**Purpose**

The purpose of this file is to illustrate the ability to connect and use a PIR sensor to RPi and use OSC messaging through Python script to address Supercollider to produce sound, all within RPi.

**Prerequisites**

1. Raspberry Pi 2
2. sd card with 2015-11-21-raspbian-jessie-lite.img or newer raspbian jessie
3. router with ethernet internet connection for the rpi
4. plastic case for RPi (optional)
5. usb soundcard
6. power cable for RPI
7. laptop connected to same network as the rpi
8. patience

**Setup Steps**

**(1) Compiling SC master natively on Raspberry Pi Raspbian Jessie**

**A (hardware setup)**

1. connect an ethernet cable from the network router to the rpi

2. insert the sd card and usb soundcard

3. last connect usb power from a 5V@1A power supply

**B (login & preparations)**

1. ssh pi@raspberrypi.local #from your laptop, default password is raspberry

2. sudo raspi-config #change password, expand file system, reboot and log in again with ssh

**C (update the system, install required libraries & compilers)**

1. sudo apt-get update

2. sudo apt-get upgrade

3. sudo apt-get install alsa-base libicu-dev libasound2-dev libsamplerate0-dev libsndfile1-dev libreadline-dev libxt-dev libudev-dev libavahi-client-dev libfftw3-dev cmake git gcc-4.8 g++-4.8

**D (compile & install jackd (no d-bus) )**

1. git clone git://github.com/jackaudio/jack2.git --depth 1

2. cd jack2

3. ./waf configure --alsa #note: here we use the default gcc-4.9

4. ./waf build

5. sudo ./waf install

6. sudo ldconfig

7. cd ..

8. rm -rf jack2

9. sudo nano /etc/security/limits.conf #and add the following two lines at the end:

@audio - memlock 256000

@audio - rtprio 75

Ctrl-o to save Ctrl-x to exit

**E (compile & install sc master)**

1. git clone --recursive git://github.com/supercollider/supercollider.git supercollider

2. cd supercollider

3. git submodule init && git submodule update

4. mkdir build && cd build

5. export CC=/usr/bin/gcc-4.8 #here temporarily use the older gcc-4.8

6. export CXX=/usr/bin/g++-4.8

7. cmake -L -DCMAKE\_BUILD\_TYPE="Release" -DBUILD\_TESTING=OFF -DSSE=OFF -DSSE2=OFF -DSUPERNOVA=OFF -DNOVA\_SIMD=ON -DNATIVE=OFF -DSC\_ED=OFF -DSC\_WII=OFF -DSC\_IDE=OFF -DSC\_QT=OFF -DSC\_EL=OFF -DSC\_VIM=OFF -DCMAKE\_C\_FLAGS="-mtune=cortex-a7 -mfloat-abi=hard -mfpu=neon -funsafe-math-optimizations" -DCMAKE\_CXX\_FLAGS="-mtune=cortex-a7 -mfloat-abi=hard -mfpu=neon -funsafe-math-optimizations" ..

8. make -j4 #leave out flag j4 on single core rpi models

9. sudo make install

10. sudo ldconfig

11. cd ../..

12. rm -r supercollider

13. sudo mv /usr/local/share/SuperCollider/SCClassLibrary/Common/GUI /usr/local/share/SuperCollider/SCClassLibrary/scide\_scqt/GUI

14. sudo mv /usr/local/share/SuperCollider/SCClassLibrary/JITLib/GUI /usr/local/share/SuperCollider/SCClassLibrary/scide\_scqt/JITLibGUI

**F (start jack & sclang & test)**

1. jackd -P75 -dalsa -dhw:1 -p1024 -n3 -s -r44100 & #edit -dhw:1 (to match your soundcard. usually it is 1 for usb)

2. sclang (should start sc and compile the class library with only 3 harmless class overwrites warnings)

3. s.boot #should boot the server

4. a= {SinOsc.ar([400, 404])}.play (should play sound in both channels)

5. a.free

6. {1000000.do{2.5.sqrt}}.bench (benchmark: ~0.89 for rpi2, ~3.1 for rpi1)

7. a= {Mix(50.collect{RLPF.ar(SinOsc.ar)});DC.ar(0)}.play (benchmark)

8. s.dump (avgCPU should show ~19% for rpi2 and ~73% for rpi1)

