CLASS II - OSC

Open Sound Control flexibility and control

OSC

While working with Serial you may have noticed that it is a very rigid communication protocol. Meaning that it can only communicate with one device or program at the same time.

If you want to share data between programs while using Serial you would manually activate or deactivate connections. Which is something we don’t want in our ecosystem, at least not when it isn't our own choice.

OSC

Incoming OSC, Open Sound Control, a communication protocol originally designed at CNMAT department at University of Berkely. It was a way of sending performance data like gestures, parameters or note sequences in between computers, synthesizers and other media devices. OSC is sometimes used as alternative for MIDI when a higher resolution or richer parameter space is required.

OSC

OSC uses a very specific format for its messages which includes addresses. These addresses are extremely useful because you can collect parameters for different devices, format them as one package and parse the messages according to their OSC addresses and route them to their relevant devices.

Because OSC is end to end and network based the format of text stays within the ASCII or human readable domain so it saves us the hassle of conversion which we had to do for Serial.

OSC ADDRESS

The addresses for OSC as mentioned before is very specific and can look like this:

```
/oscillator/4/frequency
```

Each dash is the indicator of a subsection within the address. For example you are working with multiple oscillators within your device you can route them as follows:

/oscillator/4 <message>
/oscillator/5 <message>

For each oscillator you can declare subsections that are device dependable.

OSC ADDRESS

`/oscillator/4/frequency`

Did you notice the slash symbol on the start of the address? That is a convention that has to be uphold otherwise OSC will get confused.

Touchdesigner prepends the starting slash when creating OSC messages, so you do not have to include this.

Sending a message from Touchdesigner would look something like this:

Oscillator/4/filter

Where the first slash is added by the OSC protocol on default.

OSC NETWORK

- OSC sends its messages over network, this can either be a local network, meaning everything stays on the same machine, or a forked network where multiple clients, computers, are connected to the network. This can either be wireless like WiFi or wired network through a network switch or router.
- To find the appropriate machine, OSC uses two arguments which is the ip-address and the network port.

DIFFERENCES AND OVERSIGHT OF WIRED AND WIRELESS NETWORK

- Wired network:
- Stable
- Fast, depending on network switch but easy 1 gb per second
- Static ip-addresses
- Physical wires
- Hardware depedent

- Wireless network:
- Stable as the weakest acces point in the network
- Can be slower due to shared bandwidth with other devices connected to the acces point.
- Dynamic ip-addresses
- No wires or cables
- Can be achieved with hotspot on smartphone

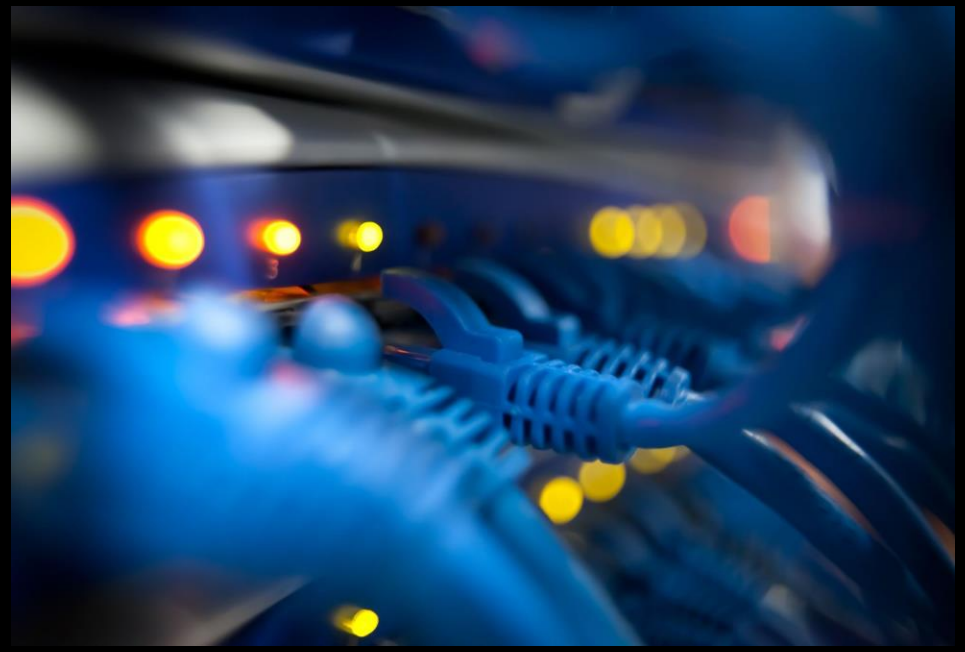
SWITCH

ACCESS POINT/WIFI

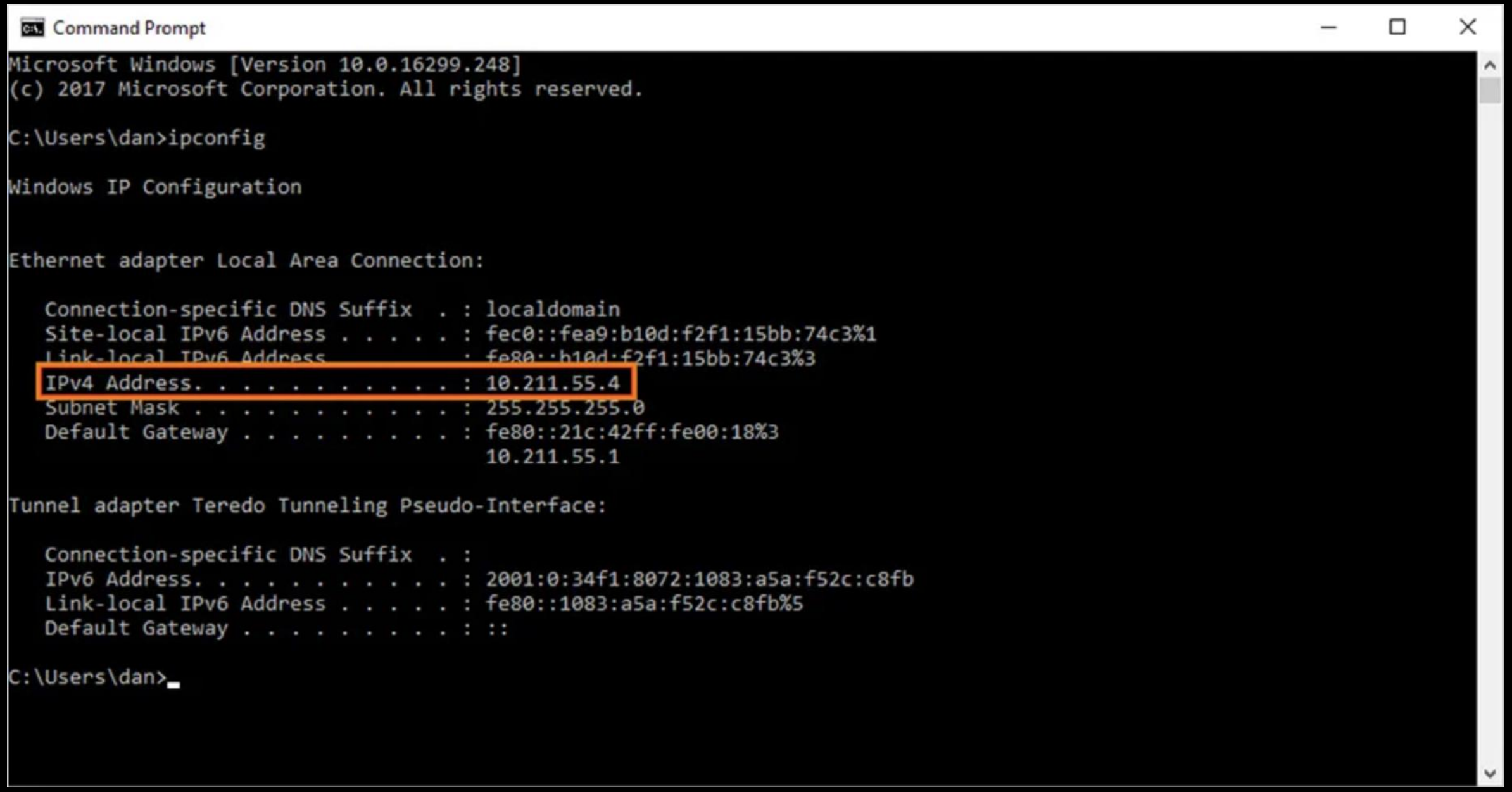
IP-ADDRESS AND NETWORK PORT

IP-ADDRESS:

- The ip-address is the address for the computer assigned by the network switch/ router. It is unique for each computer but can change when disconnected and reconnected.
- To find your ip-address on Windows you can use the command: ip-config in the command line. On macOS or Unix you can use ifconfig in the terminal.



WINDOWS COMMAND PROMPT



MACOS / UNIX

```
LinuxConfig.org:~$ ifconfig
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.122.222  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:35:a2:cc  txqueuelen 1000  (Ethernet)
        RX packets 16648  bytes 92190850 (92.1 MB)
        RX errors 0  dropped 1338  overruns 0  frame 0
        TX packets 9045  bytes 827750 (827.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 212  bytes 20344 (20.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 212  bytes 20344 (20.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

LinuxConfig.org:~$
```

IP-ADDRESS

IP-ADDRESS:

Usually the ip-address format looks something like 192.xxx.xx.xx , however the numbers can and will change depending on the kind of network you are connected to. Eduroam for example looks like: 10.xx.xx.xx This is for networking reasons we're not going to discuss.

Note that Eduroam is not well suited for sending OSC due to the vast amount of safeguards built into the network.

LOCALHOST

So what about when all the OSC-messages stay on the same machine? Than we use a term which is called: localhost, meaning: this computer.

