POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria del Cinema e dei Mezzi di Comunicazione

Tesi di Laurea Magistrale

Progettazione di un suono pilotato da un sistema avanzato per sonorizzare i veicoli elettrificati

Relatore

prof. Marco Masoero

Correlatore aziendale

Ing. Massimiliano Curti

Candidato

Andrea Greco

Anno Accademico 2020-2021

Alla mia famiglia

Sommario

L’industria automobilistica negli ultimi decenni sta cambiando, i veicoli elettrici sono ormai sempre più diffusi per le strade e ognuna delle principali case automobilistiche ha lanciato almeno un modello di veicolo elettrico o ibrido. Questo trend è probabilmente destinato ad aumentare nel tempo.

Una caratteristica del motore elettrico è la poca rumorosità a basse velocità. Questo a primo impatto sembrerebbe essere un vantaggio, in quanto il rumore emesso dai veicoli contribuisce ad aumentare l’inquinamento acustico, che ha effetti negativi sulla salute delle persone e sull’ambiente.

Tuttavia, la silenziosità di un veicolo elettrico può risultare pericolosa per i pedoni. Gli “indizi” utilizzati dalle persone per rilevare un veicolo in avvicinamento si dividono in visivi e uditivi. Utilizzando solo indizi visivi, si è verificato che i pedoni tendono a sottostimare le velocità basse e questo può essere pericoloso. Inoltre i pedoni con difficoltà visive possono basarsi quasi esclusivamente su indizi uditivi, per questi motivi risulta ancora più importante la progettazione di un sistema che aumenti la facilità di rilevazione dei veicoli elettrici. [1]

La soluzione attualmente usata è quella di aggiungere un suono esterno al veicolo in modo da aumentarne la rilevabilità. Questo suono viene chiamato AVAS (Acoustic Vehicle Alerting System) ed è stato regolamentato ad Ottobre 2016 dal “Regolamento n.138 della Commissione economica per l’Europa delle Nazioni Unite (UNECE)” e dalla “Regolamentazione delegata dalla Commissione EU 2017/157” che ha stabilito che dal 1 Luglio 2021 tutte le automobili elettriche potranno circolare solo se hanno a disposizione un AVAS che abbia le caratteristiche indicate nel suddetto regolamento.

Lo scopo di questo progetto è quello di progettare un suono che sia conforme alla normativa europea e di esplorare le diverse opportunità a disposizione durante il processo di sound design. Inizialmente sono stati studiati i risultati di alcune ricerche già effettuate sull’argomento che hanno analizzato l’effetto percettivo di alcune caratteristiche del suono, in particolare quelle che rendono l’AVAS più facilmente rilevabile. Sono stati inoltre considerati anche parametri soggettivi come “appropriatezza” e “piacevolezza”.

Una volta definite queste caratteristiche, si sono vagliate le possibili tecniche per la progettazione del suono. La prima tecnica, più semplice, consiste nel sintetizzare un suono composto da diverse forme d’onda semplici. Come spunto iniziale si è utilizzato uno dei preset del software “Igniter”, sviluppato da “AIR Music Technology”, utilizzato come base su cui lavorare. Come risultato si è ottenuta la possibilità di aggiungere un numero indefinito di componenti utilizzando le principali forme d’onda (sinusoidale, quadra, dente di sega, triangolare) o eventualmente una via di mezzo tra due di queste forme d’onda base.

L’altra tecnica utilizzata è quella della sintesi granulare, che consiste nell’utilizzare campioni sonori registrati e riprodurne solo piccole porzioni in loop, agendo su diversi parametri e introducendo un fattore di casualità. Questa tecnica verrà approfondita nel dettaglio ed è quella più utilizzata dalle varie case automobilistiche, in quanto consente un approccio più creativo e risultati molto vari. Dopotutto, la necessità di aggiungere un AVAS al veicolo è anche un’opportunità di brandizzare il prodotto, generando suoni innovativi che possano essere facilmente associabili ad una casa automobilistica. I principali brand hanno già investito cercando di sfruttare al meglio questa occasione, in particolare BMW ha ingaggiato il famoso compositore di colonne sonore e produttore discografico Hans Zimmer per la progettazione del proprio AVAS.

Il progetto, oltre ad esplorare queste tecniche e progettare un suono tramite un approccio sperimentale, consiste nella progettazione di un sistema per leggere segnali dalla rete del veicolo e utilizzare i dati per la sintesi sonora in tempo reale.

Il protocollo utilizzato per la comunicazione con il veicolo è il CAN bus (Controller Area Network bus), che è diventato ormai uno standard per veicoli elettrici. Grazie a questo protocollo, utilizzando un device chiamato "CAN bus Shield", è possibile mandare e ricevere messaggi tramite socket. I segnali vengono inviati dal veicolo ad un Raspberry PI 4. Uno script Python ha il compito di interpretare questi messaggi ed estrapolare i dati utilizzando un database in formato dbc che associa ogni segnale generato dal veicolo ad un messaggio corrispondente nel protocollo CAN.

La generazione sonora avviene tramite un linguaggio di composizione algoritmica chiamato "supercollider", che riceve i dati tramite un protocollo chiamato OSC (open sound control) e li usa per generare e modificare il suono in tempo reale, in modo da produrre un risultato che sia percettivamente facile da associare al classico rumore dei motori a combustione interna.

Il Raspberry è collegato ad un altoparlante, che solitamente è montato nella fascia anteriore dell'automobile.

Il progetto è stato svolto in remoto, pertanto il veicolo è stato simulato da un laptop. Nella fase di sound design è stato usato un sensore ToF con una funzione di trasferimento in grado di simulare la pressione del pedale dell'acceleratore e il conseguente aumento di velocità.

Il Raspberry PI 4, il sensore ToF e il Can bus Shield sono stati forniti da Teoresi S.p.A., azienda che ha fornito assistenza durante lo svolgimento del progetto.

L'ultima attività svolta è stata la registrazione del suono in camera anecoica, messa a disposizione dal Politecnico di Torino. Le diverse misurazioni effettuate seguendo le metodologie esplicate nel "Regolamento n.138 della Commissione economica per l'Europa delle Nazioni Unite (UNECE)", hanno permesso di verificare se il suono progettato fosse conforme alla normativa stessa.

Indice

Lista delle tabelle

Lista delle figure

1 Introduzione generale

1.1 Le motivazioni dietro al progetto

L'industria automobilistica è in fase di transizione alla mobilità elettrica, son sempre più numerosi i modelli elettrici in commercio e in circolazione.

I motori elettrici sono silenziosi a basse velocità, questo è un problema per i pedoni in quanto è più difficile rilevare un veicolo in avvicinamento, specialmente per persone ipovedenti.

La soluzione è la generazione di un suono da parte di un sistema chiamato AVAS, acronimo di Acoustic Vehicle Alerting System. Il suono viene prodotto esternamente al veicolo in modo da aumentarne la rilevabilità. In media un veicolo elettrico a basse velocità (sotto i 20 km/h), viene identificato 1,5 secondi dopo un veicolo a benzina alla stessa velocità, mentre dopo l'aggiunta dell'AVAS i risultati sono comparabili. [2] Secondo un altro studio, è stato stimato che un veicolo elettrico che viaggia a 10 km/h può essere identificato a soli 5 metri di distanza dal pedone, mentre alla stessa velocità un automobile a benzina viene rilevata anche a 50 metri. [3]

Il suono esterno al veicolo è stato regolamentato in varie parti del mondo, in questo progetto verrà tenuta in considerazione la normativa europea riguardante gli AVAS.

1.2 Stato dell'arte

Le case automobilistiche che hanno almeno un modello elettrico o ibrido in commercio hanno dovuto impegnarsi su questo argomento, in quanto dal 1 Luglio 2021 è diventato obbligatorio per tutti i veicoli elettrici o ibridi essere provvisti di un AVAS, come specificato dal Regolamento n.138 della Commissione economica per l'Europa delle Nazioni Unite.

Tra le varie aziende BMW ha investito molto sulla progettazione di questo suono, ingaggiando il famoso compositore di colonne sonore Hans Zimmer, vedendo questo "problema" come un'ottima opportunità di branding.

Gli esempi in commercio sono risultati della combinazione di varie tecniche, l'utilizzo di vari strati di complessità sonora fa ottenere un effetto realistico, nei casi più riusciti il suono sembra proprio derivare intrinsecamente dal movimento del veicolo, aggiungendo un carattere futuristico al modello che, associato ad un design estetico moderno, contribuisce alla percezione di una nuova generazione di veicoli, più avanzata e "robotica".

Nonostante i vari esempi presenti, c'è ancora poca letteratura in merito. Il seminario gratuito online della Siemens intitolato "Creare il suono adatto per i veicoli elettrici con l'Active Sound Design" è stata la fonte più preziosa delle idee esposte in questo documento. [4]

Tra i vari esempi in commercio analizzati, un'eccezione è costituita dal modello di 500 elettrica prodotto dalla FIAT. In questo caso, l'AVAS intona le prime note della celeberrima colonna sonora del film "Amarcord" composta da Nino Rota, si tratta del primo caso in cui un'automobile "canta" e invece di emettere rumore, fa musica. Questa modalità può essere disattivata dall'utente, inoltre è possibile solo oltre una certa velocità in quanto prima dei 20 km/h il suono deve essere comunicativo dell'incremento della velocità del veicolo, come stabilito dalla normativa europea. Questo esempio ci dà un'idea di quanto sia vario il panorama di opzioni possibili nella realizzazione dell'AVAS, che, pur dovendo rispettare alcuni parametri, consente la sperimentazione di novità assolute per l'industria automobilistica.

Tra gli esempi in commercio analizzati, uno dei punti in comune è la tendenza alla ricerca di un suono innovativo e futuristico, che, in combinazione ad un design estetico moderno e accattivante, comunichi ai pedoni la presenza di un tipo di veicolo diverso che rappresenta la novità. I modelli elettrici in questo modo acquistano un "carattere" molto diverso rispetto a quelli con motore a combustione interna.

Un'altra caratteristica che è comune ai vari esempi riguarda la posizione dell'altoparlante, che si trova solitamente nel paraurti anteriore, diretto verso il basso. Questo probabilmente è collegato al fatto che per rilevare un veicolo è molto più importante il livello del rumore emesso che la direzione da cui proviene. [1]

2 Specifiche previste dalla normativa europea

2.1 Regolamentazione delegata dalla Commissione EU 2017/157

La Regolamentazione delegata dalla Commissione EU 2017/157 stabilisce che ogni nuovo veicolo dal 1 Luglio 2019 deve essere disposto di un AVAS, mentre dal 1 Luglio 2021 è obbligatorio per tutte le automobili elettriche o ibride, anche i modelli già in commercio hanno dovuto aggiornarsi.

