BUILD A STEP SEQUENCER USING PYTHON

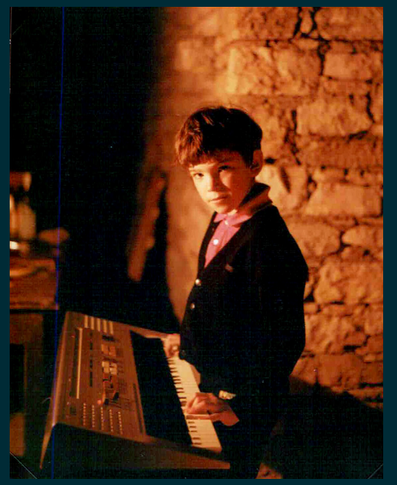
FOSDEM '17

Brussels
4 & 5 February 2017

WHO AM I?

Yann Gravrand (@ygravrand)

- Techie
- Musician



PART 1: BACKGROUND

- Musical instruments
- Synthetizers and samplers
- Sequencers
- Step sequencers

MUSICAL INSTRUMENTS

- Can be played by humans

- Some can be "played" by computers:
  - Synthetizers
  - Samplers
  - ...

SYNTHETIZERS

- Sound generators
- Lots of parameters can be tweaked

FAMOUS SYNTHETIZERS

Minimoog (analog)

DX7 (digital)

FAMOUS SYNTHETIZERS

Nord Lead
(analog modeling)

Mininova
(analog modeling)

