



**POLITECNICO**  
MILANO 1863

IMAGE AND SOUND  
**ISPG**  
PROCESSING GROUP

# CREATIVE PROGRAMMING AND COMPUTING

Protocols

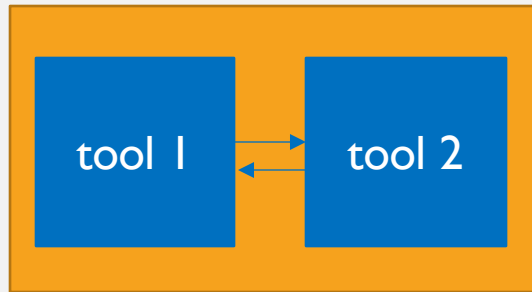
# INTRODUCTION

- Very often in creative computing there is the need to use different tools or different machines
  - In a project several people are involved with different used tools
  - Tools are used for different tasks
  - People are not in the same place
- Hence, there is the need to let them to communicate
- Two issues:
  - Communication: exchange of information
  - Synchronization: exchange of control messages with purpose to make the tools or machines work in a synchronous way

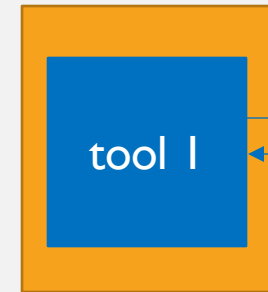
# INTRODUCTION

- Different ways of communication and synchronization

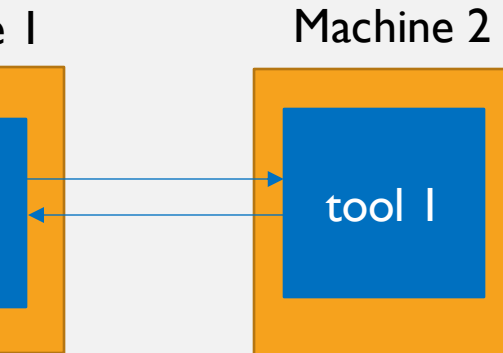
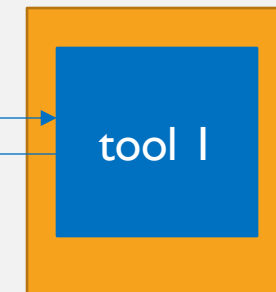
Machine



Machine 1



Machine 2

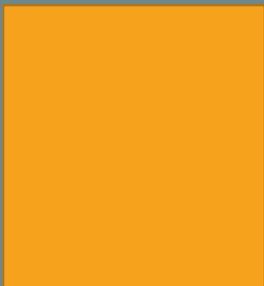


Dedicated Net

Machine 2



Machine 1



Machine 2



Machines in the same place

Internet

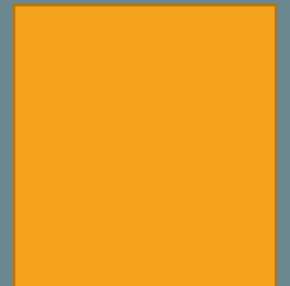
Machine 2



Machine 1



Machine 2

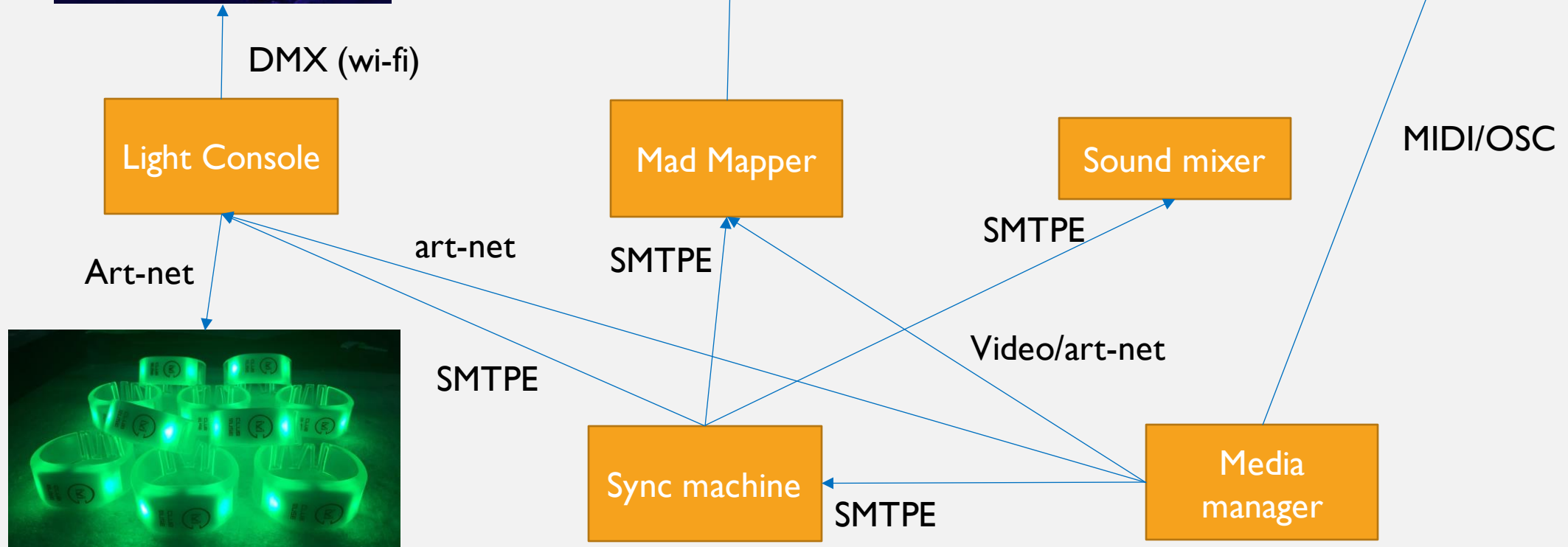


Geographically distributed machines

MUSE







## SMTPE - TIME CODE



# SMPTE TIMECODE

- Society of Motion Picture and Television Engineers (SMPTE) timecode
- Originally used to sync audio and video signals
- It is sequence of data sent at specific time intervals
- Timecodes contains:
  - binary coded decimal ***hour:minute:second:frame*** identification
  - 32 bits for use by users (U-bit)
  - *Color frame flag*
  - *Drop frame flag*
  - Binary group flags for the interpretation of U-bit
- It can be or not embedded in the media (audio, video)
- If it is embedded in the media it is embedded in each frame

