PYACQ

A python framework for acquisition and monitoring experiment: the distributed way

OUTLINES

- Goals
- Overview
- Nodes list
- Demo

GOALS

BUILD SETUPS IS LONG

- Drive several devices in parralel
- Make severals materials speaking together
- Deeling with large data
- Having a good monitoring during experiment

EXPERIMENTAL SETUP EXAMPLE

Charaterize place cells during experiments:

- Grab multichannel
- Grab cam images
- Do rougth spike sorting
- Do rougth animal tracking
- Estimate and plot cells density

INGENEER POINT OF VIEW

- Making differents software working together is hard.
- Lab softwares are very specifics. They often do not exists.

  So : I do not need Software but modules to build them.

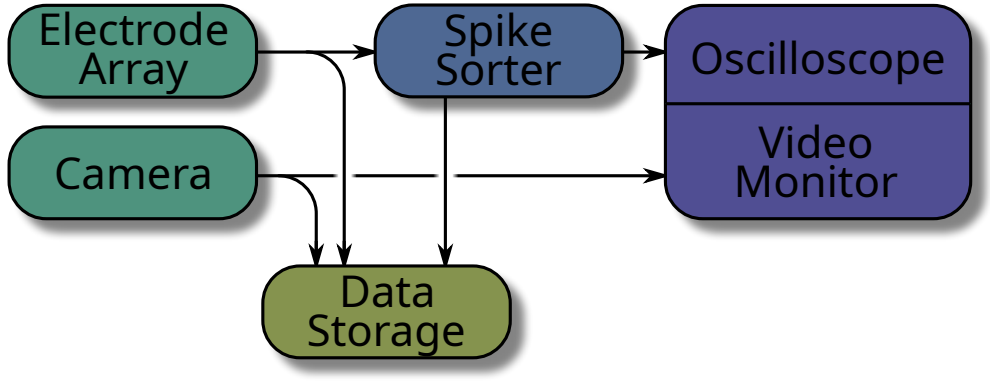
HIGH COMPUTATIONS

- Need of several machine or CPU/GPU
- Need of scaling the computational cost

PYACQ OVERVIEW

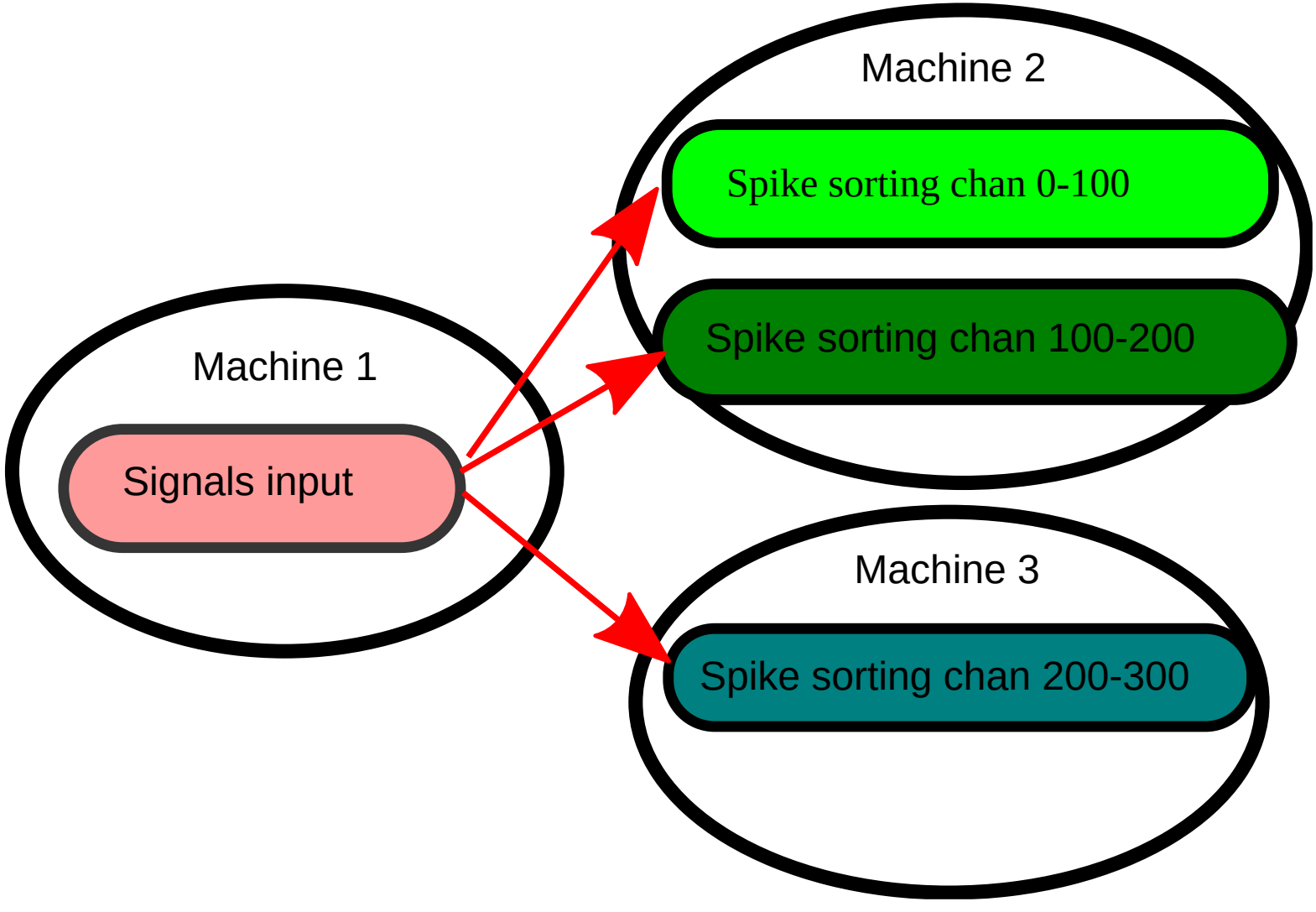
DISTRIBUTED DATA ACQUISITION + STREAM PROCESSING SYSTEM

Graph of Nodes



DISTRIBUTE NODES ACROSS

- thread
- process
- machine


Machine 2
Spike sorting chan 0-100
Spike sorting chan 100-200
Machine 1
Signals input
Machine 3
Spike sorting chan 200-300

SIMPLE END USER'S CODE

```
import pyacq

# Connect to a remote host and create a new process there
manager = pyacq.create_manager('rpc')
worker_host = manager.add_host('tcp://10.0.0.103:5678')
worker = worker_host.create_nodegroup()

# Create nodes for data acquisition, analysis, storage, and display
device = manager.create_node('NiDAQmx')
analyzer = manager.create_node('Spikesorter', host=worker)
recorder = manager.create_node('HDF5Recorder', host=worker)
viewer = manager.create_node('QOscilloscope')

# Connect all nodes together
analyzer.input.connect(device.output)
recorder.input.connect(analyzer.output)
viewer.input.connect(analyzer.output)

# Begin acquiring and streaming data
manager.start_all()
```