VST

VST Plugins

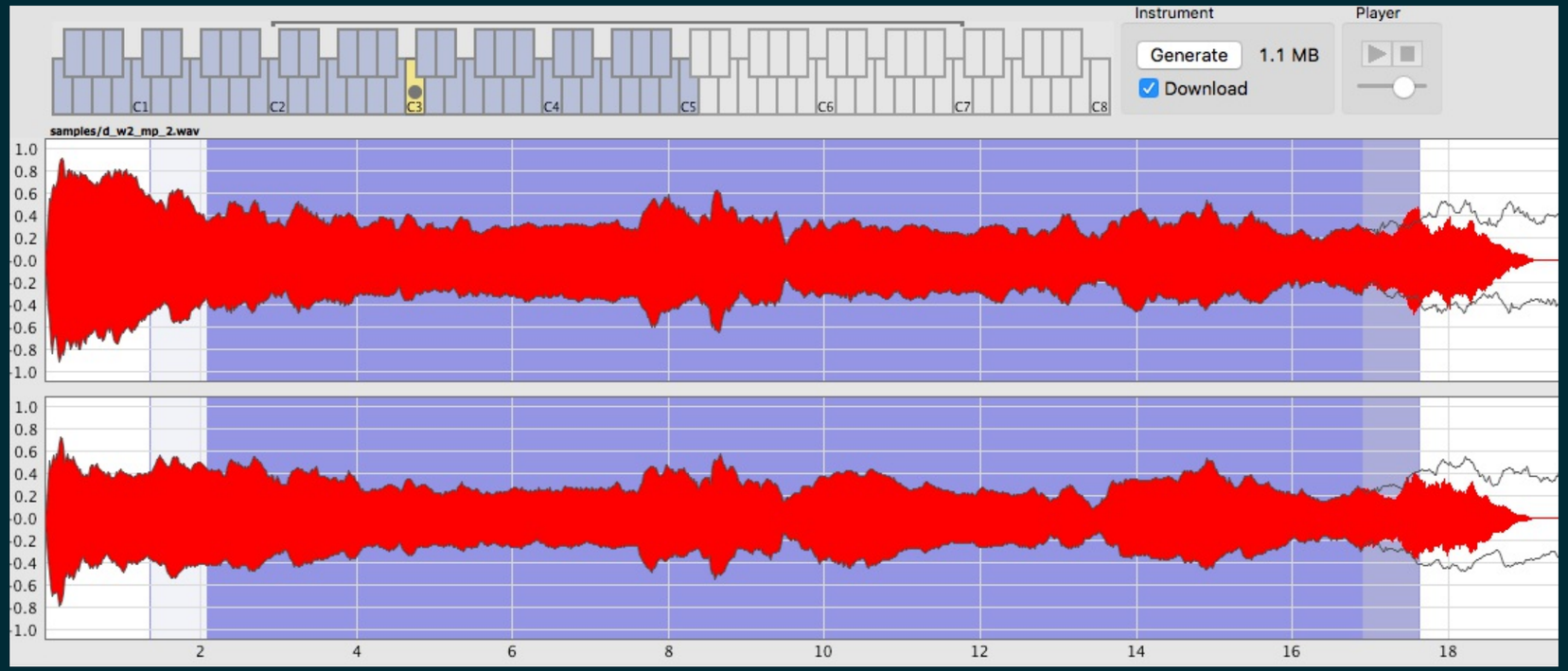
SAMPLERS

Do not *generate* sounds themselves

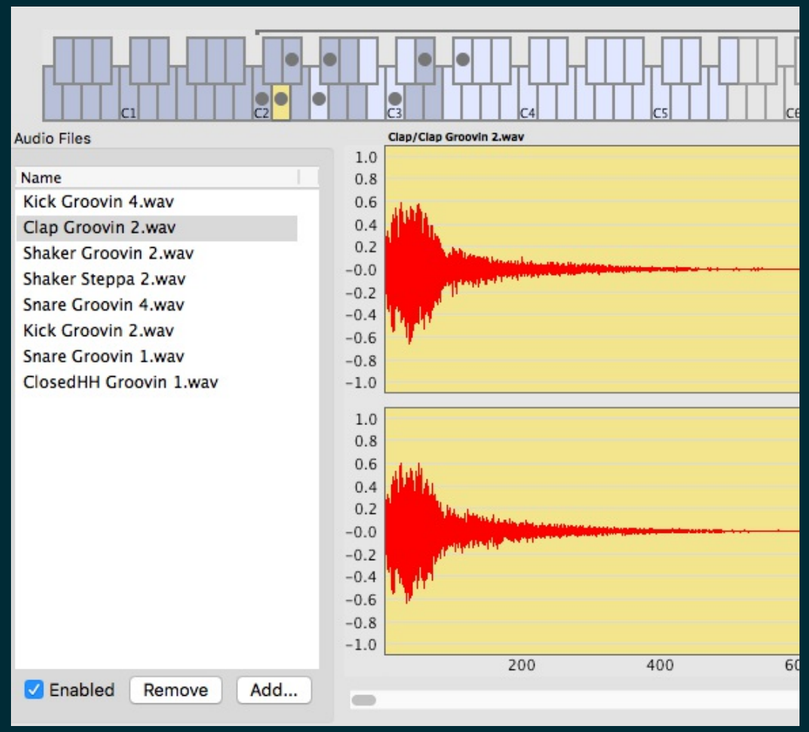
Play samples (little chunks of sound)

SAMPLES / NOTES:

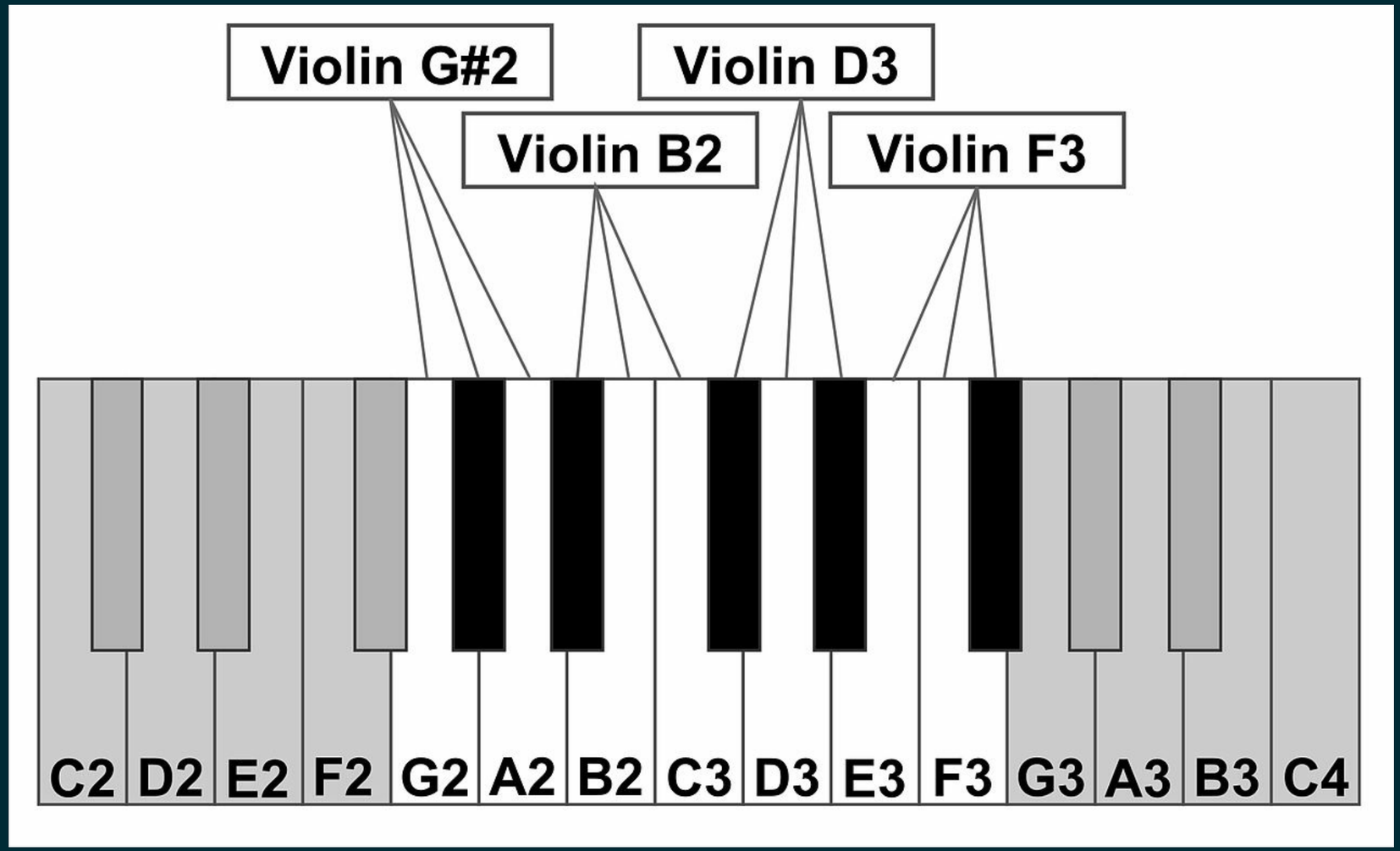
- One sample for the whole keyboard (pitch adjusted or not)



