

Titan Tricks

## sACN Unicast

The idea for this is from a post in the [Facebook group](#):

**Has anyone successfully controlled the Mission Ballroom disco ball with an Avo? Need to unicast sACN to the nodes but can't find a way to do that.**

sACN - or Streaming ACN - is usually used as multicast protocol: the console sends it to a network switch with a special header so that the switch can define groups, one per universe. Nodes and fixtures in turn announce to the switch which universe(s) they need - they **subscribe** to the respective group. The switch then makes sure that each node/fixture receives the data they want, and are not receiving data they do not want. This reduces network traffic and computing load. This mechanism is called **Multicast**.

There is a fallback mechanism in place if some details are missing. In that case the switch defaults to forwarding all sACN data to all nodes/fixtures which requires much more bandwidth and strains the network interfaces but makes sure that all participants have a chance to get their data. This is called **Broadcast**.

However it looks like there are fixtures in the market which do not support this mechanism but require **Unicast**: they only work with data explicitly sent to them.

*(Please excuse this very simplistic explanation. I greatly encourage you to read more on this, and there are plenty of websites dedicated to introducing you to network computing. But for the purpose of this article this might suffice.)*

Avolites Titan at the moment (Titan v15, 2022) sends sACN only as Multicast (automatically falling back to Broadcast if no IGMP capable switch is present). While it has been requested to implement sACN unicast into Titan a solution is required to circumvent this for the time being.

---

## MIDIMonster

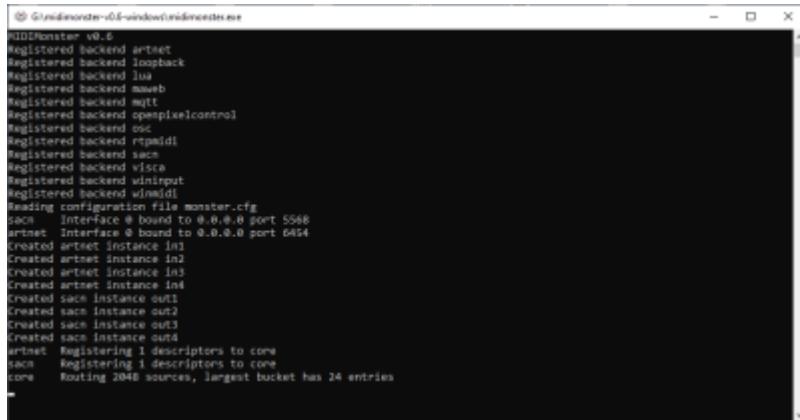
... and along comes MIDIMonster. Available at <https://midimonster.net> this is something like the Swiss Army Knife for lighting protocols. It is a lightweight console application, available for Windows, Linux and OSX, where you simply write a configuration as a text file and then start the program. This comes at the extra benefit that you don't need drivers, nothing being installed, and this can even be run from a USB thumbdrive.

If you download MIDIMonster from the above link, start it, and get this message

Failed to load plugin backends\lua.dll, check that all supporting libraries are present...

then you find [the solution here](#), download [LUA binaries here](#), unzip it, and copy the file lua53.dll into MIDIMonster's main folder.

In order to run it you simply doubleclick `midimonster.exe` in which case it loads its settings from the file `monster.cfg`. Subsequently you can either edit this directly with a text editor, or you write your own file with a different name and drop it onto `midimonster.exe` in order to start it with your configuration.