La normativa definisce alcune caratteristiche che ogni AVAS deve avere per poter essere messo in circolazione:

- Il dispositivo deve generare suono quando il veicolo si mette in moto e si muove fino alla velocità di 20 km/h, anche in retromarcia.
- l'AVAS può essere munito di un interruttore per mettere attivare e disattivare la riproduzione del suono. Quanto il veicolo parte, lo switch deve essere sempre ON. Se presente, l'interruttore deve essere posizionato in modo da essere facilmente azionabile dal conducente nella normale posizione di guida. Inoltre, il costruttore deve indicare al proprietario del veicolo che la funzione di pausa per l'AVAS non deve essere utilizzata a meno che non si è certi che non vi siano pedoni nelle vicinanze.
- Il livello del suono deve poter essere attenuato quando il veicolo è in funzione. Il suono attenuato deve comunque stare sopra il livello minimo specificato dal Regolamento n.138 della UNECE**,** riportati successivamente.
- Il suono deve essere continuo e deve essere informativo del comportamento del veicolo in movimento. Inoltre deve essere simile a quello di un veicolo con motore a combustione interna appartenente alla stessa categoria. Quando il veicolo è fermo, può emettere suono. Possono essere previsti diversi suoni selezionabili dal conducente, purché siano tutti conformi alle normative.
- Il livello sonoro complessivo dell'AVAS non deve eccedere i 75 dB (A) se il veicolo procede nella direzione di marcia. Questo valore deve essere rispettato quando si misura il suono dell'AVAS a 2 metri di distanza dal suono stesso, nelle modalità specificate dalla Regolamentazione n.138 della Commissione economica per l'Europa delle Nazioni Unite.

2.2 Regolamentazione n.138 della Commissione economica per l'Europa delle Nazioni Unite (UNECE)

Il Regolamento n.138 della Commissione economica per l'Europa delle Nazioni Unite (UNECE), entrato in vigore il 5 Ottobre 2016, si applica a veicoli di categoria M e N.

In questo documento viene descritto il metodo per omologare un suono progettato per un AVAS, in particolare vengono riportati i livelli di pressione sonora minimi necessari. Le prove per l'omologazione vengono effettuate a 10 e 20 km/h.

Frequenza (Hz)		Prova a velocità costante 10 km/h (dB (A))	Prova a velocità costante 20 km/h (dB (A))	Prova di retromarcia (dB (A))
Colonna 1	Colonna 2	Colonna 3	Colonna 4	Colonna 5
Complessiva		50	56	47
Bande in terzi d'ottava	160	45	50	/
	200	44	49	
	250	43	48	
	315	44	49	
	400	45	50	
	500	45	50	
	630	46	51	
	800	46	51	
	1000	46	51	
	1250	46	51	
	1600	44	49	
	2000	42	47	
	2500	39	44	
	3150	36	41	
	4000	34	39	
	5000	31	36	

Tabella 2.1: requisiti di livello sonoro minimo in dB (A) [6]

Un suono di un sistema AVAS per essere omologato deve rispettare le seguenti condizioni:

- Ha un livello minimo di pressione sonora globale per la velocità di prova conforme alla *Tabella 2.2***.**
- Abbia almeno due delle bande in terzi d'ottava secondo la *Tabella 2.2***.** Almeno una di queste bande deve essere inferiore o inclusa nella banda in terzi di ottava corrispondente a 1600 Hz.
- Nelle due bande identificate il suono deve avere livelli minimi di pressione sonora per le due velocità di prova corrispondenti a quanto indicato nella *Tabella 2.2*.
- Per la prova in retromarcia, bisogna considerare il livello sonoro complessivo, che non deve superare il valore indicato in *Tabella 2.2*.
- Almeno un tono deve variare in frequenza in proporzione della velocità in ciascun singolo rapporto di una media di almeno lo 0.8 % per 1 km/h nell'intervallo di velocità compreso tra 5 km/h e 20 km/h.
- Il livello sonoro complessivo dell'AVAS non deve eccedere i 75 dB (A) se il veicolo procede nella direzione di marcia.

Come stabilito dalla Regolamentazione delegata dalla Commissione EU 2017/157, il suono deve essere simile a quello di un veicolo con motore a combustione interna appartenente alla stessa categoria. Siamo abituati ad un suono che aumenta in frequenza all'aumento di velocità, e viceversa. Il tono che varia in proporzione alla velocità è fondamentale in quanto comunica ai pedoni l'accelerazione e la decelerazione del veicolo. In questo progetto i toni saranno fatti variare in funzione della velocità in modo esponenziale, infatti la frequenza di un suono viene percepita in modo logaritmico (è facile riconoscere la differenza tra 50 e 60 Hz mentre 1000 e 1010 Hz sono molto difficili da distinguere).

3 Considerazioni sulle caratteristiche del suono dell'AVAS

3.1 Rilevabilità del veicolo

I veicoli elettrici producono poco rumore a basse velocità, mentre oltre i 20 km/h il rumore dei pneumatici è sufficiente.

In un precedente lavoro sono stati considerati gli effetti di alcune caratteristiche sonore sulla rilevabilità di un veicolo elettrico. Durante un esperimento è stato riprodotto lo scenario di attraversamento della strada, il pedone è stato sostituito da una testa artificiale con un microfono. Il veicolo era in movimento alla velocità di circa 20 km/h. [2]

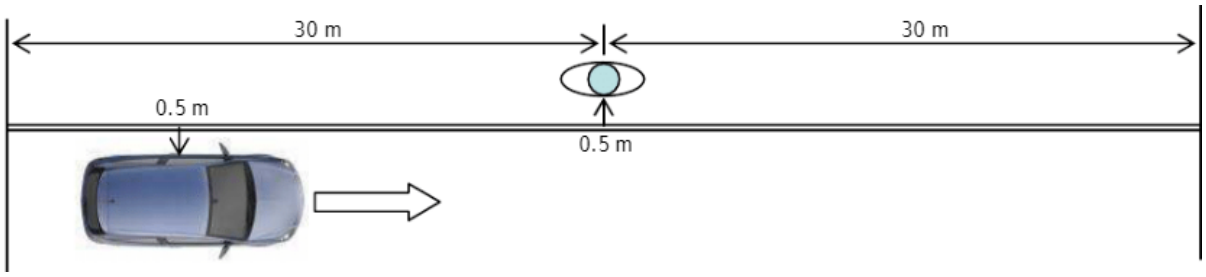

Figura 3.1: rappresentazione grafica dello scenario di attraversamento della strada. [2]

I segnali utilizzati durante l'esperimento erano composti da una combinazione di toni puri. È stata studiata l'influenza del numero di toni utilizzati e della presenza di una modulazione temporale, sia in frequenza che in ampiezza. La larghezza di banda complessiva dei suoni generati era tra i 300 a 1500 Hz, sotto i 300 Hz infatti si concentra la maggior parte del rumore ambientale e il segnale sarebbe stato mascherato acusticamente. Il limite dei 1500 Hz è stato scelto per focalizzare l'energia in una banda di frequenza limitata, che teoricamente dovrebbe rendere il segnale più facilmente rilevabile a confronto di un segnale con stessa energia complessiva ma larghezza di banda maggiore. È stata creata una libreria di segnali diversi aventi differenti numeri di toni e diversi livelli di modulazione in frequenza e/o ampiezza. [2]

La modulazione in frequenza si verifica quando la frequenza di un segnale (portante) viene pilotata da un altro, detto modulante. Per quanto riguarda la modulazione in ampiezza invece, l'ampiezza della portante varia in modo proporzionale a quella del segnale modulante.

Gli stimoli generati sono stati processati in modo da rappresentare una sorgente omnidirezionale che passa a 20 km/h davanti al pedone, il risultato è stato aggiunto a quello di un veicolo elettrico registrato nelle stesse condizioni ma senza AVAS. I campioni ottenuti sono stati fatti sentire a 162 persone che hanno partecipato all'esperimento, 53 delle quali ipovedenti. I partecipanti hanno ascoltato in cuffia un rumore ambientale continuo e ogni tanto è stato simulato il passaggio di un veicolo con uno dei segnali generato. Per ogni segnale si è considerato il tempo di reazione medio che ha consentito la rilevazione del veicolo. [2]

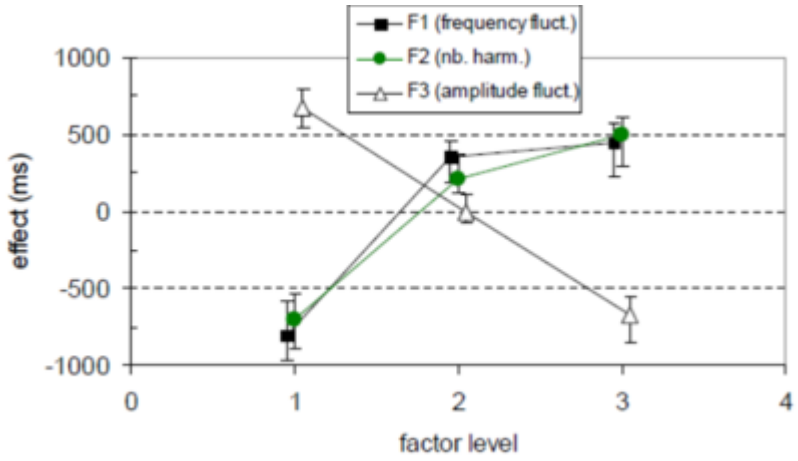

Figura 3.2: effetti delle caratteristiche del suono sul tempo di reazione. [2]

In *Figura 5.2* sull'asse y è rappresentato l'effetto delle tre diverse caratteristiche sul tempo di reazione, sull'asse x la quantità di modulazione o il numero di toni (livello di fattore 1 corrisponde a bassa modulazione e pochi toni, livello 3 corrisponde a tanti toni e grande modulazione). Si nota che una maggiore modulazione in ampiezza ha contribuito a diminuire il tempo di reazione. L'effetto di una maggiore modulazione in frequenza e maggior numero di toni invece è stato opposto, livelli minimi di entrambi questi fattori corrispondono a tempo di reazione minore. [2]

Queste considerazioni saranno utili nel corso del progetto, sarà utilizzata una modulazione in ampiezza e si farà modo di generare un suono limitato in banda per aumentare la rilevabilità dello stesso.

3.2 Appropriatezza e piacevolezza del suono

Il futuro sarà sempre più elettrico, è ragionevole pensare che le nostre strade fra non troppi anni saranno colme di veicoli elettrici o ibridi e saremo circondati da mezzi di trasporto che suonano in modo diverso da quanto siamo correntemente abituati.

Per questa ragione, è bene anche considerare aspetti percettivi soggettivi nella fase preliminare della progettazione del suono.

In una relazione del 2013 sono stati fatti sentire 17 diversi suoni artificiali a 40 partecipanti (di cui 6 esperti di sound design). Sono stati misurati la piacevolezza e l'appropriatezza dei vari campioni proposti. Essendo un test soggettivo, si sono riscontrate differenze tra alcuni individui anche elevate. [9]

Secondo la maggior parte degli utenti, il suono generato da un AVAS dovrebbe essere simile a quello di un motore a combustione interna.