- One sample for each note



- One sample for a group of notes, pitch is ajusted



DRUM MACHINES?

Sound generator (drum oriented) + step sequencer

TR 909

Tempest

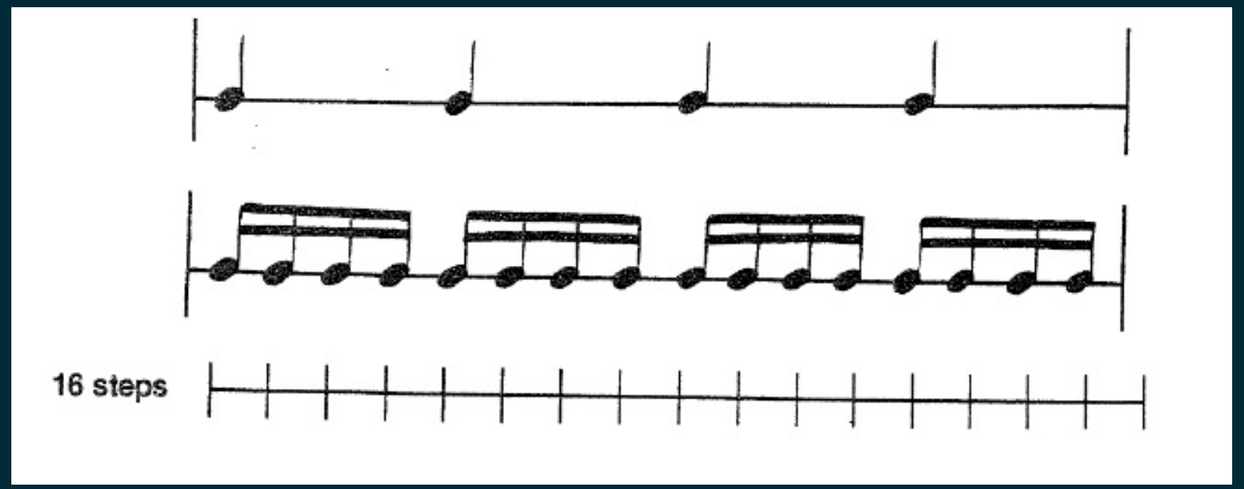
SEQUENCERS

- Play a sequence of notes
- Several tracks, instruments...