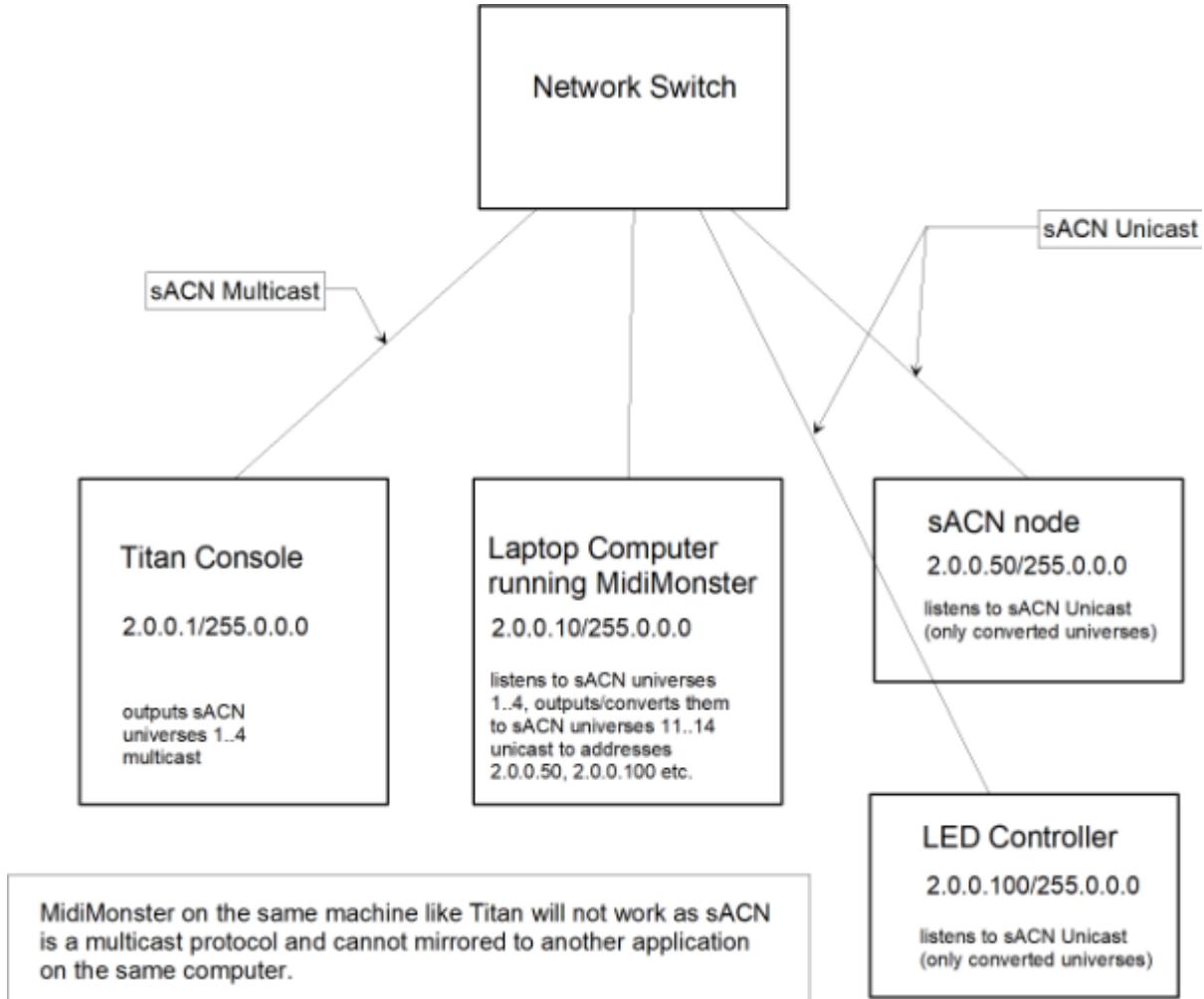
```
© Gmidimonster v0.6
Midimonster v0.6
Registered backend artnet
Registered backend loopback
Registered backend lxa
Registered backend mab
Registered backend mgt
Registered backend smppixelcontrol
Registered backend ssc
Registered backend rtpmddi
Registered backend sncm
Registered backend vlsca
Registered backend wininput
Registered backend winmddi
Reading configuration file monster.cfg
sacn  Interface @ bound to 0.0.0.0 port 5568
artnet  Interface @ bound to 0.0.0.0 port 6454
Created artnet instance in1
Created artnet instance in2
Created artnet instance in3
Created artnet instance in4
Created sacn instance out1
Created sacn instance out2
Created sacn instance out3
Created sacn instance out4
artnet  Registering 1 descriptors to core
sacn  Registering 1 descriptors to core
core  Routing 2048 sources, largest bucket has 24 entries
```

Depending on the situation I successfully tested two scenarios: converting multicast sACN to unicast sACN (sending the data out on different universes) which requires a separate computer, and converting broadcast Art-Net to unicast sACN which may as well run directly on a console. I tried both scenarios with a bunch of computers, checking sACN with [sACNView](#) and - to be sure - verifying the sent data with [Wireshark](#).