A lot of objects that initialize OSC accept the argument localhost however sometimes it doesn't work. In that case you have to enter the ip-address of the localhost which is:

127.0.0.1
Using this will always work.

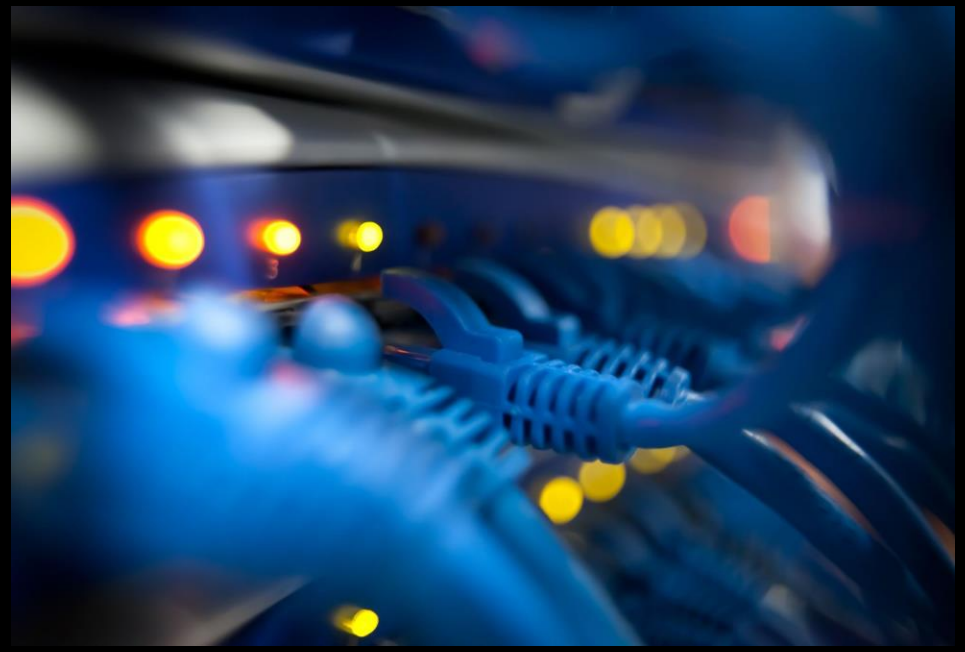
NETWORK PORT

HOUSENUMBERS:

Let's compare networking to a hotel. Our computer is a hotel and when placed within a network our hotel lives on a street with multiple hotels built along it.

The ip-address states to what hotel the message has to be delivered to. But the hotels exists out of a huge collection of rooms and the message has to be delivered to the right room number.

The network port is virtually the same as the room number. So with these two arguments the network switch or router understands to what hotel and what room the message has to be delivered to.



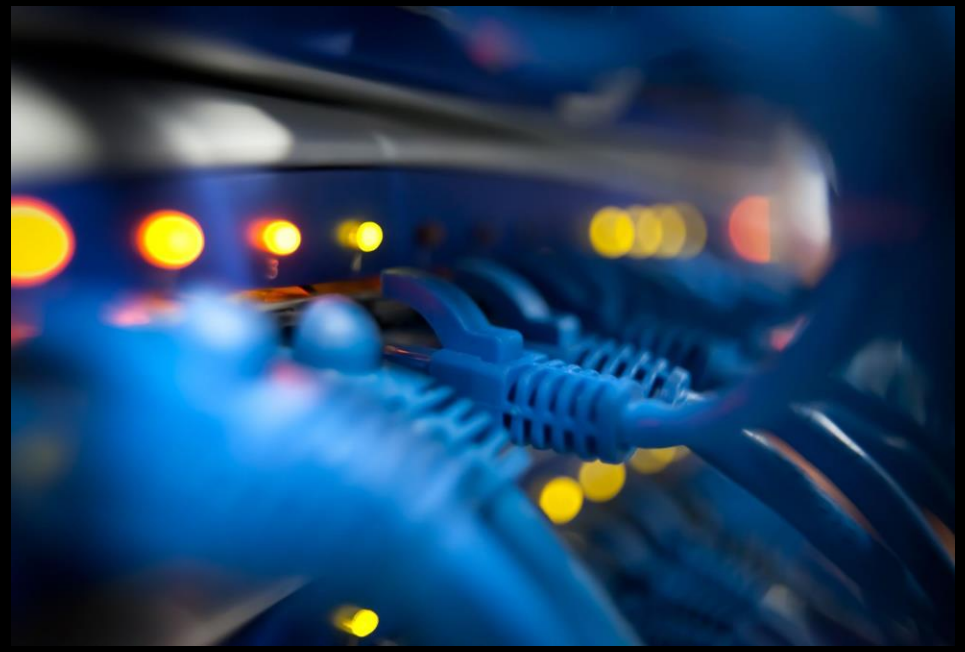
NETWORK PORT

PORTNUMBERS:

In the early days of computers the network port referenced a physical port within the computer. Nowadays the port is virtual and represents an endpoint within a network, the room number.

However, certain port numbers are occupied by processes that are always in use by your computer and they tend to be 'lower' value numbers like: 21, 44, 88, 10 and so on.

Often used port numbers for OSC are: 8888, 7777, 9999 or any inbetween. Stick to the higher side of the numbers and the port should be free to listen to your OSC messages.



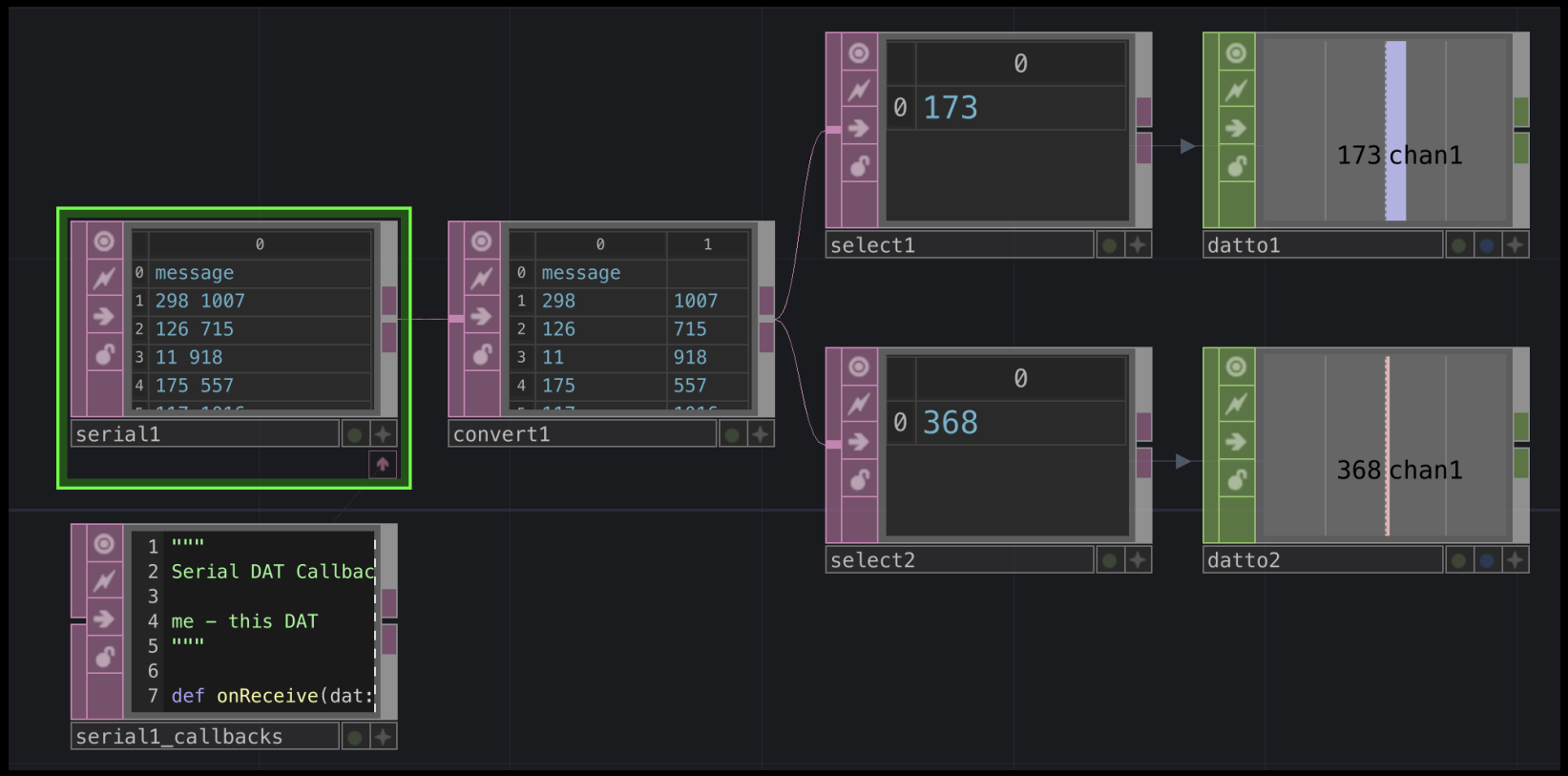
START OF THE ECOSYSTEM

- Why are we going to utilize OSC?
- It will be the beginning of our ecosystem. We can start importing data from serial into Touchdesigner and use OSC to relay the data to M4L to control instruments, filters or other parameters. Also it will allow us to send parameters from Touchdesigner to M4L so we can have RGBA-pixel information control parameters.
- Due to the flexibility of OSC we can send messages back and forth. So not only will Touchdesigner be able to control parameters in M4L but the other way also. This way parameters from instruments, filters or sound analysis can direct control visual parameters and goes beyond basic audio reactivity.

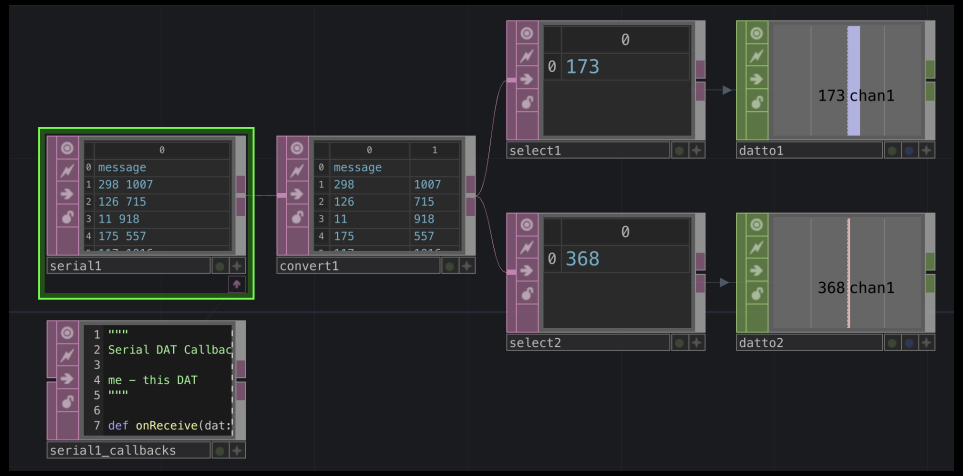
LET'S DIVE INTO IT HEADFIRST!

