available_ports_in = midiin.get_ports()

#Prompting user to select output MIDI port
if available_ports_out:
    print('OUTPUT - Found ports: ', available_ports_out)
    user_input = input('Enter port number or \'q\' to quit: ')
    if user_input == 'q':
        print('ok bye bye!')
        quit()
    else:
        midiout.open_port(int(user_input))
    print('Connection established')
else:
    print('No available ports. Please ensure that a virtual MIDI port is available')
    quit()
```



```python
while True:
    if keyboard.is_pressed('q'):
        print("Exiting")
        break

    if keyboard.is_pressed('a'):
        midiout.send_message([NOTE_ON, 48, velocity]) #send MIDI note 48
        midiout.send_message([NOTE_OFF, 48, 0]) #send MIDI note 48
    if keyboard.is_pressed('w'):
        midiout.send_message([NOTE_ON, 49, velocity]) #send MIDI note 49
        midiout.send_message([NOTE_OFF, 49, 0]) #send MIDI note 49
    if keyboard.is_pressed('s'):
        midiout.send_message([NOTE_ON, 50, velocity]) #send MIDI note 50
        midiout.send_message([NOTE_OFF, 50, 0]) #send MIDI note 50
```



```python
print("Closing MIDI port connection")
midiout.close_port()
del midiout
```