---

## sACN to sACN, multicast to unicast

Network structure:



This was the original intention as everything was already prepared on sACN. All it needs is another computer hooked up to the same network, MIDIMonster, and a suitable configuration like this:

## monster.cfg

```
; This configuration maps sACN Multicast universe 1 to sACN Unicast
universe 11
;   for the receiver (node) IP address 2.0.0.100.

[backend sacn]
bind = 0.0.0.0 5568
detect = verbose

[sacn in1]
universe = 1

[sacn out1]
universe = 11
priority = 100
destination = 2.0.0.100
unicast = 1

[map]
```

```
in1.{1..512} > out1.{1..512}
```

An example file for mapping 4 universes

is attached here

All you need to do is adjust this to your needs (in and out universes, target network addresses, maybe more mappings etc.), save this as monster.cfg (or drop this on midimonster.exe, or call it like this: midimonster <path/to/configfile.cfg> (personally I prefer saving this as monster.cfg as I can simply doubleclick midimonster.exe)).

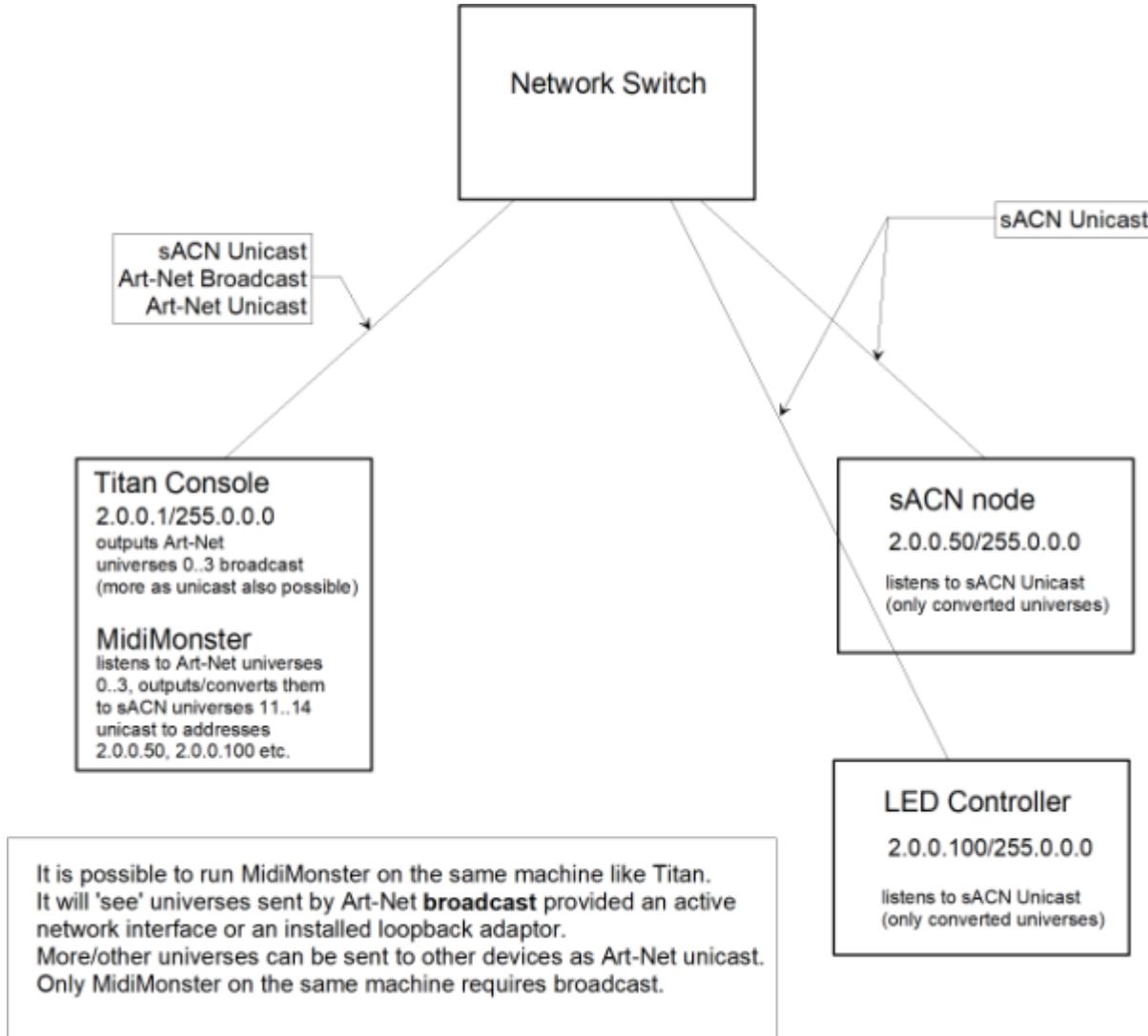
Hints:

