UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria dell'Informazione e
dell'Automazione

Analisi ed implementazione del protocollo MIDI MPE per il controllo di strumenti musicali virtuali

The MIDI MPE protocol for the control of virtual musical instruments: analysis and implementation

Candidato:
Giacomo Di Giovanni

Relatore:
Prof. SQUARTINI Stefano

Correlatore:
Prof. GABRIELLI Leonardo

Anno Accademico 2020-2021

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria dell'Informazione e
dell'Automazione

Analisi ed implementazione del protocollo MIDI MPE per il controllo di strumenti musicali virtuali

The MIDI MPE protocol for the control of virtual musical instruments: analysis and implementation

Candidato:
Giacomo Di Giovanni

Relatore:
Prof. SQUARTINI Stefano

Correlatore:
Prof. GABRIELLI Leonardo

Anno Accademico 2020-2021

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria dell'Informazione e dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

A mamma e papà

Sommario

La capacità di potersi esprimere attraverso strumenti elettronici non è mai stata paragonabile a quella degli strumenti fisici. Tuttavia tramite lo sviluppo di alcuni dispositivi e protocolli, come la Roli Seaboard e la specifica MPE del protocollo Midi, c'è stato un grande avanzamento nel campo dell'espressività. Unendo il tutto ad un software di sintesi modulare virtuale: VCV rack, l'espressività percorre un'inifnità di strade uniche tra loro. La tesi mostrerà i vari processi di *analisi* del protocollo, *implementazione* dello stesso, ricostruire sullo *strumento virtuale* un modello per emulare la sintesi audio di una tastiera già presente sul mercato e *validazione* del modello tramite spettrogrammi.

Indice

Capitolo 1

Introduzione

L'interesse parallelo per la musica e l'informatica sono i genitori di questo progetto, che nasce per sfruttare strumenti elettronici a favore della creatività dei musicisti. Il progetto si basa sul protocollo MIDI e sull'utilizzo di un software di sintetizzazione audio, che nel corso della tesi verrano analizzati e verrà spiegata la loro necessità. Questi due elementi consentiranno di sviluppare un software capace di far trasparire, nella sintetizzazione audio, l'espressività del musicista. Una volta costruito il software, verrà affrontata una parte di Sound Design che è stata utilizzata per creare una catena di sintetizzazione. Infine verrà spiegato il processo di validazione del software e della catena di sintetizzazione.

1.1 Obiettivo

L'obiettivo principale è creare un software capace di utilizzare il protocollo MIDI MPE, usato da alcuni strumenti elettronici, per comunicare al programma di sintetizzazione audio i parametri espressivi.

1.2 Strumenti

Per sviluppare il programma è stato necessario l'uso dei seguenti software:

IDE VS Code

Version Control GitHub

Debugger gdb

Ambiente di Compilazione MinGW

Editor Grafico Inkscape

Software di calcolo MATLAB

Capitolo 2

Protocollo Midi

Colonna portante del progetto è il *Protocollo Midi*, fu ideato nel 1981 e presentato nel 1983 al fine di codificare i gesti esecutivi che il musicista compie quando suona una tastiera elettronica sensibile al tocco e trasmetterli in tempo reale ad altri dispositivi [4]. Nel 2018 la Midi Manufacturers Association ha ufficialmente adottato la specifica del Midi MPE, che analizzeremo nel dettaglio nel capitolo 2.2. Il protocollo si basa sulla comunicazione di segnali digitali per comunicare le seguenti informazioni:

- Quale nota è stata premuta
- Il momento in cui viene premuta o lasciata
- Quanto forte è premuta (Velocity)
- Quanto varia la pressione sul tasto (Aftertouch)
- Vibrato (Modulation Wheel)
- Pitch Bend

Tutte queste informazioni vengono incapsulate in quello che si chiama *Midi Message*

2.1 Standard Midi

Figura 2.1: Una Tastiera Midi

Alla sinistra è presente un esempio di Tastiera Midi, basandosi su questo protocollo la tastiera trasmette le informazioni elencate in precedenza e non elabora un segnale audio. Per produrre un suono dalla tastiera midi bisogna connetterla ad un'interfaccia capace di comprendere i segnali midi e trasformarli in un segnale audio. Per fare ciò bisogna innanzitutto comprendere il protocollo a basso livello.

2.1.1 Informazioni Generali

Qualsiasi Interfaccia Midi ha un hardware che consente di mandare messaggi asincroni, che sono composti da un bit di inizio, 8 bits per i dati e, infine, un bit di stop. Si hanno 10 bits in totale, per un periodo di circa 320 microsecondi. Lo start bit è sempre 0 mentre lo stop bit è sempre 1. I bit a seguire vanno dal LSB al MSB. [1].

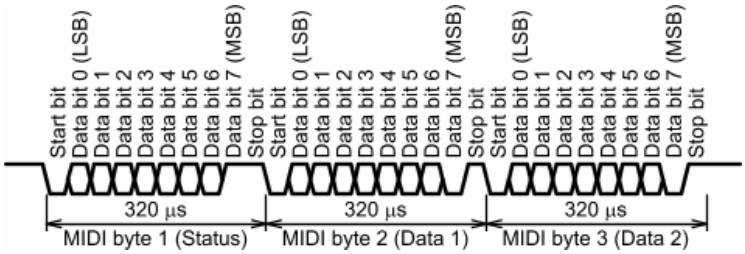


Figura 2.2: Messaggio Midi Generale

2.1.2 Tipi di Message

Il protocollo delinea due macro categorie di messaggi: *Channel Message* e *System Message*.

Channel Message

Un Channel Message usa quattro bits nello Status Byte per indirizzare il messaggio in uno dei sedici canali Midi e quattro bits per definire il messaggio. Possono esserci due tipi di Channel Message: *Voice* e *Mode*.

System Messages

Possono essere presenti tre tipi di System Message: Common, Real-Time e Exlusive, tuttavia per il progetto in esame non è stato necesario approfondire questo tipo di messagio, il quale trova ampia descrizione su midi.org [1].

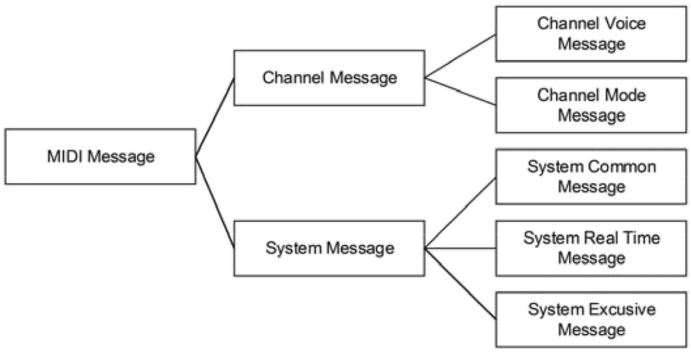


Figura 2.3: Diagramma Tipologie Messaggi Midi

2.1.3 Tipi di Byte

Nei Channel Message sono presenti due tipi di bytes: *Status Bytes* e *Data Bytes*.

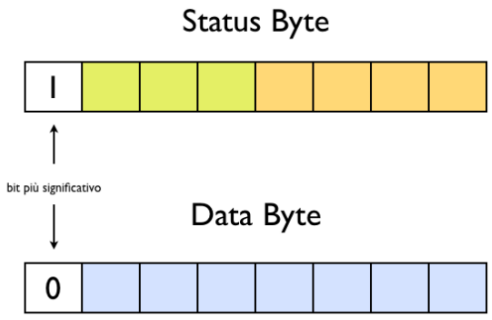


Figura 2.4: Status e Data a confronto

Gli **Status Bytes** sono composti da 8 bit, nei quali il Most Significant Bit è fissato ad 1. Il loro scopo è identificare il tipo di messaggio e comunicare il channel.
I **Data Bytes** seguono sempre uno Status Byte, possono esserci uno o due Data Byte nell'intero messaggio midi. Anche i Data Byte sono composti da 8 bit, nei quali il Most Significant Bit è fissato a 0.

