Open Sound Control

Michael Lazarski
Studiengang Media Systems

28. August 2013

Inhaltsverzeichnis

1 Geschichte

1.1 Musical Instrument Digital Interface

Die Geschichte von Open Sound Control beginnt mit MIDI (Musical Instrument digital Interface). Also am Anfang der 1980er Große Synthesizer Manufakturen beginnen MIDI als Standard Protokoll zu Implementieren in ihre Hardware wurde dies damals als eine Revolution in der Musik Industrie angesehen. Nur mit einem Computer, einem Midi Controller und der dazugehörigen Software konnte ein Musiker aufnehmen, komponieren und mischen ohne zusätzliches Werkezug. Midi wird bis heute noch in der Musikindustrie benutzt.

1.1.0.1 Wie Midi Funktioniert MIDI ist eine unidirektionale Schnittstelle zur seriellen Datenübertragung. Midi hat keine Datenflusskontrolle. Die Übertragungsgeschwindigkeit beträgt 31250 Bit/s. [Association(1995)]

Es gibt 3 verschiedene Midi Anschlüsse

- MIDI-IN: Hiermit empfängt der MIDI-Controller Signale
- MIDI-OUT: Hiermit sendet der MIDI-Controller Signale
- MIDI-THRU: Sendet die ankommenden Signale vom MIDI-IN Port einfach weiter ohne sie weiter zu verarbeiten

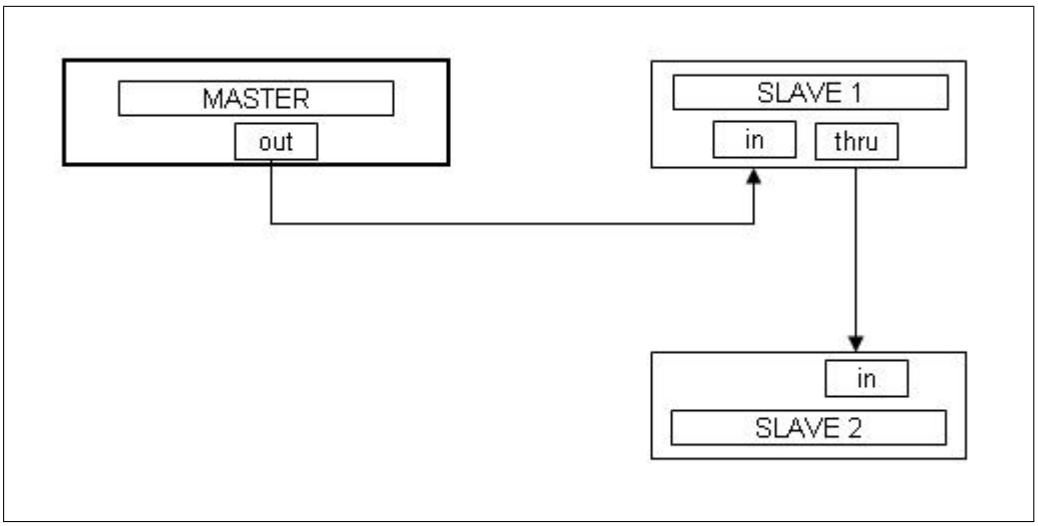

Abbildung 1: MIDI Master Slave

Midi funktioniert nach einem Master-Slave-Prinzip 1. Will man mit einem Midi-Keyboard(Master) einen Synthesizer (Slave) steuern, so verbindet man die MIDI-OUT Buchse des Masters mit der MIDI-IN Buchse des Slaves. Möchte man nun noch einen zweiten Slave hinzufügen, so schließt man ein Kabel zwischen dem MIDI-THRU des ersten Slaves mit der MIDI-IN Buchse des zweiten Slaves.

1.1.1 Nachteile von MIDI

MIDI hat ein paar sehr frustrierende Nachteile:

- Serieller Transport der Daten. Die Daten müssen erst sequenziell geordnet werden.
- Langsame Übertragenes raten. Sehr problematisch bei großen Datenmengen.
- Pitch wird in Integern dargestellt.
- Tendiert zu Keyboard Controllern da Wind, Gittern und andere Instrumente schwer zu Implementieren sind.
- Integer Repräsentation von Controller values. Dies kann zu Ungenauigkeit führen beim einstellen von feinen paramtern.
- Ungenau Zeitauflösung. Diese wird auch wieder in Integern angegeben.
- MIDI benötig Spezielle Hardware.

