Title: Lecture 3
Credit: Taught by Professor Mikael Giordi
Draft: 1

FADE IN:

INT. STANFORD CLASSROOM - DAY

The classroom buzzes with excitement as the students prepare to present their MIDI device projects. Professor Giordi stands at the front of the room, eager to listen and provide constructive feedback on their work.

Clyde steps up first to present his MIDI device.

CLYDE
For my project, I built a MIDI foot controller using an Arduino Uno, a few footswitches, and potentiometers. I programmed it to send MIDI note on/off and control change messages. I encountered some difficulty with debounce, but I managed to resolve it by implementing a software debounce solution.

PROFESSOR GIORDI
(nodding)
Great job, Clyde! Your choice of the Arduino Uno as a microcontroller is a solid starting point. I'm glad you tackled the debounce issue, as it's a common challenge when working with switches. Keep refining your code and hardware design!

Noah is up next, presenting his MIDI device.

NOAH
I designed a MIDI wind controller using a Raspberry Pi, a pressure sensor, and a touch-sensitive strip for pitch control. The most challenging part was converting the pressure sensor data into accurate MIDI messages for velocity and note on/off.

PROFESSOR GIORDI
(smiling)
Excellent work, Noah! Your choice of the Raspberry Pi allows for more advanced features, and I like the idea of incorporating a pressure sensor for a more expressive instrument. Keep experimenting with the sensor data conversion to improve the accuracy of your MIDI messages.

Raj takes the floor, presenting his MIDI device.

RAJ
I created a MIDI drum pad using an Arduino Mega, piezo sensors, and rubber pads. I programmed it to send note on/off messages with velocity based on the piezo sensor output. The hardest part was filtering the sensor data to avoid false triggers and noise.

PROFESSOR GIORDI
(enthusiastic)
Well done, Raj! I like your choice of piezo sensors for capturing the drum hits. Filtering sensor data is indeed a challenge but also an essential skill when designing MIDI devices. Keep refining your filtering algorithm to improve your drum pad's performance!

The remaining students present their MIDI devices, each showcasing unique designs and technologies. Professor Giordi provides constructive feedback, commending their creativity and offering suggestions for improvement.

PROFESSOR GIORDI
(beaming)
I'm truly impressed by the variety and innovation in your MIDI device projects! You've all demonstrated a solid understanding of the concepts we've covered, and I'm excited to see how you'll continue to apply this knowledge throughout the course. Keep up the great work!

PROFESSOR GIORDI
(cheerfully)
Now that we've seen the fantastic work you've all done on your MIDI devices, let's move on to today's lecture. We'll be discussing MIDI clock synchronization and MIDI System Exclusive messages. These topics will help you create even more advanced and interactive MIDI devices.

Giordi switches on the projector and starts a presentation to visually explain the concepts.

PROFESSOR GIORDI
(continuing)
First, let's talk about MIDI clock synchronization. This feature allows multiple MIDI devices to sync their internal clocks, ensuring that events such as note sequences and tempo-based effects occur in perfect time with each other.

PROFESSOR GIORDI
(enthused)
Let's take a closer look at the various types of MIDI clock messages and how they work together to ensure accurate synchronization between MIDI devices.

Giordi switches on the projector and displays a diagram illustrating the different MIDI clock messages.

PROFESSOR GIORDI
(continuing)
First, we have the MIDI Clock message itself, also known as the timing clock or tick. This message is sent 24 times per quarter note to maintain a steady tempo. The MIDI Clock message is represented by the status byte 1111 1000 in binary. Devices use these ticks to keep their internal clocks in sync, ensuring that sequencers, arpeggiators, and other tempo-based devices work in harmony.

Giordi elaborates on the importance of maintaining a consistent tempo, as it allows multiple devices to stay in sync and perform together seamlessly.

PROFESSOR GIORDI
(continuing)
Next, we have the MIDI Start message. This message, represented by the status byte 1111 1010 in binary, signals the beginning of a sequence. When a device receives this message, it starts playback from the beginning of its sequence. This ensures that all connected devices start playing simultaneously and maintain a synchronized performance.

Giordi explains that the MIDI Start message is essential for coordinating the playback of multiple devices in a MIDI setup.

PROFESSOR GIORDI
(continuing)
Now let's discuss the MIDI Stop message. Represented by the status byte 1111 1100 in binary, this message signals the end of a sequence. When a device receives this message, it stops playback immediately. This allows you to control the stopping of multiple devices at once, ensuring a synchronized conclusion to the performance.

Giordi emphasizes the importance of the MIDI Stop message for managing the playback of connected devices.

PROFESSOR GIORDI
(continuing)
Finally, we have the MIDI Continue message. This message, represented by the status byte 1111 1011 in binary, resumes playback from the last stopped position. When a device receives this message, it picks up where it left off in its sequence. This is particularly useful in live performance situations, where you may want to pause and then resume playback without starting from the beginning.