Il Most Significant Bit è necessario per mantenere una macchina a stati coerente nel lato recezione.(Esempio: Dopo uno Status di Note ON, mi aspetto due Data Bytes, se dopo il primo vedo un mesaggio di Status la macchina ricevente percepirà un errore). Dalla figura 2.4 si può notare una suddivisione in caselle di Status e Data Byte, dove ogni casella rappresenta un bit. Nello Status Byte è presente una divisione in due nibble, dove il primo ha un bit fisso a 1 e quindi rimangono 2^3 modi di indicare uno Status, il quale inizia necessariamente da 1000, ovvero 8 nel sistema esadecimale, per arrivare a 1111 cioè F. Mentre abbiamo un intero nibble (quindi pari a 2^4) per indicare il channel a cui è riferito lo status, quindi da 0 a 15. Nel Data Byte invece non abbiamo suddivisioni, per cui il data può variare da 0 a 2^7, cioè 127.

Byte	Binario	Esadecimale	Decimale
Status Byte Min	**1**000	8	8
Status Byte Max	**1**111	F	15
Channel Min	0000	0	0
Channel Max	1111	F	15
Data Min	**0**0000000	0	0
Data Max	**0**1111111	7F	127

Abbiamo dunque a disposizione 8 tipologie di messaggio (*Status*), 16 canali e una risoluzione di 2^7 bit.

2.1.4 Messaggi Esempio e Limiti

Prenderemo in Esempio tre tipologie di messaggio: *Note On*, *Pitch Bend* e *Channel Pressure*.

I dati nella tabella sono in binario

Tipologia	Status Byte	Data Byte 1	Data Byte 2
Note On	1001cccc	0nnnnnnn	0vvvvvvv
Pitch Bend	1110cccc	0vvvvvvv	0vvvvvvv
Channel Pressure	1101cccc	0vvvvvvv	non presente

Tabella 2.1: Tipologie di Messaggio

c bit utilizzati per indicare il channel

n bit utilizzati per indicare la nota

v bit utilizzati per comunicare un valore

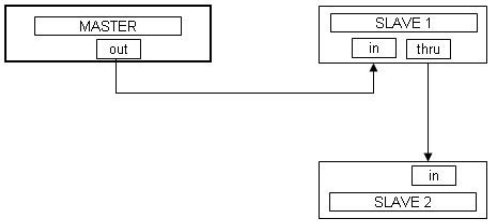


Figura 2.5: Master e slaves

Spesso i controller midi sono dotati di due ruote espressive (figura 2.6): *Pitch Wheel* e *Modulation Wheel*. Queste permettono di inviare dei messaggi midi per modificare il suono. La prima invia delle informazioni riguardo la modifica dell'intonazione delle note suonate, la seconda può assumere diverse funzioni in base alla scelta del parametro, spesso si usa per variare la frequenza del LFO nella catena del sintetizzatore. Un approfondimento sui sintetizzatori verrà affrontato nel capitolo 4.2.

Figura 2.6: Pitch e Mod Wheels

Come si può vedere dalla tabella 2.1, i messaggi per il Pitch Bend e il Channel Pressure non inviano alcun Data Byte per indicare la nota a cui sono riferiti. In più il pitch bend anzichè usare un Data Byte per la nota e un Data Byte per il valore, utilizza i due byte per avere una risoluzione più alta (2^{14}), il Channel Pressure invece prevede solo un Data Byte. Questi parametri dunque agiscono su tutte le note attive in egual modo, ovvero in maniera **monofonica**, comunicando le informazioni sul channel 0. Inoltre i bit per indicare il channel, nel Midi Standard, sono utilizzati per sfruttare più strumenti alla volta (disposti in *Daisy Chain*) detti slave (figura 2.5), con un singolo controller, detto master. Il master scrive su molteplici channel e ogni

slave legge un channel diverso dagli altri. Ogni slave, tramite un output chiamato **Thru**, inoltra i messaggi agli slave successivi. Questo, insieme alle precedenti motivazioni comporta delle notevoli limitazioni a livello espressivo. Per evitare ciò, la Midi Association ha adottato una nuova specifica (Midi MPE, capitolo 2.2) che è implementata in nuove interfacce capaci di inviare parametri espressivi in modalità **polifonica**.

2.2 Midi MPE

La specifica del **MIDI MPE** è progettata per i controller Midi che permettono a chi li suona di variare individualmente il *pitch* e il *timbre* di note singole. In molti di questi controller, il pitch varia con un movimento laterale su una superficie continua, mentre la variazione individuale del timbre è dovuta alla variazione di pressione oppure spostando le dita in avanti o indietro, rispetto al musicista [2].

Figura 2.7: Seaboard by Roli

La seaboard in figura 2.7 dell'azienda Roli è un ottimo esempio di controller che utilizza a pieno la specifica MPE, avendo una superficie continua il musicista riesce ad aumentare l'espressività musicale eseguendo un vibrato attraverso movimenti laterali diversi per ogni nota oppure l'aftertouch polifonico sfruttando la pressione sulla superficie.

Un altro esempio di controller MPE è in figura 2.8, il continuum fatto dall'azienda Haken Audio. É stato uno dei primo controller ad implementare un controllo tridimensionale, ovvero che lo spostamento lungo un asse del dito provoca la variazione di un parametro. Ai tempi l'MPE non era ancora realtà, ma evidenzia come il mercato si stesse già muovendo verso le necessità dei musicisti.

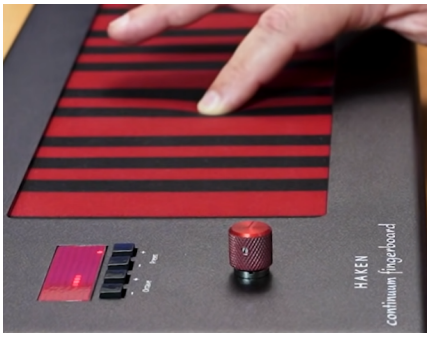


Figura 2.8: Continuum Fingerboard by Haken

2.2.1 Caratteristica Midi MPE

Essendo il Midi MPE una specifica, essa si basa comunque sul protocollo Midi, con una differenza sostanziale: **ad ogni nota suonata viene assegnato un proprio canale**. Questo permette di avere pitch bend dedicati ad ogni nota e in più viene semplificata la gestione dell'aftertouch polifonico. Si potrebbe pensare che questo

limiti la polifonia a 16 note, in realtà esauriti i 16 channel a disposizione, la specifica assegna più note su uno stesso channel, quindi la quantità di note che possono essere utilizzate è considerevole. La limitazione è nell'espressione polifonica, dato che i parametri espressivi vengono comunicati sui channel; se su un channel fossero presenti due o più note, una modifica al pitch coinvolgerebbe tutte le note su quel channel.

2.3 Parametri D'Interesse

Non tutte le funzioni del Protocollo Midi sono necessarie allo sviluppo di questo progetto, per cui verranno presentati qui di seguito gli Status primari da cui verranno estrapolati i parametri utilizzati nel progetto.

Status Byte	Valore	Significato
Note ON	0x 9	Indica la pressione di un tasto
Note OFF	0x 8	Indica il rilascio di un tasto
Control Change	0x B	Indica i valori di manopole o altri parametri
Pitch Bend	0x E	Indica la modifica di intonazione
Poly Pressure	0x A	Indica il valore di pressione una volta premuto il tasto

2.3.1 Parametri Midi Standard

Nel progetto ci interessano i seguenti parametri del Midi Standard

Parametro	Significato
Velocity	Indica l'intensità della pressione iniziale
Pitch Bend	Indica il movimento della ruota espressiva associata
Modulation Wheel	Indica il movimento della ruota espressiva associata
Aftertouch	Indica l'intensità di pressione dopo la pressione iniziale
Release Velocity	Indica la velocità di rilascio del tasto

2.3.2 Parametri MPE

I parametri MPE possono essere molteplici, per cui nel progetto sono stati implementati i parametri della Seaboard Roli usando i loro rispettivi nomi.

Parametro	Significato
Strike	Indica l'intensità della pressione iniziale
Glide	Indica il movimento laterale delle dita
Slide	Indica il movimento verso avanti o verso sè delle dita
Press	Indica l'intensità di pressione dopo la Strike
Lift	Indica la velocità di rilascio del tasto

2.3.3 Lettura Dati

Una volta individuati i parametri d'interesse è necessario estrapolarli dai messaggi. Siccome si ha una corrispondenza tra i parametri Midi Standard e Midi MPE, proseguirà solamente come si leggono i dati Midi MPE e dove necessario i dati Midi Standard.

