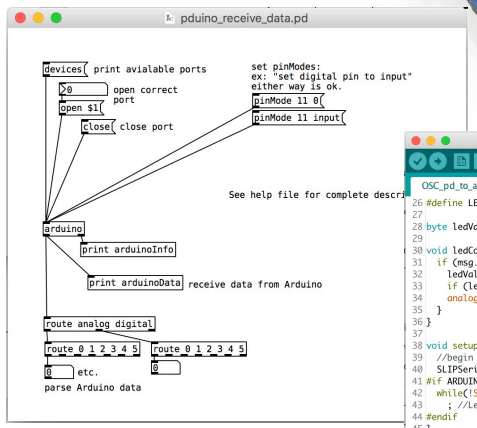
Extended topics

- Open Sound Control via Arduino
- Indirect gesture acquisition
- Inter-application audio routing



John Sullivan
Centre for Interdisciplinary Research in Music Media and Technology
Input Devices and Music Interaction Laboratory
john.sullivan2@mail.mcgill.ca

Open Sound Control (OSC)

- REVIEW:
  - network communication protocol developed at CNMAT (U. C. Berkley)
  - Can send/receive many different data types, most importantly: float, int, string, lists of data
  - custom, human readable, and hierarchical address structure (URL style)
  - Can communicate over network via UDP or TCP protocols
    - We are using UDP (User Datagram Protocol)
  - ..or can use serial port (with Arduino)

- Why OSC?
  - Handle more complex communications
  - Manage many data streams
  - Send data over networks
  - Multi-user performance
  - Working with non-standard (non-MIDI capable) instruments
  - OSC has become a widely supported industry standard for music and multimedia applications where flexible networked communications is needed.

Open Sound Control (OSC)

Some example OSC messages:

/knob 255
/myController/button 1
/myController/faders/1 0.0002
/myController/faders/2 0.91273
/myController/faders/3 0.26473
/myController/XYZdeg 99 340 201
/myController/status hello
/myController/status Everything ok.