I suoni che hanno ricevuto più commenti positivi in termini di piacevolezza erano corrispondenti a frequenze basse, mentre alte frequenze sono state considerate negativamente. Un suono che comunica l'accelerazione del veicolo è considerato sia appropriato che piacevole. La presenza di modulazione in ampiezza da un gruppo numeroso di partecipanti è considerata appropriata, mentre altri non l'hanno trovata piacevole se troppo ampia e veloce (frequenza e ampiezza della modulante elevate). L'autore della ricerca sottolinea come un suono simile a quello del motore a combustione interna è considerato spesso piacevole per inerzia psicologica, è familiare e quindi risulta piacevole. [9]

3.3 Caratteristiche del suono dell'AVAS

In base alle considerazioni precedenti, il suono da progettare avrà le seguenti caratteristiche:

- almeno un tono varia al cambiare della velocità del veicolo, in modo che comunichi l'accelerazione o decelerazione del veicolo.
- deve essere percettivamente simile al suono di un motore a combustione interna, in modo da essere facilmente riconoscibile come rumore di un automobile in avvicinamento. Sono da evitare rumori come quello di una sirena, o suoni discontinui di avviso in quanto di solito non vengono associati all'avvicinamento di un veicolo (oltre al fatto che possono essere considerati fastidiosi).
- presenza di modulazione in ampiezza, per aumentare la rilevabilità del veicolo. Da evitare modulazione ampia e veloce perché può risultare fastidiosa. La modulazione in frequenza invece non verrà utilizzata.
- limitato in una determinata banda di frequenza, in modo da essere più definito e quindi più facile da rilevare.
- alte frequenze (indicativamente sopra i 4-5 kHz) sono da evitare, in quanto risultano fastidiose, come riportato dai test soggettivi.
- basse frequenze (indicativamente sotto i 150/200 Hz) sono da evitare, in quanto vengono mascherate acusticamente dal rumore ambientale (che si concentra soprattutto alle basse e medio-basse frequenze).
- nella valutazione soggettiva dei campioni proposti ai soggetti, aggettivi come "futuristico" e "innovativo" sono stati considerati spesso attributi piacevoli dei suoni proposti. Gli esempi in commercio analizzati hanno colto questa opportunità di branding e hanno investito molto per ottenere un risultato originale.

Una delle sfide di questo progetto è riuscire a raggiungere un risultato abbastanza simile al rumore dei veicoli con motore a combustione interna, in modo da permettere ai pedoni di comprendere che il suono percepito è quello di un veicolo in avvicinamento, ma allo stesso tempo abbastanza diverso da renderlo chiaramente associabile ad un nuovo tipo di veicoli, quelli elettrici, che rappresentano il futuro e suonano diversamente rispetto a quello a cui siamo abituati.

I segnali provenienti dalla rete del veicolo utilizzati al fine della generazione del suono in tempo reale, sono i seguenti:

- velocità: il segnale più importante in quanto è proprio grazie alla variazione di velocità che il suono dovrà cambiare in frequenza, in questo modo i pedoni percepiscono un veicolo elettrico in movimento. Il suono progettato non cambia in relazione alla frenata, nei vari esempi analizzati e nel seminario della Siemens viene considerata sufficiente la variazione del suono in base alla velocità, infatti quando un veicolo decelera la frequenza del suono diminuisce, questo indica un veicolo in frenata o decelerazione.
- coppia motrice: per includere la percezione della potenza. [4] Nel prototipo realizzato con sensore ToF è stata utilizzata la posizione del pedale dell'acceleratore per semplicità.
- accensione e spegnimento motore: quando il motore viene acceso, l'AVAS si aziona e il suono viene generato. Quanto viene spento, il suono svanisce.

- retromarcia: il veicolo deve emettere rumore anche in retromarcia. In questo progetto il suono del veicolo in retromarcia è lo stesso utilizzato in avanzamento. Nei lavori svolti precedentemente non sono stati trovati riferimenti riguardo diverse caratteristiche sonore necessarie in retromarcia. Tra gli esempi in commercio, un veicolo elettrico utilizza lo stesso rumore sia in avanzamento che in retromarcia: l'unica differenza è che il veicolo da fermo con la retromarcia inserita emette un suono ad un livello leggermente più elevato. Questo probabilmente perché un veicolo fermo che sta uscendo da un parcheggio in retromarcia può essere pericoloso. Per la retromarcia viene utilizzato un segnale che è informativo dello stato della retromarcia, di norma è ad un valore che indica che la retromarcia non è inserita, ma quando varia lo script supercollider deve intercettare questo valore e modificare il suono di conseguenza.
- attenuazione: come specificato dalla normativa, l'utente deve avere a disposizione un metodo per attenuare il livello sonoro anche quando il veicolo è in movimento. Si suppone la presenza di un bottone, facilmente accessibile durante la guida, che mandi un segnale alla rete del veicolo, quando questo viene ricevuto dal server supercollider il suono viene attenuato.

4 Composizione algoritmica con Supercollider

Ai fini di questo progetto occorreva uno script che fosse in grado di generare un suono e cambiarne alcune caratteristiche in tempo reale. Tra le varie possibilità è stato scelto Supercollider, un linguaggio scritto e non modulare, a differenza della maggior parte delle alternative. Nonostante sia più complicato ottenere certi risultati, questa caratteristica permette una maggiore libertà espressiva in quanto si può andare a manipolare manualmente ogni singolo parametro.

Uno dei fattori che ha contribuito alla scelta di questo linguaggio, riguarda la compatibilità con il Raspberry PI, questo consente di poter creare un prototipo di un sistema embedded utilizzando una scheda Raspberry collegata alla rete del veicolo.

Supercollider è un linguaggio di programmazione ottimizzato per la sintesi audio in tempo reale e la composizione algoritmica. È composto da tre parti:

- **scsynth**: il server audio che si occupa di generare il suono in tempo reale. È composto da più di 400 "unit generators" (UGens) a disposizione degli utenti per la sintesi sonora. Grazie alla sua struttura è possibile combinare diverse tecniche in parallelo, cosa che sarà molto utile nel corso di questo progetto.
- **sclang**: il linguaggio di programmazione. É orientato agli oggetti. Controlla il server scsynth attraverso il protocollo Open Sound Control (OSC).
- **scide**: è l'editor testuale per sclang, integrato con delle guide molto utili su tutte le funzioni e le UGens a disposizione. [7]

Di seguito è riportata la generazione di una semplice sinusoide. La UGen SinOsc genera un'onda sinusoidale. Senza specificare ulteriori parametri viene generato un segnale con frequenza 440 Hz, fase 0 e con ampiezza oscillante tra -1 e 1.

```
{SinOsc.ar}.play;
```

Una UGen è un oggetto che può processare o generare audio e segnali di controllo. Viene creata utilizzando il metodo "ar" o "kr":

- Metodo "ar": crea una UGen che genera 44100 campioni al secondo (ar sta infatti per audio rate);
- Metodo "kr": crea una UGen che genera 689 campioni al secondo, 44100/64 dove 64 è la grandezza di un blocco (control rate). Viene utilizzato per segnali che non generano audio ma controllano determinati parametri.

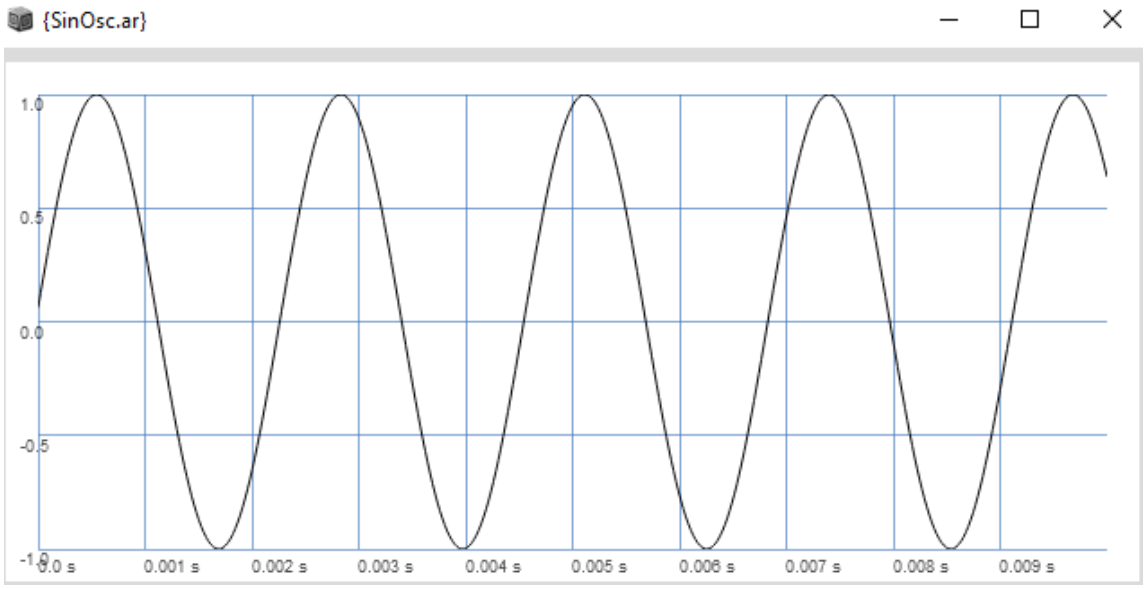

Figura 4.1: grafico del segnale sinusoidale generato dalla UGen SinOsc (audio rate)

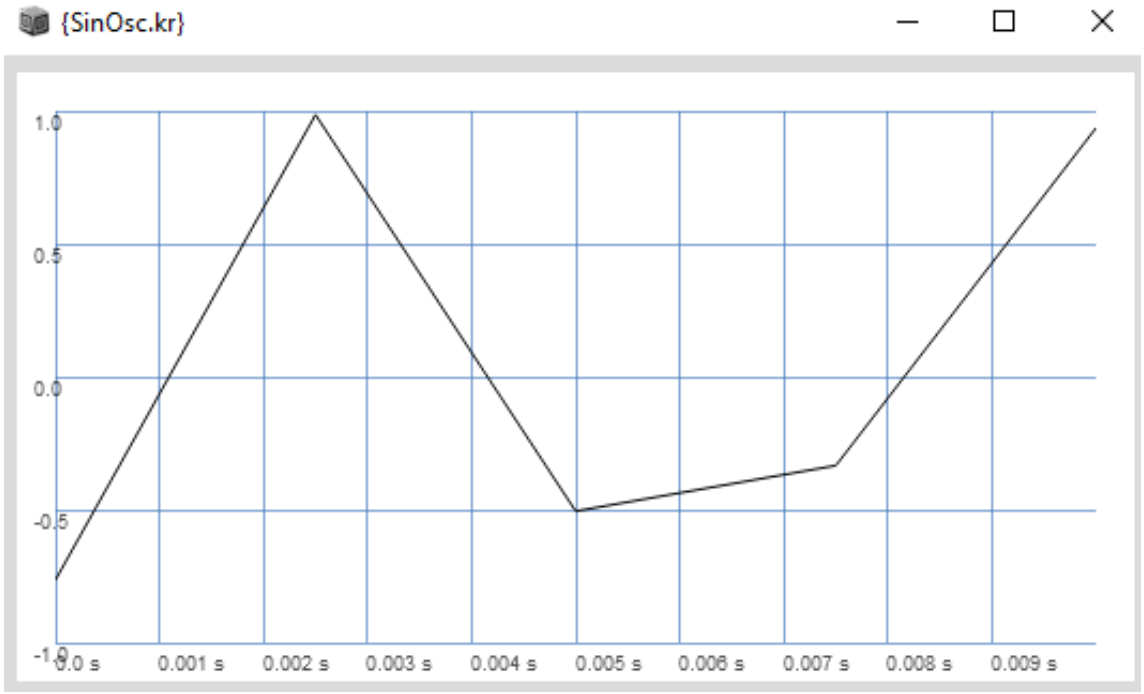

Figura 4.2: grafico del segnale sinusoidale generato dalla UGen SinOsc (control rate)