STEP SEQUENCER

A 4/4 measure is divided into:

- 4 quarter notes
- Each quarter note is divided into 4 steps --> A sequence like this is 16 steps long



STEP SEQUENCER

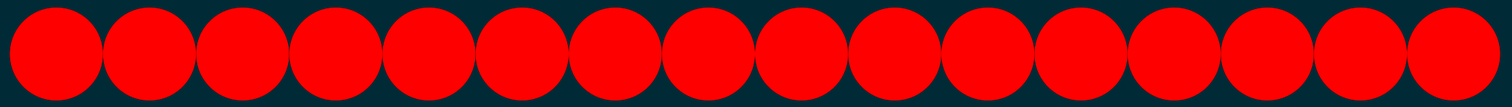
For each step, we define:

- the note / pitch
- other attributes: length...
- ... and activate it or not

EXAMPLES

Daft punk - Aerodynamic @ 1:03

4 * 16-step patterns



EXAMPLES

Daft punk - Aerodynamic @ 2:28

4 * 16-step patterns, some notes off

USING A STEP SEQUENCER

- "Step by step" mode: for each step, define the note attributes. No timing, no rush
- "Live" mode: turn steps on and off in real time, adjust pitch, length...

PART 2: THE PROJECT

- Project goals
- MIDI
- Using mido
- The Dirty Part: blocking, threads, asyncio...

I HAD

A cool synth

Colorful (and empty) pads

AND

A snake

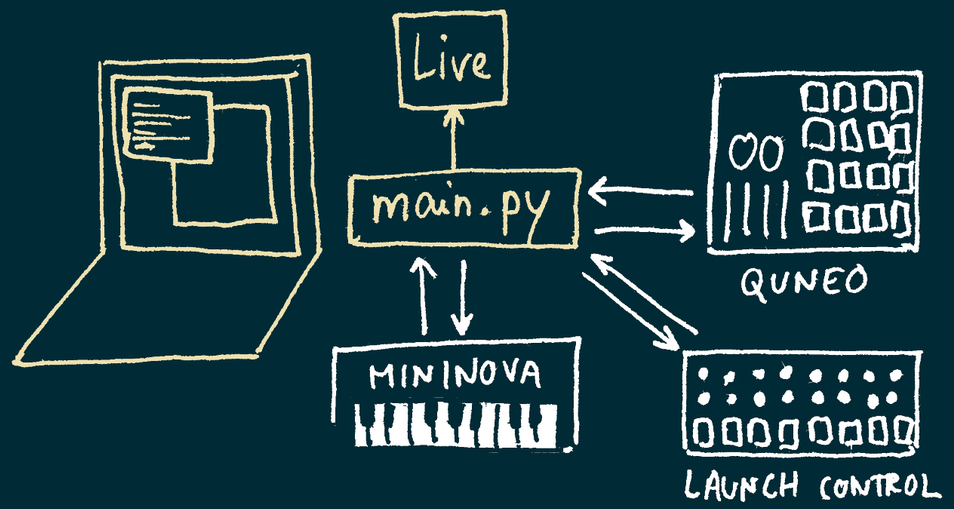
PROJECT GOALS

- Make the synthetizer play notes using Python
- Modify and turn notes on / off to create a sequence
- Implement "step by step" and "live" modes
- Change tempo in real time
- Make interactions possible with *any* controller...
- ... Starting with mine, of course :)
- No GUI, focus on usability with hardware (live oriented)

MIDI: MUSICAL INSTRUMENT DIGITAL INTERFACE

- Extremely old standard: 1983!
- Still largely in use today
- To synchronize and communicate between devices
- Message types:
  - Notes (NOTE ON, NOTE OFF)
  - Control Change (Ex: Filter resonance, Hold pedal...)
  - Program Change (Change instrument)
  - Sys ex
  - ...

WE WILL NEED TO SPEAK MIDI WITH DEVICES



- Midi input: pads pressed, keys pressed, knobs turned...
- Midi output: play a note, turn a LED on...

MIDI INPUT: RECEIVING MESSAGES

```
inport = mido.open_input()
msg = inport.receive()   # Blocking call
```

Message reception blocks

So if we want to do something else in parallel, we have to handle this in a thread or coroutine or...?

MIDI OUTPUT: PLAYING NOTES

```
import mido

outport = mido.open_output()
msg = mido.Message('note_on', note=100, velocity=3)
outport.send(msg)
```

--> BEEEEEEEEEEEEEEEEEEEE...

```
outport.send(mido.Message('note_off', note=100))
```

--> ... EEEP.

To play notes, we need a timer between NOTE_ON and NOTE_OFF (note duration). time.sleep?