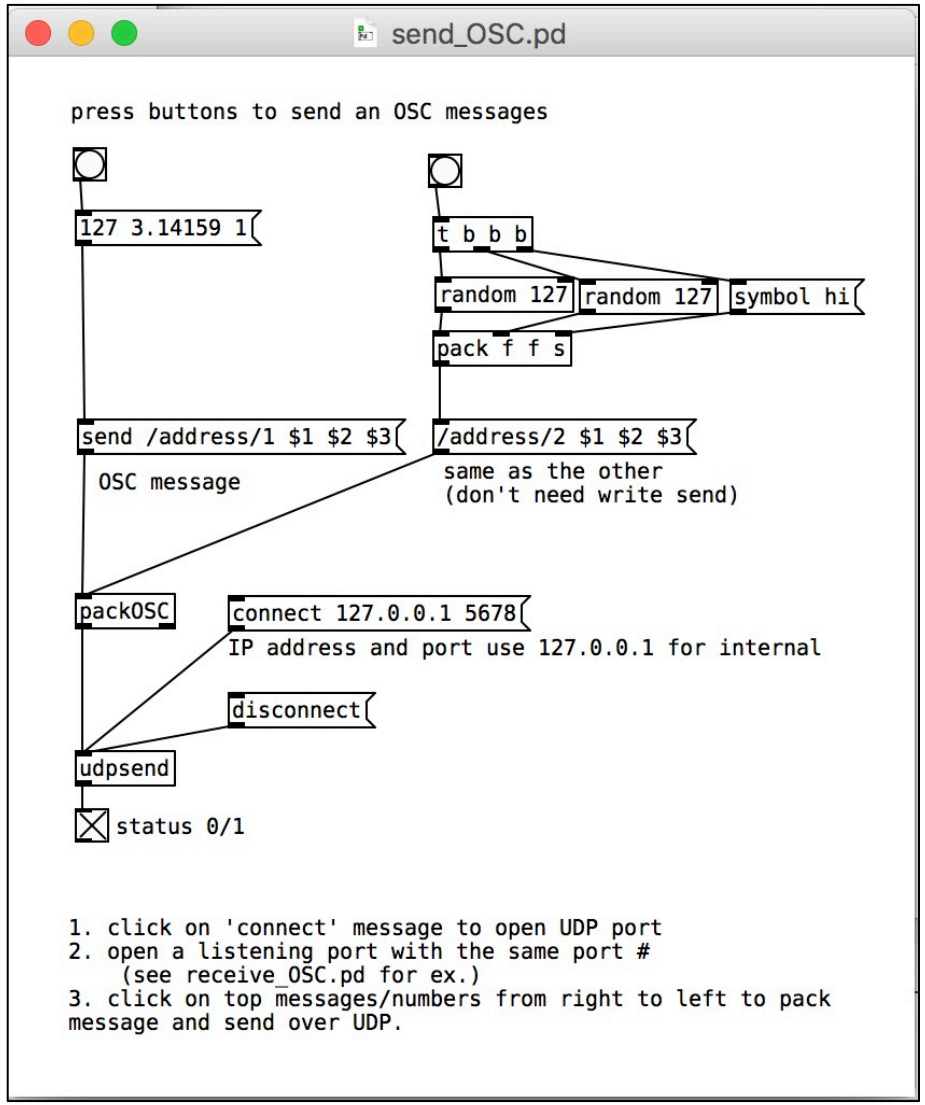
**will send as 2 strings*

/player1/buttons/1 1	/player3/buttons/1 0
/player1/buttons/2 0	/player3/buttons/2 0
/player1/buttons/3 0	/player3/buttons/3 1
/player1/knobs/1 255	/player3/knobs/1 100
/player1/knobs/2 127	/player3/knobs/2 0
/player1/knobs/3 0	/player3/knobs/3 0
/player2/buttons/1 0	/player4/buttons/1 1
/player2/buttons/2 1	/player4/buttons/2 1
/player2/buttons/3 0	/player4/buttons/3 1
/player2/knobs/1 11	/player4/knobs/1 255
/player2/knobs/2 255	/player4/knobs/2 255
/player2/knobs/3 200	/player4/knobs/3 255

etc...

OSC in Pure Data

- In Pd, we are using the ‘mrpeach’ external package, which provides a suite of OSC objects.
- Sending over UDP:
  - [packOSC] to format data
  - [udpsend] with:
    - Recipient IP address (can use 127.0.0.1 for internal routing)
    - Recipient port # (between 0 and 65535)
    - https://www.lifewire.com/popular-tcp-and-udp-port-numbers-817985