Prima di generare suono, occorre sempre inizializzare il server audio scsynth. Per farlo, viene utilizzata una variabile globale “s” che per comodità è assegnata ad un’istanza del server, in questo modo risulta facilmente accessibile dall’interprete.

```
//avvio del server scsynth
s.boot;
```

Gli oggetti più utilizzati per la sintesi sonora appartengono alla classe SynthDef. Un oggetto SynthDef è la definizione di un synth, dentro di questo vengono specificate le UGen da generare. Questa configurazione può viene aggiunta in memoria, l'utente può quindi costruirsi una libreria di SynthDef e quando necessario chiedere al server di istanziare un Synth a partire da una determinata SynthDef.

In sostanza, definire una SynthDef è paragonabile alla definizione di un progetto di un Synth analogico, lo schema elettrico. Quando istanziamo un Synth costruiamo il sintetizzatore vero e proprio. [8]

Le SynthDef sono di solito parametrizzate, in questo modo si possono istanziare diversi Synth aventi caratteristiche differenti, quali frequenza e volume. Nell'esempio riportato, gli argomenti hanno assegnato un valore di default.

```
SynthDef.new (\esempio, {
    arg out=0, freq=440, volume=0.01;
    var signal;
    signal = SinOsc.ar(freq);
    Out.ar(out, volume*signal!2);
    }).add;
```

Quando viene istanziato un Synth l'utente può specificare uno o più argomenti in modo da sovrascrivere i valori di default. Il metodo "add" applicato all'oggetto SynthDef serve ad aggiungere la definizione in memoria, una volta compilato questo codice sarà possibile istanziare un Synth. Nell'esempio, il Synth "x" genera una sinusoide con frequenza 600 Hz, mentre "y" ne genera una con frequenza 440 Hz (in quanto non specificata).

```
x = Synth.new(\esempio, [\freq,  600, \volume, 0.02]);
y = Synth.new(\esempio);
```

Attraverso il metodo "set" è possibile modificare dinamicamente alcune caratteristiche di un Synth già creato. Questo consente di leggere i dati provenienti dalla rete del veicolo e utilizzarli per modificare alcune caratteristiche del suono in tempo reale.

```
y.set(\freq, 800, \volume, 0.04));
```

5 Prototipo per la simulazione della velocità del veicolo

5.1 Prototipo con velocità simulata da posizione del mouse

Ai fini della progettazione del suono, è stato necessario trovare un modo per simulare la variazione della velocità del veicolo, in modo da poter facilmente ascoltare il risultato ed apportare modifiche al codice di sintesi sonora in tempo reale.

In prima istanza, è stato utilizzata su supercollider la UGen "MouseX" che intercetta la posizione del mouse nell'asse X per generare un segnale di controllo. I parametri passati al metodo "kr" sono il valore minimo e valore massimo, corrispondenti al margine sinistro e destro dello schermo.

```
MouseX.kr(minval: 0, maxval: 1);
```

A questo punto la frequenza della UGen SinOsc, che si occupa di generare una semplice sinusoide, può essere pilotata dalla UGen MouseX. In questo modo la frequenza della sinusoide cambia muovendo il mouse sull'asse orizzontale dello schermo.

In particolare quando viene compilata la riga di codice nell'esempio riportato di seguito, viene generata una sinusoide con frequenza 100 Hz (supponendo che il mouse si trovi in corrispondenza del margine sinistro dello schermo). Spostando il mouse verso destra, la frequenza aumenta fino ad arrivare a 500 Hz a fine schermo. La frequenza di un segnale percettivamente ne definisce il pitch (in italiano, tono o intonazione). Una frequenza maggiore corrisponde ad un suono più acuto, in questo esempio il movimento del mouse simula l'andamento della velocità che corrisponde ad un aumento del tono percepito.

```
sin = SinOsc.ar(MouseX.kr(100, 500));
```

Il limite principale di questo metodo è che viene generato un solo segnale alla volta. Si può leggere anche il MouseY per avere due possibili segnali con cui pilotare i diversi parametri, il problema è che i due segnali MouseX e MouseY non sono correlati, è possibile variarne uno senza variare l'altro ed è molto difficile generare un effetto realistico.

5.2 Prototipo con sensore ToF

Successivamente è stato utilizzato un prototipo più evoluto. Teoresi S.P.A ha fornito una scheda di sviluppo della STMicroelectronics (NUCLEO-F401RE), combinata con X-NUCLEO-6180A1 (sempre STMicroelectronics), scheda di espansione che comprende il sensore di prossimità ToF.

Figura 5.1: scheda di sviluppo combinata con scheda di espansione con sensore di prossimità

Un sensore di prossimità ToF (time of flight) funziona misurando il tempo necessario alla luce per raggiungere l'oggetto più vicino e riflettersi di nuovo nel sensore. Un emettitore infrarossi invia impulsi luminosi, un rilevatore di luce legge gli impulsi riflessi e una circuiteria elettronica si occupa infine di misurare con precisione il tempo che intercorre tra l'emissione dell'impulso e la ricezione del segnale riflesso. [10]

Avvicinando la mano al sensore, si può simulare la pressione del pedale. Il primo dato che possiamo leggere è quindi la posizione dell'acceleratore, che è stato usato per approssimare la coppia motrice e darà indicazione della potenza del motore.

La scheda di sviluppo NUCLEO-F401RE viene utilizzata per integrare il sensore al prototipo, è collegata al laptop tramite USB e comunica tramite porta seriale. Oltre alla posizione del pedale dell'acceleratore, viene generato un altro segnale, che rappresenta la velocità, applicando un gradino nel dominio discreto del tempo. Questo introduce una sorta d'inerzia, quando viene premuto l'acceleratore la velocità cresce con un leggero ritardo che aggiunge realisticità al prototipo.

La funzione di trasferimento a tempo discreto utilizzata è la seguente

$$y = \frac{0.002059z + 0.001686}{z^2 - 1.511z + 0.5488} * u$$

Utilizzando il seguente codice su Matlab, sono stati generati i grafici di risposta al gradino (*Figura 4.2*) e all'impulso (*Figura 4.3*) della funzione di trasferimento utilizzata:

```
z=tf("z")
sysdiscr =  (0.002059*z + 0.001686)/(z^2-1.511*z+0.5488)
step (sysdiscr)
impulse (sysdiscr)
```

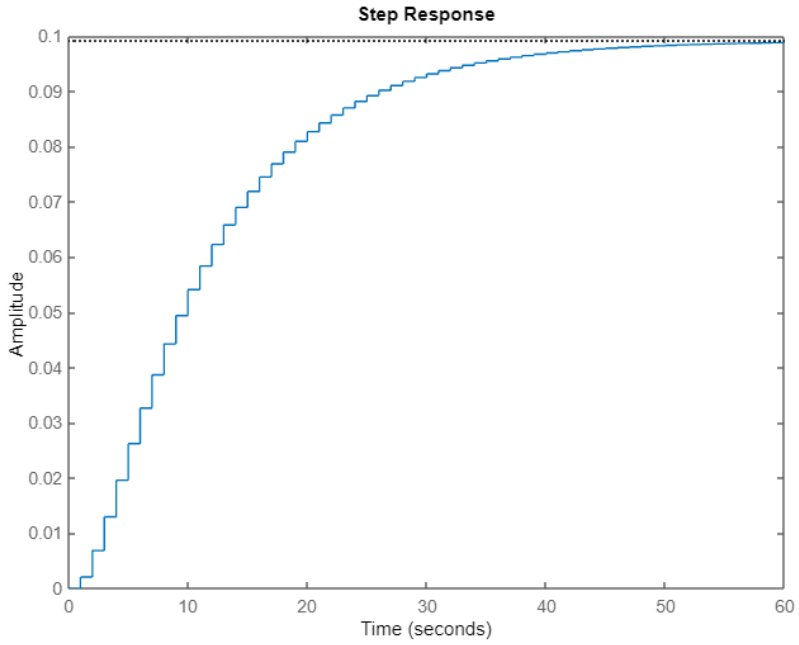

Figura 5.2: risposta al gradino della funzione di trasferimento utilizzata

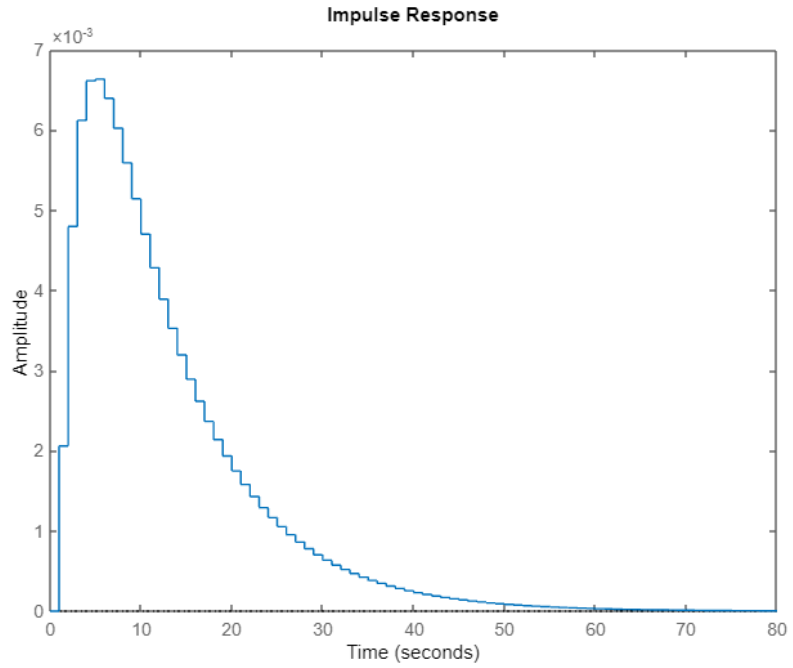

Figura 5.3: risposta all'impulso della funzione di trasferimento utilizzata

I segnali generati sono incapsulati in un messaggio di 4 byte ed inviati tramite porta seriale. Uno script Python si occupa di ricevere i dati dalla porta seriale e spacchettarli. Una volta letti, la velocità (speed) e la posizione del pedale (throttle) vengono inviati tramite protocollo OSC in modo che siano ricevuti dallo script di supercollider e utilizzati per modificare in tempo reale il suono. Il primo dei quattro byte è 10101011 (171 in base 10) e serve ad identificare l'inizio del pacchetto. Il secondo e il terzo byte rappresentano rispettivamente la posizione del pedale e la velocità.

```
while (1==1):
        if (ser.in_waiting >= 4):
            serialString = ser.read(4)
            if (serialString[0]==171):
                throttle = serialString[1]
                speed = serialString[2]
                send_osc(client, "/speedEngine", speed)
                send_osc(client, "/gaspedalpos", throttle)
```

Il sensore ToF consente una approssimazione accettabile dell'andamento della velocità in relazione alla posizione del pedale dell'accelerazione. Questo ha permesso in fase di sound design di avere una maggiore realisticità del cambiamento del suono generato a seconda dell'aumento di velocità del veicolo, dando la sensazione di stare effettivamente premendo un acceleratore.