Strike

Il valore *Strike* viene estrapolato dal secondo Data Byte dello Status "Note ON".

Esempio Esadecimale:
9n 3C **5F**

Questo è un messaggio che ha uno Status Note ON, su un channel 'n' che ci da le seguenti informazioni:

- Il primo Data Byte è 3C, dunque la nota a cui è riferito il Note On è il Do Centrale (che corrisponde al valore 60 in decimale)
- Il secondo Data Byte indica il valore Strike è 5F che corrisponde a 95

Glide

Il valore *Glide* viene estrapolato dai due Data Bytes dello Status Pitch Bend.

Esempio Esadecimale:
En **09 40**

Con il messaggio di Status Pitch Bend, su un channel 'n', bisogna però eseguire delle operazioni di *bit shift*. In quanto il secondo Data Byte porta l'informazione del Most Significant Byte. Quindi per ricostruire l'informazione è necessario prendere l'informazione del secondo Data Byte (40), fare un left shift di 7 bit (diventa 2000) ed effettuare la somma del primo Data Byte, per cui abbiamo 2009, che corrisponde in decimale a 8201, un valore poco sopra il valore di mezzo (8192) dato che si possono avere pari valori in negativo e in positivo in quanto il Pitch Bend ha una risoluzione di 16384.

Slide

Il valore di *Slide* viene estrapolato da uno Status "Control Change".

Esempio Esadecimale:
Bn 4A **5D**

Nei messaggi di tipologia Control Change (Status B) il primo Data Byte ci dice da che Controller viene il messaggio, nel caso dello Slide è il 4A (74 in decimale). Il secondo Data Byte ci da invece informazioni sul valore.

Press

Il valore *Press* viene estrapolato dai messaggi con Status Poly Presure.

Esempio Esadecimale:
An 3C **5F**

Il funzionamento è simile alla tipologia Note On, viene indicato il channel di riferimento, la nota a cui è riferito i messaggio nel primo Data Byte e il valore nel secondo Data Byte.

Lift

Il valore *Lift* viene estrapolato dai messaggi con Status Note OFF.

Esempio Esadecimale:
8n 3C **5F**

Ha esattamente la funzione contrapposta dello Strike. Troviamo la nota sul primo Data Byte e il valore sul secondo.

Modulation Wheel

Il valore della *Modulation Wheel* viene mandato tramite un Control Change.

Esempio Esadecimale:
An 01 **5F**

Per la Modulation Wheel il controller è 01, il valore è dato dal secondo Data Byte.

2.4 NRPN

Il protocollo MIDI inlcude comandi che soddisfano quasi tutti i bisogni dei diversi controlli musicali. Tuttavia ci sono alcuni strumenti che hanno dei controller particolari, per gestirli il protocollo MIDI ha introdotto i **Non-Registered Parameters**,(*NRPNs*([3]. Il loro funzionamento sfrutta i messaggi di Control Change, utilizzando due Controller per definire il Controller stesso e altri due per definire il Valore del messaggio.

Prendendo in esame un esempio abbiamo quattro messaggi (Esadecimale):

- Bn 63 XX
- Bn 06 YY
- Bn 62 xx
- Bn 26 yy

Il primi due indicano il parametro rispettando l'ordine XXxx, mentre gli ultimi due comunicano il valore sempre nell'ordine YYyy. Per comporre i byte ovviamente è necessario effettuare il bitshift del MSB XX e del MSB YY, aggiungendo rispettivamente i LSB xx e yy.

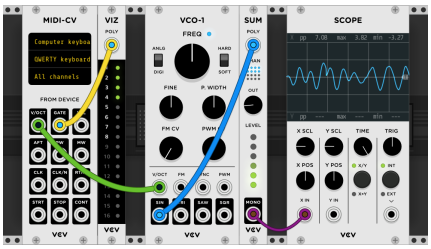
Capitolo 3

VCV Rack

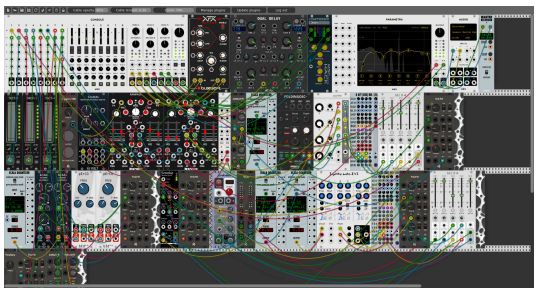
VCV Rack è un programma di sintesi modulare virtuale, è una riproduzione software del sintetizzatore modulare hardware **Eurorack** (figura 3.1).

Figura 3.1: Esempio di sintetizzatore Eurorack

Essendo VCV Rack un software gratis Open-Source il vantaggio rispetto alla versione HW è subito evidente: il sintetizzatore può essere esteso senza limiti di budget. Tuttavia è necessario, proporzionalemente alla grandezza della catena di moduli, avere un calcolatore che sappia sostenere le computazioni necessarie. Possiamo infatti avere della catene molto semplici come quella in figura 3.2a che riproducono sintetizzatori semplici. Dando spazio alla fantasia è invece possibile concatenare svariati moduli per dare frutto a sintetizzatori unici, un esempio di ciò che si può arrivare ad ottenere è in figura 3.2b.



(a) Catena di moduli semplice



(b) Catena di moduli complessa

Figura 3.2: Due catene di moduli

3.1 Introduzione al software

Alla prima apertura del software ci troveremo davanti alla schermata introduttiva (figura 3.3) dove sono presenti diversi "blocchi" che chiameremo *"moduli"*. Ogni modulo interagisce con gli altri attraverso segnali **analogici virtuali**. Possiamo fare una classificazione approssimativa delle varie tipologie di questi segnali.

- **Audio** questi segnali possono essere uditi dall'utente.
- **CV** (Control Voltage) segnali che possono modulare parametri di altri moduli.
- **1V/Oct** sono dei segnali CV che rappresentano un pitch o una nota. 1V corrisponde ad un'ottava.
- **Gate** Utilizzato come segnale di ON/OFF. (0V = OFF e 10V = ON)
- **Trigger** sono segnali ON/OFF ma estremamente brevi (circa 1 millisecondo) Da non confondere con il Gate. Utilizzati per causare un evento, come il colpo di una percurssione.
- **Clock** sono dei trigger, ma con una cadenza temporale regolare. Utilizzati per regolare il tempo musicale della sessione.

I segnali possono essere trasmessi tra i moduli tramite cavi virtuali, il software utilizza il Volt come unità di misura. Il range è di 10V *peak-to-peak*. Nel codice sarà semplicissimo utilizzare questa unità di misura, successivamente verrà mostrato.

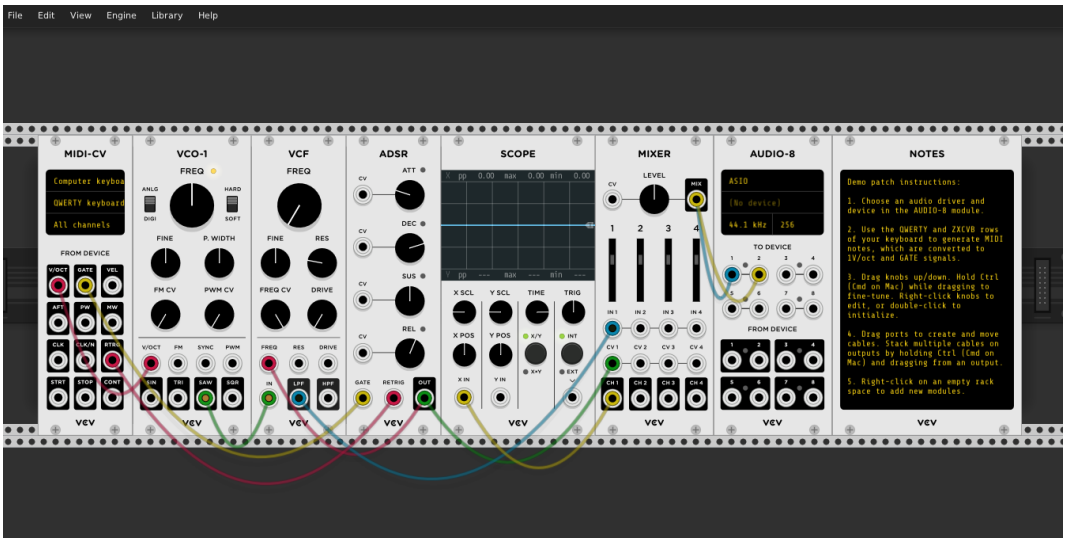


Figura 3.3: Schermata Iniziale

3.2 Sviluppo Plugin

In VCV Rack il concetto più importante da apprendere prima di iniziare lo svilippo software è la differenza tra *plugin* e *module*. Il primo è un raccoglitore di tanti module, la tesi riguarda lo sviluppo di un singolo module.