FAST DATA TRANSFER BETWEEN NODES

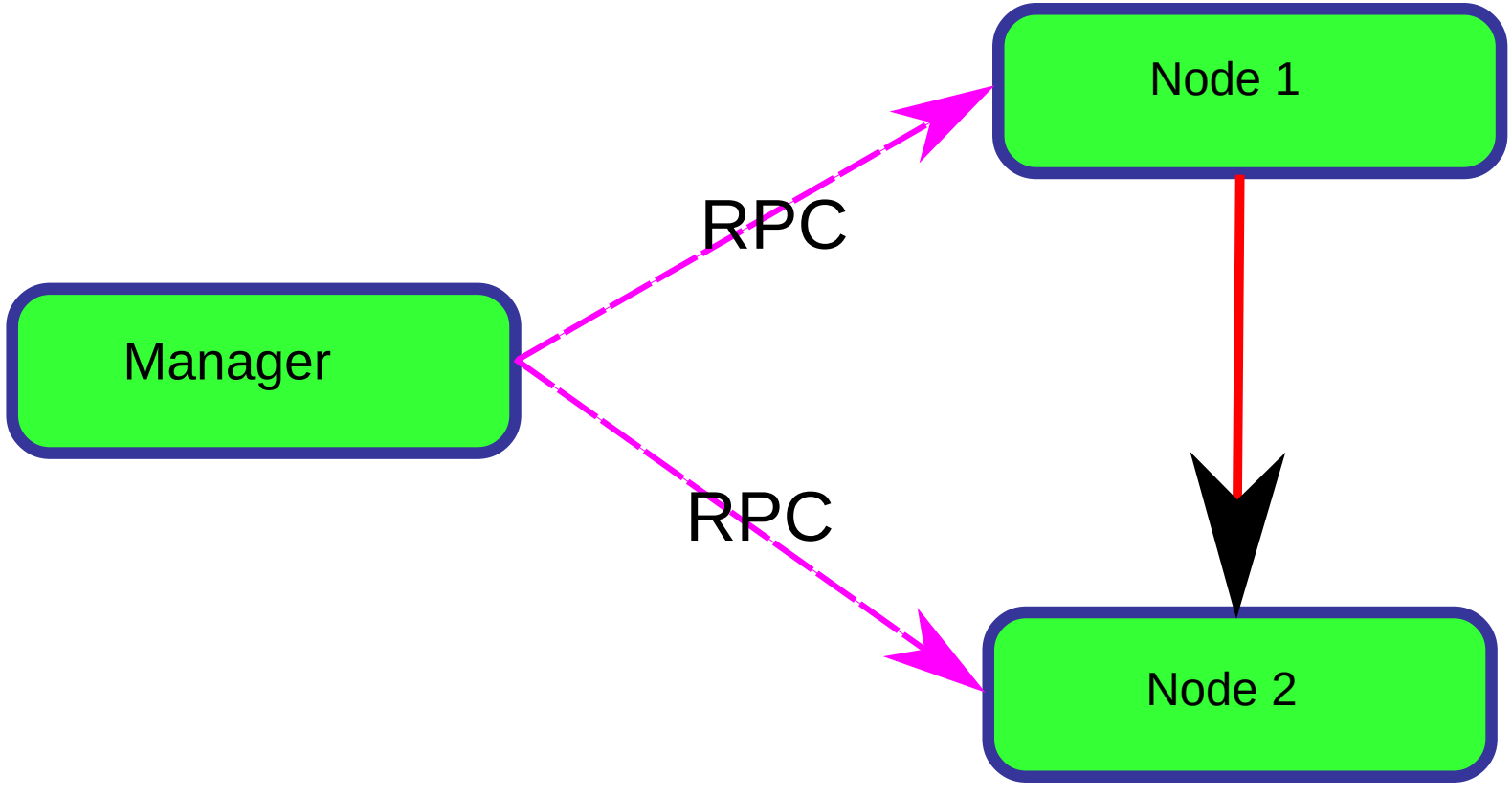
On top on zmq.

Several senarios for data transfer:

- full data in message
- compressed or not
- no copy if in same thread
- shared memory in same machine

ALL NODES ARE REMOTE CONTROLLED

RPC = Remote Precedure Call


Node 1
RPC
Manager
RPC
Node 2

PERFORMANCE

Pyacq focused on:

- Easy for dev
- Distribution

If high performance needed, Node can be writeen in C cuda, opencl, ...

NODE LIST: ACQUISITION

- National Instruments IO cards
- Measurement computing IO cards
- Webcam
- EEG systems : Emotiv, BrainAmp, (OpenEEG)
- Blackrock
- Multichannel system

To be done : Intan board (and OpenEphys)

NODE LIST: PROCESSING

- Filters (with scipy or OpenCL)
- Threshold detection (=Trigger)
- Trigger average