We'll start on the Touchdesigner side and steadily work our way into M4L. We start with Touchdesigner because it will act as the middleman or the spider in the web.

SERIAL TO TOUCHDESIGNER

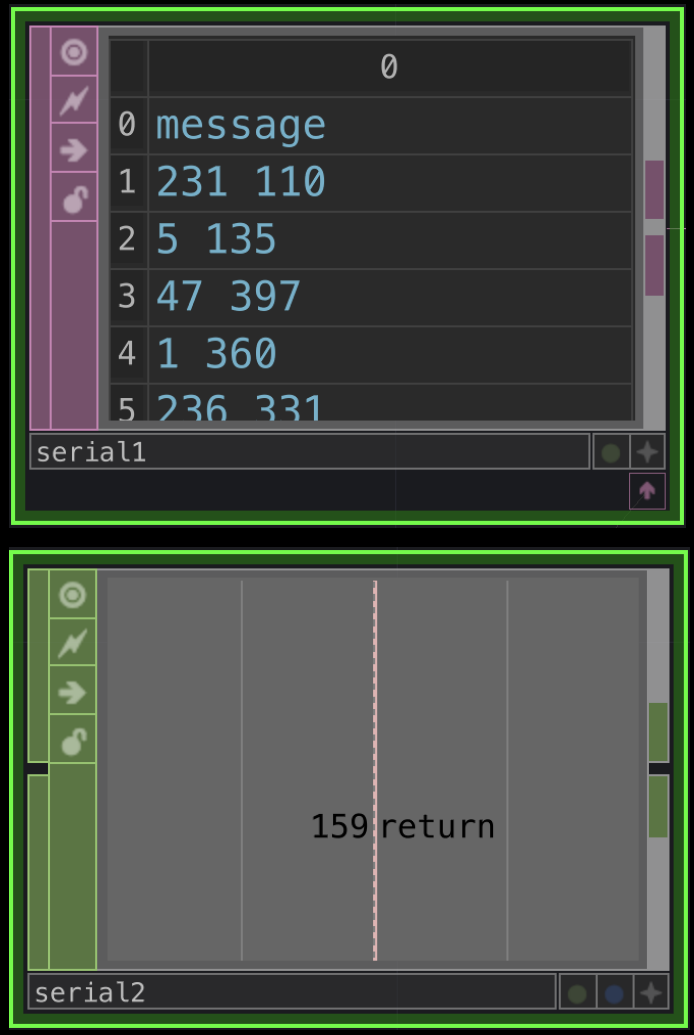


SERIAL TO TOUCHDESIGNER



- The workflow in Touchdesigner is as follows:
- Serial DAT opens serial connection -> a convert is used to split the data into two collumns -> select allows you to select a row and collumn -> DAT to CHOP allows you to convert strings into integers.

WHAT!? NO ASCII CONVERSION HASSLE?



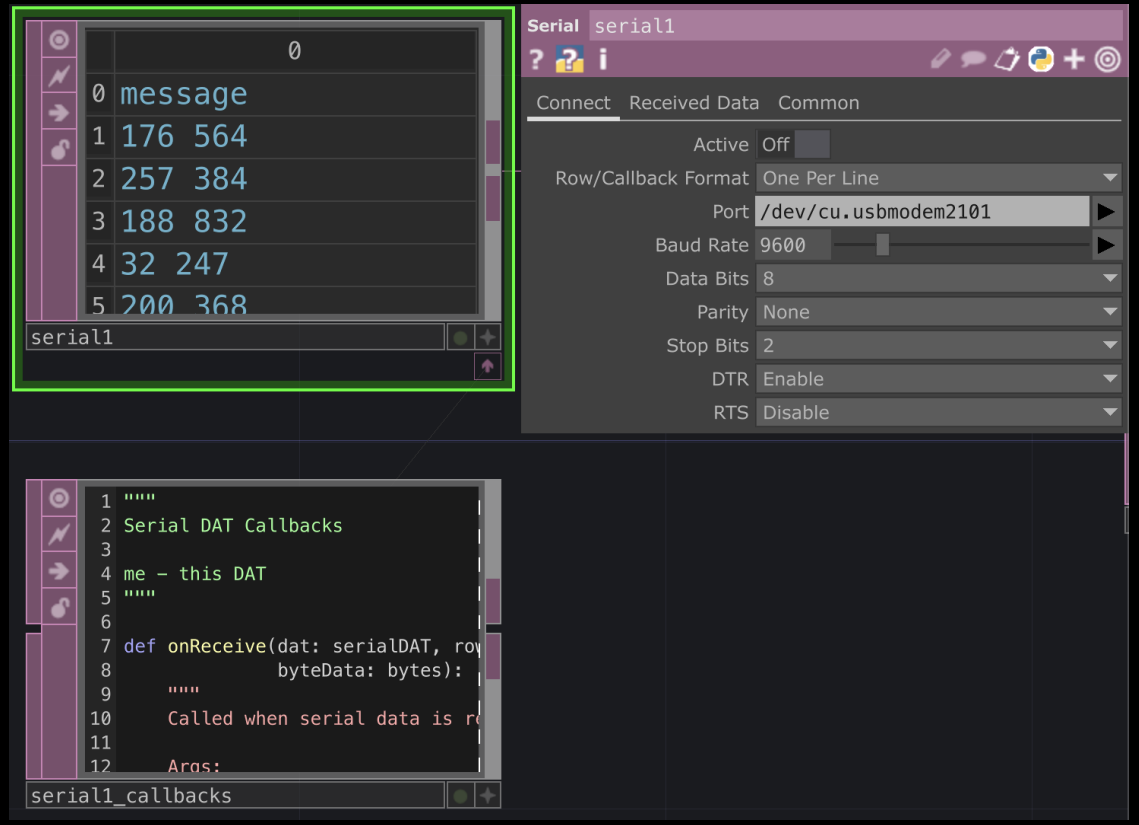
- In this case we work with DAT's because they are already string-based. See how the values are different inbetween Serial CHOP and Serial DAT? That is because CHOP is integer based and we would have to convert numbers back into ASCII again, but thankfully DAT is available to us and we only have to convert strings into integers.

SERIAL DAT

On the parameter window of Serial DAT you can declare the port and the baud rate. Both should be familiar.

Baud rate: the speed of communication; this is declared on Arduino side but usually 9600.

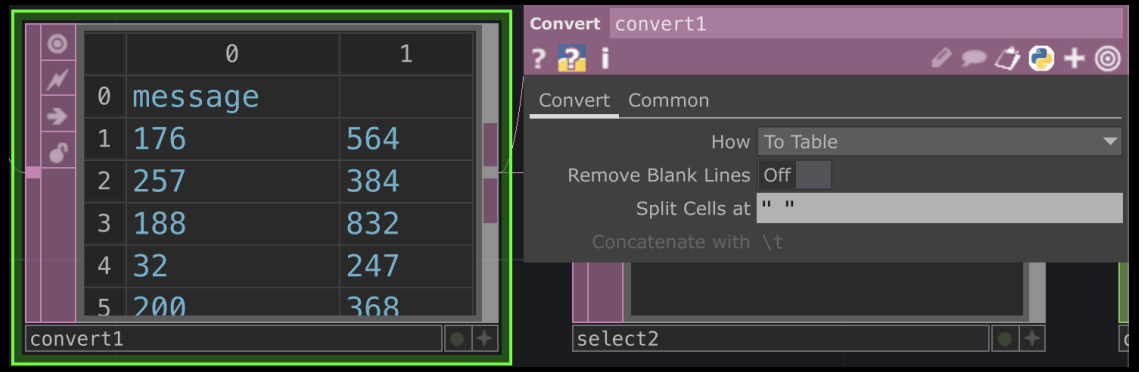
The port references the physical address of the connected peripheral, in this case the Arduino, usually it is listed as an usbmodem.



CONVERT DAT

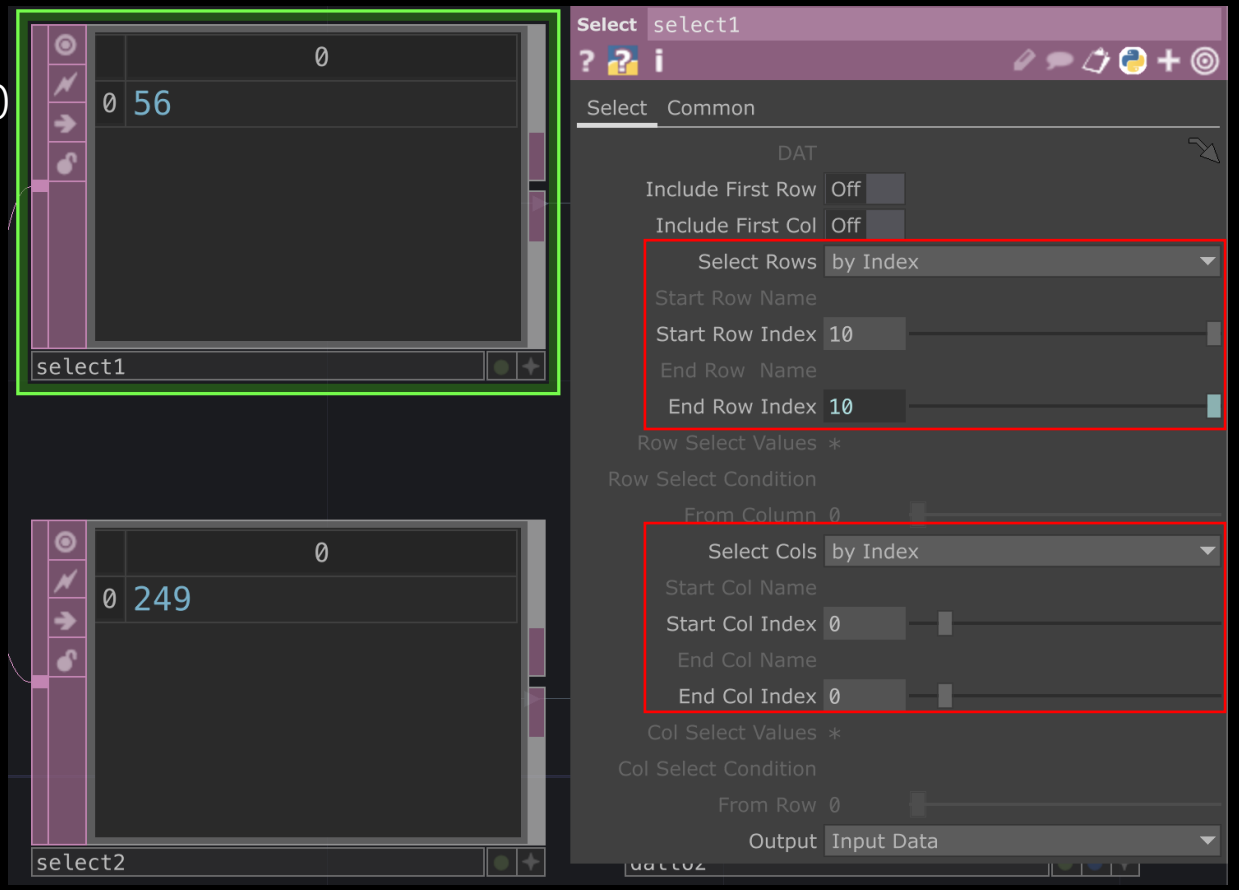
The convert DAT is there to change the lines of strings into two collumns. See at the parameter window how there is a " " in the 'Split cells at' fieldbox?