1.2 Zeta Instrument Processor Interface

1994 wurde von der Firma Zeta Instruments und einer Research Gruppe der University of California das Zeta Instrument Processor Interface (kurz ZIPI) vorgestellt. Es sollte MIDI ablösen und war auf dem OSI-Model aufgebaut. ZIPI konnte sich nie wirklich durchsetzen. MIDI hat eine Peer-to-Peer-Architektur. ZIPI baute auf eine Stern Architektur mit einem HUB in der Mitte als zentrale Anlaufsstelle. Das hatte den Vorteil, dass man einfach Geräte aus dem HUB entfernen konnte. ZIPI ist auch unabhängig von einer physikalischen Implementierung. Das alles hat aber nicht geholfen, da das Adressierungsschema zu komplex war. Es mussten 1016127 Synth zustände geregelt werden im ZIPI Controller. Zum Vergleich hatte midi nur 16 Kanäle, die zwischen 12 und 128 zuständen hatten. Es gibt bis heute keine kommerziell vertriebenen Geräte die ZIPI unterstützen.

2 Open Sound Control

1997 Kündigten die ZIPI Entwickler Matt Wright und Adrian Freed das Open Sound Protokoll an was mehr bekannt ist unter dem Namen Open Sound Control ist ein gutes Beispiel von modernen Netzwerk datenübertragungs Kontrollsystemen. Wie andere Transport Protokolle erlaubt Open Sound Control es das die Kommunikation zwischen einem Computer und anderen Medien Geräten stattfinden kann, darüber hinaus ermöglicht Open Sound Control auch die Möglichkeit das Programme die auf einem Gerät laufen Daten miteinander austauschen können. Open Sound Control wurde speziell für Musiker entwickelt es kann aber auch gut in anderen Gebieten der Netzwerk basierten Kontrolle von System angewendet werden.

2.1 Technik

Einer der Vorteile von Open Sound Control ist, dass man dafür keine spezielle Hardware braucht, wie es z.B. bei MIDI der Fall ist. Open Sound Control unterstützt das Transmission Control Protocol kurz TCP und das User Datagram Protocol kurz UDP, die allgemein bekannt sind und z.B. im Internet oder im Heimnetzwerk schon verwendet werden. Dadurch hat Open Sound Control keine Geschwindigkeitsgrenze. Wir das Netzwerks schneller in dem Open Sound Control eingesetzt wird schneller so wird auch das proktoll schneller verarbeitet. Durch diese Architektur könnte man Open Sound Control auch Open Stuff Control nennen, da theoretisch damit alles gesteuert werden kann, was eine Netzwerkschnittstelle hat.

2.1.1 OSC Packets

Die Nachrichten werden in sogenannten OSC Packets verschickt. Eine Applikation die OSC Pakete verschickt nennt man OSC-Client. Jede Applikation die OSC Packete empfängt nennt man OSC Server.
Ein OSC Packet

2.1.2 Transmission Control Protocol

TCP ist ein verbindungsorientiertes Protokoll. Das heißt jeder Teilnehmer eines Netzwerkes kann exakt zurück verfolgt werden. Der direkte Austausch zweier Stellen ist gewährleistet und Datenverluste können behoben werden, da Daten neu angefordert werden können.

2.1.3 User Datagram Protocol

UDP ist ein verbindungsloses Protokoll. Heißt, Daten werden als “Stream” übertragen. Das heißt ein Sender beginnt “auf Glück” mit dem senden von Daten. Der Empfänger hat jedoch keine Möglichkeit eine Korrektur anzufordern. Der Vorteil von UDP: Es wird an alle Teilnehmer eines Netzwerkes gleichzeitig gesendet.

2.2 Syntax

2.2.1 OSC-Message

EditierenEine sogenannte OSC Message ist aufgeteilt in:

- Address Pattern
- Type Tag String
- Arguments

2.2.2 Adress Pattern

Ein Open Sound Control Adress Pattern beginnt immer mit einem ”/”(Forward Slash) gefolgt von einem String. Möchte man nun also einen Cutoff Filter auf einem Synthesizer ändern, so kann man diesen mit:

/synthesizer/filter/cutoff

ansprechen.