Il limite di questo modello è che non è stato considerato il cambio di marcia, più viene "schiacciato" il pedale più cresce la velocità mentre in una situazione più realistica un valore della posizione del pedale più alto (equivalente ad un pedale maggiormente "schiacciato") non corrisponde per forza ad una velocità del veicolo maggiore.

5.3 Protocollo OSC

Il protocollo OSC (Open Sound Control) è utilizzato per la comunicazione in tempo reale tra diversi hardware e/o applicazioni.

I dati sono trasmessi in forma di pacchetti, chi invia messaggi OSC fa da client mentre chi riceve è un server, in questo caso lo script di supercollider. È un protocollo di alto livello, non si occupa del meccanismo di trasmissione dei dati. Tipicamente i pacchetti OSC sono inviati e ricevuti utilizzando socket UDP. [11]

Un messaggio OSC è composto da:

- Indirizzo: specifica a quale entità all'interno del server è destinato il messaggio. [8] Inizia sempre col carattere "/".
- Tipo: specifica il tipo di dati, può essere omesso.
- Argomenti: i dati effettivi.

Per permettere la comunicazione tra client e server, occorre specificare la porta corretta e l'indirizzo IP del server. In particolare la porta per comunicare con l'interprete sclang è la 57120, mentre l'indirizzo IP è 127.0.0.1.

```
client = udp_client.SimpleUDPClient("127.0.0.1", "57120")
```

In questo esempio, viene inviato il valore contenuto nella variabile "speed" all'entità "/speedEngine", definita poi nel server.

```
send_osc(client, "/speedEngine", speed)
```

OSC su supercollider

Supercollider supporta il protocollo OSC. In particolare in questo progetto è stato utilizzato l'oggetto OSCdef, concettualmente simile ad un SynthDef.

Quando viene istanziato un nuovo OSCdef ci sono tre principali parametri che devono essere specificati:

- chiave: identifica l'OSCdef in modo univoco.
- funzione: una funzione che verrà eseguita in risposta al messaggio ricevuto corrispondente all'OSCdef. La funzione riceve alcuni parametri in maniera automatica, tra questi un array "msg" che contiene l'indirizzo e gli argomenti ricevuti.
- percorso: è l'indirizzo OSC dell'oggetto, che inizia sempre col carattere "/". Ogni messaggio OSC ricevuto sull'indirizzo corrispondente a questo valore, scatenerà l'esecuzione della funzione specificata precedentemente.

Quando viene compilato l'OSCdef viene aggiunto in memoria (ogni volta che viene riavviato il server scsynth occorre ricompilare tutti gli OSCdef e i SynthDef per aggiungerli nuovamente in memoria). Da quel momento in poi, quando l'interprete riceve un messaggio OSC sulla porta 57120 con indirizzo corrispondente a quello di una OSCdef, viene eseguita la funzione specificata all'interno dell'oggetto.

Nell'esempio successivo, quando viene ricevuto un messaggio OSC con indirizzo "/speedEngine" viene generata una sinusoide con frequenza dipendente dall'argomento ricevuto. Se ad esempio viene rilevato il messaggio "/speedEngine 20" viene generata una sinusoide con frequenza 420 Hz.

```
OSCdef ('speedlistener', {
    arg msg;
   {SinOsc.ar(400+msg[1], 0, 0.1)}.play;
}, "/speedEngine");
```

Lo script su supercollider sarà costituito da vari OSCdef, ognuno risponde ad uno dei segnali ricevuti dalla rete del veicolo:

- l'OSCdef "ignitionlistener" si occupa di istanziare i vari synth. La normativa stabilisce che il veicolo elettrico acceso ma non in movimento non deve obbligatoriamente emettere suono. Tuttavia nella maggior parte degli esempi in commercio esaminati, il veicolo emette suono anche quando è fermo, per questo motivo anche nel corso di questo progetto è stato deciso di generare il suono direttamente all'accensione. Questo è importante soprattutto in retromarcia quando il livello del suono di solito è leggermente più elevato, perché un veicolo che sta uscendo da un parcheggio in retromarcia può rappresentare un pericolo maggiore per i pedoni o altri eventuali veicoli circolanti nello stesso tratto di strada.
- l'OSCdef "speedlistener" riceve il valore della velocità del veicolo e apporta modifiche in tempo reale al suono precedentemente generato. In particolare vengono cambiati il tono (frequenza) delle varie componenti sonore e il volume.
- l'OSCdef "gaspedallistener" che modifica alcune componenti del suono in relazione alla posizione del pedale dell'acceleratore. Nel prototipo presentato in questo documento, la posizione del pedale approssima la coppia motrice che dà un'indicazione della potenza del motore.
- l'OSCdef "shutdownlistener" si attiva allo spegnimento del motore del veicolo elettrico, in questo caso il suono viene interrotto e tutti i synth precedentemente istanziati vengono liberati mediante il metodo "free".
- l'OSCdef "attenuationlistener" si attiva quando il guidatore preme il pulsante per diminuire il livello del suono. Come stabilisce la normativa europea, deve essere possibile attenuare il suono quando il veicolo è in funzione.
- l'OSCdef "reverselistener" entra in azione quando viene attivata la retromarcia. Un segnale del protocollo CAN generalmente indica lo stato della retromarcia, quando viene attivato è inviato un messaggio OSC che attiva una funzione definita nell'OSCdef. In questo studio non è stato progettato un suono a parte, l'unico effetto dell'inserimento della retromarcia è l'aumento del rumore generato a veicolo fermo. Quando viene inserita la prima marcia, il volume deve tornare al livello originale.

Il protocollo OSC viene usato anche internamente in maniera implicita per la comunicazione tra l'interprete dei comandi sclang e il server di generazione sonora scsynth.

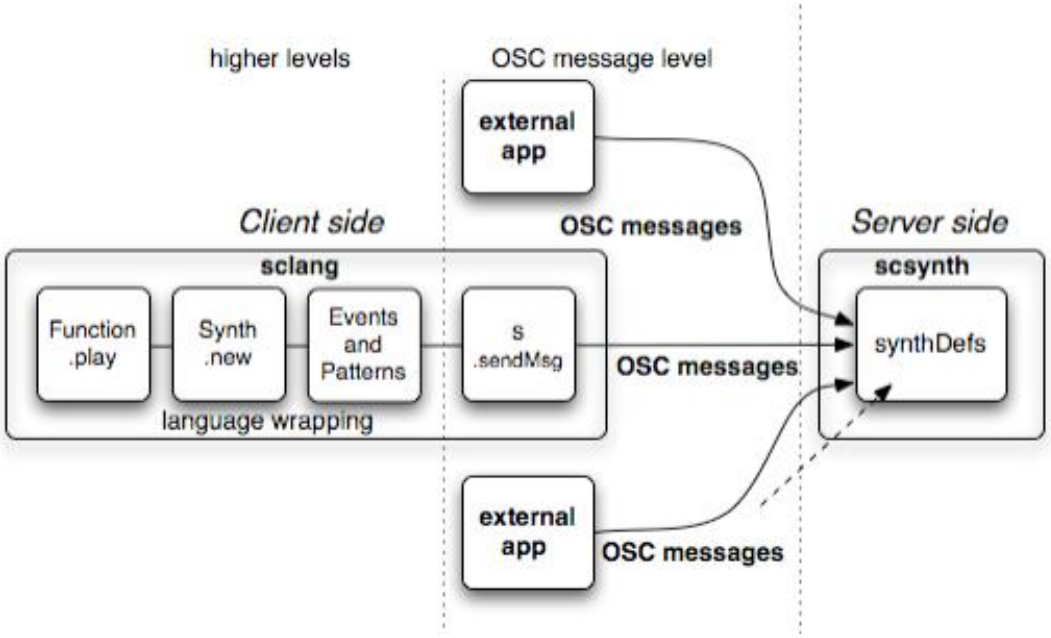

Figura 5.4: comunicazione tra sclang e scsynth tramite OSC [11]

6 Sound design

Una volta identificate le risorse a disposizione che hanno permesso la realizzazione del primo prototipo e definite le specifiche del suono che si vuole ottenere, occorre stabilire una strategia per la generazione sonora. Riguardo questo argomento in letteratura non si trovano molte indicazioni.

Ai fini di questo progetto, le fonti principali per le idee e le metodologie applicate sono le seguenti:

- Considerazioni generali di sound design tratte da corsi precedentemente svolti e dal libro "Designing Sounds" di Andy Farnell. [12]
- Seminario Siemens gratuito, con slide annesse. [4]
- Ascolto e analisi spettrale (tramite il software Audition della Adobe) del suono di veicoli elettrici già in commercio.
- Software "Igniter" sviluppato da Krotos Audio con preset già svolti e replica su Supercollider di uno script che emulasse lo stesso comportamento, dando la possibilità all'utente di variare i parametri.

6.1 Strati sonori

Il seminario Siemens riporta la possibilità di diversi approcci nella progettazione di un suono per un veicolo elettrico.