The " " is telling the convert DAT to split cell at a space within the string. This refers to how we've setup the printing of data on the Arduino side.



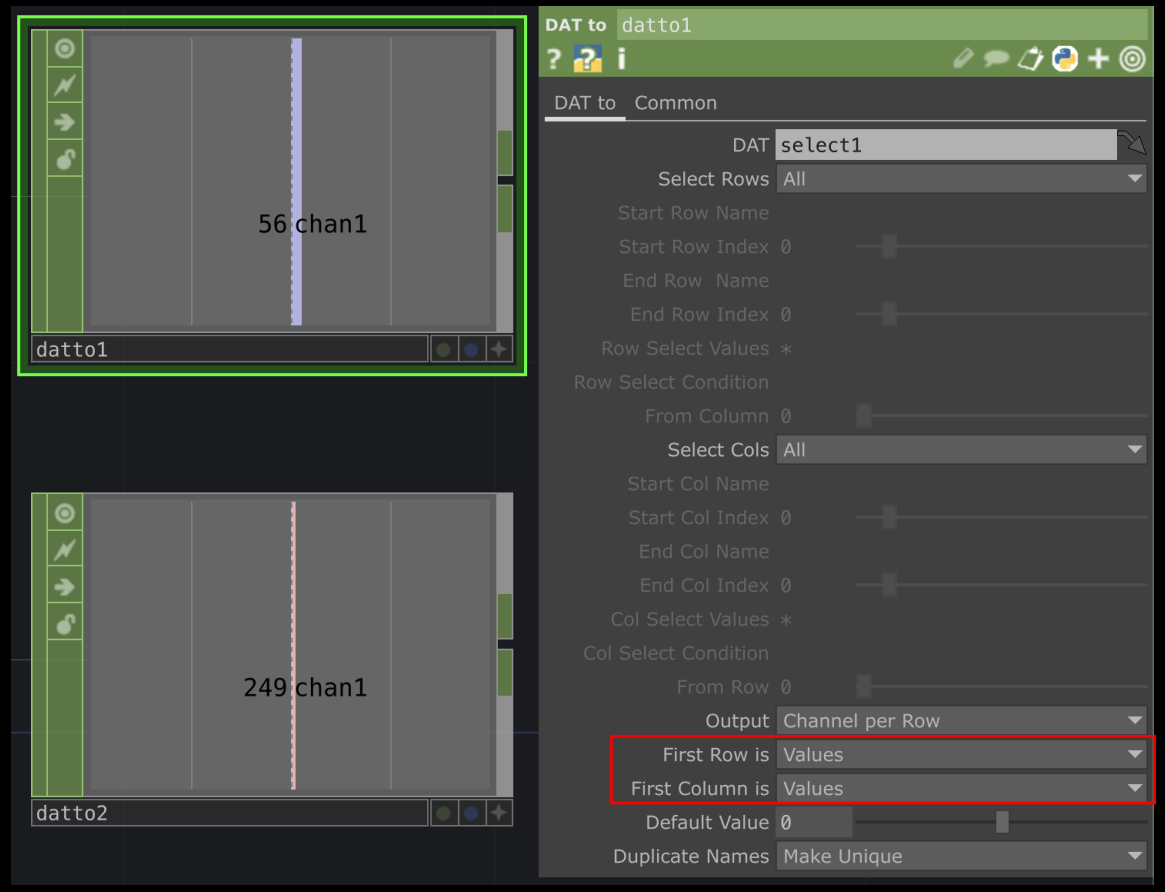
SELECT DAT

- For the columns we want to select index 0 on select1 and index 1 at select2. The order is arbitrary and it speaks for itself that each select should hold a different column in order to split the data sufficiently.
- Both select DATs now have index 0,0 which is due to auto-updating of the cells whenever others are negated.



DAT TO CHOP

- Finally we have to convert the string information into integer or number based data again. We do this with a DAT to CHOP operator.
- Take a note at the 'First row is' and 'First column is', they should both be set to 'Values', otherwise the name of the channel will be the values you want to access.

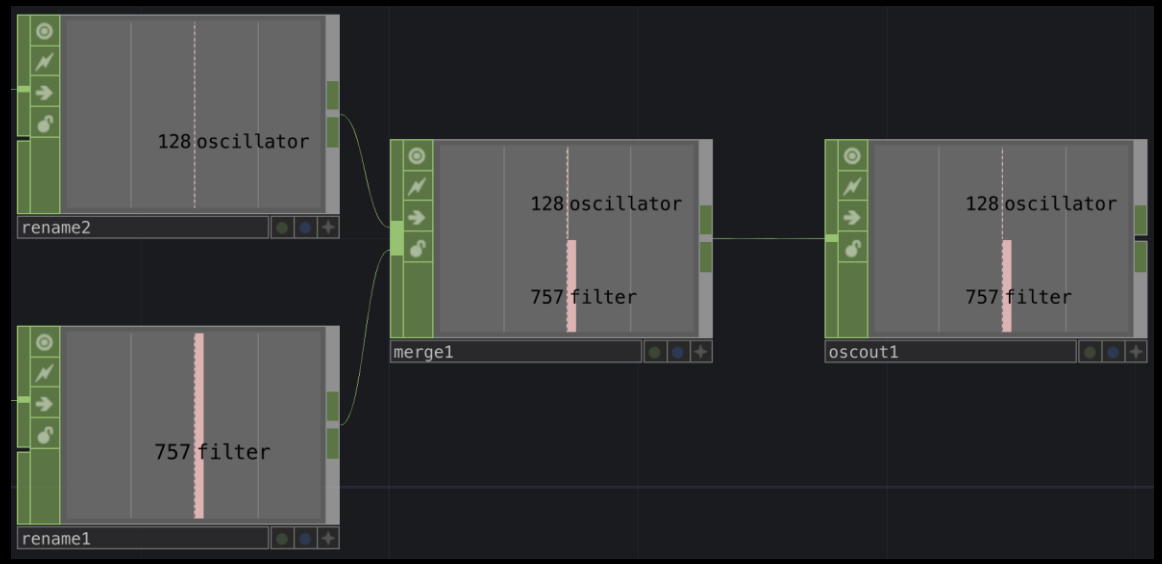


NICE, FIRST SECTION DONE!

- You could do a number of things to process the data now. But since that is very project dependent we're not going to zoom into that.
- Instead, we're skipping right ahead into formatting and sending it as OSC to M4L. For this we're going to adjust our first device. You may want to resave your device so both will stay intact.

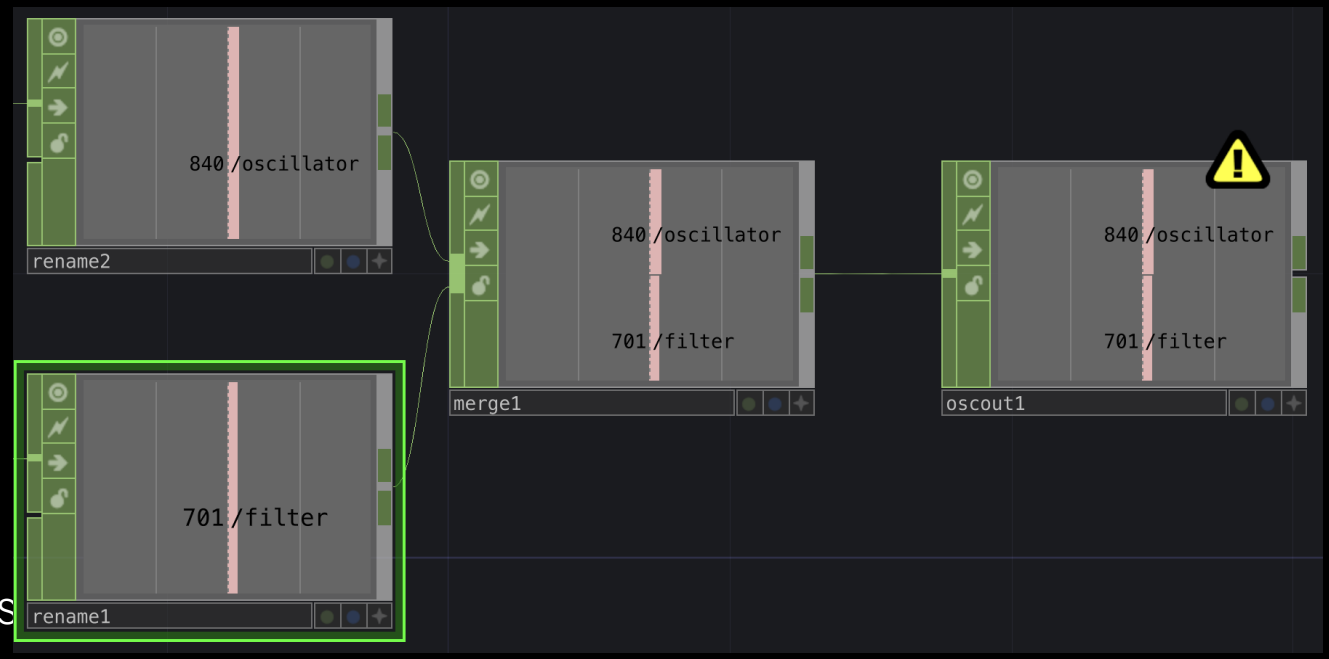
FORMAT FOR OSC

- Because OSC allows us to do pattern matching by name we want to rename our channels into something more meaningful than chanX. These names are arbitrary yet good practice to do so.
- After renaming we want to merge the data together so they can be sent as a package to one network port, efficiency!