2.2.3 Type Tag String

Es gibt fünf Ätomic Data Types”[Wright(2002)]. Diese sollten in jeder Open Sound Control Implementierung vorhanden sein.

- int32 - 32-bit big-endian two’s complement integer
- OSC-timetag - 64-bit big-endian fixed-point time tag, semantics defined below
- float32 - 32-bit big-endian IEEE 754 floating point number
- OSC-String - A sequence of non-null ASCII characters followed by a null, followed by 0-3 additional null characters to make the total number of bits a multiple of 32.

- OSC-blob - An int32 size count, followed by that many 8-bit bytes of arbitrary binary data, followed by 0-3 additional zero bytes to make the total number of bits a multiple of 32.

Es gibt auch Type Tags die in vielen Implementierungen vorhanden sind aber nicht vom Standard gefordert werden. Hier eine kleine Auswahl[Wright(2002)]:

- c - n ascii character, sent as 32 bits
- r - 32 bit RGBA color
- m - 4 byte MIDI message. Bytes from MSB to LSB are: port id, status byte, data1, data2
- T - True. No bytes are allocated in the argument data.
- F - False. No bytes are allocated in the argument data.
- N - Nil. No bytes are allocated in the argument data.
- I - Infinitum. No bytes are allocated in the argument data.
- [-Indicates the beginning of an array. The tags following are for data in the Array until a close brace tag is reached.
-] - Indicates the end of an array.

Somit ist jeder Typ ein vielfaches von 32 bit. Ein paar beispiel:

/foo,f 440
/foo,iisff 1000 -1 "hello"1.23 4.56

Wie man an den Beispielen sehen kann, kann man auch Mehrer Argumente in einer Nachricht übergeben. Im zweiten Beispiel sind die Argumente 1000, -1, "hell"1.23 4.56. In einer Programmiersprache wie Java würde so ein eine Übergabe z.B. so aussehen:

foo(new Integer(1000), new Integer(-1),new String("hello"), new Float(1.23), new Float(4.56)

Die Nachricht in Byte Darstellung:

2f	(/)	66	(f)	6f	(o)	6f	(o)
0	()	0	()	0	()	0	()
2c	(,)	69	(i)	69	(i)	73	(s)
66	(f)	66	(f)	0	()	0	()
0	()	0	()	3	()	e8	(è)
ff 68	(ÿ) (h)	ff 65	(ÿ) (e)	ff 6c	(ÿ) (l)	ff 6c	(ÿ) (l)
6f	(o)	0	()	0	()	0	()
3f	(?)	9d	()	f3	(ó)	b6	(¶)
40	(@)	b5	(µ)	b2	(”)	2d	(-)

An der Tabelle lässt sich gut die 32 Bit Aufteilung erkennen. Das aligment ist wichtig für die Verarbeitung in der CPU. Nur so kann man eine hohe performence erhalten die für die Realtime Verarbeitung notwendig ist.

3 Implementierungen

3.1 Hardware Implementierungen

3.1.1 The Missing Link OSC/MIDI Translator

The Missing Link ist eine Kleine unabhängig Box die ihre eignes Wifi Netzwerk aufbaut. So kann man mit mobilen Geräten auf die Box zugreifen. Sie übersetzt dann die speziellen OSC Nachrichten in MIDI Nachrichten, mit der man z.B. Synthesizer kontrollieren kann. The Missing Link ist für schnelle Übertragungen, Flexibilität und Konfigurierbarkeit ausgelegt. Es braucht keinen Computer, der die Steuerung übernimmt. So kann OSC fähige Wireless Lan Geräte direkt mit der Box Kommunizieren. Es können sogar mehrere OSC Geräte auf die Box zugreifen.