## SMPTE TIMECODE

- Time code can have a multiple number of frame rates, common ones are:
  - 24 frame/sec (film High Definition, 2k, 4k, 6k)
  - 25 frame/sec (PAL Europe System, Brazil, Argentina and SECAM)
  - 29.97 ( $30 \times 1000 \div 1001$ ) frame/sec (NTSC American System(US, Canada, Mexico, Colombia, etc...))
  - 30 frame/sec (HDTVSD)

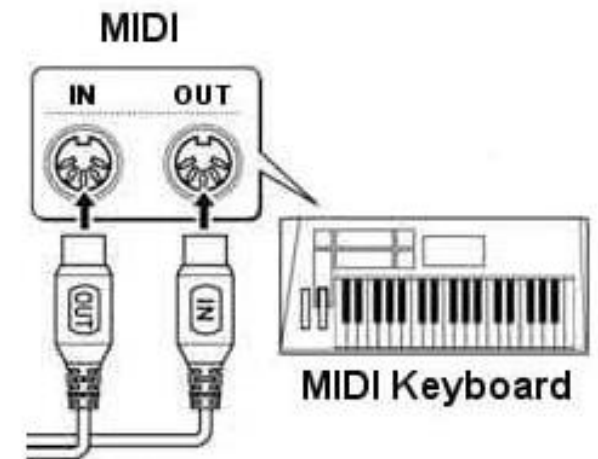


## MIDI - REVIEW



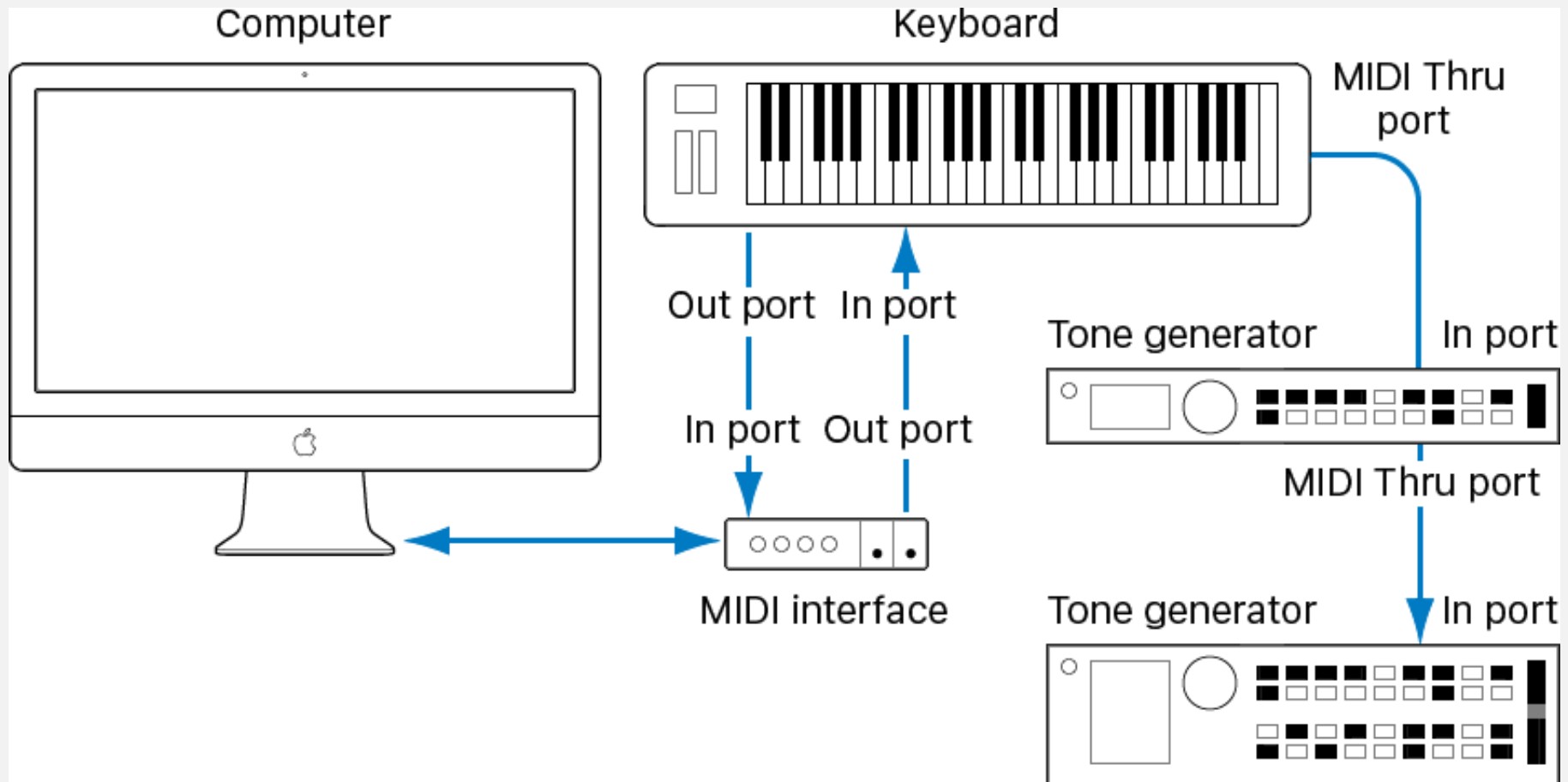
# MIDI

- Presented in 1983 by North American Music Manufacturers Show
- It is a communication protocol to exchange musical information among machines
- The MIDI data stream is a unidirectional asynchronous bit stream at 31.25 Kbits/sec. with 10 bits transmitted per byte (a start bit, 8 data bits, and one stop bit)
- Based on **Channels** and **Messages** to send through the channels
- A device can send messages to a specific channel. All the devices that listen to that channel will receive the message



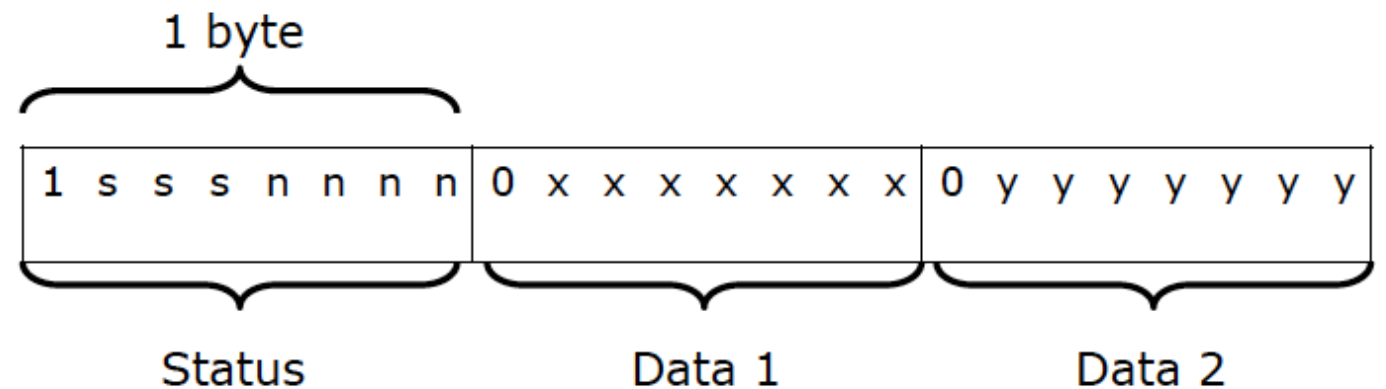
# MIDI

- MIDI communication systems is a point to point communication but it is possible to create a chain thanks to MIDI Thru