3.2.1 Requisiti

I macro requisiti del *module* possono essere sintetizzati come segue:

- Interagire con i dispositivi Midi connessi
- Fornire in output i parametri di V/Oct e Gate
- Fornire in output i parametri Midi MPE
- Fornire in output il parametro NRPN selezionato

Il secondo requisito classifica il module nella categoria *Midi/CV* ovvero convertire i messaggi midi in segnali CV (Controlled Voltage). Inoltre per sfruttare la polifonia ogni parametro, riferito ad un output, verrà dichiarato come vettore con dimensione pari a 16. Questo per garantire che ogni outpute abbia i suoi 16 canali gestibili indipendentemente.

Esempio:
Dichiarazione di un vettore su cui vengono salvate le note
`uint8_t notes[16]`

3.2.2 Implementazione

Figura 3.4: Widget Midi

Dispositivi Connessi

É possibile poter osservare i dispositivi connessi (figura 3.4) dichiarando un widget di tipo MidiWidget (`Widget::MidiWidget* midiWidget`) per poi caricare il widget sul module (`createWidge<MidiWidget>`), sarà così possibile scegliere il Midi Driver e successivamente i dispositivi Midi disponibili del driver. Dichiarando invece due variabili di tipo `midi::InputQueue midiInput` e `midi:Message msg` è possibile incapsulare in `midiIput` i messaggi che vengono ricevuti e tramite il metodo ereditato `midiInput.shift(&msg)` ritroveremo su `msg` un singolo messaggio midi che possiamo interpretare con dell'altro codice. Il metodo che si occupa di ciò è `processMSG(msg)`.

Essendo la variabile msg di tipo Message eredita quattro metodi essenziali.

- `getStatus()` restituisce in valore esadecimale lo **status** del messaggio
- `getChannel()` restituisce in valore decimale il **canale** del messaggio
- `getNote()` restituisce in decimale il valore del **primo Data Byte**
- `getValue()` restituisce in decimale il valore del **secondo Data Byte**

V/Oct e Gate

Come già anticipato, VCV Rack basa i suoi segnali virtuali su un range di 10 Volt. Il **V/Oct** è un tipo di segnale che per ogni incremento di un Volt si ha un incremento di un'ottava. Avendo un range di 10V, possiamo avere al massimo 10 Ottave. A causa della divisione in intervalli musicali occidentale, abbiamo 12 note all'interno di un'ottava. Dal protocollo midi sappiamo che il Data Byte che ci indica la nota ha un range di 128 note (da 0 a 127). Utilizzando i metodi ereditati `msg.getNote()` e `msg.getChannel()` riusciamo ad assegnare alla variabile notes il numero di nota tramite la seguente assegnazione: `notes[msg.getChannel()] = msg.getNote()`. Se posizioniamo al centro (0V) la nota 60 (C4) e dividendo i valori Midi per 12, avremo una divisione in V/Oct di ogni nota.

$$Voltage = \frac{notes[channel] - 60}{12} \tag{3.1}$$

Per il **Gate**, essendo un segnale binario, avremo un vettore di tipo `bool gates[16]` e ogni qual volta arriva un messaggio di note ON attiveremo il gate sul rispettivo canale(`gates[msg.getChannel()] = true`), mentre verrà disattivato ai segnali di NoteOFF (`gates[msg.getChannel()] = false`).
Per quanto riguarda la conversione in voltaggio, l'operazione è semplice:

$$Voltage = \texttt{gates[channel] ? 10 : 0}$$

Se è true assegna il valore 10 altrimenti assegna il valore 0.

Infine per fornire i valori assegnati di voltaggio in output, vanno dichiarate le variabili di tipo output come enumeratori

```
enum OutputsIds {
    VOCT,
    GATE,
      }
```

Per poi sfruttare il metodo ereditato per settare il votlaggio dell'output.

```
outputs[VOCT].setVoltage(Voltage, channel)
```

Qui viene passato un generico Voltage al metodo, in realtà va sostituita l'espressione 3.1.

Midi MPE

I parametri espressivi da fornire in output sono:

- Strike
- Press
- Glide
- Slide
- Lift

Ognuno di questi parametri deriva da messaggi con un particolare status. Per poter interpretare il messaggio correttamente, il parametro `msg` viene passato alla funzione `processMSG(msg)`, all'interno di questa funzione troviamo il costrutto `switch(msg.getStatus())`, quindi analizzerà gli status di ogni singolo messaggio, distinguendo i vari *case*.
Per il parametro **Strike** ci troveremo nel case `0x9` in quanto questo dato si estrae dal messaggio di NoteON e verrà effettuato un assegnamento alla variabile vettoriale in questo modo:

```
strike[msg.getChannel()] = msg.getValue()
```

Questa tipologia di assegnamento è ricorrente in tutti i parametri espressivi, l'unico che trova differenze è il Glide. Per quanto riguarda il **Glide**, che ha status `0xe`, è neccsario effettuare un bitshift perchè i due Data Bytes del messaggio trasportano il primo il LSB e il secondo il MSB dell'intero dato. Quindi dichiariamo la variabile Glide come un intero senza segno dal valore di 16 bit:
`uint16_t glide[16]`
E l'assegnamento del valore avverà in questo modo:

```
glide[msg.getChannel()] = (uint16_t) msg.getValue() << 7 | msg.getNote()
```

in pratica: si prende il Value che corrisponde al LSB del dato, essendo un dato a 8 bit, si esegue il casting da uint8_t a uint16_t, se esegue poi un bitshift di 7 bit, infine si aggiunge a questo valore l'informazione del primo Data Byte: msg.getNote().

Per la conversione di queste informazioni in voltaggio si usa la seguente formula generale:

$$Voltage = yMin + \frac{X - xMin}{(xMax - xMin)(yMax - yMin)} \tag{3.2}$$

Dove:

X Valore del parametro

xMin il valore minimo possibile in entrata

xMax il valore massimo possibile in entrata

yMin il valore minimo possibile in uscita

yMax il valore massimo possibile in uscita

NRPN

Per la gestione del parametro NRPN è nessaria la suddivisione in due sottoproblemi:

- acquisire l'informazione dall'utente di quale parametro si vuole utilizzare all'interno di VCV Rack
- fornire in output il parametro

Per la prima parte è stato inserito uno slider orizzontale, che modifica il parametro NRPNDATA. Qusando quest'ultimo viene configurato all'avvio del modulo, vengono impostati il valore minimo e il massimo, che in questo caso sono 0 e 16383. Successivamente a runtime, il valore viene assegnato alla variabile `nrpnParam` come segue:

```
nrpnParam = (int) params[NRPNDATA].getValue()
```

questo per evitare l'assegnamento di parametri con la virgola che lo slider prevede, ma non necessari per la sua funzione nel modulo.

Per il **secondo requisito** era necessario ricomporre il messaggio ricevuto. Questo è stato possibile attraverso una macchina a stati finiti (figura 3.5).