3.1.1.1 Spezifikation

- 802.11b WiFi: adhoc oder infrastructure Modus und open, WEP, WPA oder WPA2 Sicherheit
- OSC über wireless UDP: OSC zu MIDI, MIDI zu OSC, Multiple gleichzeitige Verbindungen
- MIDI: MIDI IN/OUT (Standard 5 Pin). Konfigurierbares internes Routing mit soft MERGE/THRU. Alle MIDI befehle sind Implementiert.
- Strom: 9V-12V DC, tip +/-, ¿=250mA oder über USB

- Abmessungen: 3.3”x 2.2”x 1.6”

Abbildung 2: The Missing Link

3.2 Software Implementierungen

3.2.1 osc-ruby

osc-ruby ist eine simple OSC 1.1 Implementierung für Ruby 1.9/2.0 und für JRuby. Sie kann sowohl auf dem Client als auch auf dem Server eingesetzt werden. Schauen wir uns eine simple Implementierung an, wo ein Server und ein Client auf dem gleichen Host laufen.[Funaba(2013)]

```
# einbinden der Libary
require 'rubygems'
require 'osc-ruby'
require 'osc-ruby/em_server'
# Erstellen eines Servers objektes
@server = OSC::EMServer.new( 3333 )
# Erstellen eines Client Objektes der sich mit dem Server verbindet
@client = OSC::Client.new( 'localhost', 3333 )
# Implementierung einer simplen Willkommens Nachricht
@server.add_method '/greeting' do | message |
  puts "#{message.ip_address}:#{message.ip_port} -- #{message.address}
end

# Starten des Server Loops
Thread.new do
  @server.run
end
# Ein "Willkommen!" an alle schicken
@client.send( OSC::Message.new( "/greeting" , "Willkommen!" ))
# nach 3 millisekunden das Program beenden
sleep( 3 )
```

3.2.2 oscmex

Mit oscmex gibt es eine Libary die Funktionen für Matlab bereitstellt. So können mit oscmex Nachrichten empfangen und gesendet werden zwischen OSC Endpunkten. Die Libary ist dafür gedacht Musik zu analysieren. oscmex ist in C geschrieben. Eine möglicher Einsatz wäre streng mathematisch genau Gestensteuerung oder die genaue Berechnung von Timings für Präsentationen. Auf ein Beispiel wird verzichtet.[Schmeder(2013)]

3.2.3 TouchOSC

TouchOSC ist eine Applikation für Apples iOS und Googles Android. Für beide Systeme kostet sie 4,99$. Das Besondere an TouchOSC ist das es Nachrichten über Wireless LAN verschicken und Empfangen kann. Dadurch eignet sich TouchOSC besonders für Live Auftritte oder Live Shows. Ein voller Midi und Apples Logic Pro Support ist implementiert. TouchOSC ist komplett modular so kann es z.b. Als DJ pult oder als Effektgeräte oder als Drum Synthesizer genutzt werden. Frei erhältlich ist ein Editor für das Interface

Abbildung 3: TouchOSC

so kann man am Computer sein Interface erstellen und abspeichern. Damit man es zu einem Späteren Zeitpunkt wiederbenutzen kann.

REFERENCES

[Association(1995)] MIDI Manufacturers Association. Midi 1.0 detailed specification, 1995. URL `http://madamebutterface.com/assets/documents/MIDI%201.0%20Detailed%20Specification.pdf`.

[Funaba(2013)] Tadayoshi Funaba. osc-ruby. https://github.com/aberant/osc-ruby, 2013. URL `https://github.com/aberant/osc-ruby`.

[hexler.net(2011)] hexler.net. touchosc. http://hexler.net/, 2011. URL `http://hexler.net/`.

[LLC(2002)] Jabrudian Industries LLC. The missing link. http://wifimidi.com/, 2002. URL `http://wifimidi.com/`.

[Relandini(2001)] Silvio Relandini. Midi master and slave, 2001. URL `http://upload.wikimedia.org/wikipedia/commons/5/5d/MIDI_master_and_slave.JPG`.

[Schmeder(2013)] Andy W. Schmeder. oscmex. http://sourceforge.net/projects/oscmex/, 2013. URL `http://sourceforge.net/projects/oscmex/`.

[Wright(2002)] Matt Wright. Opensound control specification. http://archive.cnmat.berkeley.edu/OpenSoundControl/OSC-spec.html, 2002. URL `http://archive.cnmat.berkeley.edu/OpenSoundControl/OSC-spec.html`.

Abbildungsverzeichnis