- midimonster takes very low processing power (at least in my tests with 4 universes converted)
- at least in my tests it was not possible to make this work with MIDIMonster directly on the console: it does run but doesn't output as required - the nodes see the universes but there is no data. I believe it is not possible to have two applications on the same machine which both attempt to send to the same port, and one also listens to the other.
- make sure that all network addresses are correct, and that firewalls are off or have the required rules set.

---

## Art-Net broadcast to sACN unicast

Network structure:



The question arose if the additional computer could be spared and if MIDIMonster could be run on the same machine Titan runs on. As explained in the section above this does **not** work with sACN - the external nodes see that MIDIMonster sends something but the values are always 0. I suspect this is due to some fancy network routing problem: two programs attempting to send on the same port and on the same machine, AND one program trying to listen to the other.

The solution to this dilemma is using Art-Net: we route Titan's DMX-lines to Art-Net broadcast which is sent to everyone in the network, even to another program on the same computer. Of course the MIDIMonster config needs to be slightly altered as now we have to listen to Art-Net. N.B. in Titan you can adjust per universe whether Art-Net should be sent as unicast (= distinct individual IP address) or broadcast (= IP address 255.255.255.255). There is no problem to send unicast to other nodes as well - only the lines for MIDIMonster need to be broadcast.

MIDIMonster config:

[monster.cfg](#)

```
; This configuration maps Art-Net universe 0 to sACN universe 11,
; Unicast, to node 2.0.0.100
; In order to run on the same computer as Titan Art-Net needs to be
; sent as broadcast.
```

```
[backend sacn]
bind = 0.0.0.0 5568
detect = verbose

[backend artnet]
bind = 0.0.0.0
detect = verbose

[artnet in1]
net = 0
universe = 0

[sacn out1]
universe = 11
priority = 100
destination = 2.0.0.100
unicast = 1

[map]
in1.{1..512} > out1.{1..512}
```

An example file for mapping 4 universes

is attached here

All you need to do is adjust this to your needs (in and out universes, target network addresses, maybe more mappings etc.), save this as monster.cfg (or drop this on midimonster.exe, or call it like this: midimonster <path/to/configfile.cfg> (personally I prefer saving this as monster.cfg as I can simply doubleclick midimonster.exe).

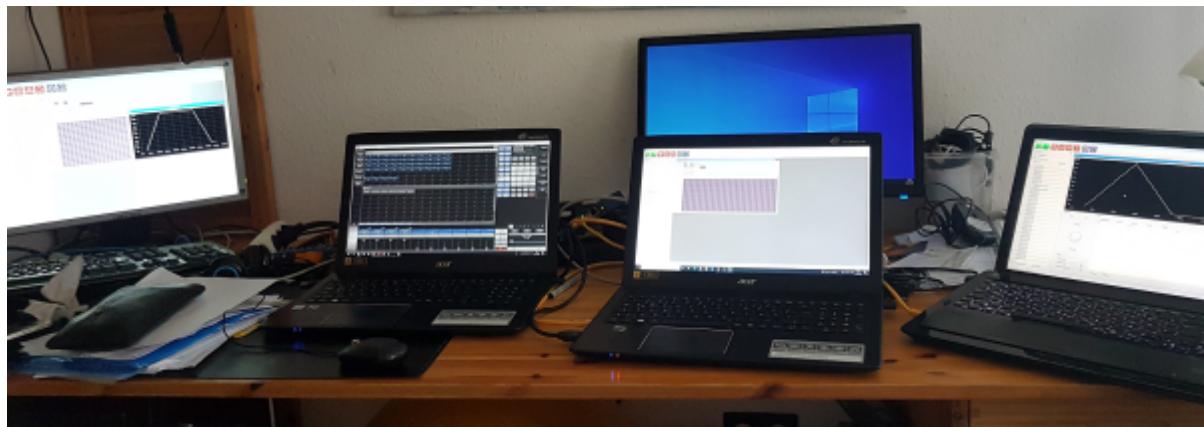
Hints:

- midimonster takes very low processing power (at least in my tests with 4 universes converted)
- make sure that all network addresses are correct, and that firewalls are off or have the required rules set.
- running anything else on the same computer Titan runs on is greatly discouraged for the Titan PC Suite, let alone for Avolites Titan consoles.
- if you need this on your console frequently then you might consider [starting this automatically when the system starts](#).