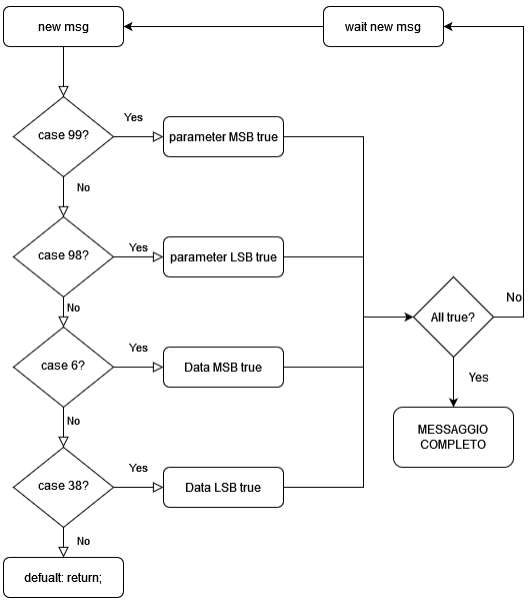


Figura 3.5: Macchina a stati finiti

Sono presenti 4 stati di controllo, che al loro interno modificano una propria variabile booleana, dopo la visita in uno stato si controlla nel quinto stato se tutte le variabili sono vere, nel caso sono vere il messaggio è completo, altrimenti bisogna aspettare che sia completo. Se non si rientra in uno dei 4 stati di controllo si esce dalla macchina a stati.

Capitolo 4

Sound Design

Il **design del suono** è l'arte e la pratica di creare tracce sonore per molteplici necessità. Il design del suono comunemente include la produzione e la modifica di audio precedentemente composti o registrati, come effetti sonori e dialoghi, ma può anche comportare la **creazione di suoni da zero** attraverso i **sintetizzatori**. [5]. Per quanto riguarda la realizzazione è necessario comprendere il funzionamento dei sintetizzatori e dei loro parametri fondamentali, per poi comprendere la loro suddivisione in moduli.

4.1 Obiettivo

In fase di progettazione era necessario testare le funzioni del plugin sviluppato. Per cui sono state registrate delle tracce audio con due tastiere: *ASM Hydrasynth* e *Seaboard Roli*. Per la prima sono state registrate due clip audio e salvati i corrispettivi file midi, mentre è stata registrata solamente una clip audio e rispettiva traccia midi per la Seaboard della Roli. Successivamente tramite due software: *Loop MIDI* e *MIDI OX* la traccia midi veniva reindirizzasta su VCV Rack per essere convertita in segnali CV dal plugin sviluppato.

4.2 Componenti Sintetizzatore

Il suono di qualsiasi strumento può essere caratterizzato tramite la **Sintesi sottrattiva**. Ovvero a partire da un segnale ricco di armoniche si interviene con una serie di filtri per modificare timbro e forma d'onda. Molto spesso all'interno di un sintetizzatore questo processo è standard, ci sono suoni già predefiniti; questa tipologia prende il nome di *sintetizzatori normalizzati*. É invece importante capire il funzionamento dei **sintetizzatori modulari**, ovvero ogni modulo controlla singolarmente un parametro di sintesi.

I moduli *essenziali* possono essere racchiusi in questo elenco:

- VCO (Voltage Controlled Oscillator)
- VCF (Voltage Controlled Filter)
- VCA (Voltage Controlled Amplifier)
- LFO (Low Frequency Oscillator)
- ADSR (Attack Decay Sustain Release)

Questi elementi sono il core di qualsiasi sintetizzatore modulare, analizzeremo il loro funzionamento all'interno del software di sintesi modulare virtuale VCV Rack. Vale però una regola per tutti i moduli:

Gli output usualmente hanno un background nero.
Gli altri sono ingressi di ***controllo CV****.*

4.2.1 VCO

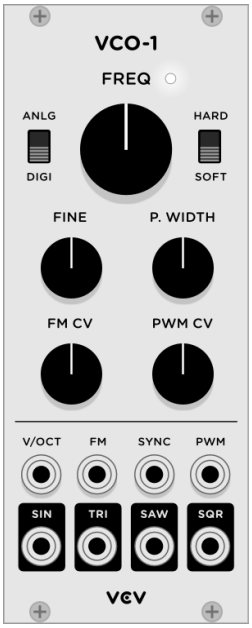
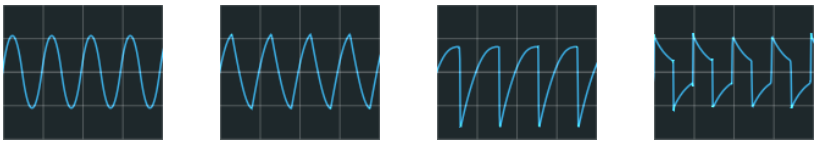


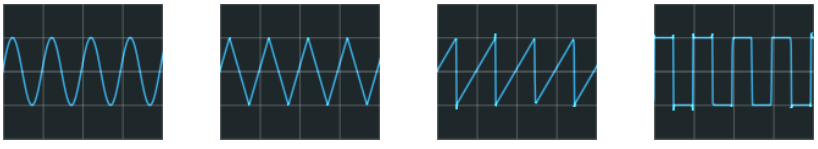
Figura 4.1: VCO by VCV Rack

L'oscillatore controllato in tensione è un dispositivo elettronico con lo scopo di produrre un'oscillazione periodica, il cui periodo, e di conseguenza la frequenza, è determinato dal valore assoluto di una **tensione applicata ad un ingresso di controllo**[6]. Guardando al modulo VCO di VCV Rack possiamo notare alcuni parametri fondamentali. Il modulo è composto da varie manopole, quella fondamentale nominata **FREQ** regola il valore della frequenza di oscillazione. Questo valore è fisso a meno che non viene inserito un *Cavo Virtuale* nell'ingresso nominato **V/OCT**. In questo caso la frequenza varia in base al voltaggio presente sul rispettivo ingresso. É possibile sfruttare l'ingresso **FM** per effettuare ulteriori modifiche all'intonazione (frequenza di output), con intensità regolate dalla manopola denominata **FM CV**. Per esempio collegando a tale ingresso un LFO, che oscilla a frequenza basse, è possibile ricreare un effetto di vibrato. Abbiamo infine le quattro tipologie di **output**, le quali in base alla modalità analogica o digitale del plugin, forniscono quattro tipologie di onde.



(a) Sin (b) Tri (c) saw (d) sqr

Figura 4.2: Segnali Analogici



(a) Sin (b) Tri (c) saw (d) sqr

Figura 4.3: Segnali Digitali

4.2.2 VCF

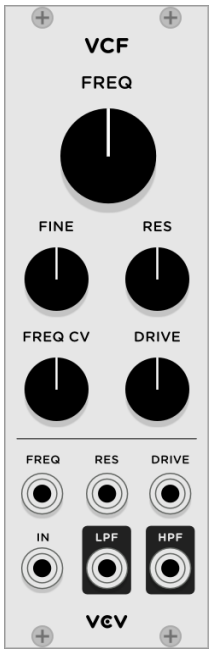


Figura 4.4: VCF by VCV Rack

Il VCF è un filtro che viene controllato tramite segnali elettrici (fisici o virtuali). É caratterizzato da tre fattori: Frequenza di taglio, fattore di risonanza e modalità di filtro (passa basso, passa alto, passa banda)[7].
Come nel VCO troviamo una manopola principale, questa regola la *frequenza di taglio*, ovvero la frequenza alla quale avviene il filtraggio. Questo parametro può essere controllato anche dall'input **FREQ** che modifica la frequenza di taglio. Troviamo poi sulla destra la manopola **RES**, aumenta o diminuisce il fattore di risonanza, nella figura 4.5 troviamo curve con identica frequenza di taglio, ma con diversi fattori di risonanza. Anche questo parametro può essere modificato da un opportuno ingresso CV.
Infine abbiamo due output, restituiscono il segnale dopo un filtro passa basso oppure passa alto.