Innanzitutto occorre tenere conto di quali sono a livello fisico le grandezze fisiche che influiscono maggiormente nella percezione del suono generato da un veicolo. Queste grandezze sono associate a diversi strati, in particolare ne vengono identificati tre:

- Strato "velocità": è la componente che è influenzata dalla variazione della velocità del veicolo, si traduce percettivamente in un aumento in frequenza di almeno un tono appartenente al suono.
- Strato "potenza": la componente che dà la percezione della potenza del veicolo in avvicinamento, nel corso di questo progetto è stata utilizzata la posizione dell'acceleratore, di solito viene utilizzata la coppia motrice.
- Strato di "realismo aggiuntivo": componente che rimane fissa, generalmente a frequenze basse o medio basse e contribuisce ad aggiungere realisticità al suono prodotto dal veicolo. [4]

La componente che varia con la velocità è quella dove c'è più spazio d'azione, sono possibili diverse tecniche:

- Approccio tradizionale: sintesi sonora più classica, l'obiettivo è quello di emulare il comportamento del motore a quattro tempi. Comporta la creazione di un modello che può essere più o meno complesso.
- Approccio creativo: di solito prevede l'utilizzo della sintesi granulare, vengono utilizzati campioni registrati opportunamente e, riproducendo piccole porzioni del segnale a velocità elevata, si ottengono suoni ricchi e complessi. La generazione sonora in questo caso non è collegata a quella di un veicolo classico. [4]

- Approccio misto: prevede la combinazione dei due metodi elencati precedentemente e consente di ottenere un suono innovativo ma allo stesso tempo ancora legato al classico rumore dei motori a combustione interna.

Il suono, come stabilisce la normativa, deve variare in relazione alla velocità nel range tra 0 e 20 km/h, oltre questo valore l'AVAS non è più obbligatorio in quanto il rumore emesso dal rotolamento degli pneumatici sulla strada è sufficiente.

Nel nostro caso, il suono da 0 a 20 km/h cambia conformemente a quanto stabilito dalla normativa, oltre i 20 km/h inizia ad attenuarsi gradualmente.

6.2 Strato "velocità"

6.2.1 Approccio tradizionale

L'approccio più classico, consiste nell'emulazione del suono riprodotto da un motore a quattro tempi, per farlo è necessario individuare i fenomeni fisici che causano indirettamente la generazione sonora.

Nel caso del rumore dei veicoli dotati di motore a combustione interna, si parla di suono consequenziale, in quanto viene prodotto come conseguenza di una determinata funzione. Il suono dell'AVAS è invece un suono intenzionale, progettato e aggiunto ad un prodotto da un designer. [9]

All'interno di un motore a combustione interna ci sono vari elementi in movimento che determinano il funzionamento del sistema, tra questi processi le principali fonti sonore individuate sono le seguenti:

- impulsi irradiati direttamente dal blocco motore (scoppio).
- rumore dovuto alla vibrazione della carrozzeria del veicolo.
- irradiazione sonora dalla superficie del tubo di scarico, il design del sistema di scarico e in particolare il comportamento della valvola di scarico sono i principali responsabili del carattere del suono di un motore. Anche il numero dei cilindri è un fattore importante che determina il rumore emesso.
- pulsazioni dalla bocca del tubo di scappamento.
- suoni addizionali dovuti a pneumatici, cinghia, turbocompressore. [12]

La principale componente del rumore generato è dovuta al movimento dei pistoni, che è anche fonte dell'energia utilizzata dal sistema. Nel modello di generazione sonora questo fenomeno viene emulato da un fasore, un segnale di controllo che pilota la fase di un segnale oscillante. Motore, scarico e corpo del veicolo sono una rete di eccitazioni secondarie in serie. [12]

Uno script supercollider che si basa su un modello di generazione sonora basato su questi principi era già disponibile in rete. [13] Il prototipo è stato testato utilizzando questo modello, che in seguito è stato provato in combinazione alle altre tecniche utilizzate. Lo script definisce un SynthDef chiamato "engine" che riceve come parametro la velocità del veicolo e genera un segnale in tempo reale.

6.2.2 Approccio creativo

L'approccio creativo consiste nell'abbandonare l'idea di suoni associabili a quelli di un vecchio motore a combustione interna, alla ricerca di risultati innovativi e futuristici.

Una prima tecnica possibile consiste nell'utilizzo di oscillatori in parallelo. È stato quindi creato una sorta di sintetizzatore su supercollider che permettesse di aggiungere un numero indefinito di componenti, ognuna avente diverse caratteristiche come volume, frequenza o modulazione in ampiezza. Queste diverse componenti variano in un determinato range al variare della velocità del veicolo.

Per ottenere un risultato percettivamente accettabile si è partiti da una base svolta, è stato emulato un preset del plugin Igniter.

Igniter

Igniter è un plugin sviluppato da Krotos Audio, il suo utilizzo è quello di progettare il suono di veicoli (automobili, motociclette, aeroplani, elicotteri). È pensato soprattutto per il sound design di videogiochi o cortometraggi in quanto permette tramite effettistica l'emulazione del suono di veicoli in movimento all'interno di video o giochi.

Questo plugin può essere utilizzato all'interno di una DAW (digital audio workstation) e permette di registrare la performance in tempo reale, utile nel caso di sound design per un cortometraggio (infatti mentre il video viene riprodotto si possono manipolare alcuni parametri che simulano effetti sonori coerenti con le immagini e registrare la performance). [14]

Con Igniter diverse tecniche possono essere combinate per ottenere un risultato: la sintesi granulare, la riproduzione di campioni sonori registrati e la sintesi sonora tramite oscillatori in parallelo. Queste diverse tecniche possono essere combinate e bilanciate in modo da creare un mix che soddisfi le esigenze del designer.

Una libreria di preset è a disposizione dell'utente, a partire da questa si può lavorare ed ottenere i propri suoni personalizzati. Tra questi preset ne è disponibile uno appositamente creato per il suono esterno di veicoli elettrici.

Il plugin offre la possibilità di utilizzare fino a cinque oscillatori in parallelo. Per ogni oscillatore è possibile modificare i seguenti parametri:

- forma d'onda: si possono scegliere le forme d'onda base (sinusoidale, triangolare, dente di sega e quadra).
- livello sonoro: ogni oscillatore ha un diverso controllo del gain (guadagno), l'utente può così bilanciare le diverse componenti del segnale a suo piacimento.
- frequenza: il tono fondamentale di ogni oscillatore. Per ogni oscillatore su un pannello sono definiti i range tra i quali varia la frequenza al variare della velocità. I valori massimi e minimi di frequenza possono essere diversi per ogni oscillatore.
- modulazione in frequenza: è possibile introdurre una modulazione in frequenza su ogni oscillatore. Si possono specificare frequenza della modulante e l'ammontare della modulazione.
- modulazione in ampiezza: per ogni oscillatore si può introdurre una modulazione in ampiezza. Due controlli permettono di cambiare la frequenza della modulante e l'ammontare della modulazione in ampiezza.
- "blend": è un controllo che permette di miscelare tra loro diverse forme d'onda semplici per crearne più complesse. Funziona specificando due diverse forme d'onda e un valore tra 0 e 1 che rappresenta la percentuale di miscelazione delle due forme d'onda.

Il preset del suono per veicoli elettrici è composto da quattro oscillatori in parallelo, inizialmente nello script di supercollider sono stati riportati esattamente gli stessi parametri utilizzati in questo esempio.

Creazione sintetizzatore su supercollider

Per ottenere gli stessi risultati su Supercollider occorre creare un SynthDef che permetta di rendere personalizzabili tutti i parametri elencati precedentemente in fase di istanziazione di un nuovo Synth.

Il problema principale è che non esiste un modo semplice per miscelare diversi tipi di forme d'onda, è stato necessario quindi trovare una soluzione alternativa.

Sintesi Wavetable

La tecnica della sintesi Wavetable consente di memorizzare in una tabella cicli unici associabili a diverse forma d'onda (un solo periodo). Per generare il suono viene specificato un indice che rappresenta quale tra le forma d'onda memorizzate si vuole riprodurre. Essendo memorizzato un solo ciclo, il segnale viene riprodotto in modo periodico.

Una funzionalità interessante della sintesi Wavetable è la possibilità di miscelare forme d'onda adiacenti nella tabella, in questo modo è possibile ottenere il "blend".

Il primo passo necessario è la creazione di un inviluppo: il primo parametro del metodo "new" di un oggetto "Env" (inviluppo) è un array che rappresenta i livelli (ampiezza), il secondo è un array con gli intervalli temporali tra i vari livelli mentre il terzo è la curvatura, che determina la forma dell'inviluppo stesso.

Dopo aver creato quattro inviluppi, uno per ogni forma d'onda base (sinusoidale, triangolare, quadrata e dente di sega), viene usato il metodo "asSignal" che ritorna un segnale della grandezza specificata come parametro, campionando l'inviluppo con un numero di intervalli pari al parametro stesso, in questo caso 1024. Il segnale viene poi convertito in un oggetto Wavetable tramite il metodo "asWavetable".

```
~sin = Signal.sineFill(1024,1!1,0!1).asWavetable;
envtri = Env([0,1,0,-1,0]);
sig=envtri.asSignal(1024);
~tri=sig.asWavetable;
envsq = Env([1,1,-1], curve: \step);
sig=envsq.asSignal(1024);
~square=sig.asWavetable;
envsaw = Env([0,1,-1,0],[1,0,1], curve: \linear);
sig=envsaw.asSignal(1024);
~saw=sig.asWavetable;
```

Le due classi “Signal” e “Wavetable” sono molto simili, entrambe sono un array di Float ma l’oggetto Wavetable è caratterizzato da un formato particolare, non è una interpolazione lineare standard: [11]

```
Signal: [a0, a1, a2...]

Wavetable: [2*a0-a1, a1-a0, 2*a1-a2, a2-a1, 2*a2-a3, a3-a2...]
```

A questo punto i 4 oggetti Wavetable (sin, tri, square, saw) vengono caricati in un Buffer nel server (identificato dalla variabile “s”).

Un oggetto Buffer è un’astrazione a lato client di un buffer lato server. Un buffer è un metodo utilizzato solitamente per memorizzare file audio, ma può essere utilizzato per altri tipi di dato. Tecnicamente parlando un buffer nel server è un array multicanale di numeri in virgola mobile a 32 bit. [11]

Ogni buffer è identificato univocamente da un numero (bufnum) associato durante l’allocazione nel server. Se non viene specificato dall’utente, il numero assegnato automaticamente è il primo disponibile al momento dell’allocazione, in questo modo quando vengono caricati consecutivamente diversi buffer è noto che saranno identificati da numeri crescenti (il primo è il numero 0, il secondo 1, il terzo 2 e così via). Questo è importante perché la miscelazione di forme d’onda (”blend”), è possibile solo con buffer aventi numero adiacente. Per questo motivo sono stati caricati in memoria 8 buffer, ogni forma d’onda è presente due volte in memoria per permettere tutte le combinazioni possibili.

```
~triBuf=Buffer.loadCollection(s, ~tri);              // bufnum 0
~squareBuf=Buffer.loadCollection(s, ~square);        // bufnum 1
~sawBuf=Buffer.loadCollection(s, ~saw);              // bufnum 2
~sinBuf=Buffer.loadCollection(s, ~sin);              // bufnum 3
~tri2Buf=Buffer.loadCollection(s, ~tri);             // bufnum 4
~saw2Buf=Buffer.loadCollection(s, ~saw);             // bufnum 5
~sin2Buf=Buffer.loadCollection(s, ~sin);             // bufnum 6
~square2Buf=Buffer.loadCollection(s, ~square);       // bufnum 7
```

Per la generazione del segnale audio occorre una UGen chiamata “VOsc”, un oscillatore Wavetable che può leggere dalla tabella in memoria un singolo ciclo del segnale e riprodurlo periodicamente a seconda degli altri parametri, ad esempio la frequenza che determina la velocità di riproduzione, quante volte il periodo viene ripetuto in un secondo.