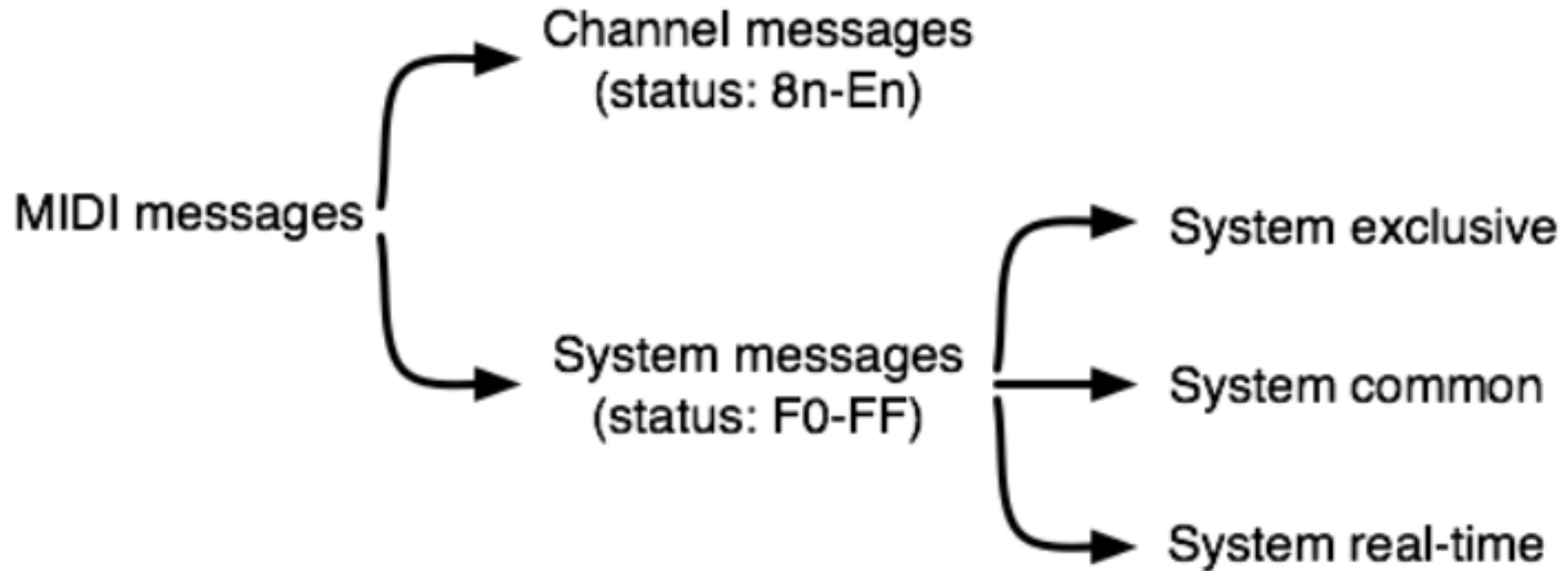
# MIDI MESSAGE FORMAT

- 2 types of MIDI message bytes: the **status byte** and the **data byte**
- Status bytes always begin with 1, and data bytes with 0. That leaves only 7 bits per byte to represent the message (128 possible values).
- MIDI messages begin with the status byte, where 3 bits (sss) are used to denote the type of message, and 4 bits (nnnn) to denote the channel number to which the message apply (max. 16 channels).





# MESSAGE



- 2 types of messages: referred to the channel or to the whole system

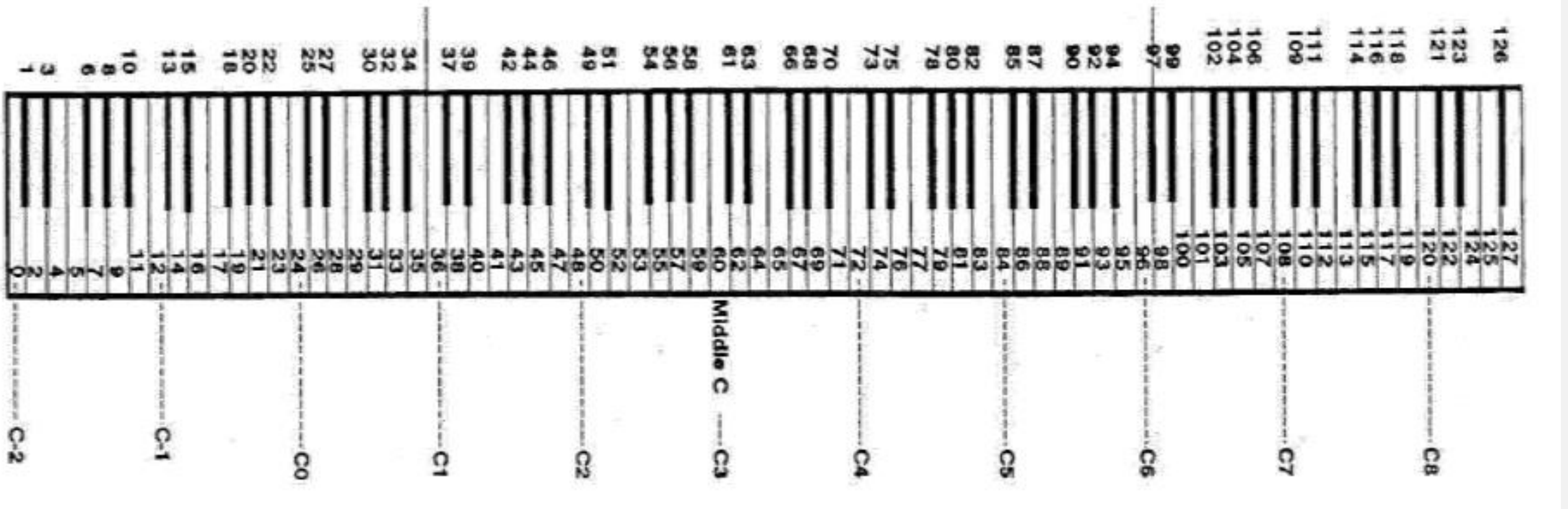
## CHANNEL MESSAGE

- MIDI channel numbers (n) are referred as 1 to 16, while in reality they are represented by binary values 0 to 15 (0-F).
- Example: the status byte of a note off message for channel 7 is “86”
- MIDI channel messages