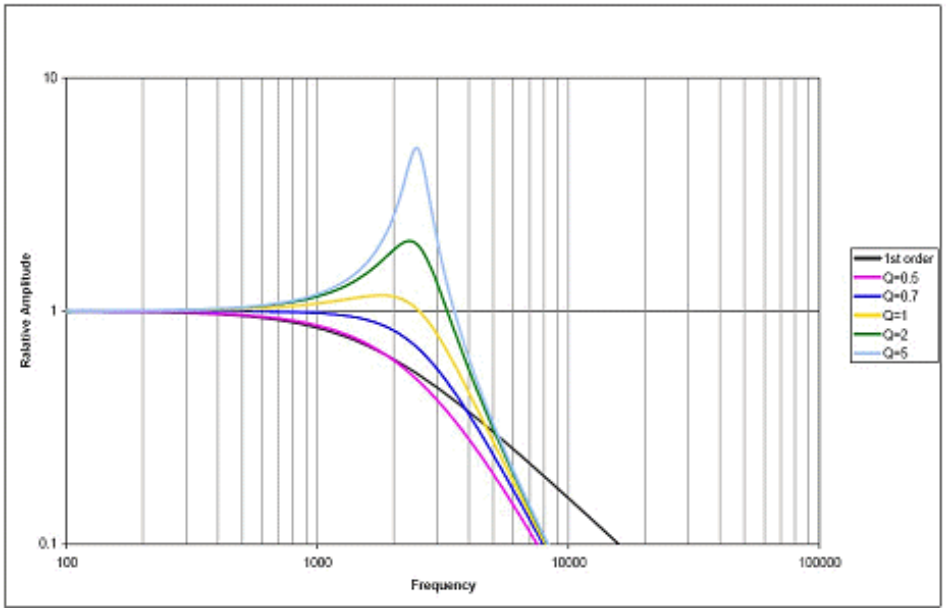
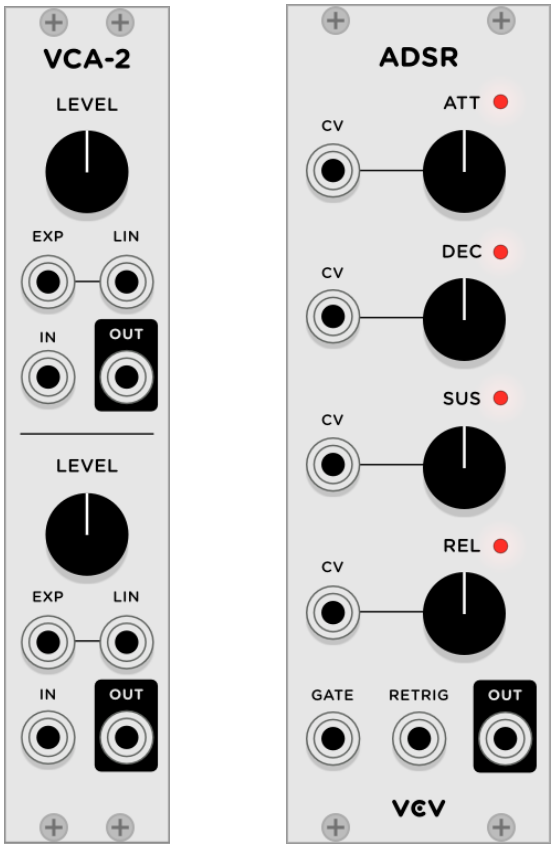


Figura 4.5: Curve generiche di di Filtri Passa Basso



(a) Sin (b) Tri

4.2.3 VCA

Il VCA è un modulo che consente di controllare il volume di un segnale in ingresso variando il volume in uscita.
Senza alcun input collegato può agire da amplificatore oppure attenuatore, spostando la manopola **LEVEL**.
Abbiamo un input e un output obbligatori che sono l'ingresso del segnale e l'uscita amplificata o attenuata.
Abbiamo poi due input di controllo che hanno lo stesso ruolo, modulare il livello. Quello a sinistra computa in maniera *esponenziale* mentre quello a destra in maniera *lineare*.
Usato insieme al modulo ADSR costituisce una parte fondamentale per l'*envelope* (figura 4.6) dei segnali.

4.2.4 ADSR

L'ADSR è quello che caratterizza tutte le fasi di un singolo suono, dal nascere al morire (figura 4.6).

Attack È il tempo che impiega il volume per passare da zero al suo valore massimo [8].

Decay Rappresenta il tempo che il suono impiega a passare dal volume massimo raggiunto durante la fase di attack al volume di sustain [8].

Sustain Volume che si mantiene dopo la fase di attack finché il tasto resta premuto [8].

Release Tempo che impiega il volume per arrivare al volume nullo.

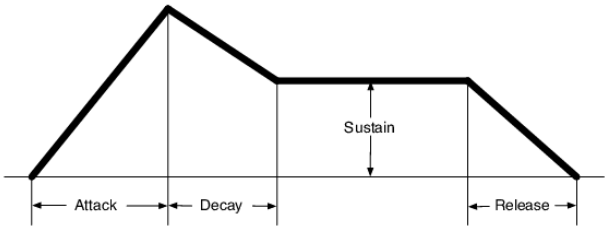


Figura 4.6: Grafico Temporale di un *Envelope*

4.2.5 LFO

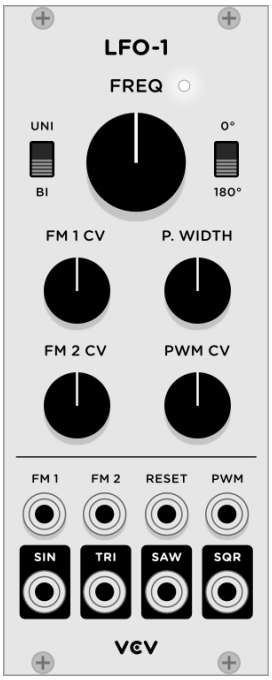


Figura 4.7: LFO Module by VCV Rack

A differenza dell'oscillatore VCO, che lavora a frequenza udibili (20 Hz - 20kHz), l'LFO è un oscillatore a frequenza infrasonica (al di sotto di 20Hz). Viene usato per modulare effetti, come per esempio modulare l'intonazione di una nota per simulare un vibrato, oppure modulare l'ampiezza per simulare un effetto tremolo. Il modulo VCV è caratterizzato da un range che va da 0.06 Hz a 1 kHz, regolabile dalla manopola **FREQ**. Oltre le regolazione di Frequency Modulation, utilizzabili anche con segnali V/Oct, verranno utilizzato i 4 output identici al modulo VCO (figura 4.3).

4.3 Catena Moduli

Per la validazione del progetto verrà analizzata la catena di moduli (Figura 4.8) progettata per riprodurre fedelmente una clip della ASM Hydrasynth creata in fase di sviluppo dal motore di sintesi proprietario.

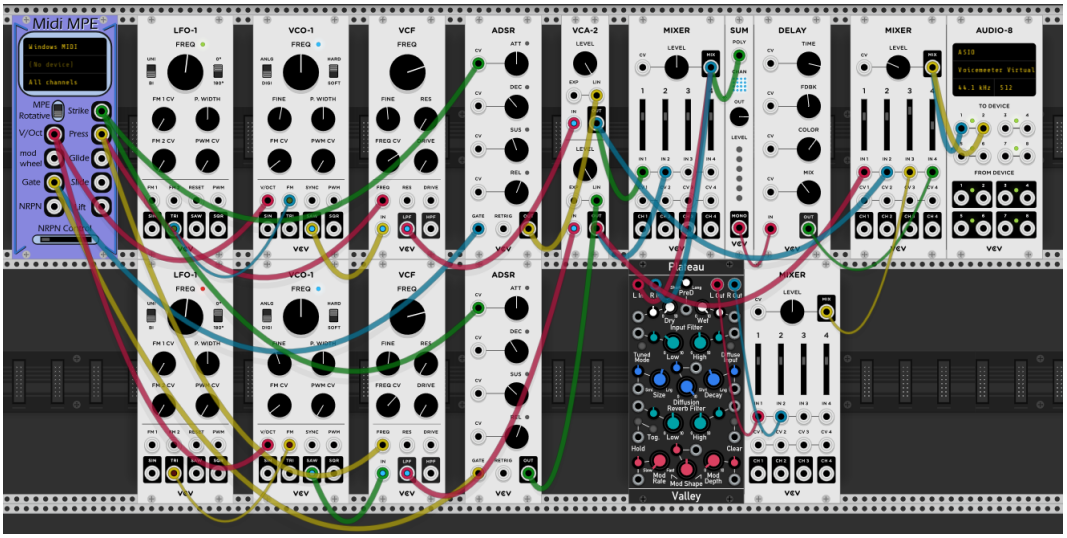


Figura 4.8: Catena di Moduli

La prima parte della catena è composta da quattro moduli sdoppiati in parallelo che costituiranno i due *side* della stereofonia. In ordine troviamo un LFO tarato a circa 2 Hz, che entra nel modulatore di frequenza del VCO con un segnale triangolare per ricreare l'effetto vibrato che si sente nella clip. Nel modulo VCO entra il segnale **V/Oct** del plugin sviluppato, al fine di variare la frquenza in base alla nota. Troviamo poi un VCF in configurazione passa basso con frequenza di taglio a 1300 kHz, con la possibilità di aumentarla con il segnale **Press**, e una risonanza del 0%. Troviamo poi un ADSR per definire l'envelope delle singole note, caratterizzato dal segnale di attack originario (**Strike**), un decay di 28 ms, sustain del 43% e release 197 ms. Infine il suono viene caratterizato da un delay e un riverbero.