ALIGNING NOTES (STEPS) WITH TEMPO

Naive implementation:

```
while True:
    outport.send(mido.Message(...))
    time.sleep(tempo.step_duration)
```

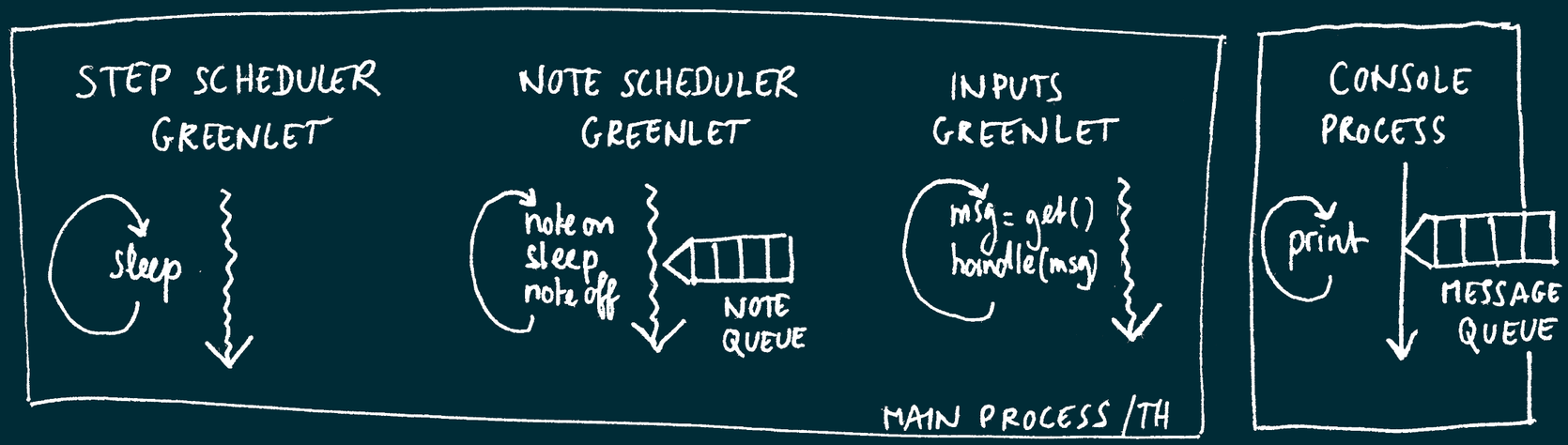
Two problems:

- time.sleep also blocks, so we have to handle it in a thread or coroutine or...
- Waking up, sleeping for X seconds, waking up...: the tempo slowly drifts. Calculate absolute times

SOLUTIONS

- Threads
  - Many queues to avoid shared state
- Coroutines with asyncio
  - Everything in a single thread, less concurrency issues
  - Ok since our app is I/O bound
  - ...But we have to modify mido to insert yield from or await...
- Greenlets with gevent
  - Monkey patches time.sleep so we can use mido as is and have greenlets

PROPOSED DESIGN



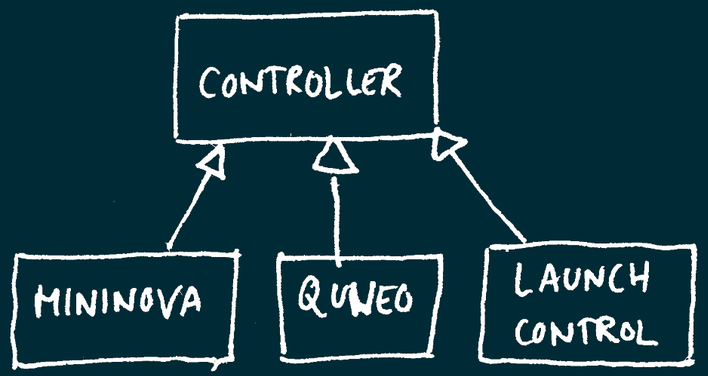
- Main process is I/O bound
- Console process is CPU bound!