FORMAT FOR OSC

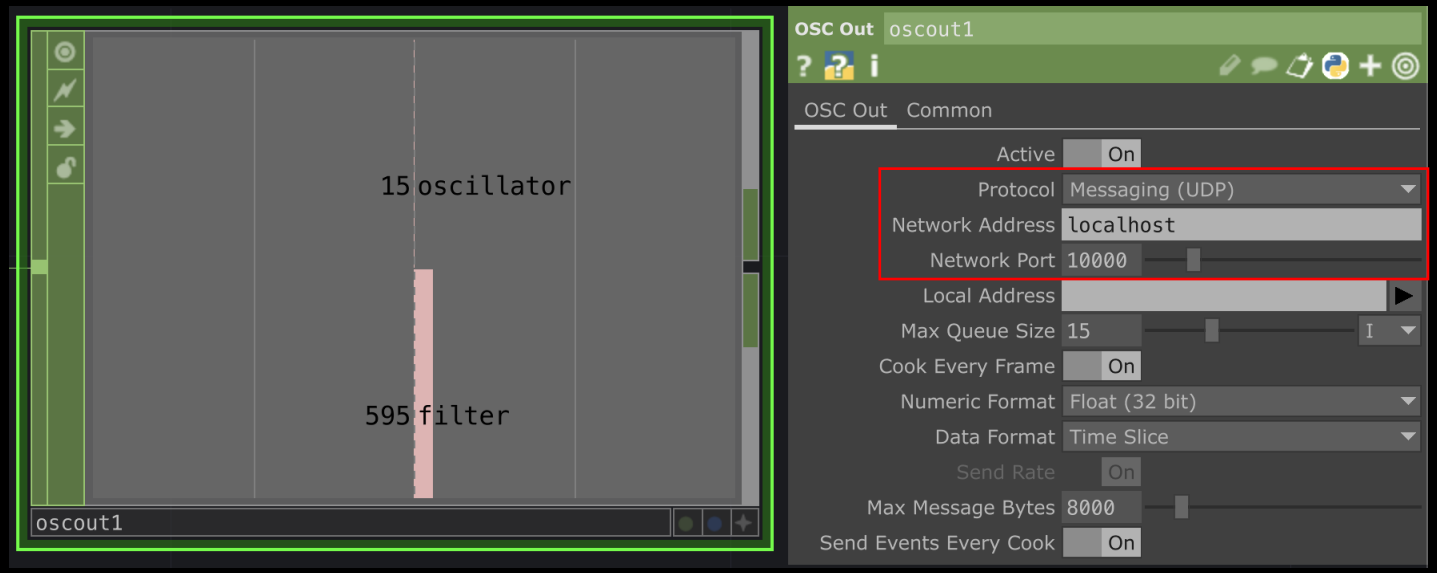
- Earlier I spoke about the address formatting for OSC. /device/4/filter for example. If we include the '/' in the rename OSCOUT1 gives us a warning, which is stated below.
- Touchdesigner prepends the '/' for us so we do not have to do this manualy. However if you want to make subsections than you do.




```
Warning: Channel name starting with / detected. The required starting / for OSC address name is always prepended to channel names, manually specifying it is not required.
(/project1/oscout1)
```

OSC OUT CHOP

- The parameter window of OSC OUT CHOP gives us many options but actually only two are important for now and a third is more of a clue.
- The network address would be the ip-address, for now it may remain localhost or 127.0.0.1 but when you send to a different machine this is where you want to put in the ip-address of that machine.
- The network port is the roomnumber where the data is sent to. So on the receiving end you want to listen in on this specific port.
- The protocol is set to UDP by default, this is nice to know because on M4L the object is called [udpreceive] to access OSC communication.



ALRIGHT, FOR NOW WE'RE DONE ON TOUCHDESIGNER AND MOVING ON TO M4L.

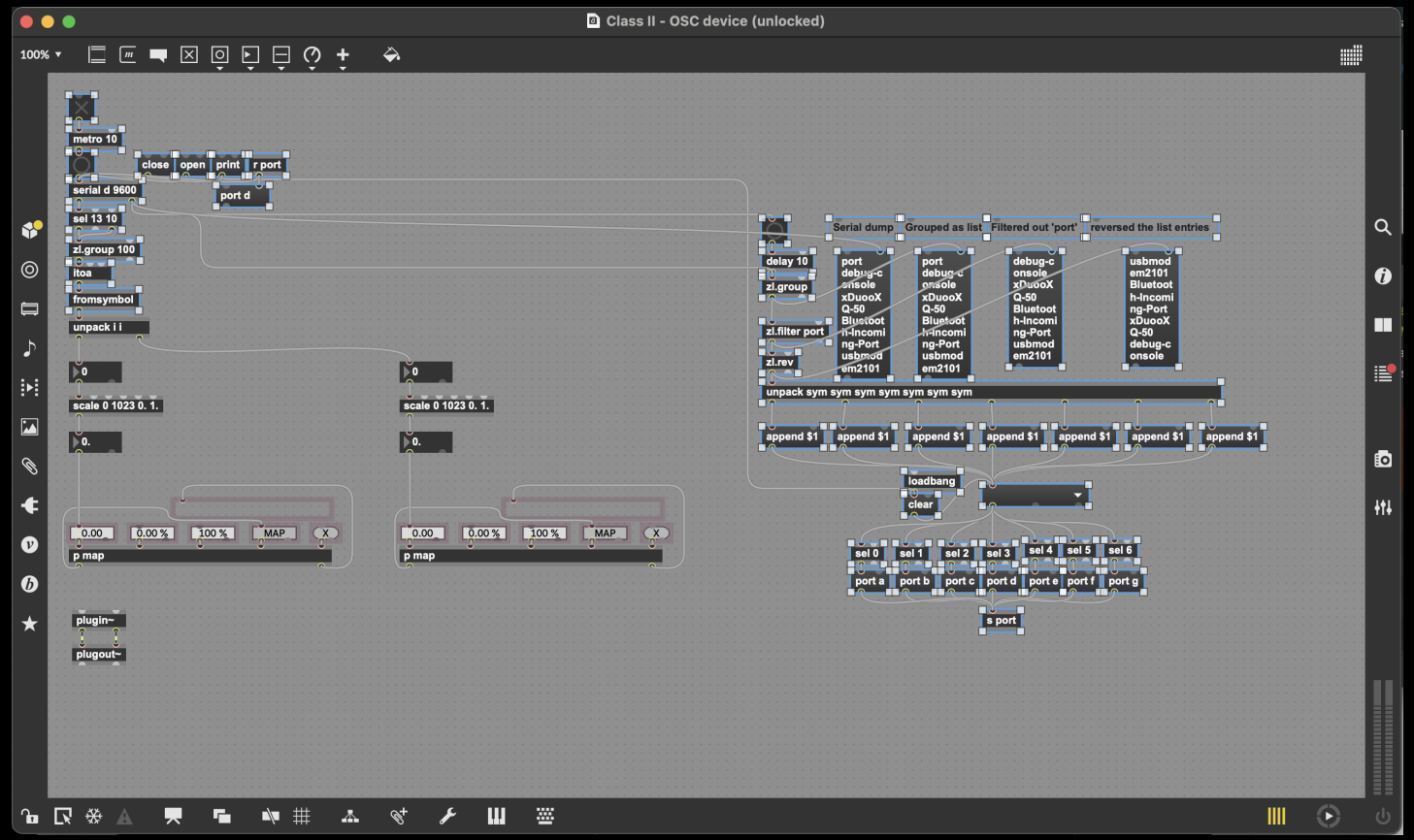
We will gut the M4L device we made last week so that it doesn't listen to seria anymore but to OSC. As said before, open up the device and resave it as a new device so both will remain intact.

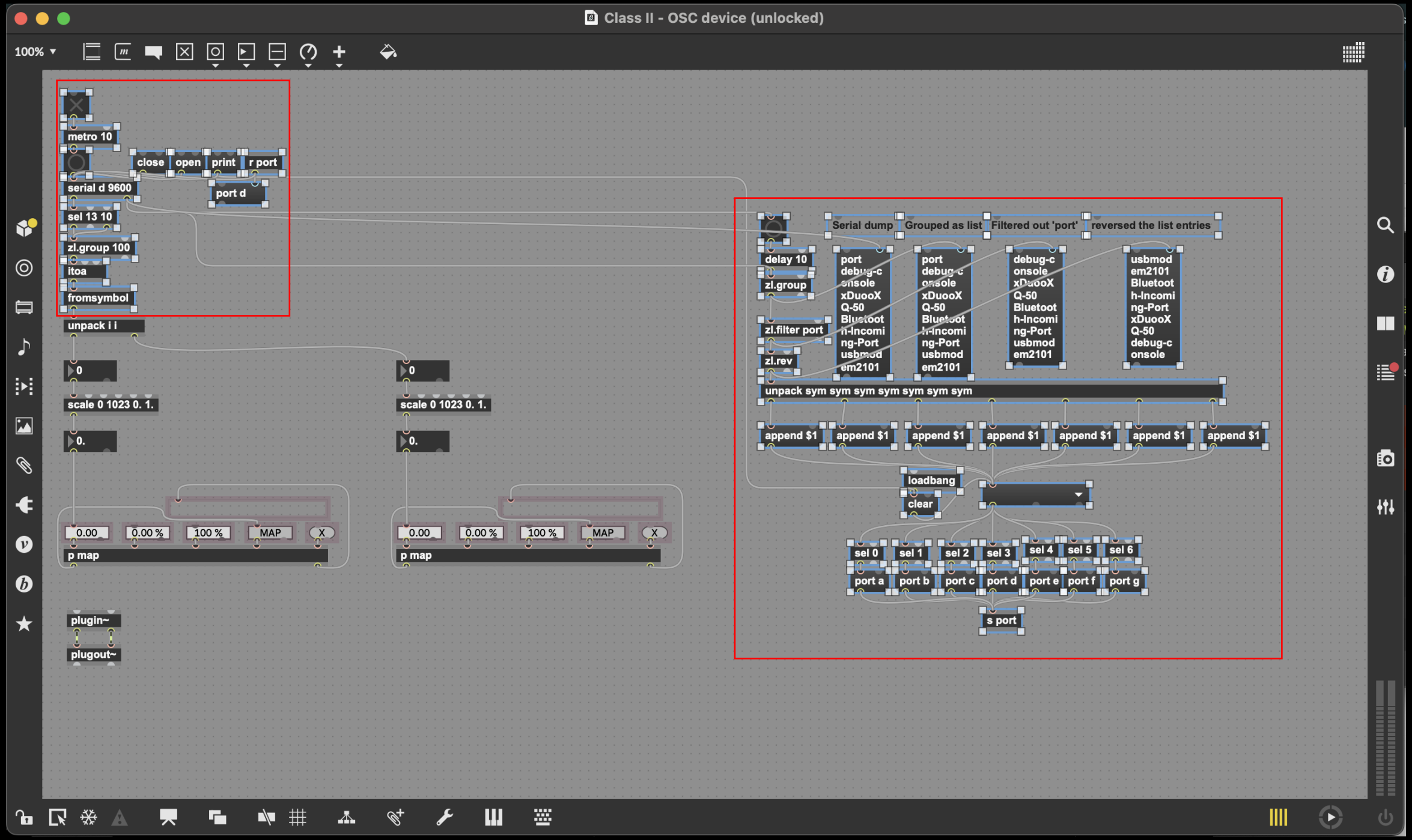
KILL OUR CHILD

So we can almost completey strip our device, everything related to Serial can be deleted which would be the:

- Serial connection and ASCII conversion
- The dynamic menu portion

We can leave in the mapping section because we want to reuse it anyway.