Capitolo 5

Risultati

Per validare il plugin e la catena di moduli su VCV Rack è stato sviluppato uno script matlab per elaborare uno spettrogramma per compararlo con lo spettrogramma della clip originale e migliorare gli aspetti e le caratteristiche del sound design come frequenza di taglio, delay e tempo di caduta del riverbero. Una volta eseguite varie iterazioni di miglioramento confrontando i vari spettrogrammi con quello originale, si è arrivati ad un risultato finale soddisfacente.

5.1 Matlab Script

Per avere lo spettrogramma è stato utilizzato *Matlab* con la funzione `spectrogram`.

```
audiofile = 'FileAudioSelezionato.wav';

[y, Fs] = audioread(audiofile);
%%
x = y(:,1);
%%
spectrogram(x, blackman(1024), 128, 1024,Fs,'minThreshold',
-100, 'yaxis')
title('Spectrogram of signal')
```

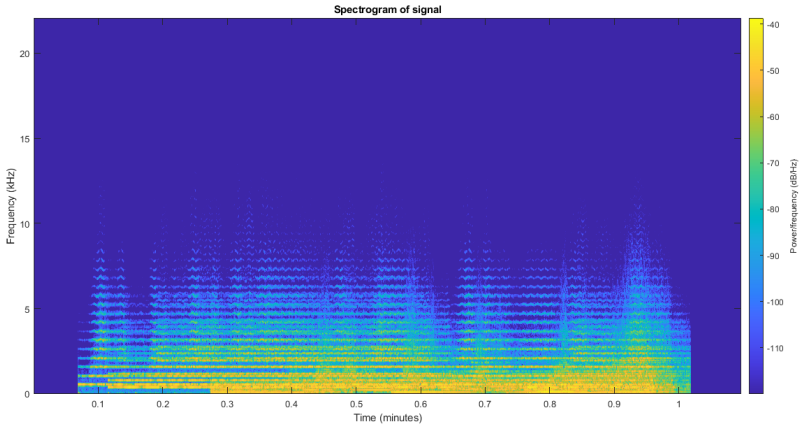
Sulla variabile *audiofile* viene salvato il file wav, che successivamente viene assegnato ad una matrice n*2, dove n è il numero di campioni presenti nel file audio (campionatmento a 44100 Hz, Fs). La presenza di due colonne è dovuta al fatto della stereofonia, successivamente eliminata e considerato solo uno dei due *side*. Infine viene graficato lo spettrogramma tramite il comando specifico.

5.2 Analisi Spettrogramma

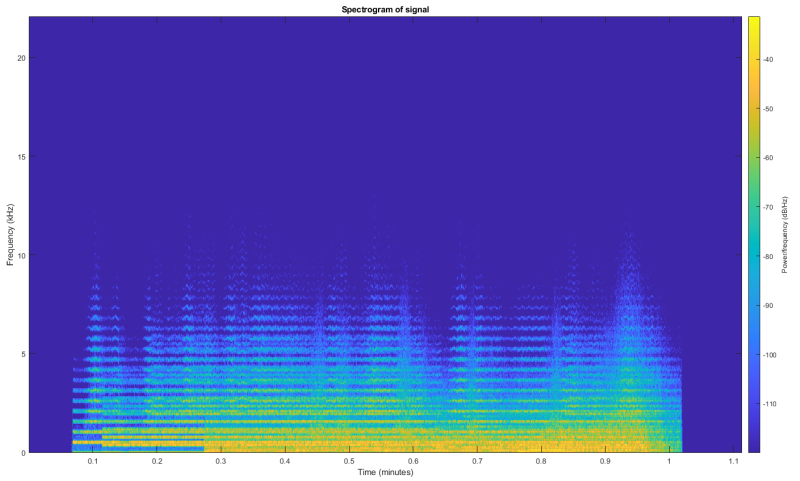
Per arrivare al risultato finale è stato necessario condurre 8 iterazioni, una volta eseguita l'iterazione 0, dove veniva costruita una catena grezza, le iterazioni consistevano in poche fasi:

1. Analisi differenze tra gli spettrogrammi
2. Comprensione errori
3. Studiare la soluzione
4. Modificare i parametri VCV Rack
5. Graficare lo spettrogramma

Da un'analisi visiva complessiva i due spettrogrammi sono molto simili, validando complessivamente il lavoro di sviluppo del plugin e del sound design.



(a) Spettrogramma clip originale

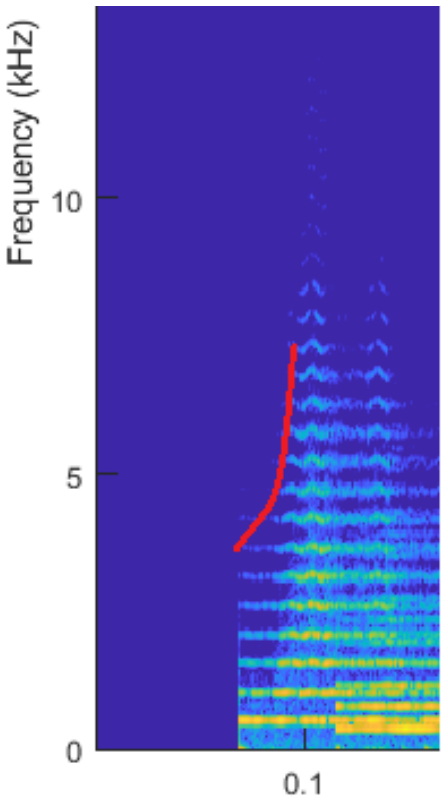


(b) Spettrogramma VCV Rack

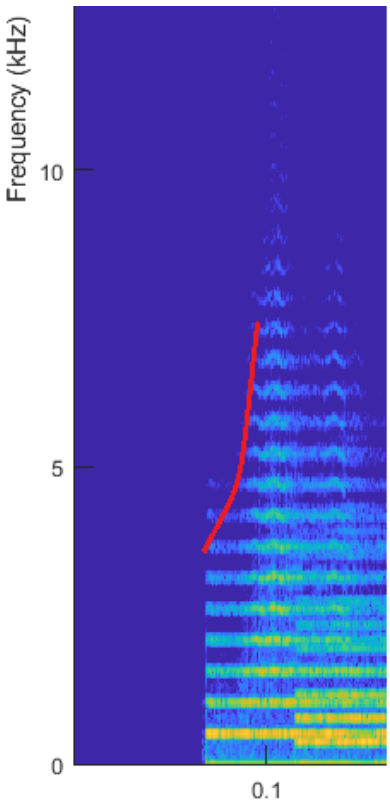
Figura 5.1: I due spettrogrammi a confronto

5.2.1 Analisi Frequenza di taglio

Nella prima parte dello spettrogramma è possibile notare l'effetto della frequenza di taglio tramite l'intensità e il numero di armoniche e della sua rispettiva modulazione visualizzando una curva per punti.