## Trivia

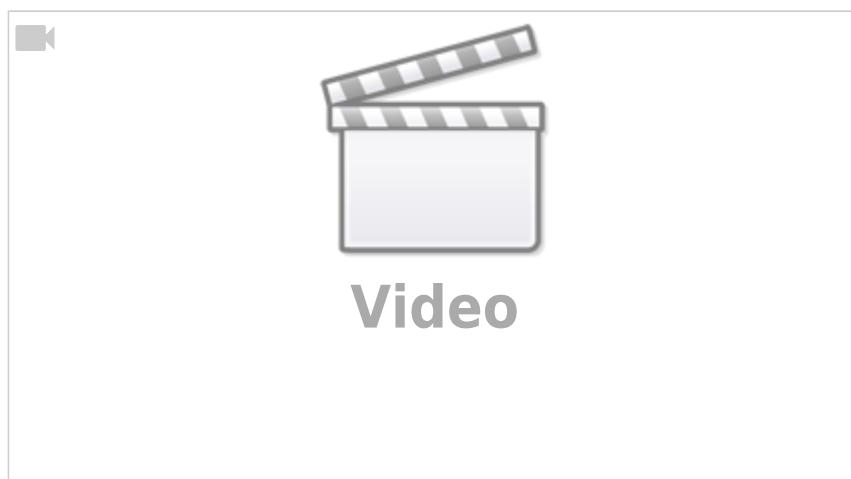
### Picture: my test drive

Yes, I really did try this with some computers (and Wireshark and sACN View - and the network switch is in the middle...)



## Video: another test, on a D9

And yes, I also tried running this directly on a console (and made one of those shaky videos with my mobile phone):



## Wireshark screenshots

This is an example of how the sACN network traffic looks in wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	2.0.0.1	239.255.0.4	UDP	680	50580 + 5568 Len=638
2	0.000202	2.0.0.10	2.0.0.100	UDP	680	5568 + 5568 Len=638
3	0.000255	2.0.0.10	2.0.0.100	UDP	680	5568 + 5568 Len=638
4	0.000286	2.0.0.10	2.0.0.100	UDP	680	5568 + 5568 Len=638
5	0.000322	2.0.0.10	2.0.0.50	UDP	680	5568 + 5568 Len=638
6	0.025799	2.0.0.1	239.255.0.1	UDP	680	50580 + 5568 Len=638
7	0.026145	2.0.0.1	239.255.0.2	UDP	680	50580 + 5568 Len=638
8	0.026145	2.0.0.1	239.255.0.3	UDP	680	50580 + 5568 Len=638
9	0.026145	2.0.0.1	239.255.0.4	UDP	680	50580 + 5568 Len=638
10	0.042179	2.0.0.10	2.0.0.50	UDP	680	5568 + 5568 Len=638
11	0.050089	2.0.0.1	239.255.0.4	UDP	680	50580 + 5568 Len=638
12	0.074718	2.0.0.1	239.255.0.4	UDP	680	50580 + 5568 Len=638
13	0.287579	2.0.0.1	2.255.255.255	UDP	110	60000 + 60000 Len=68
14	0.550179	2.0.0.1	239.255.0.1	UDP	680	50580 + 5568 Len=638

```

> Frame 1: 680 bytes on wire (5440 bits), 680 bytes captured (5440 bits) on interface \Device\NPF_{13268952-260A-450C-847C-7793CA7FC307}, id 0
> Ethernet II, Src: QuantaCo_d3:f1:c8 (d8:c4:97:d3:f1:c8), Dst: IPv4mcast_7f:00:04 (01:00:5e:7f:00:04)
> Internet Protocol Version 4, Src: 2.0.0.1, Dst: 239.255.0.4
> User Datagram Protocol, Src Port: 50580, Dst Port: 5568
> Data (638 bytes)

