MAKING LIFE EASIER WITH

DECORATORS FOR PYZMQ

YOURS TRULY

Electronics hobbyist since childhood

Active in Finlands hackerspace scene

Believer: Open Source & Open Hardware

https://github.com/rambo

http://fi.linkedin.com/in/eeroafheurlin/

ØMQ AKA Z(ERO)MQ

- FAST, **VERY** FAST
- LOW LEVEL
- NO BUILT-IN SERVICE DISCOVERY
  - RECOMMENDED PATTERNS EXIST
- MULTIPLE TRANSPORTS
  - INPROC
  - UNIX SOCKETS
  - NETWORK SOCKETS
- PLENTY OF LANGUAGE BINDINGS

BACKGROUND

- D-BUS DECORATORS AS INSPIRATION
  - BUT D-BUS HAS ANNOYING LIMITATIONS
  - SUPER-EASY TO USE THOUGH
- ORIGINALLY FROM “REACTOR SIMULATOR”
  - HTTPS://GITHUB.COM/HELSINKIHACKLAB/REACTOR

Photo: Tuula Ylikorpi

CURRENT STATUS

- WORKS
  - ASYNC ONLY
- THERE WILL BE API CHANGES
- HTTPS://PYPI.PYTHON.ORG/PYPI/ZMQDECORATORS

RPC DEMO

- ASYNC
- ROUTER/ DEALER SOCKET

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import zmq
import zmqdecorators

SERVICE_NAME="test_asyncrpc"
SERVICE_PORT=6900 # Set to None for random port

class myserver(zmqdecorators.service):
    def __init__(self, service_name, service_po
        super(myserver, self).__init__(service_
        # TODO: other init code ??

    def cleanup(self):
        print("Cleanup called")

    @zmqdecorators.method()
    def beer(self, resp, bottles, drinkers):
        bottles = int(bottles)
        drinkers = int(drinkers)
        print "Sending bottles as reply"
        # Remember, ZMQ only deals in strings,
        resp.send("Here's %d bottles of beer fo

    @zmqdecorators.method()
    def food(self, resp, arg, arg2):
        print "Sending noms as reply"
        # Remember, ZMQ only deals in strings,
        resp.send("Here's %s for the noms (for


if __name__ == "__main__":
    instance = myserver(SERVICE_NAME, SERVICE_P
    print("Starting")
    instance.run()

```

SIGNALS DEMO

- PUB/SUB SOCKET

```python
class mypublisher(zmqdecorators.service):

    def __init__(self, service_name):
        super(mypublisher, self).__init__(service_name)

        self.pcb = ioloop_mod.PeriodicCallback(self.bottles_cal
        self.pcb.start()
        self.pcb2 = ioloop_mod.PeriodicCallback(self.slices_cal
        self.pcb2.start()

        self.pcb3 = ioloop_mod.PeriodicCallback(self.noargs, 50
        self.pcb3.start()


    @zmqdecorators.signal(SERVICE_NAME, SIGNALS_PORT)
    def noargs(self, *args):
        """What this function actually does, does not matter to
        print "No args signal called"
        pass

    @zmqdecorators.signal(SERVICE_NAME, SIGNALS_PORT)
    def bottles(self, n):
        """What this function actually does, does not matter to
        pass

    def bottles_caller(self):
        """We call this function from the eventloop so we have
        n = random.randint(0,100000)
        data = "%s bottles of beer on the wall" % n
        print data
        return self.bottles(data)

    @zmqdecorators.signal(SERVICE_NAME, SIGNALS_PORT)
    def slices(self, n):
        pass

    def slices_caller(self):
        n = random.randint(0,100000)
        data = "%s slices in the box" % n
        print data
        return self.slices(data)