(a) Dettaglio Frequenza di Taglio Originale



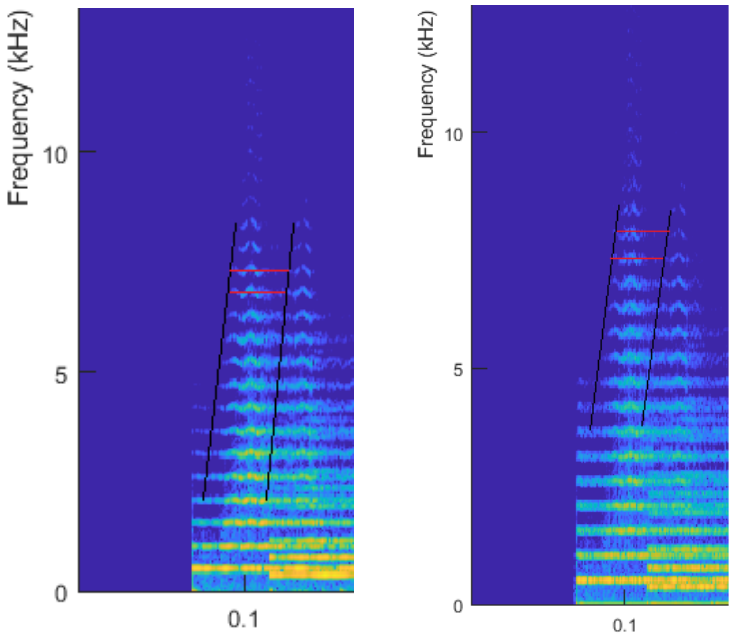
(b) Dettaglio Frequenza di Taglio VCV Rack

Figura 5.2: I due dettagli a confronto

Dalla figura 5.2a si può notare come le armoniche della frequenza fondamentale diminuiscono d'intensità fino alla sesta, mentre nella riproduzione su VCV Rack le armoniche sono più intense e presenti fino alla nona circa. Nonostante questa piccola differenza, la curva di modulazione risulta comunque identica, validando la funzionalità del parametro **Press** utilizzato per modificare la frequenza di taglio nel parametro.

5.2.2 Analisi Tempo di Delay

Considerando sempre il tratto iniziale dello spettrogramma, misurando la distanza tra due ripetizioni visibili di un armonica è possibile validare la configurazione del tempo di delay.



(a) Dettaglio Tempo Delay Originale

(b) Dettaglio Tempo Delay VCV Rack

Figura 5.3: I due dettagli a confronto

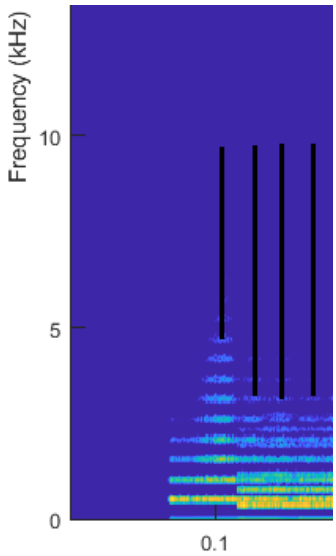
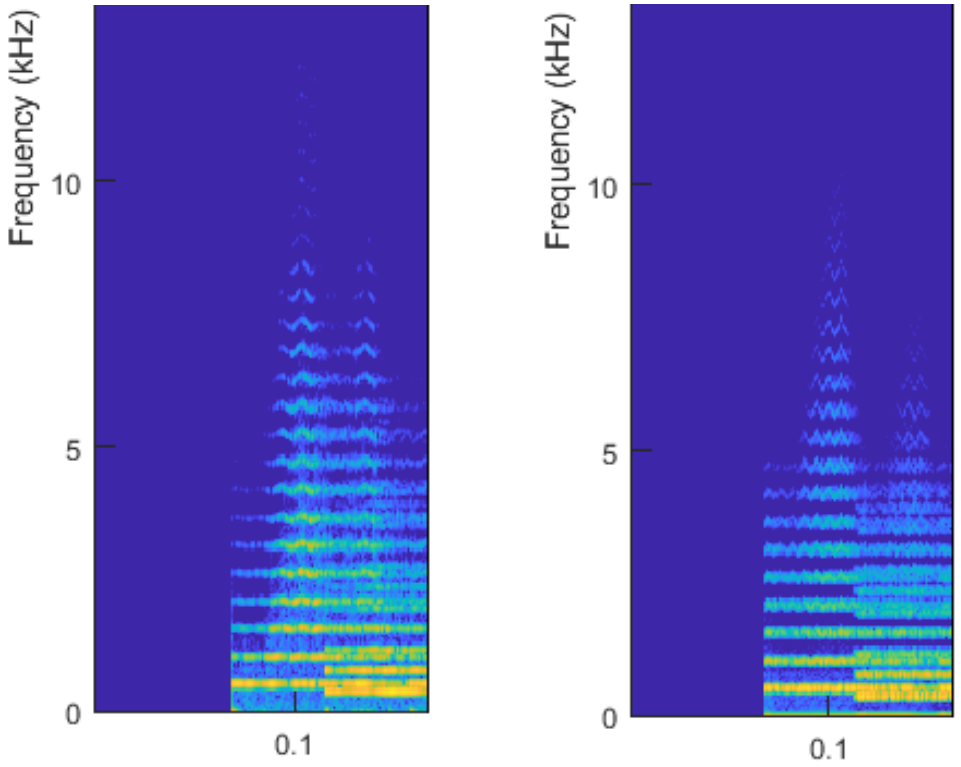


Figura 5.4: Tempi errati Delay

Dal confronto in figura 5.3 i tempi di delay risultano pressocchè simili, specialmente confrontando la distanza per punti ed inserendo un segmento di lunghezza fissa. Inoltre, durante il processo di sound design non tutto viene configurato correttamente, infatti nell'iterazione quattro è emerso che il time delay era completamente sbagliato (figura 5.4). Scendendo nel dettaglio non è solo sbagliato il tempo di delay ma anche il parametro *feedback*, ovvero il numero di ripetizioni, che invece di essere due come la clip originale, risultano essere quattro. Nelle iterazioni successive questi due parametri sono stati raffinati.

5.2.3 Analisi Modulazione Frequenza VCO tramite LFO

Nella settima iterazione è presente un difetto evidente, l'errata configurazione della modulazione sulla frequenza del VCO.



(a) Dettaglio Modulazione Originale (b) Dettaglio Modulazione VCV Rack

Figura 5.5: I due dettagli a confronto

É evidente come nella figura 5.5b è presente un'oscillazione triangolare molto evidente a differenza dello spettrogramma della clip originale. Questo evidenzia la forte influenza del modulo LFO, che modulava con un'onda triangolare il VCO. Per cui nell'iterazione successiva è stato ammorbidita la modulazione e aumentato leggermente il volume del riverbero. Arrivando infine allo spettrogramma dell'ottava iterazione(figura 5.1b) che risultando simile all'originale può considerarsi il risultato finale.

Capitolo 6

Conclusioni

Prendendo in considerazione le tastiere Midi MPE e il software introdotto VCV Rack, le prime sono caratterizzate dal termine **Ampia Espressività** ma anche dal difetto della *sintesi normalizzata*, mentre il secondo da **Libertà Espressiva** ma dall'assenza di un *modulo MPE/CV*.

Il progetto si pone tra le due caratterizzazioni, eliminando gli aspetti negativi di entrambi gli elementi. Per cui il modulo sviluppato mette in comunicazione uno strumento che permette al musicista di esprimersi in maniera completa con un software che consente di modificare il suono in totale libertà. Questo consente di avere molteplici combinazioni a livello di suono e di espressività senza essere limitati dalle scelte degli sviluppatori dei sintetizzatori commerciali.

Tuttavia è necessario che la tastiera sia perfettamente conforme al protocollo midi MPE (esempio: Roli Seaboard), altrimenti si perde quasi del tutto il lato espressivo del musicista. Il modulo rimane comunque utilizzabile, ma va utilizzata la modalità *rotative* che assegna ad ogni nota un rispettivo canale se non previsto dal sintetizzatore.

I possibili usi possono essere molteplici, dalla prototipazione per sintetizzatori normalizzati all'utilizzo in esecuzioni live di musicisti. Per quest'ultimo aspetto possono essere sviluppati in futuro moduli per l'assegnamento di *parametri di regolazioni* presenti su alcune tastiere Midi.

Bibliography

[1] *Midi Association*. URL: https://www.midi.org/specifications.

[2] *Midi Association*. URL: https://www.midi.org/midi-articles/midi-polyphonic-expression-mpe.

[3] *Philip Rees Music Tech*. URL: http://www.philrees.co.uk/nrpnq.htm.

[4] *Treccani istituto di enciclopedia italiana*. URL: https://www.treccani.it/enciclopedia/midi.

[5] *Wikipedia*. URL: https://it.wikipedia.org/wiki/Design_del_suono.

[6] *Wikipedia*. URL: https://it.wikipedia.org/wiki/Oscillatore_controllato_in_tensione.

[7] *Wikipedia*. URL: https://it.wikipedia.org/wiki/Voltage-controlled_filter.

[8] *Wikipedia*. URL: https://it.wikipedia.org/wiki/ADSR.