Message	Status	Data 1	Data 2
Note off	8n	Note number	Velocity
Note on	9n	Note number	Velocity
Polyphonic aftertouch	An	Note number	Pressure
Control change	Bn	Controller number	Data
Program change	Cn	Program number	-
Channel aftertouch	Dn	Pressure	-
Pitch wheel	En	LSbyte	MSbyte

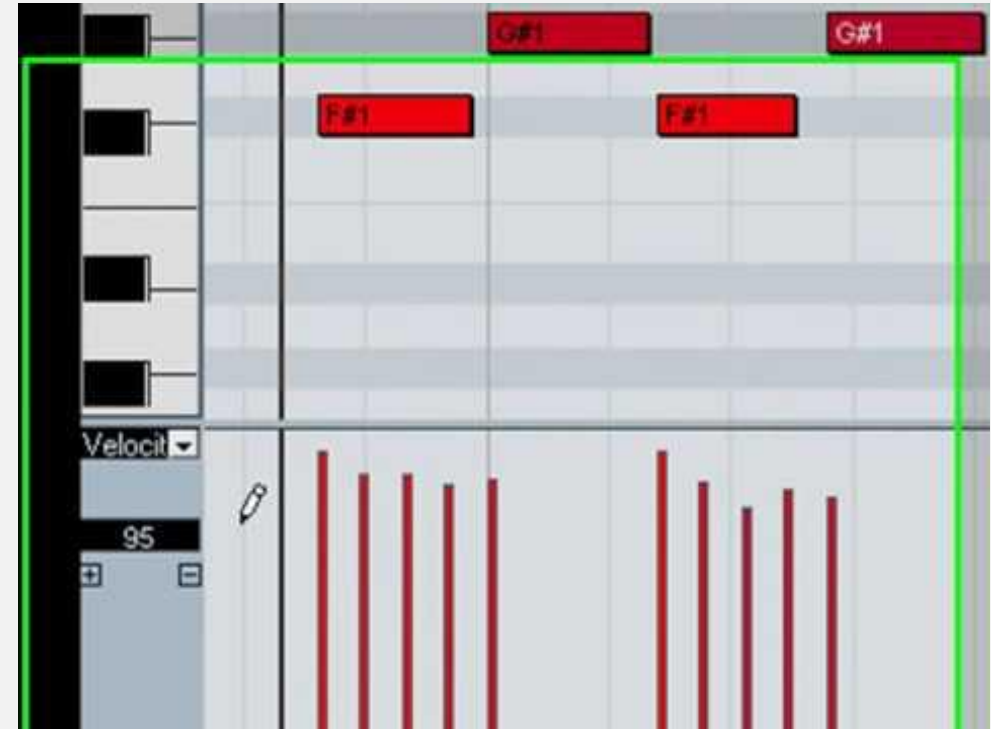
# NOTE ON/OFF

- Note on” triggers a musical note,“note off” turns it off
- There are 128 (0-127) possible note values (~10 octaves) mapped to the chromatic western music scale.



## NOTE ON/OFF

- Note on messages are also associated to a “velocity” value, which is the force with which a note is played
- Note on velocity can be used to control volume and timbre of a sound (e.g. by controlling the scaling of an envelope generator)
- Note off velocity relates to the speed at which a note was released





## AFTERTOUCH

- It refers to the amount of pressure placed on a key

Message	Status	Data 1	Data 2
Polyphonic aftertouch	An	Note number	Pressure
Channel aftertouch	Dn	Pressure	-


- Polyphonic key pressure transmits a separate value per key (thus requiring separate sensors). This is expensive.
- Most instruments use a single sensor, thus one message is sent with the approx (Channel Aftertouch)

## CONTROL CHANGE

- MIDI is also capable of transmitting orders for the controllers (e.g. pedals, switches and wheels) of a given device
- All these controllers are addressed by using the same status byte, with the first data byte determining the specific controller

Message	Status	Data 1	Data 2
14-bit controllers MSbyte	Bn	00-1F (controllers)	Data
14-bit controllers LSbyte	Bn	20-3F (same order)	Data
7-bit controllers/switches	Bn	40-65	Data
Undefined	Bn	66-77	-
Channel mode	Bn	78-7F	Data

# CONTROL CHANGE

<i>Controller number (hex)</i>	<i>Function</i>	
00	Bank select	 <p>The first 64 numbers are used for 32 physical controllers at greater control resolution</p> <p>Thus 2 data bytes (14 bits) are used to represent the controller's position (16384 possible values)</p>
01	Modulation wheel	
02	Breath controller	
03	Undefined	
04	Foot controller	
05	Portamento time	
06	Data entry slider	
07	Main volume	
08	Balance	
09	Undefined	
0A	Pan	
0B	Expression controller	
0C	Effect control 1	
0D	Effect control 2	
0E-0F	Undefined	
10-13	General purpose controllers 1-4	
14-1F	Undefined	
20-3F	LSbyte for 14 bit controllers (same function order as 00-1F)	

# CONTROL CHANGE

<i>Controller number (hex)</i>	<i>Function</i>	
40	Sustain pedal	<b>7-bit controllers use only 1 data byte for their position</b>  <b>On/off switches are represented with values 00-3F for off, and 40-7F for on.</b>
41	Portamento on/off	
42	Sostenuto pedal	
43	Soft pedal	
44	Legato footswitch	
45	Hold 2	
46-4F	Sound controllers	
50-53	General purpose controllers 5-8	
54	Portamento control	
55-5A	Undefined	
5B-5F	Effects depth 1-5	
60	Data increment	
61	Data decrement	
62	NRPC LSbyte (non-registered parameter controller)	
63	NRPC MSbyte	
64	RPC LSbyte (registered parameter controller)	
65	RPC MSbyte	