9. a.free

10. exit #quit sclang

11. pkill jackd #quit jackd

**(2) Python - pyOSC module install**

Python - pyOSC module install You need pyOSC module in Python to send OSC messages to SC

Open the terminal and type the following:

1. ssh pi@raspberrypi.local #to log into pi (default username: pi and password: raspberry)

2. sudo apt-get update

3. sudo apt-get upgrade

4. curl -O https://trac.v2.nl/raw-attachment/wiki/pyOSC/pyOSC-0.3.5b-5294.zip -k

5. unzip pyOSC-0.3.5b-5294.zip

6. sudo python ./setup.py install

**(3) Python simple code for OSC messaging to SC using a PIR sensor**

Now it is time to write the code in Python so the PIR OSC messages can be sent to SC server locally.

Open the terminal and type the following:

1. sudo nano filename.py

opens a blank sheet in Python named filename.py (I used “pirtest.py”)

1. paste the following code into Python

import RPi.GPIO as GPIO

import time

import OSC

sensor = 21 #the pin you connected the PIR to

GPIO.setmode(GPIO.BCM)

GPIO.setup(sensor, GPIO.IN, GPIO.PUD\_DOWN)

previous\_state = False

current\_state = False

lastpin = -1

sc= OSC.OSCClient()

sc.connect(('192.168.1.2', 57120)) #send locally or not to sc

def sendOSC(name, val):

msg= OSC.OSCMessage()

msg.setAddress(name)

msg.append(val)

try:

sc.send(msg)

except:

pass

print msg #debug

while True:

val= GPIO.input(sensor)

if val!=lastpin:

sendOSC("/bang", val)

lastpin=val

while True:

time.sleep(0.1)

previous\_state = current\_state

current\_state = GPIO.input(sensor)

if current\_state != previous\_state:

new\_state = "HIGH" if current\_state else "LOW"

print("GPIO pin %s is %s" % (sensor, new\_state))

ctrl o #to save ctrl x #to exit

**(4) Simple code for SC to test the osc messages coming from python using the PIR sensor. (you can put your code here).**

1. sudo nano filename.scd # choose a filename ( I used "mycode.scd")
2. enter the following SC code (or your code):

(

Server.default.waitForBoot{

Task({

SynthDef.new(\noise, {

arg freq = 800, gate = 0,

amp = 0.5, out = 0;

var sig, env;

env = EnvGen.kr(Env.adsr(0.01, 0.1, 1, 10),gate);

sig = LFNoise2.ar(freq, amp);

sig = sig \* env;

Out.ar(out, sig).dup;

}).add;

5.wait;

x=Synth.new(\noise);

}).play;

OSCdef.new(

\bang,

{

arg msg, time, addr, port;

[msg, time, addr, port].postln;

x.set(\gate, msg[1]);

},

'/bang'

);

}

)

Ctrl-o #to save Ctrl-x #to exit

**(5) Autostart instructions for RPi**

Now it is time to program RPi to autostart the scripts in Python and SuperCollider on boot. Assuming all the above steps are done correctly toy should have created 2 files :

1. filename.scd (I have mycode.scd)
2. filename.py (i have pirtest.py)

Now we will proceed to the next step. We need to create 2 files (autostart.sh and pythonlauncher.sh) In the terminal type:

1. sudo nano ~/autostart.sh #and add the following lines:

#!/bin/bash

/usr/local/bin/jackd -P75 -dalsa -dhw:1 -p1024 -n3 -s -r44100 &

su root -c "sclang -D /home/pi/mycode.scd"

(I used "mycode.scd", you can use your filename)

Ctrl-o #to save Ctrl-x #to exit

1. sudo nano pythonlauncher.sh #and add the following lines:

#!/bin/sh

pythonlauncher.sh

cd /

cd home/pi

sudo python pirtest.py

cd /

(I used "pirtest.py" you can used your file here)

Ok now we need to make these 2 files executable:

1. chmod +x !/autostart.sh

2. chmod 755 pytholauncher.sh

Next step is to tell the machine to run these files at reboot:

1. sudo crontab -e

2. paste the following lines:

· @reboot /bin/bash /home/pi/autostart.sh

· @reboot /bin/sh /home/pi/pythonlauncher.sh

Ctrl-o #to save Ctrl-x #to exit

**(6) Reboot et voilà**

Now that is all set it is time to reboot.

1. sudo reboot

(after reboot the project should be working, if not go through the steps again in case you missed something)