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
791	**pi**	**20**	**0**	**23812**	**15336**	**4424**	**R**	**99.9**	**1.6**	**1:05.86**	**python**
782	pi	20	0	21572	16476	7800	S	29.8	1.7	1:22.44	python

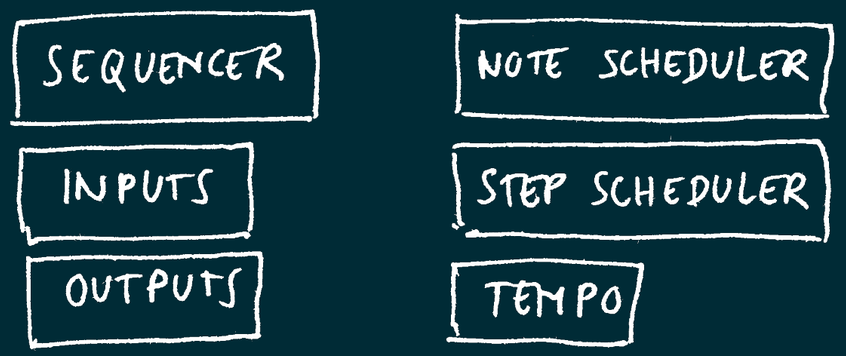
PART 3: IMPLEMENTATION & DEMO

- System overview
- Implementing a controller
- Action!

SYSTEM OVERVIEW



\+



...

IMPLEMENTING A CONTROLLER

- Map messages from controller (pad pressed) to sequencer actions (toggle step)
- Send messages to controller for feedback (LEDs...)

INTERPRETING EVENTS FROM CONTROLLERS

- Some events are represented by a single message
- Others are the result of a sequence of messages (ex: NPRN LSB, MSB)
- Solution: a RulesChain
  - Each Rule matches a message
  - A state automaton keeps track of the matched rules
  - Flexible rules evaluation engine

```
self.register('FILTER',
              self.on_cc,
              RulesChain(Rule(type_='control_change', control='74'),
                         Rule(type_='control_change', control='27',
                              value='0'))
             )
```

REACTING TO SEQUENCER EVENTS

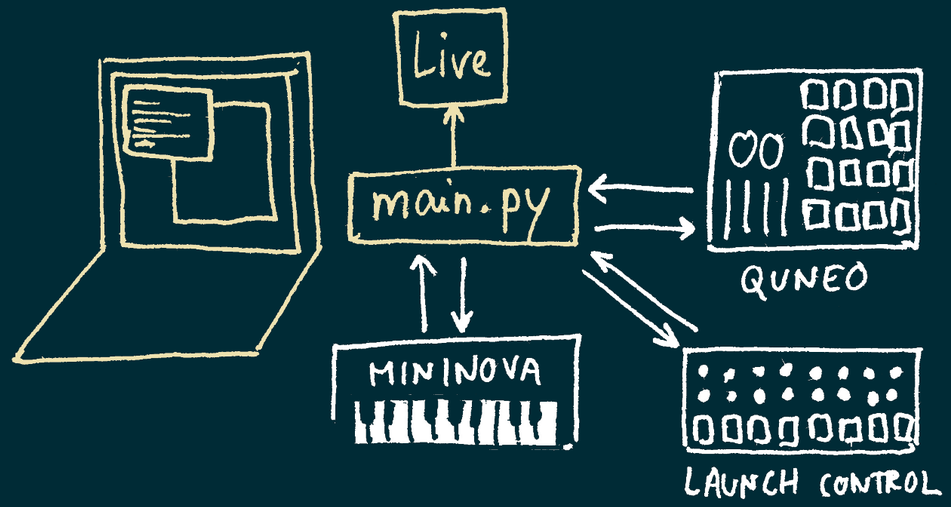
```
self.sequencer.on(SequencerEvents.STEP_BEGIN, self, self.on_step_begin)

...

def on_step_begin(self, step):
    # Turn on current step LED
    self.sequencer.output(self, *msb_lsb_output(60, 0, 32 + step.pos))
```

IN ACTION!

IN ACTION!



- Bass pattern
- Drum pattern 1
- Drum pattern 2
- Mozart pattern (32-step sequence)
- Daft punk - da funk
- Remote console

WHY PYTHON?

BENEFITS

- Easy to read, easy to write
- The dynamic features of Python and plugin system make writing controllers easy!
- Large ecosystem

CHALLENGES

- Python is not the best choice for real-time computing
- Performance on tiny devices (C.H.I.P, Rpi...)
  - Steppy was designed with simplicity in mind (gevent / single thread execution model)
  - Implies we must be "green" and use the least CPU possible

WHERE IS MY CPU?

- Rules evaluation engine:
  - Speed can be improved: PyPy, Cython, Numba...?
- Pretty printing (large characters):
  - Isolate on a core
  - Move the problem - using Websockets!

FUTURE PLANS

- Chords (especially important for a drum machine...)
- Multi track
- Load / save to midi
- External tempo sync
- Better reactive Web interface
- Web interface for rules config (like Live's mappings)
- Other protocols: DMX...

THANK YOU!

@ygravrand

github.com/ygravrand/steppy