Il parametro “bufpos” indica quale buffer riprodurre:

```
VOsc.ar(bufpos, freq: 440.0, phase: 0.0, mul: 1.0, add: 0.0)
```

Il “blend” si può ottenere usando valori non interi di “bufpos”. Sappiamo che il buffer in posizione 0 è l’onda triangolare, mentre in posizione 1 abbiamo un’onda quadra, utilizzando come bufpos 0.5 si ottiene la seguente forma d’onda:

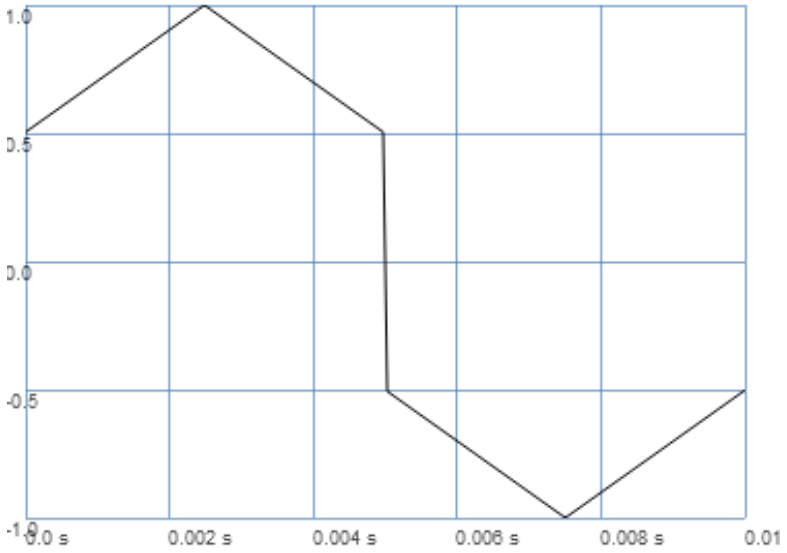

Figura 6.1: segnale a 100 Hz con forma d'onda ottenuta dal blend al 50% tra onda triangolare e onda quadra

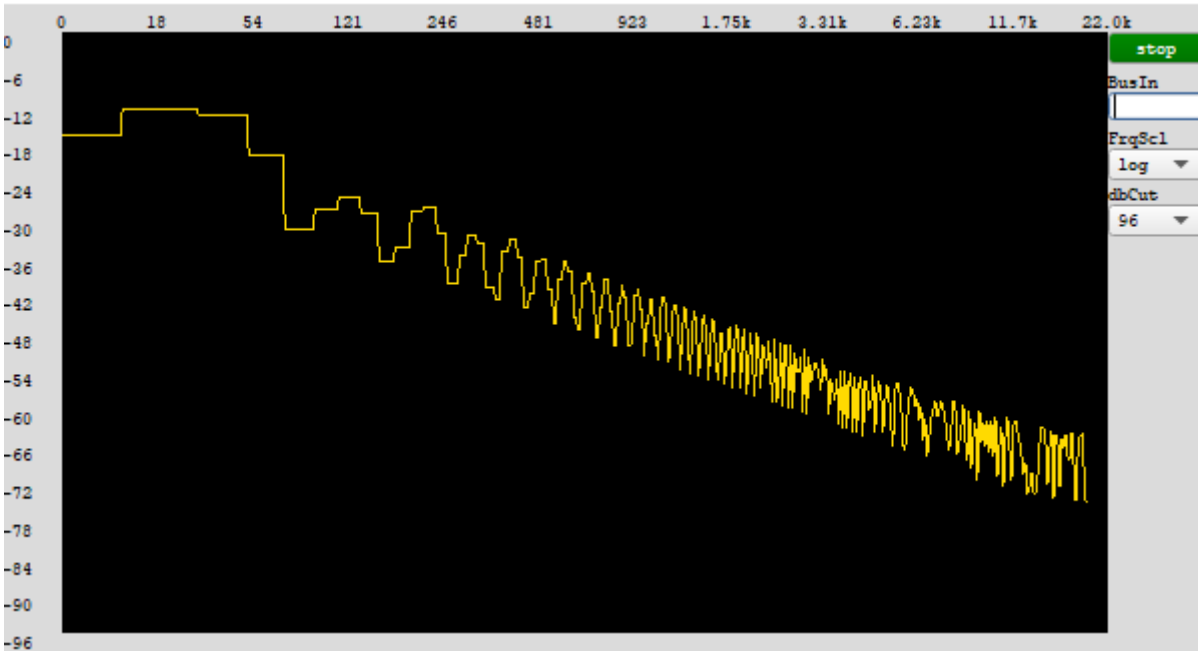

Figura 6.2: spettro in frequenza della forma d'onda risultante dal blend tra un'onda triangolare e una quadra

Infine è stato creato un SynthDef chiamato “blend” avente i seguenti parametri:

- volume: per poter settare il livello di ogni oscillatore istanziato;
- bufpos: utilizzato per scegliere la forma d’onda, è un numero intero se si vuole una delle quattro forme d’onda di base, altrimenti per miscelarle si utilizzano valori non interi.
- fm: la frequenza della modulante, in questo caso è una modulazione in ampiezza. La forma d’onda della modulante in questo esempio è fissa ed è una sinusoide. L’ampiezza della sinusoide è 0.1, è un segnale unipolare (se fosse bipolare non sarebbe una

modulazione in ampiezza ma una modulazione ad anello) in quanto viene aggiunto 1. La modulante oscilla quindi tra 1.1 e 0.9 con frequenza variabile.

- molti: è un fattore comune a tutte le componenti del segnale, utile per attenuare o aumentare il livello di tutte le componenti mantenendo il bilanciamento voluto.
- start: è la frequenza del segnale da generare.

Non è stata aggiunta la possibilità di aggiungere una modulazione in frequenza in quanto a priori si è deciso di non utilizzarla. (*Capitolo 5.3*)

Il segnale è filtrato tramite un passa banda "BPF" la cui frequenza centrale è 1000 Hz. Rq è il reciproco di Q ed è il rapporto tra larghezza di banda e frequenza centrale.

```
SynthDef.new (\blend, {
    arg out=0, bufpos=0, freq=440, start= 180, volume=0.01, molti=1 , fm;
    var modulator;
    modulator = SinOsc.ar(freq: fm,phase: 0,mul: 0.1, add:1);
    Out.ar(out, molti*volume*BPF.ar(VOsc.ar (bufpos, start, mul: modulator), freq: 1000, rq: 5 )!2);
    }).add;
```

Si possono quindi ora istanziare diversi Synth (potenzialmente infiniti) in parallelo aventi diverse caratteristiche. Nell'OSCDef "ignitionlistener" vengono istanziati i vari Synth con una frequenza iniziale, nello "speedlistener" invece viene settato dinamicamente il valore della frequenza degli oscillatori.

Le variabili "startfirst" e "endfirst" definiscono il valore minimo e massimo della frequenza del primo oscillatore (first), lo stesso vale per gli altri tre (second, third e fourth). Allo stesso modo le variabili "volumefirststart" e "volumefirstend" (e così via per gli altri Synth) definiscono valore massimo e minimo di volume.

```
OSCdef ('ignitionlistener', {
        ~first = Synth.new(\blend, [\bufpos, buftri, \start, 0.linlin(0, 20, startfirst, endfirst), \volume, 0.linlin(0, 20, volumefirststart, volumefirstend)]);
        ~second = Synth.new(\blend, [\bufpos, bufsq+ 0.4, \start, 0.linlin(0, 20, startsecond, endsecond), \volume, 0.linlin(0, 20, volumesecondstart, volumesecondend)]);
        ~third = Synth.new(\blend, [\bufpos, bufsq + 0.7, \start, 0.linlin(0, 20, startthird, endthird) ,\volume, 0.linlin(0, 20, volumethirdstart, volumethirdend)]);
        ~fourth = Synth.new(\blend, [\bufpos, bufsaw + 0.68, \start, 0.linlin(0, 20, startfourth, endfourth), \volume, 0.linlin(0, 20, volumefourthstart, volumefourthend)]);
}, "/ignition");
```

La funzione “linlin” mappa il range in entrata in un range in uscita in modo lineare. I valori in entrata sono tra 0 e 20 km/h, mentre il range in uscita è quello tra il valore minimo e il valore massimo di frequenza impostato per ogni oscillatore.

Nell’OSCDef “speedlistener” viene ricevuto il valore della velocità del veicolo dinamicamente e tramite il metodo “set” viene modificata in tempo reale sia la frequenza che il volume di ognuna delle componenti.

```
OSCdef ('speedlistener', {

        arg msg;

        ~first.set(\start, msg[1].linlin(0, 20, startfirst, endfirst), \volume, msg[1].linexp(0, 20, volumefirststart, volumefirstend));

        ~second.set(\start, msg[1].linlin(0, 20, startsecond, endsecond), \volume, msg[1].linexp(0, 20, volumesecondstart, volumesecondend));

        ~third.set(\start, msg[1].linlin(0, 20, startthird, endthird), \volume, msg[1].linexp(0, 20, volumethirdstart, volumethirdend));

        ~fourth.set(\start, msg[1].linlin(0, 20, startfourth, endfourth), \volume, msg[1].linexp(0, 20, volumefourthstart, volumefourthend));

}, “/speedEngine”);
```

Nell’OSCdef “shutdownlistener” che si attiva allo spegnimento del motore del veicolo elettrico, il suono viene interrotto e tutti i synth precedentemente istanziati vengono liberati mediante il metodo “free”.

Utilizzando i valori del preset di Igniter si ottiene un primo esempio dal quale partire per comporre diversi suoni.