## SOUND CONTROLLERS

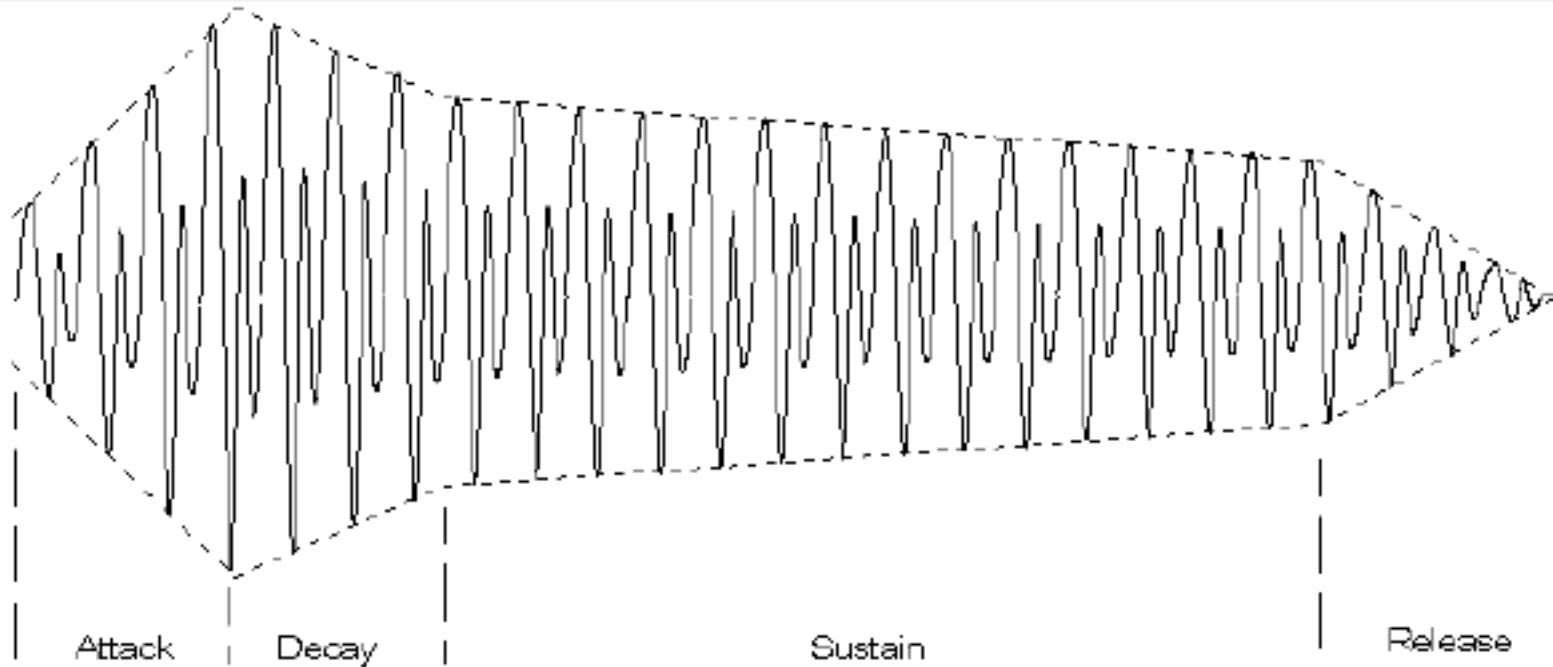
- Sound and effect controllers provide control over the sound quality of a device and the parameters of built-in effects
- They are device dependent

Controller #	Function	Controller #	Function
46	Sound variation	5B	External effects depth
47	Timbre/harmonic content	5C	Tremolo depth
48	Release time	5D	Chorus depth
49	Attack time	5E	Detune depth
4A	Brightness	5F	Phase depth
4B-4F	No default		

- Some devices can re-map these messages for other device-specific effects

# SOUND CONTROLLERS

- Timbre and brightness are used to affect the spectral content of the sound (e.g. by controlling filtering characteristics)
- Envelope controllers modify the attack and release time of a sound (following the ADSR envelope approximation).



## PROGRAM CHANGE

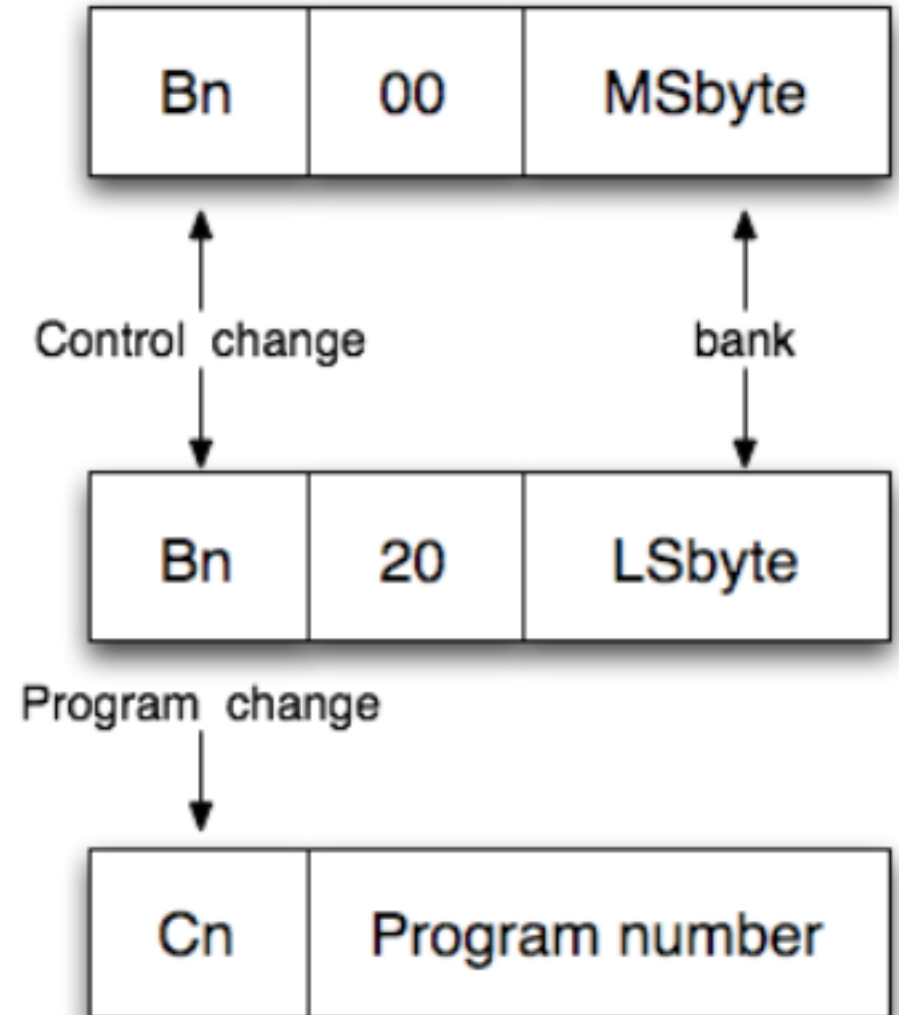
- Program change is used to change between patches or presets of an instrument
- The message is channel specific
- There is only one byte of data used (only 128 presets to choose from)