Class II - OSC device (unlocked)
100%
metro 10
close
open
print
r port
serial d 9600
port d
sel 13 10
zl.group 100
itoa
fromsymbol
unpack i i
0
scale 0 1023 0. 1.
0.
0.00
0.00 %
100 %
MAP
X
p map
0
scale 0 1023 0. 1.
0.
0.00
0.00 %
100 %
MAP
X
p map
plugin~
plugout~
Serial dump
Grouped as list
Filtered out 'port'
reversed the list entries
delay 10
zl.group
zl.filter port
zl.rev
port debug-c onsole xDuooX Q-50 Bluetoot h-Incomi ng-Port usbmod em2101
port debug-c onsole xDuooX Q-50 Bluetoot h-Incomi ng-Port usbmod em2101
debug-c onsole xDuooX Q-50 Bluetoot h-Incomi ng-Port usbmod em2101
usbmod em2101 Bluetoot h-Incomi ng-Port xDuooX Q-50 debug-c onsole
unpack sym sym sym sym sym sym sym
append $1
append $1
append $1
append $1
append $1
append $1
append $1
loadbang
clear
sel 0
sel 1
sel 2
sel 3
sel 4
sel 5
sel 6
port a
port b
port c
port d
port e
port f
port g
s port

REMEMBER OSC IS DEVELOPED BY CNMAT? TO DO PROPER PATTERN PARSING WE NEED THEIR EXTERNAL PACKAGE

File Edit View Object Arrange Options Debug Window Extras Help

New Patcher ⌘ N
New From Template
New From Clipboard ⌥⌘ N

New Project
New Text ⇧⌘ N

Open... ⌘ O
Open As Text...
Open Recent Patcher
Open Recent Project
Close ⌘ W

Show File Browser ⌘ B
Show Package Manager

Modify Read-Only ⌥⌘ M
Show in Finder
Open with External Editor ⌥⌘ O

Save ⌘ S
Save As... ⇧⌘ S
Save As Project...

Create Template...
Build Collective / Application...

Export Max for Live Device...
Export Image...

List Externals and Subpatcher Files

Package Manager: Browse Remote Packages

ax packages

Remote Packages

cnmat

FEATURED PACKAGE

Data Knot 1.4K

Data Knot is a set of machine learning tools that are optimized for low latency and real-time performance. The tools can be used with any type of audio input, including optimizations for Sensory Percussion sensors and ordinary drum triggers.

1.0.1

All Categories Alphabetical

Remote Packages 1 Package

CNMAT Externals 59.6K
by CNMAT

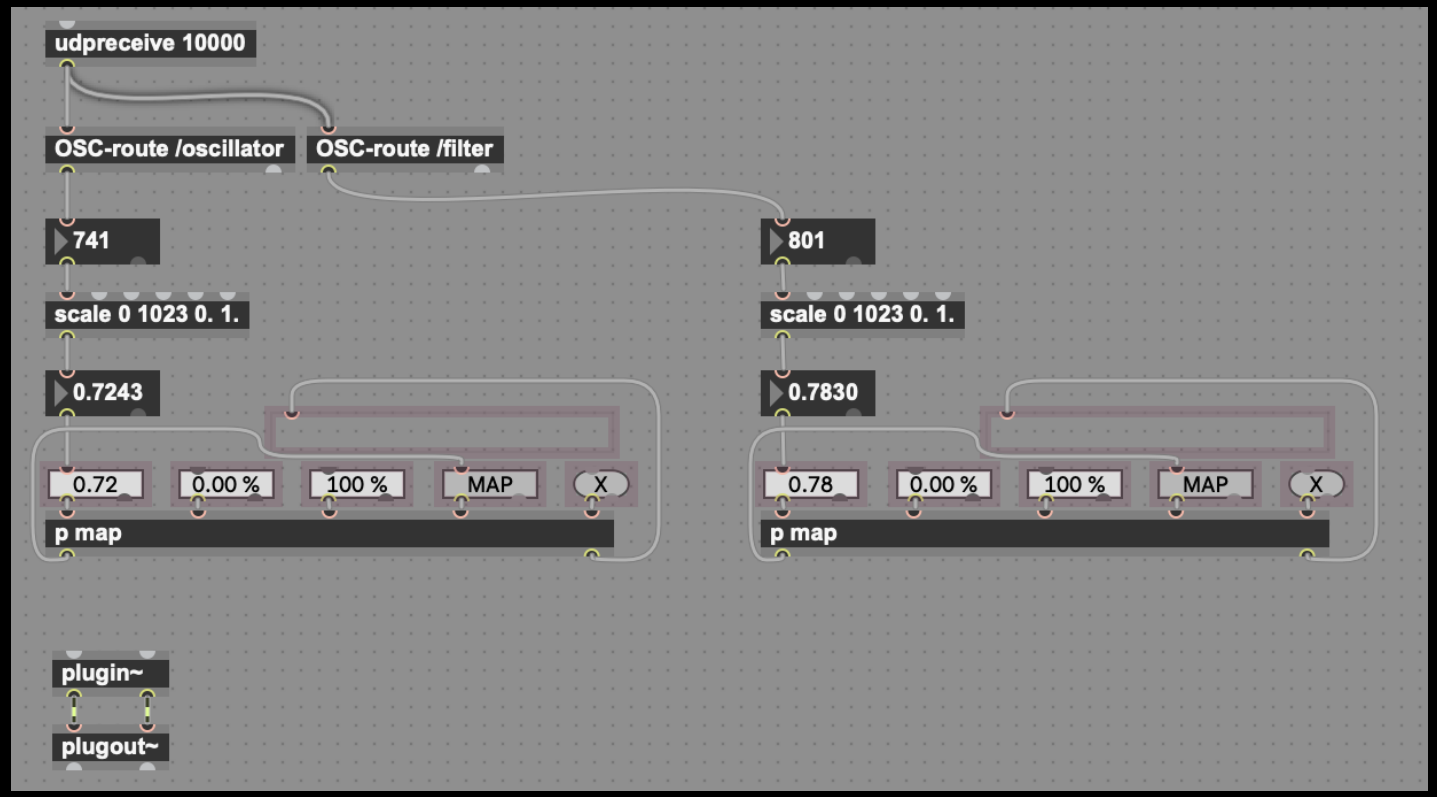
1.0.5 | audio and music

1

BEAUTIFULL SIMPLICITY

In the picture besides you can see how much our M4L patch is cleaned up by using OSC.

There is just the [udpreceive] object and the [OSC-route] objects.



[UDPRECEIVE]

Maybe confusing at first why the osc object in M4L is called [udpreceive], remember the parameter window of OSCOUT CHOP? The protocol stated 'udp', that is what [udpreceive] is referencing. In the world of network communucation there is among others UDP and TCP protocols.

UDP tries to deliver its messages withouth confirmation of establishing a connection. Meaning it is fast but no guarantee that the message actually gets delivered.

TCP always has to do a 'handshake' with the receiving end first before it will send its message. This will increase stability but adds overhead in delivery speed.

For our purposes UDP is often preferred due to its speed. [udpreceive] takes one argument which is the network port which is declared in Touchdesigner in our case. You can also send [udpreceive] a message of (port 7777) to dynamically change the port number.

[OSC-ROUTE]

[OSC-route] is the object that was installed by the CNMAT external. When creating the object make sure to type 'OSC' in capitals, otherwise it cannot be found.

Because we apply pattern matching to extract our data, both objects can be attached to the single outlet of [udpreceive]. The argument in [OSC-route] is the pattern you want to filter by.

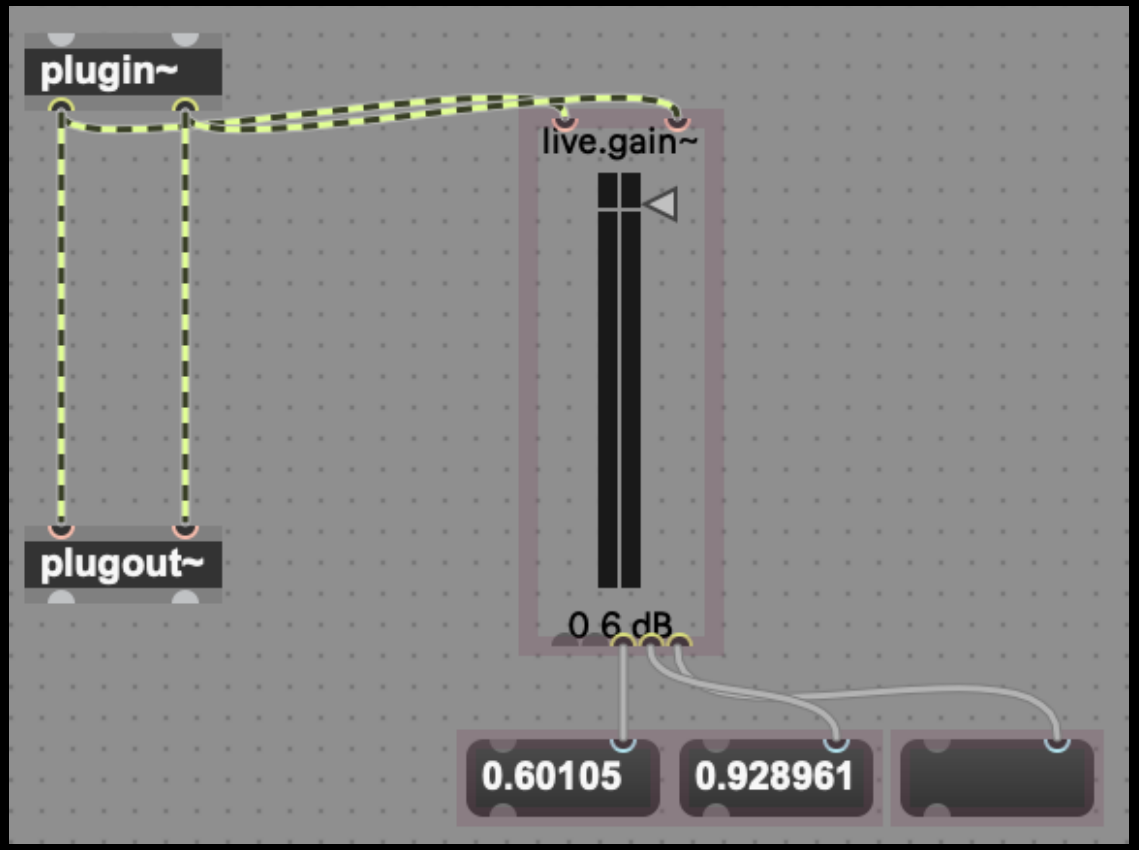
When comparing pattern matching to [unpack], this approach is superior because the order of sending the data does not matter anymore. You may dynamically add or remove message patterns without interupting the logic inside the patch. This is because the [OSC-route] object does not care about order only about patterns.