```

Network addresses in this example (port 5568 is sACN):

2.0.0.1 - Titan PC Suite

2.0.0.10 - MIDIMonster

2.0.0.50 - the 'client' to the left (sACN View)

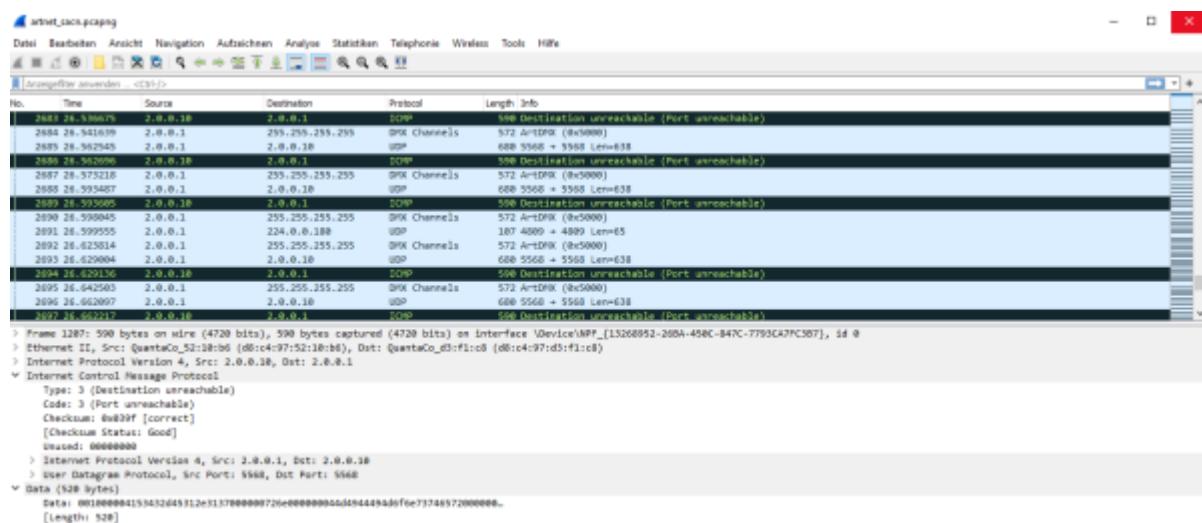
2.0.0.100 - the 'client' to the right (sACN View)

239.255.0.1, 239.255.0.2, 239.255.0.3, 239.255.0.4 are the automatically generated multicast addresses.

Packets from 2.0.0.1 to 239.255.0.1/2/3/4 are the multicast sACN sent from Titan.

Packets from 2.0.0.10 to 2.0.0.100 or 2.0.0.50 are the unicast sACN from MIDIMonster to the 'clients'.''

This is an example of how the mixed Art-Nt/sACN network traffic looks like in wireshark:



Network addresses in this example (port 5568 is sACN, port 4809 is Art-Net):

2.0.0.1 - Titan PC Suite and MIDIMonster

2.0.0.10 - the 'client' in the middle (sACN View)

2.0.0.50 - the 'client' to the left (sACN View) (no packets to this screenshot)

2.0.0.100 - the 'client' to the right (sACN View) (no packets to this screenshot)

Packets from 2.0.0.1 to 255.255.255.255 are the broadcast Art-Net sent from Titan.

Packets from 2.0.0.1 to (3.g.) 2.0.0.10 or 2.0.0.50 are the unicast sACN from MIDIMonster to the 'clients'.''

Btw. 224.0.0.180 is another special auto-generated address which is used by CITP...

## Another idea: Raspberry Pi

And very finally: [MIDIMonster](#) is available for Windows, for OSX, and for Linux. Of course the next idea would be to run this on a Raspberry Pi. But this is probably something for someone else to do. Please

let me know if you want to go down this route .

From:  
<https://www.avosupport.de/wiki/> - **AVOSUPPORT**



Permanent link:  
[https://www.avosupport.de/wiki/tricks/sacn\\_unicast?rev=1653406319](https://www.avosupport.de/wiki/tricks/sacn_unicast?rev=1653406319)

Last update: **2022/05/24 15:32**