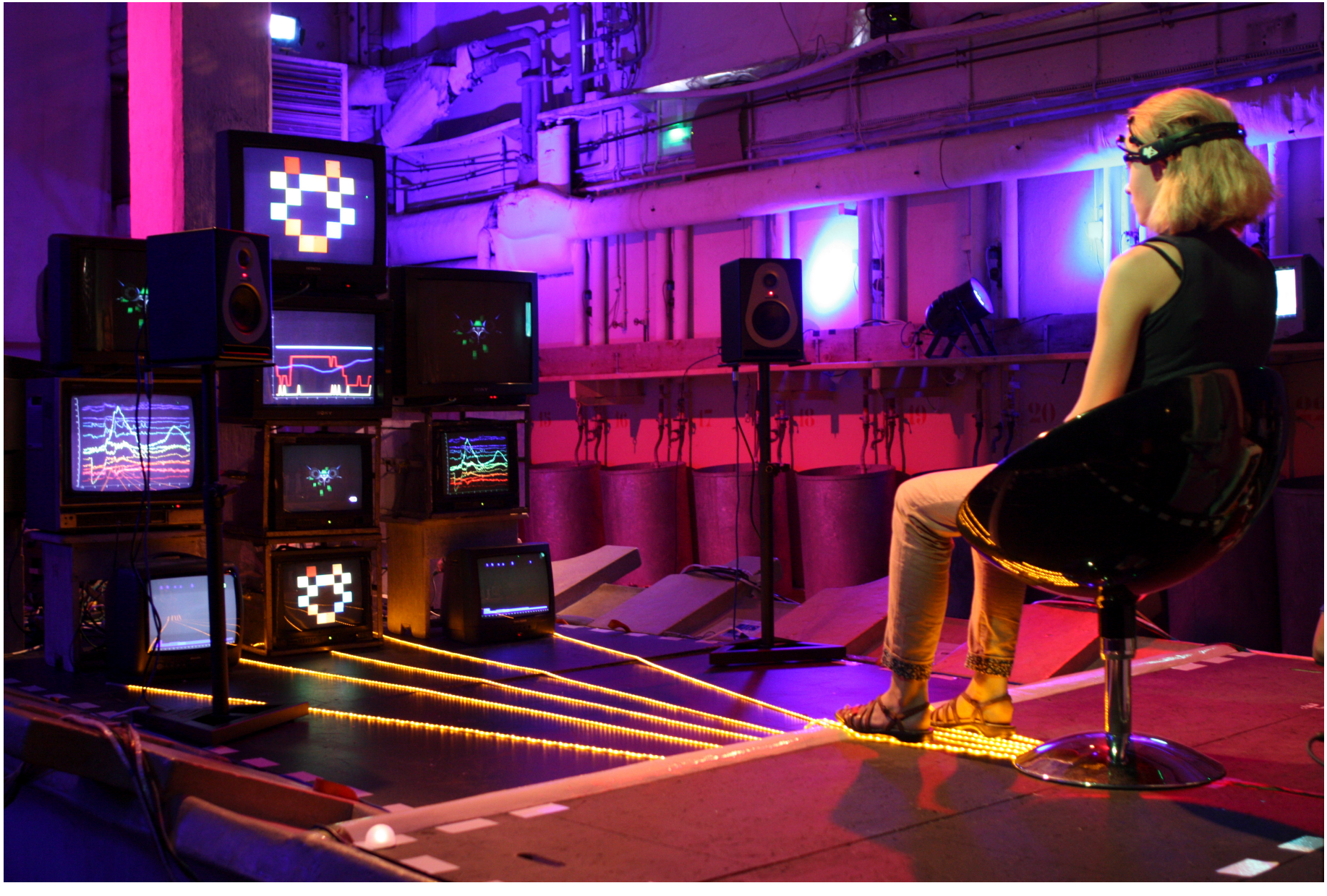
NODE LIST: VISUALISATION

- Oscilloscope
- Time Frequency (morlet) viewer
- Trigger average (ERP)