OKAY COOL, WE CAN SEND AND RECEIVE OSC. BUT IT IS SAME SAME BUT DIFFERENT

In order to actually harness the functionality of OSC let us send information from Ableton back to Touchdesigner. For now we will do a simple gain monitoring.

[PLUGIN~] AND [PLUGOUT~]

Remember these two objects? They allow the flow of sound trough the device. So we can conveniently take all through coming sound from here and plug it into [live.gain~] which is a gain monitor and will give us normalized values of the decibels of the sound.

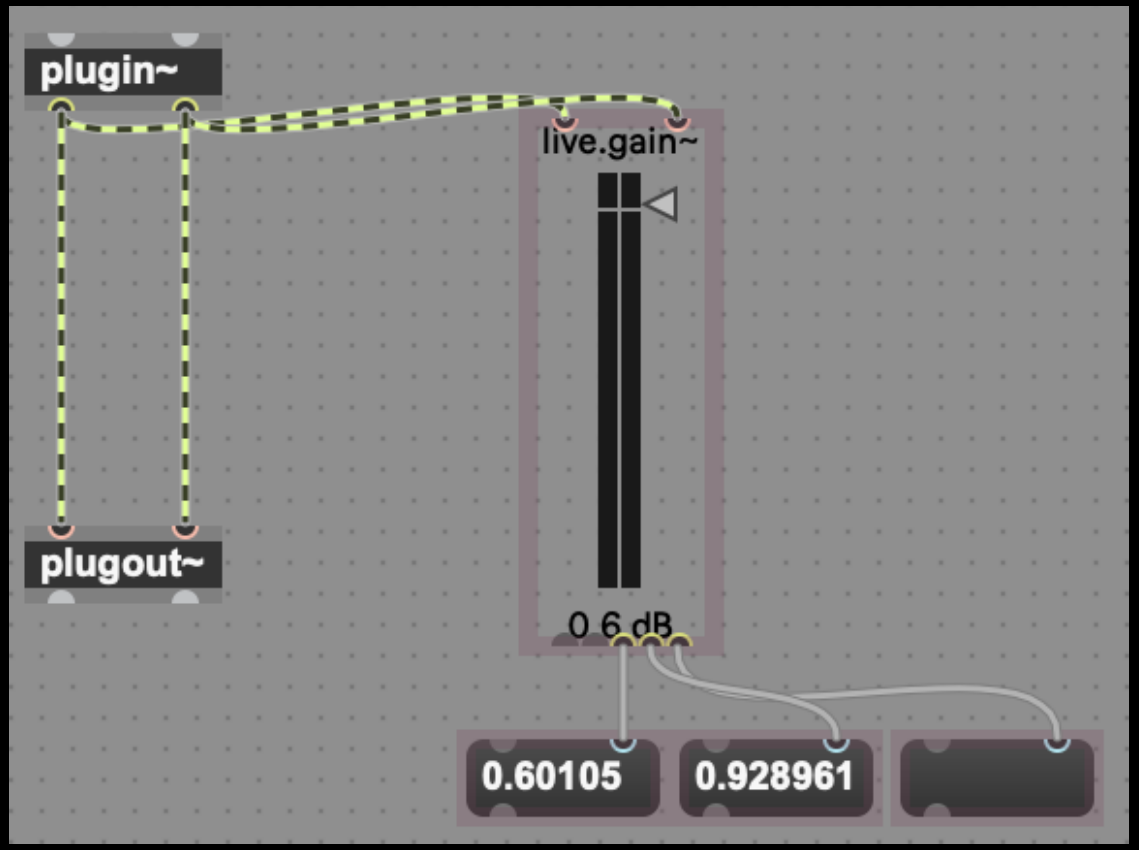


TILDES AND STRIPED WIRES

Since we're beginning to work with audio we are officialy entering the world of MSP of MAX/MSP. Max differentiates between the two worlds by two things.

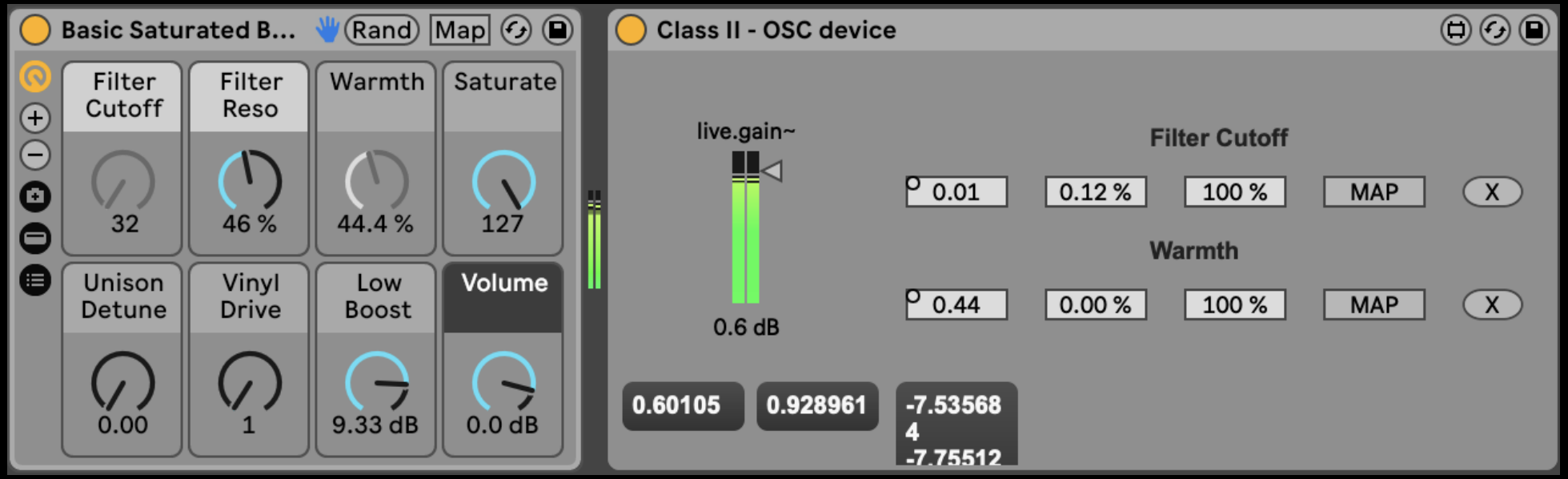
Every object that is audio related or 'signal' based is marked with the tilde-sign, '~'. Also the wires coming out of audio objects are striped as on the picture besides.

MAX or MSP objects hardly ever mix with eachother with exceptions, ofcourse.



SAVE THE DEVICE AND PLAY SOME NOTES!

Add the [live.gain~] and some message boxes to the presentation and place the device on the device rack of Ableton. When you play some notes you should see the gain meter fill with values. Those values are the peak numbers of decibels. The numbers can be confusing because they often are negative suggesting that you will hear negative sound?

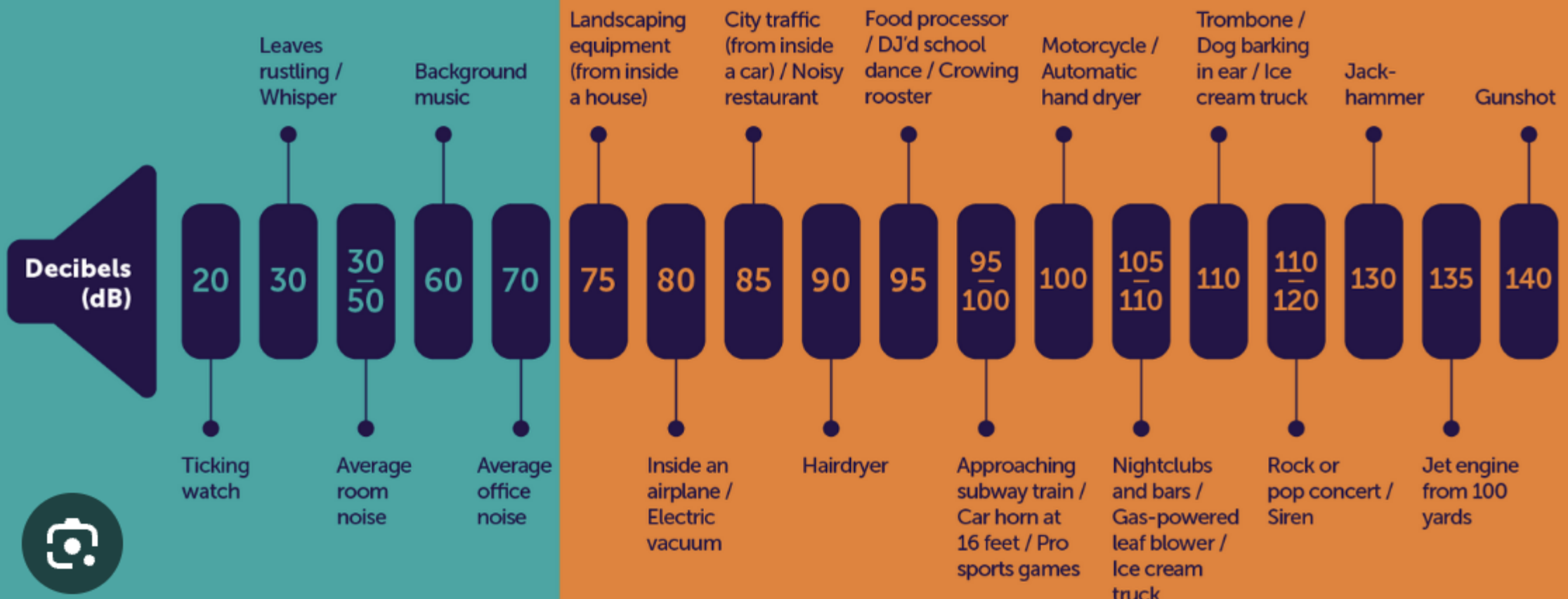


- The negative numbers of decibels on mixers or summing busses and gain meters are in relation to attenuation or amplification of safe hearing levels of dB's which would be around 50 to 60 dB.
- So when the mixer or gain device gives you values of –7 dB they actually mean that the sound is 7 decibels lower than the safe hearing range.