Giordi concludes his detailed explanation by highlighting the crucial role of MIDI clock messages in maintaining synchronization between devices.

PROFESSOR GIORDI
(continuing)
Understanding the different types of MIDI clock messages and their roles in synchronization is essential when designing MIDI devices that work together in a seamless manner. By mastering these messages, you'll be able to create more advanced and coordinated MIDI systems.

Giordi moves on to discuss MIDI System Exclusive messages.

PROFESSOR GIORDI
(continuing)
MIDI System Exclusive, or SysEx, messages allow manufacturers to create custom messages specific to their devices. These messages can be used to transmit data such as patch settings, sample data, or firmware updates between devices.

PROFESSOR GIORDI
(excited)
Let's take a closer look at the structure of SysEx messages and how they can be used to create custom functionality in your MIDI devices.

Giordi switches on the projector and displays a diagram illustrating the structure of a SysEx message.

PROFESSOR GIORDI
(continuing)
SysEx messages are unique because they allow manufacturers to create custom messages specific to their devices. Unlike standard MIDI messages, which have a fixed format, SysEx messages can vary in length and content, providing a high level of flexibility.

Giordi starts to break down the structure of SysEx messages step by step.

PROFESSOR GIORDI
(continuing)
A SysEx message begins with a status byte, which is always 1111 0000 in binary, or F0 in hexadecimal notation. This byte signals the start of a SysEx message and informs the receiving device to expect a custom message.

Giordi moves on to explain the next component of a SysEx message.

PROFESSOR GIORDI
(continuing)
After the status byte, a SysEx message includes a manufacturer ID. This ID, typically represented by one or three bytes, is unique to each manufacturer and helps devices distinguish between SysEx messages from different sources. The MIDI Manufacturers Association (MMA) manages these IDs to avoid conflicts and ensure compatibility between devices.

Giordi then discusses the core of a SysEx message: the device-specific data.

PROFESSOR GIORDI
(continuing)
Following the manufacturer ID, a SysEx message contains the actual device-specific data. This data can be anything the manufacturer wants it to be, such as patch settings, sample data, or firmware updates. The structure and content of this data are entirely up to the manufacturer, but it must be formatted in a way that the receiving device can understand and process.

Lastly, Giordi explains the final component of a SysEx message.

PROFESSOR GIORDI
(continuing)
To signal the end of a SysEx message, an End of Exclusive (EOX) byte is used. This byte is always 1111 0111 in binary, or F7 in hexadecimal notation. The EOX byte informs the receiving device that the SysEx message is complete, allowing it to process the custom data as intended.

Giordi concludes his detailed explanation of SysEx messages by emphasizing their importance in creating custom functionality.

PROFESSOR GIORDI
(continuing)
Understanding the structure of SysEx messages is crucial for creating advanced MIDI devices with custom features that go beyond standard MIDI messages. By mastering SysEx, you can develop unique and innovative MIDI devices tailored to specific needs and applications.
Giordi concludes the lecture by summarizing the key points:

PROFESSOR GIORDI
(continuing)
In summary, MIDI clock synchronization and System Exclusive messages are advanced features that can help you create more sophisticated and interactive MIDI devices. As we move forward in this course, consider how you might incorporate these concepts into your projects to achieve new levels of creativity and functionality.
As the lecture comes to an end, Professor Giordi prepares to give the students their next assignment. The students, now well-informed about MIDI clock synchronization and SysEx messages, are eager to apply their newfound knowledge.

PROFESSOR GIORDI
(smiling)
Now that you've learned about MIDI clock synchronization and SysEx messages, it's time to put that knowledge into practice. For this week's assignment, I want each of you to create a MIDI device or modify your existing project to incorporate either clock synchronization or SysEx functionality.

Giordi provides examples of potential projects for the students.

PROFESSOR GIORDI
(continuing)
For those interested in clock synchronization, you could create a simple MIDI sequencer or arpeggiator that syncs with an external MIDI clock source. Alternatively, you could build a device that sends clock messages to control other devices in your setup.

Giordi then offers suggestions for students who wish to explore SysEx messages.

PROFESSOR GIORDI
(continuing)
If you'd rather focus on SysEx messages, consider creating a MIDI device that sends custom SysEx messages to control a specific parameter on another device, or design a device that can receive and process SysEx messages to update its internal settings.

Giordi concludes by setting the deadline for the assignment and encouraging the students to ask questions and seek help if needed.

PROFESSOR GIORDI
(continuing)
Your assignment is due next week, and I'll be available during my office hours if you need any assistance or have questions. Remember, the goal is to explore these advanced MIDI features and apply them to your projects. I can't wait to see what you all come up with!

The students leave the classroom, excited to tackle their new assignment and eager to experiment with MIDI clock synchronization and SysEx messages in their projects.

FADE OUT.