Message	Status	Data 1	Data 2
Program change	Cn	Program number	-

- On some instruments, programs are organized in “banks” of 8, 16 or 32 presets

# PROGRAM CHANGE

- The program change message (Cn) only allows for the selection of one of 128 voices.
- A sound generating device may allow the user to define a program change map to decide which voice corresponds to which message
- Commonly, current devices have very large (>> 128) program memories
- A solution is to precede program changes with a 14-bit “bank select” control change message





# PITCH BEND

- The pitch wheel is the only controller with a status byte of its own.
- It uses 2 data bytes (14 bits of resolution)
- Such resolution ensures smooth changes of pitch
- It is channel specific



Message	Status	Data 1	Data 2
Pitch wheel	En	LSbyte	MSbyte

OSC



## MIDI LIMITATIONS

- Very Serial transport mechanism - Data must be ordered sequentially, problematic with heavy data streams.
- Slow transmission rate - Again problematic with heavy loads.
- Integer representation of pitch - Ignores other feasible representations.
- Bias towards 12-tone equal temperament
- Bias towards keyboard controllers - Difficult to implement on wind instruments, guitars, other instruments.
- Integer representation of controller values - Results in insufficient granularity during controller movement.
- Insufficient timing resolution - Again an integer representation.
- Requires special hardware - Needed for external connections

## MIDI LIMITATIONS

- MIDI messages have a very static predefined format
- No complex or personalized data structures are allowed
- It not possible to identify devices in a chain (MIDI is not net oriented)
- No complex network of devices are supported
- No mechanisms for flow control or error management is implemented

MIDI has been updated to follow modern requirements, but the basic structure of the protocol is not appropriate

# OSC

- OSC → Open Sound Control
- Networking protocol for real-time musical control information
- Introduced by Center for New Music and Audio Technologies in 1997
- Transport-independent (today used with UDP, TCP, WiFi, serial connections, and within applications)
- OSC is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology
- OSC is a digital media content format for streams of real-time audio control messages



# OSC

- Open-ended, dynamic, URL-style symbolic naming scheme
- Symbolic and high-resolution numeric argument data
- Pattern matching language to specify multiple recipients of a single message
- High resolution time tags
- "Bundles" of messages whose effects must occur simultaneously
- Query system to dynamically find out the capabilities of an OSC server and get documentation
- Network-based system utilizes common UDP/TCP transport mechanisms

## OSC - PACKAGES

- The unit of transmission of OSC is the OSC Packet.
- Applications that send packets are clients, applications that receive them are servers.
- An OSC Packet consists of its contents and its size. The size is always a multiple of 4 bytes, to guarantee the alignment of the packet with the payload of transmission protocols.
- An OSC packet can be represented by a datagram in UDP or TCP protocols.
- An OSC packet can contain different data types

## OSC - MESSAGES

- **OSC Messages:** consist of an OSC Address Pattern followed by an OSC Type String and followed by zero or more OSC Arguments.
- **OSC Address Pattern:** is a string beginning with “/” and specifying the recipient of a OSC Packet – URL style address
- Address of an OSC message can be a regular expression pattern:
- E.g., /filters/[2-4]/cutoff-freq 2354.1
- **OSC Type Tag String:** is a string beginning with the “,” character and followed by a sequence of characters that specify the type of data transmitted in OSC Arguments
- Some older implementations of OSC may omit the OSC Type Tag string

## OSC - MESSAGES

- Atomic type of arguments

OSC Type Tag	Type of corresponding argument
i	int32
f	float32
s	OSC-string
b	OSC-blob

- **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.

# OSC - MESSAGES

OSC Type Tag	Type of corresponding argument
h	64 bit big-endian two's complement integer
t	OSC-timetag
d	64 bit ("double") IEEE 754 floating point number
S	Alternate type represented as an OSC-string (for example, for systems that differentiate "symbols" from "strings")
c	an 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.



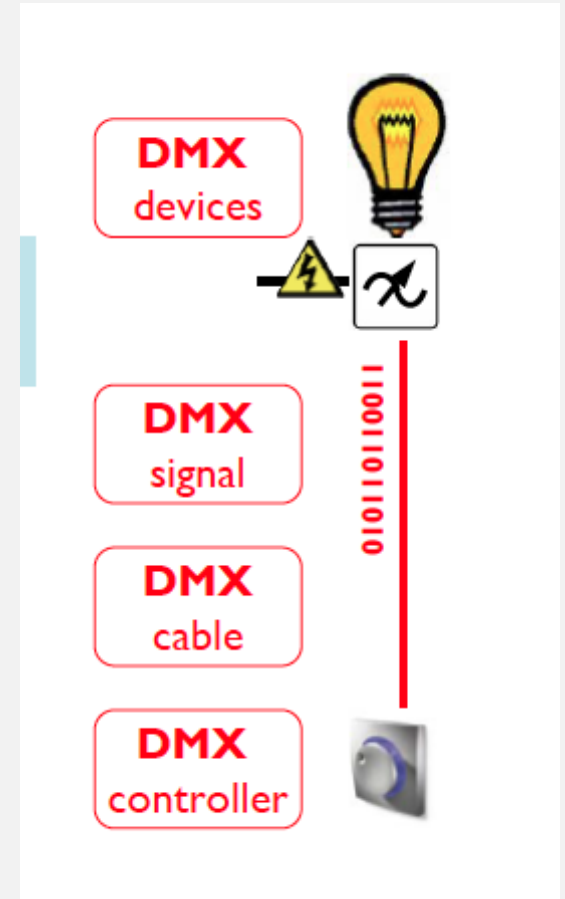
**Image Sonification by O3LAB**

DMX



# DMX – DEFINITION

- **DMX - Digital Multiplexed signal**
- It is a protocol, in which a DMX controller communicates to DMX luminaires
- It can be transmitted on dedicated hardware (XLR reversed cable) or through ethernet and WiFi connection



# DMX – DEFINITION

## Controllers



## Receivers





# DMX – DEFINITION

- D by USITT in 1986. DMX is used mainly for “controlling lighting equipment and accessories” in entertainment applications (theatre, staging, concerts etc)