NOISE LEVELS

Sounds at or below 70 dB are safe.

Sounds above 70 dB are harmful.

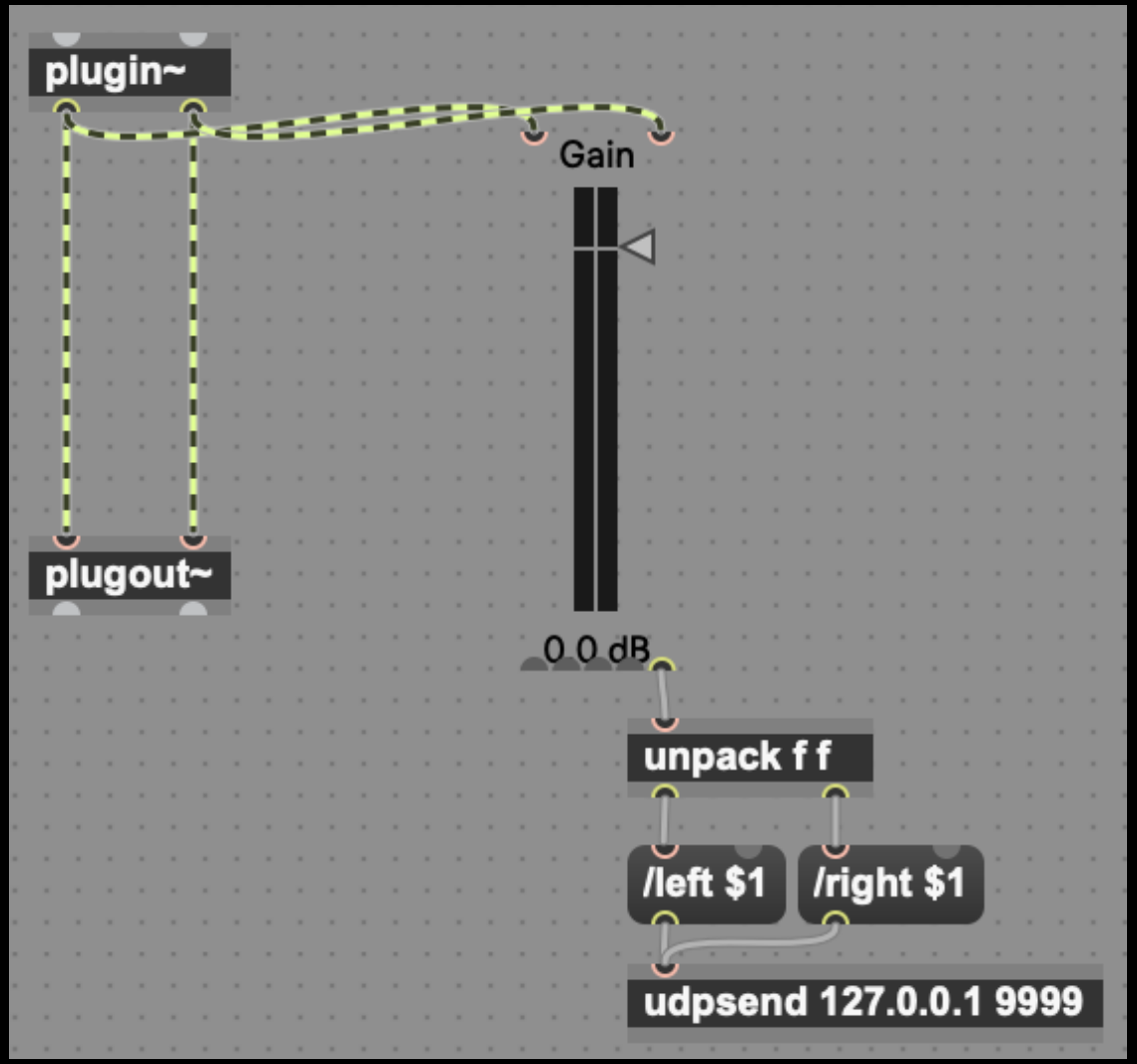


SIMPLE RETURN VALUES

So we take the audio signal from the [plugin~] object and feed that into [live.gain~]. From [live.gain~] we take the right-most outlet and connect it to an unpack.

The outlet gives a list of two floating point numbers, or floats, for each channel; left and right.

Before sending them we want to add information which float belongs to which channel, hence the (/left $1) and (/right $1) message. The $1 will be substituted by any value coming in on the first inlet of the message box. Do you see how the '/' is prepended to 'left' and 'right'? That means that [udpsend] does not automatically prepend it for us like Touchdesigner does.

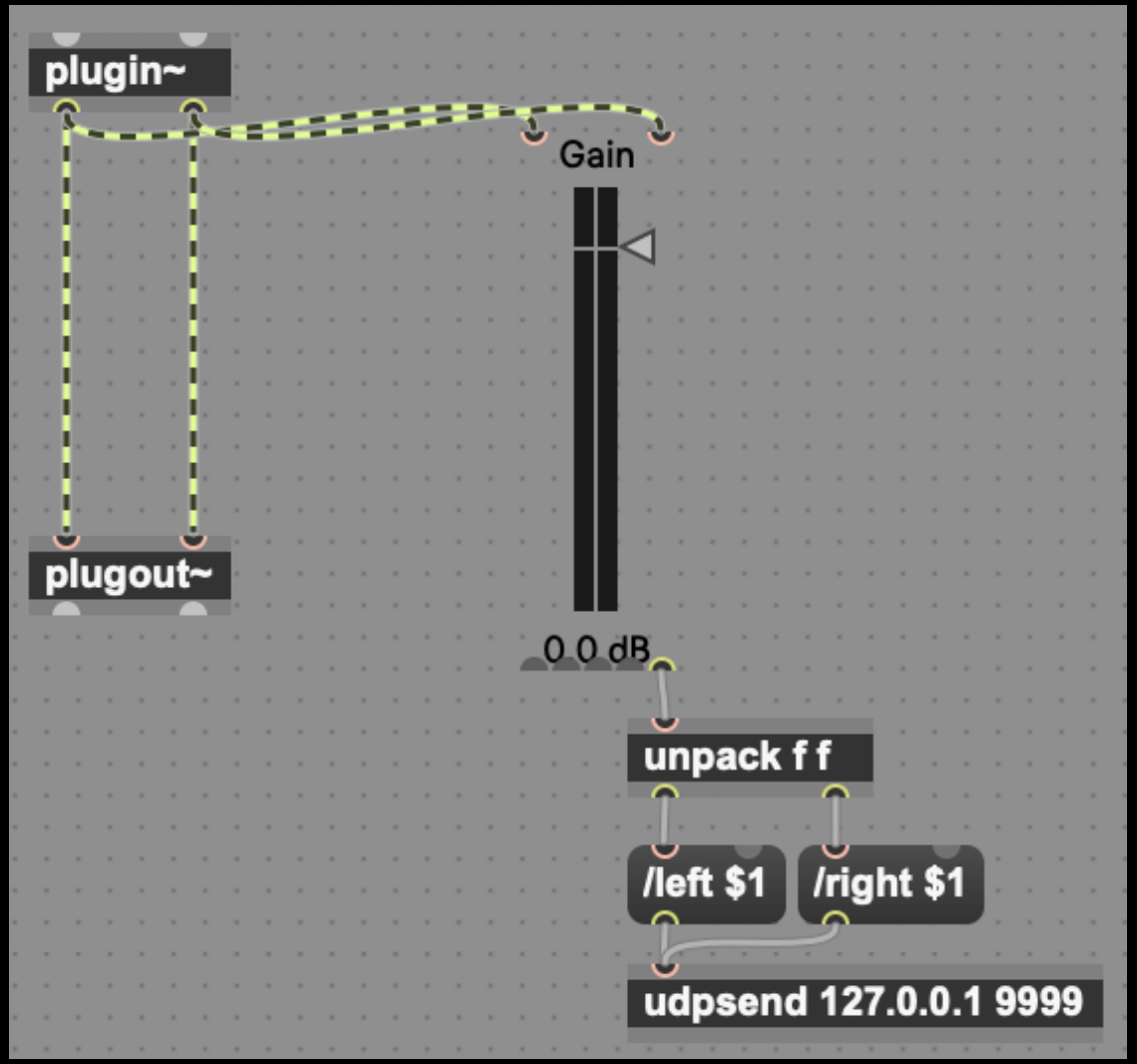


SIMPLE RETURN VALUES

The [unpack] object has two arguments because we know that a list of two float values is exported from [live.gain], therefore [unpack] should be initialized to receive two floating numbers, or 'float float' or abreviated 'f f'.

The [udpsend] object also takes two arguments, the same as OSCOUT CHOP does on Touchdesigner. It needs to know to what machine it is sending the messages and to what port or "room number".

In this case the communication is on the same machine so the ip-address is 127.0.0.1 or 'localhost' and the port number is 9999, which again is arbitrary.



SIMPLE RETURN VALUES

On Touchdesigners' side we only have to drop an OSCIN CHOP and set the right port-number. As soon as some notes or sound is playing on the track in Ableton with the loaded device it should relay the decibel peak information formatted to left and right. From here you can process and reference the values to whatever you want within Touchdesigner.

You see the 'OSC Address Scope' field box? Now it is filled with an *, asterisk, meaning 'everything'. However if you want to filter messages this is where you would do the pattern matching as with the [OSC-route] object.