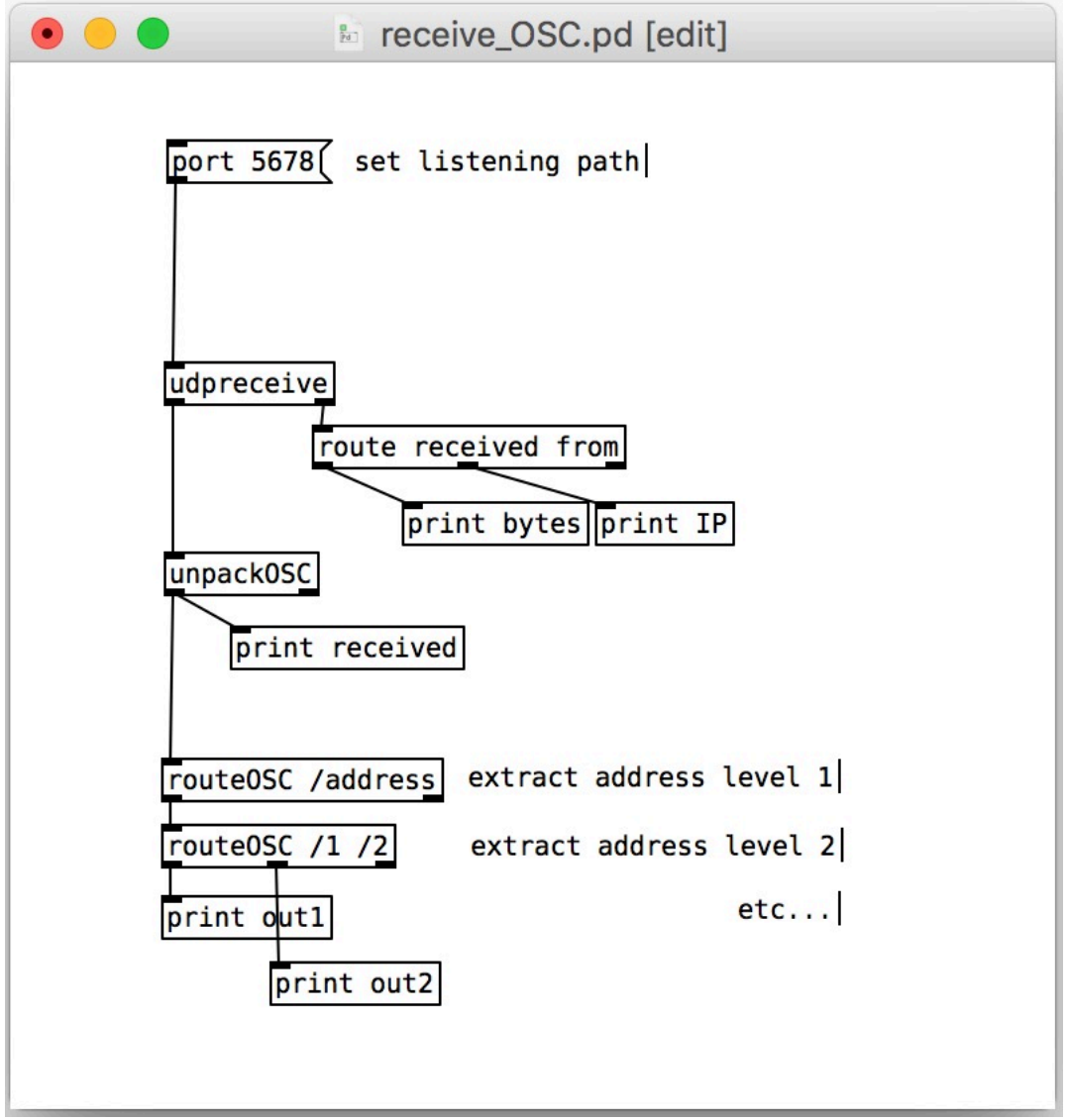


Ex: lecture-3/send_OSC.pd

OSC in Pure Data

- Receiving:
  - [udpreceive] with:
    - listening port #
    - (optional) IP address
  - [unpackOSC] to get data
  - [routeOSC] to parse data

- Optional:
  - You can stream motion data from a smartphone with OSC.
  - Free apps:
    - Android: hookOSC
    - iOS: Mrmr
    - or TouchOSC (Android/iOS but not free)



Ex: lecture-3/receive_OSC.pd

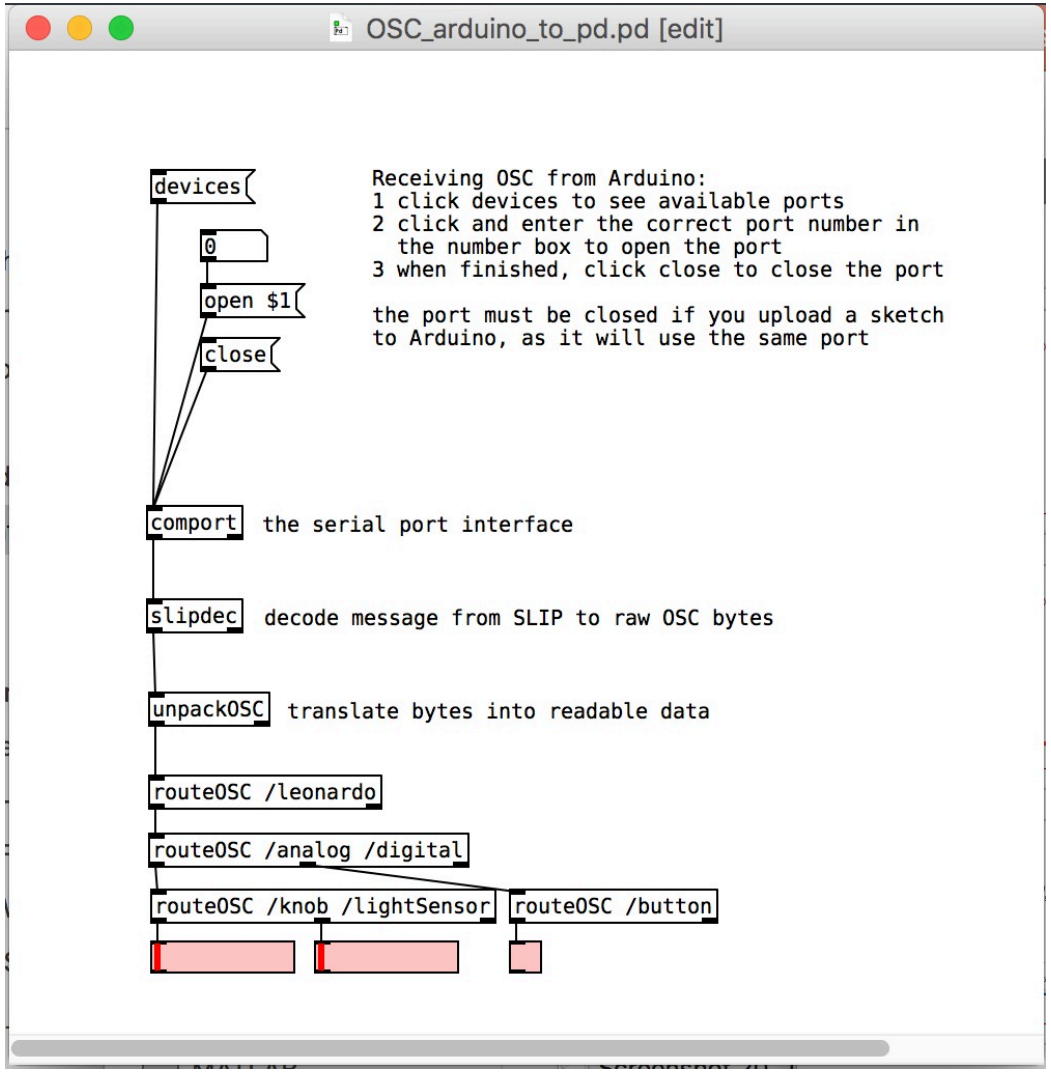
OSC with Arduino (via serial)