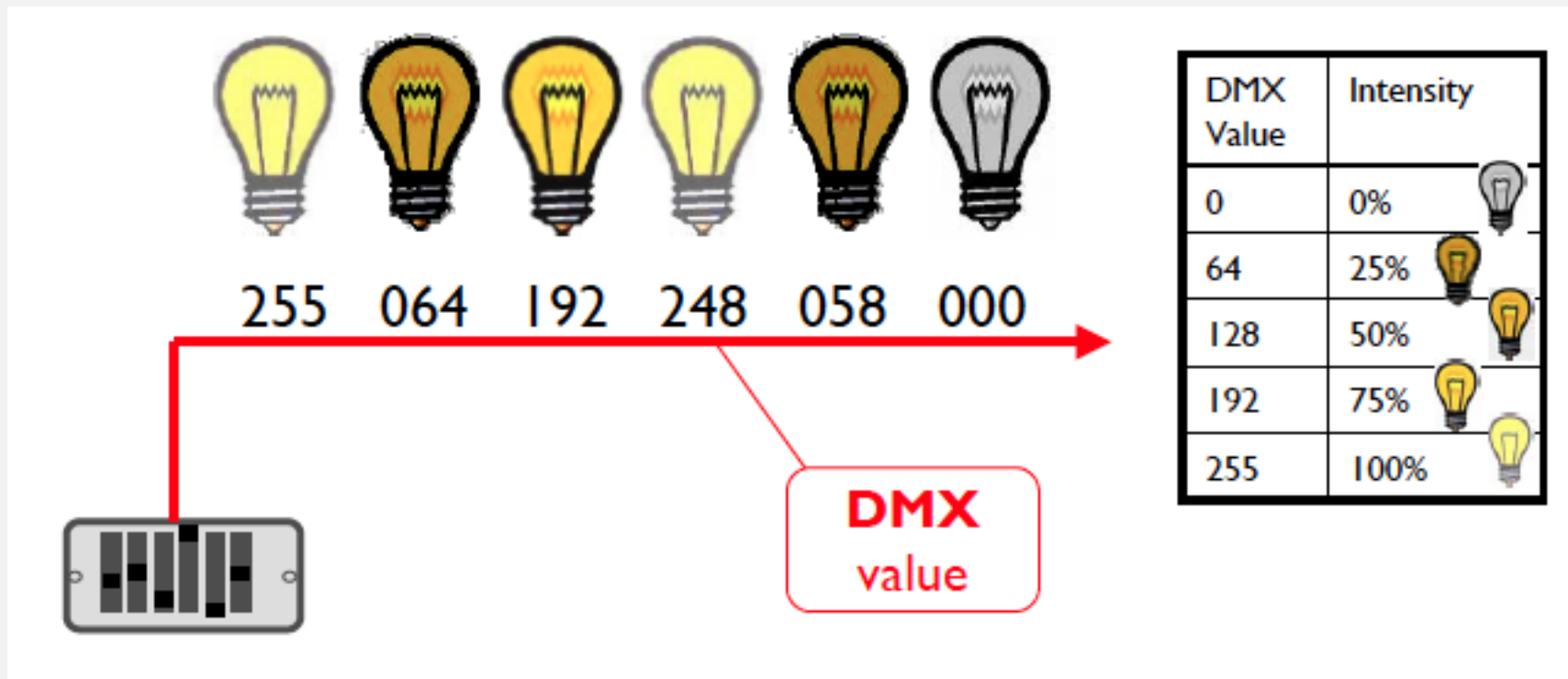
- Nowadays DMX is used more and more in architectural scene setting applications as well.





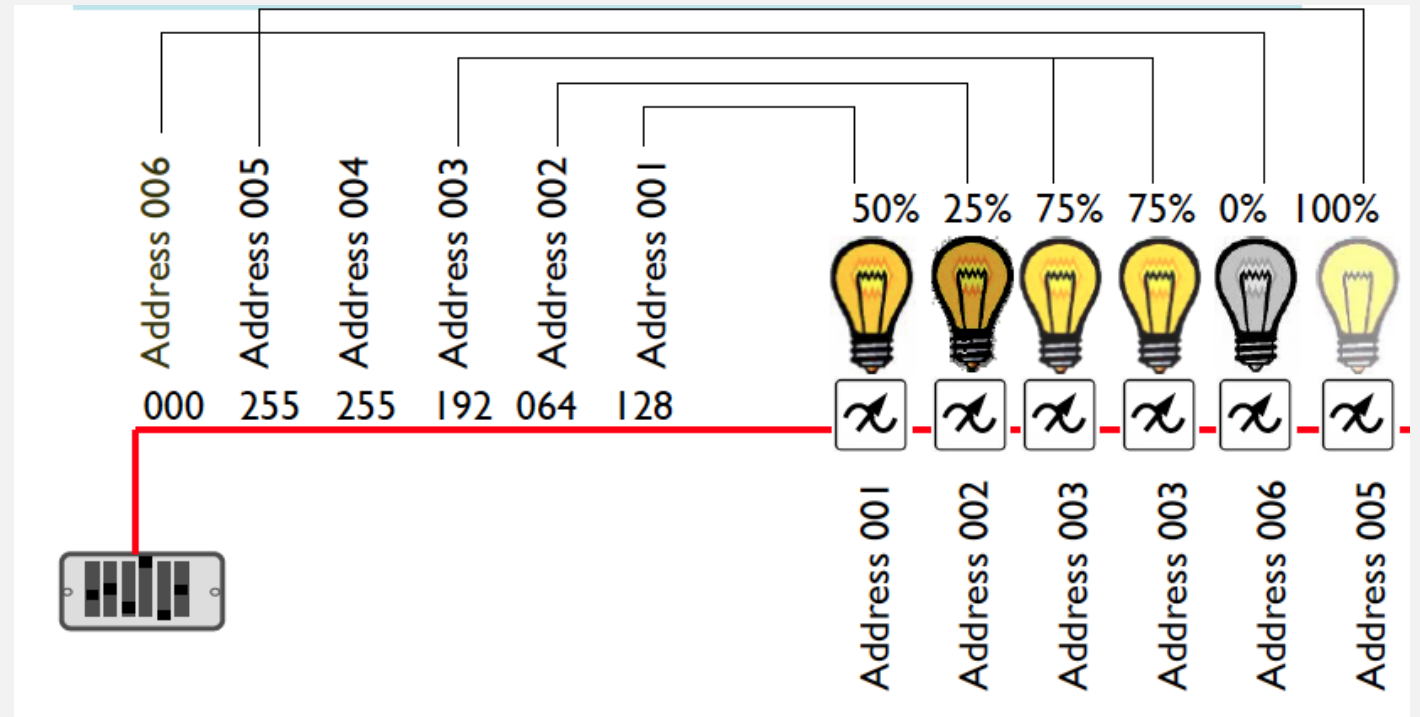
## DMX – MESSAGES

- A DMX controller sends DMX values.
- This is a 8-bit value (value between 0-255) originally corresponding to a 0-100% intensity