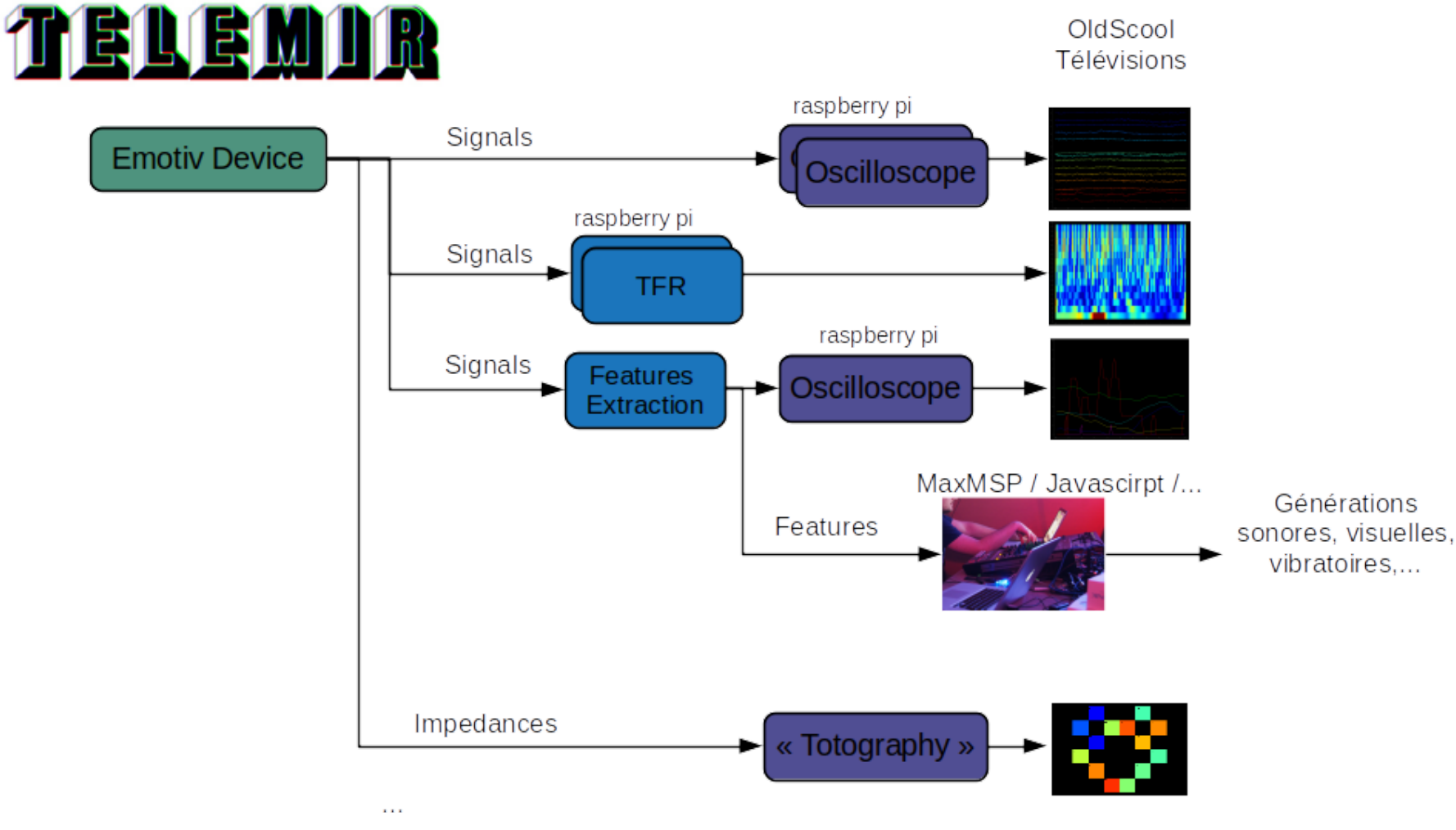
AND OF COURSE: CREATE YOUR CUSTUM NODE

Et soyez, vous aussi le roi des noeuds.

EXAMPLE : TELEMIR



EXAMPLE : TELEMIR



DEMO

CONLUSION

- Download it: https://github.com/pyacq/pyacq
- Build your own software
- Share your pyacq nodes
- It is open source!!!

OTHER DEVS

Luke (Alen Institute)