- We can read and write OSC messages directly on the Arduino as well.
  - To transmit to a connected computer via USB, it is easiest to use the serial port.
  - Arduino:
    - Install the OSC external library
      - Uses SLIP encoding (Serial Line Internet Protocol) to format the OSC messages for serial transmission (included in OSC lib)
    - Include “OSCmessage.h” and “SLIPEncodedSerialUSB.h” in your sketch
      - See examples:
        - lecture-3/Arduino/OSC_arduino_to_pd
        - lecture-3/Arduino/OSC_pd_to_arduino
  - Pure Data:
    - Use [comport] object to send/receive from serial port
    - Arduino -> Pd: use [slipdec] -> [unpackOSC] from mrpeach to decode data
    - Pd -> Arduino: use [packOSC] -> [slipenc] to encode data

OSC Arduino to Pd

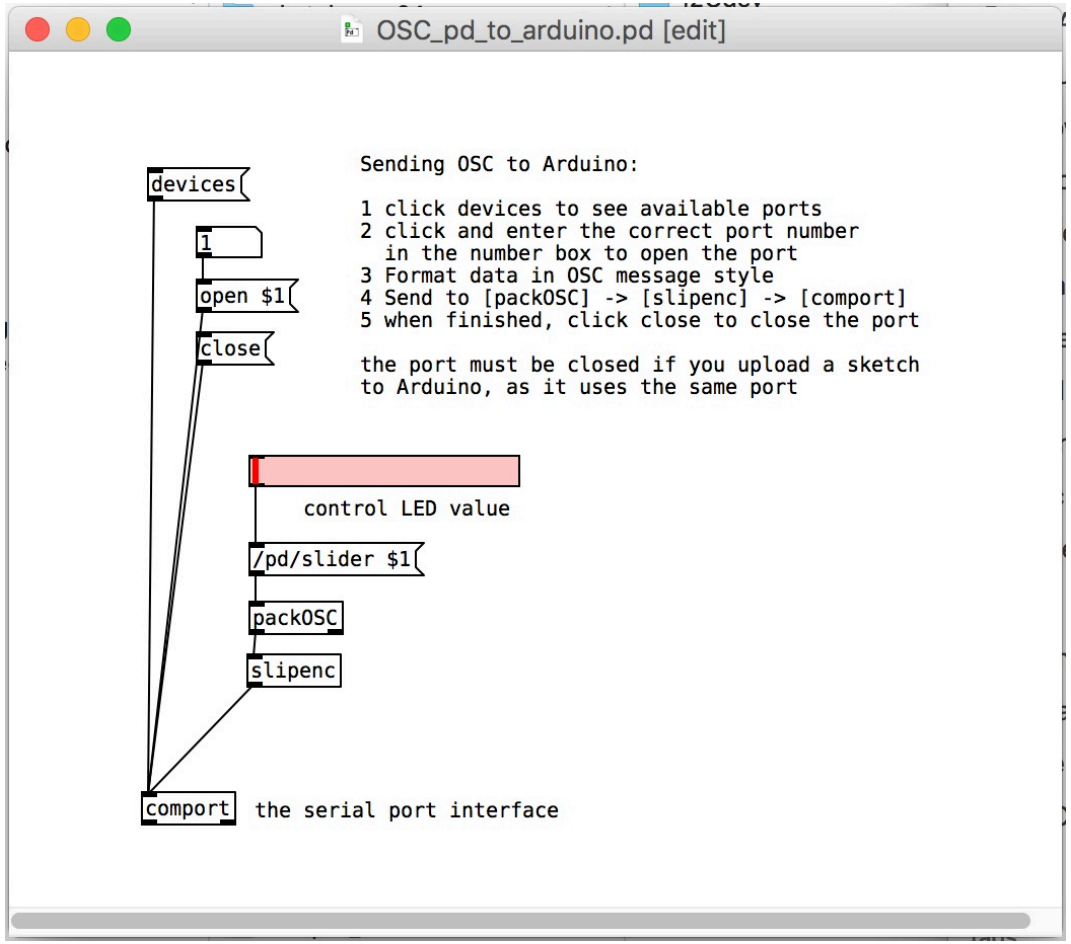

```
void setup() {
  //begin SLIPSerial just like Serial
  SLIPSerial.begin(9600);   // set this as high as you can reliably run on your p
#if ARDUINO >= 100
  while(!Serial)
    ; //Leonardo "feature"
#endif
}

void readSensors() {
  // gets sensor values and normalizes the analog values between 0 and 1
  knobVal = analogRead(KNOB);
  knobVal = knobVal/1024;
  lightVal = analogRead(LIGHT);
  lightVal = lightVal/1024;
  buttonVal = digitalRead(BUTTON);
}

void loop(){
  // read the sensors
  readSensors();

  // *** Potentiometer ***
  float diff = knobVal - lastKnobVal; // difference between current and last sent
  if (abs(diff) > threshold) {

    //the message wants an OSC address as first argument
    OSCMessage msg("/leonardo/analog/knob");
    msg.add(knobVal); // add data as additional argument(s)
    // can add more arguments using the msg.add() method

    // then send the OSC message
    SLIPSerial.beginPacket();
    msg.send(SLIPSerial); // send the bytes to the SLIP stream
    SLIPSerial.endPacket(); // mark the end of the OSC Packet
```

Ex: lecture-3/Arduino/OSC_arduino_to_pd



Ex: lecture-3/Pure Data/OSC_arduino_to_pd.pd

OSC Pd to Arduino



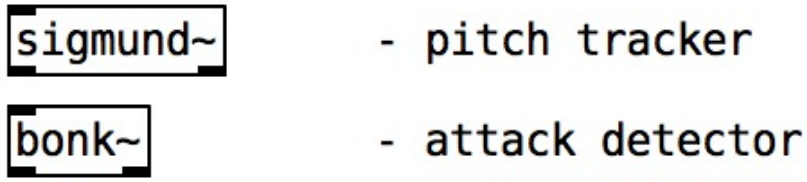
Ex: lecture-3/Pure Data/OSC_pd_to_Arduino.pd