# DMX – MESSAGES

- A DMX message is composed by strings of 512 values
  - According to network transmission rate, a DMX message is sent with 40 fps
  - The location of a DMX value is referred to as “address”
  - By addressing the DMX device, it knows which DMX value to use
- 
- No explicit address is sent, the first 8-bit message has address 001 and so on



## DMX – MESSAGES

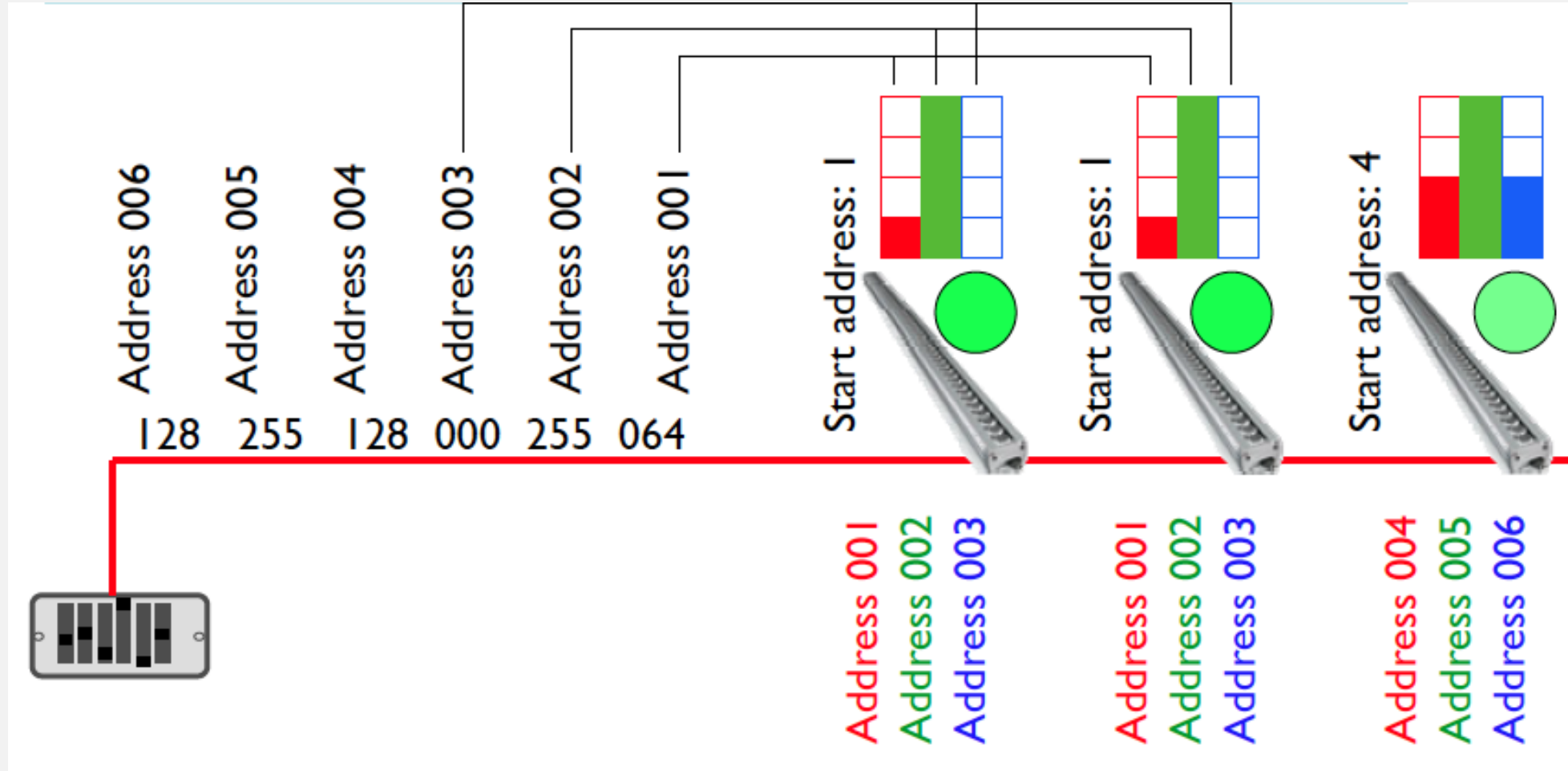
- In nowadays applications more controls are needed, not only light intensity
- DMX is very flexible you can merge the bytes of data
- For RGB light DMX uses 3 addresses
- The DMX-start address is the first DMX value used, (DMX-start address+1) the second value, etc.

Start address:	1	4	22	215
DMX addr. Red intensity:	1	4	22	215
DMX addr. Green intensity:	2	5	23	216
DMX addr. Blue intensity:	3	6	24	217

- One DMX line can control  $512/3=170$  individual RGB units

# DMX – MESSAGES

- Note: luminaires with the same address will always react the same!



## DMX – MESSAGES

- An example of DMX for a modern moving light
- Address 1: Intensity
- Address 2: R
- Address 3: G
- Address 4: B
- Address 5: Lens focus
- Address 6: Horizontal rotation
- Address 7: Vertical rotation
- In the case 8-bit allows a too small data resolution, more addresses can be joint for higher resolution data

# DMX

- In order to have a correct mapping of addresses between the controller and the receiver factories should release appropriate update software
- A DMX channel (with 512 addresses) is called *universe*
- More universes can be used at the same time





## ART-NET

- Different protocols allows to send DMX packages over the ethernet
- The most popular is Art-net
- It is a simple implementation of DMX512-A protocol over UDP in which lighting control information is conveyed in IP packets
- Allows also management functions such as detecting nodes, updating node control parameters, and transmitting timecodes; and functions that allow nodes to "subscribe" to "publisher" nodes so that, for example, nodes A and B can subscribe to node C (C will unicast information to A and B).
- It allows to transmit 32768 universes at the same time
- Today is used also for more complex data like video or moving scenography control

# MATERIALS

- **References**

- Open Sound Control(<http://opensoundcontrol.org>)
- Art-Net4 - Specification for the Art -Net 4 Ethernet Communication Protocol

- **Further readings**

- Davin E. Gaddi, **Media Design and Technology for Live Entertainment**, Routledge 2018