if __name__ == "__main__":
    instance = mypublisher(SERVICE_NAME)
    print("Starting")
    instance.run()
```

FUTURE PLANS

- CODE CLEANUP
- AUTOMAGICAL TRANSACTION TRACKING
- BETTER ERROR HANDLING
- REQ/REP SUPPORT ?

ALL DONE, 2MIN Q&A

- HTTPS://PYPI.PYTHON.ORG/PYPI/ZMQDECORATORS
- HTTPS://GITHUB.COM/RAMBO/PYTHON-ZMQDECORATORS
- OTHER QUESTIONS ?

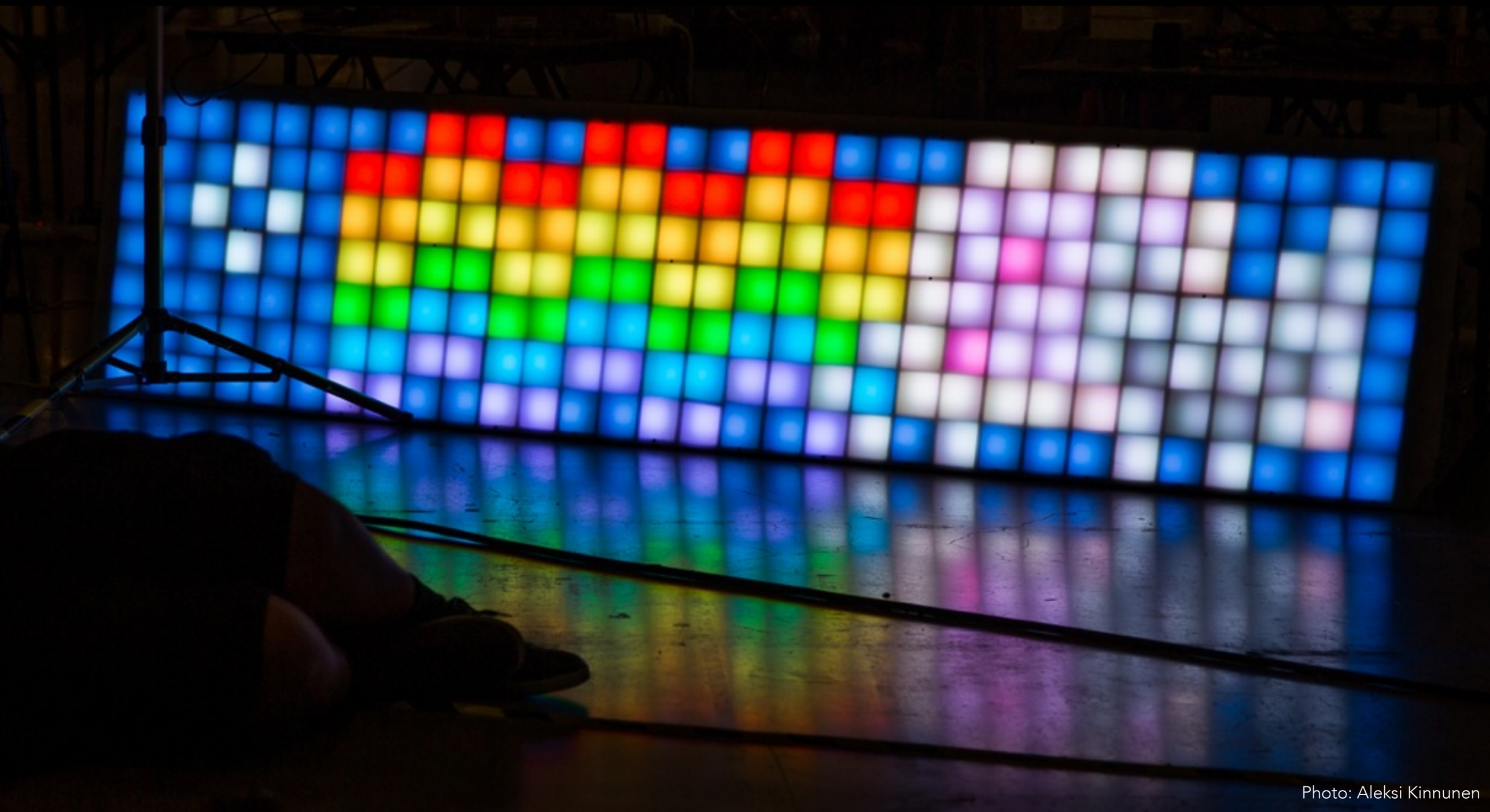

Photo: Aleksi Kinnunen