Ex: lecture-3/Arduino/OSC_pd_to_arduino

Indirect Gesture Acquisition

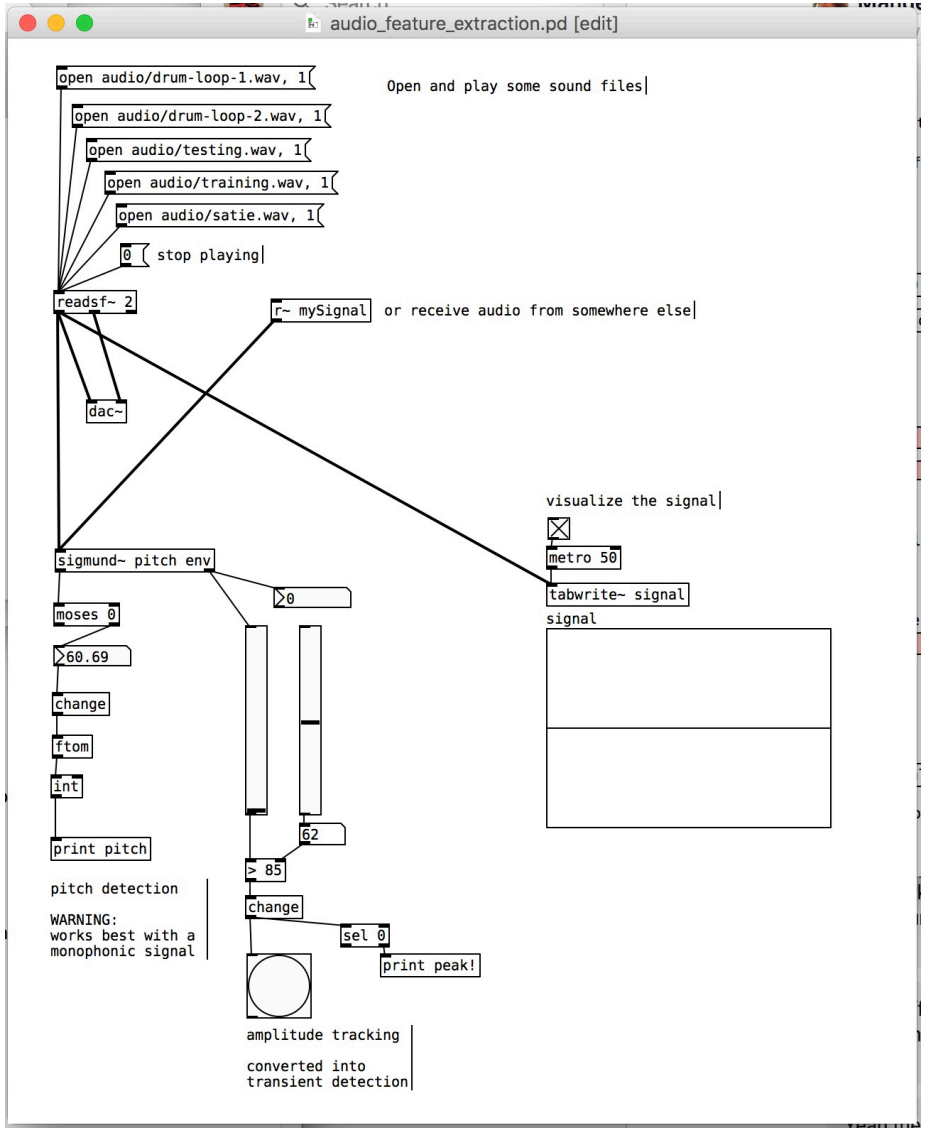
- Gesture can be captured in at least 3 different ways:
  - direct acquisition (using sensors to measure the physical actions of the performer)
  - indirect acquisition (analyze the structural properties of sound being produced by the instrument/performer)
  - physiological acquisition (biosignals – brain (EEG), neuromuscular (EMG), skin conductance (GSR), etc.)

- Sophisticated audio feature extraction requires a solid knowledge of using the Fourier Transform and FFT analysis techniques which are beyond the scope of this course.
- However, Pure Data provides 2 objects that let us get some basic information from an audio signal.



sigmund~

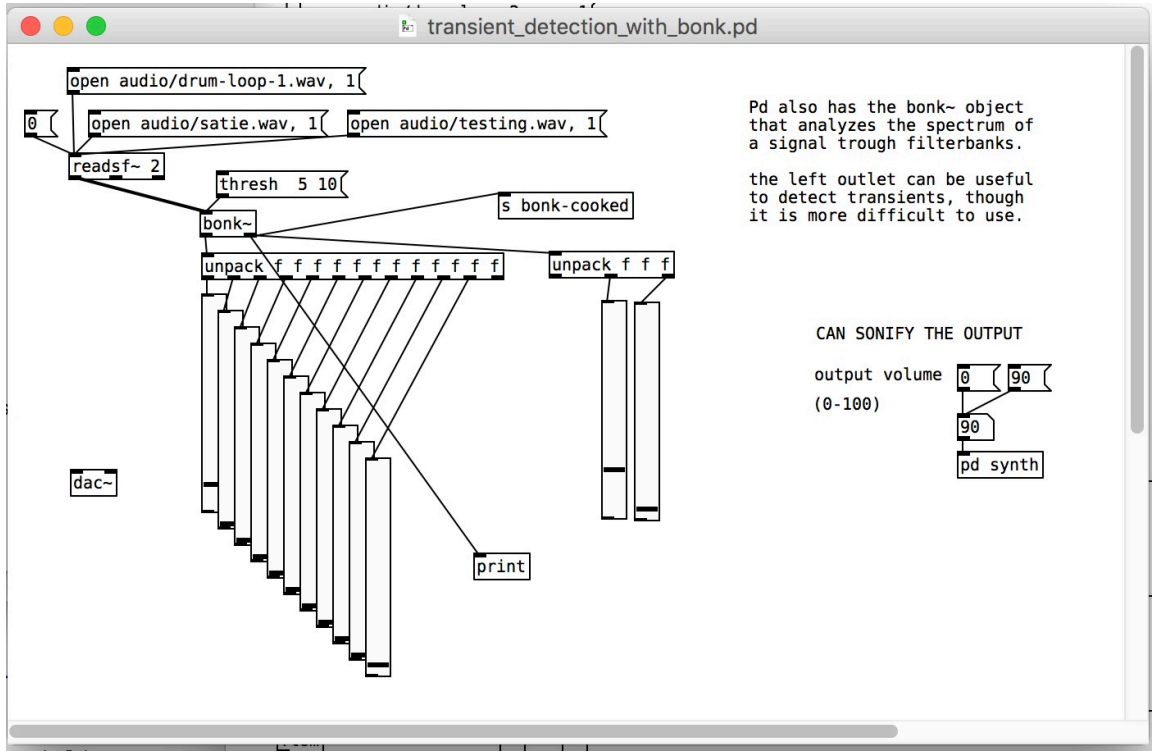
- in Pd, we can read a sound file or take an audio stream and analyze it for pitch and amplitude using [sigmund~].
- Pitch tracking will be most accurate with a monophonic signal
- We can set an amplitude threshold and detect transients as well, or follow general amplitude contours.
- These could be used as control variables so that live instrumental performance can in turn control other musical parameters, connected to other software or hardware using MIDI or OSC messages.
- This is an example of ‘indirect gesture acquisition’.



Ex: lecture-3/Pure Data/audio_feature_extraction.pd

bonk~

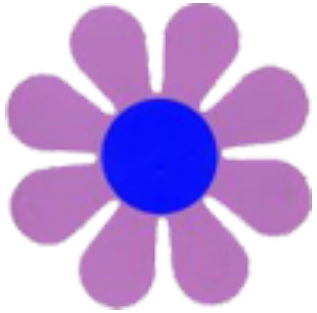
- Pd also provides the [bonk~] object for attack detection and spectral analysis.
- By playing with the high and low thresholds, you can get good results for attack detection.



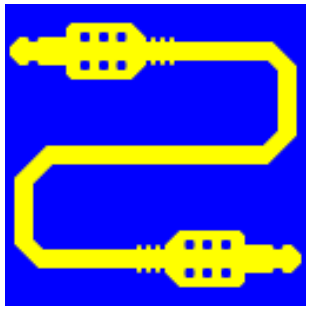
Ex: lecture-3/Pure Data/transient_detection_with_bonk.pd

Inter-application audio routing

- A typical usage of this method of control will be to analyze data from another application (a DAW, DJ software, Spotify, etc.)
- To achieve this you need a utility application that allows audio to be flexibly routed internally between applications.
- Recommended:
  - MacOS: SoundFlower (free)
  - Windows: Virtual Audio Cable (free trial)
    - There are other Windows utilities that are free, but will require more configuration and experimentation. Try a Google search for windows route audio between applications.
- Once installed you should see input and output devices in your computers audio settings.
  - Example: route audio from Ableton Live to Pure Data on a Mac:
    - In Live Preferences, select Soundflower (2ch) as output device
    - In Pure Data, select Soundflower (2ch) as input device
    - In Pd, use the [adc~] object to get the audio device's signal into your patch.



Soundflower(Mac)
https://rogueamoeba.com/freebies/soundflower/



Virtual Audio Cable (Windows)
https://vac.muzychenko.net/en/index.htm