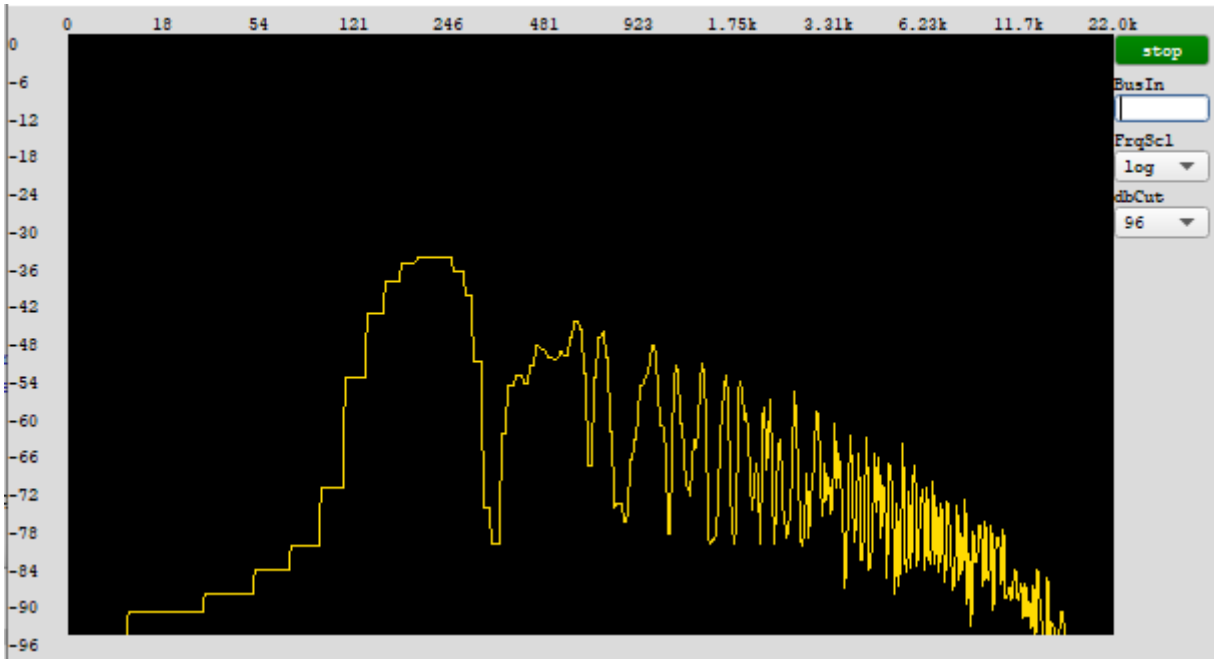

Figura 6.3: spettro in frequenza del suono del preset di Igniter a veicolo fermo

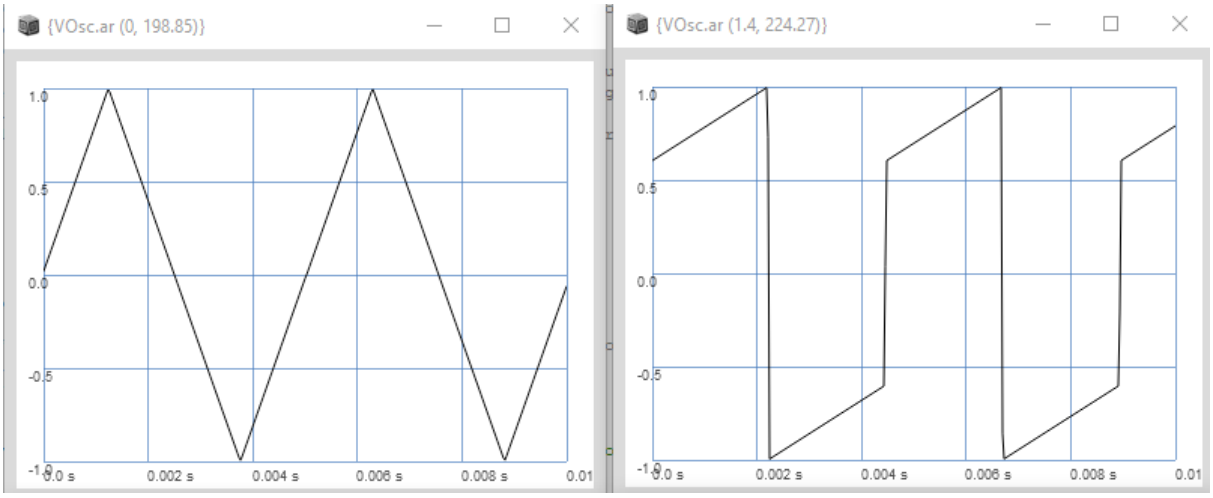

Figura 6.4: forme d'onda degli oscillatori utilizzati nel preset, a sinistra "first" a destra "second".

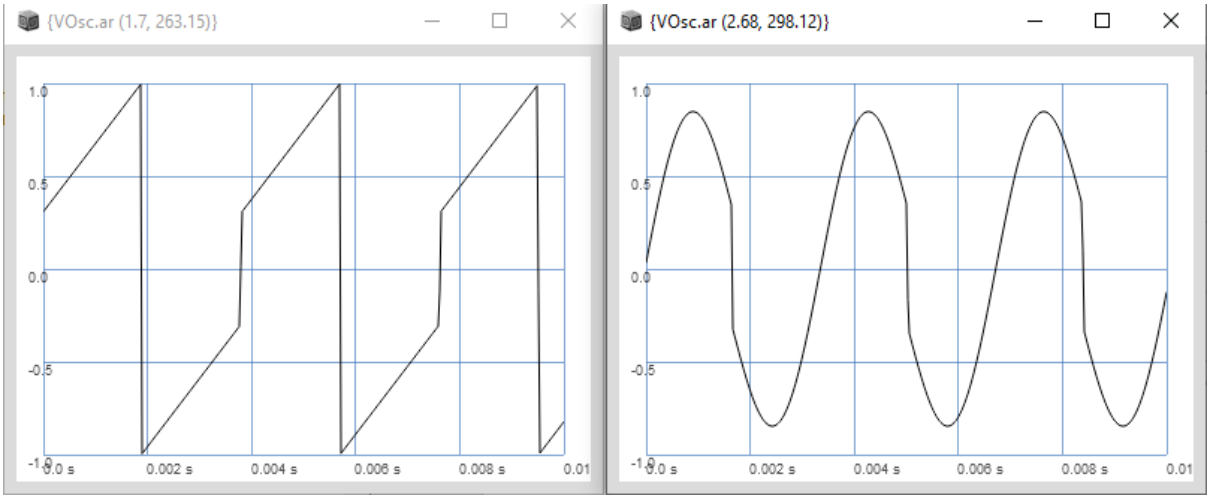

Figura 6.5: forme d'onda degli oscillatori utilizzati nel preset, a sinistra "third" a destra "fourth. I valori nella barra superiore sono rispettivamente “bufpos” e “start”, la frequenza del segnale

Variando i valori di “bufpos” (e quindi la forma d’onda) di questi quattro oscillatori, i range di frequenza e volume si sono svolte prove che hanno portato ad un risultato ritenuto soddisfacente. Teoricamente si sarebbero potuti aggiungere altri oscillatori in parallelo ma in questo progetto sono stati provati empiricamente combinazioni diverse di massimo quattro o cinque segnali.

I segnali sono stati testati solo in cuffia o tramite le casse del laptop, sarebbe stato opportuno lavorare in una sala trattata acusticamente con altoparlanti con risposta in frequenza lineare per avere una resa più fedele e maggior possibilità di raffinare il suono ottenuto.

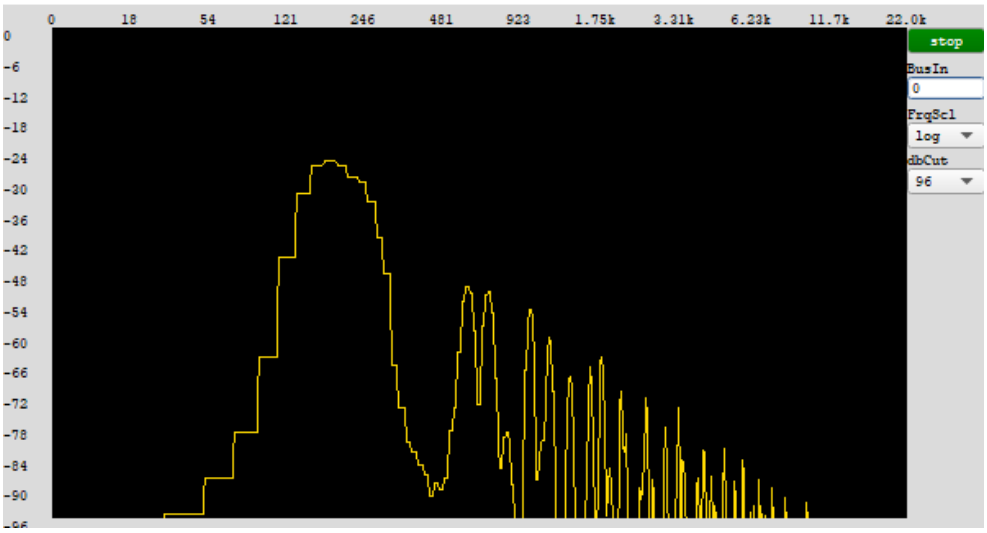

Figura 6.6: spettro in frequenza del risultato ottenuto dalla combinazione dei quattro segnali a veicolo fermo

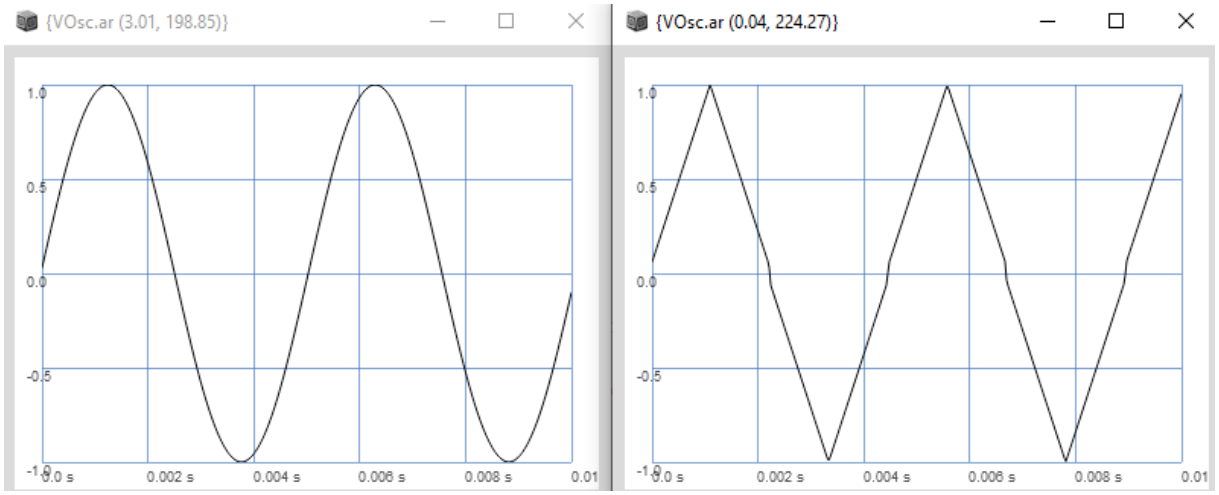

Figura 6.7: forme d'onda degli oscillatori utilizzati per il risultato ottenuto, a sinistra "first", a destra "second".

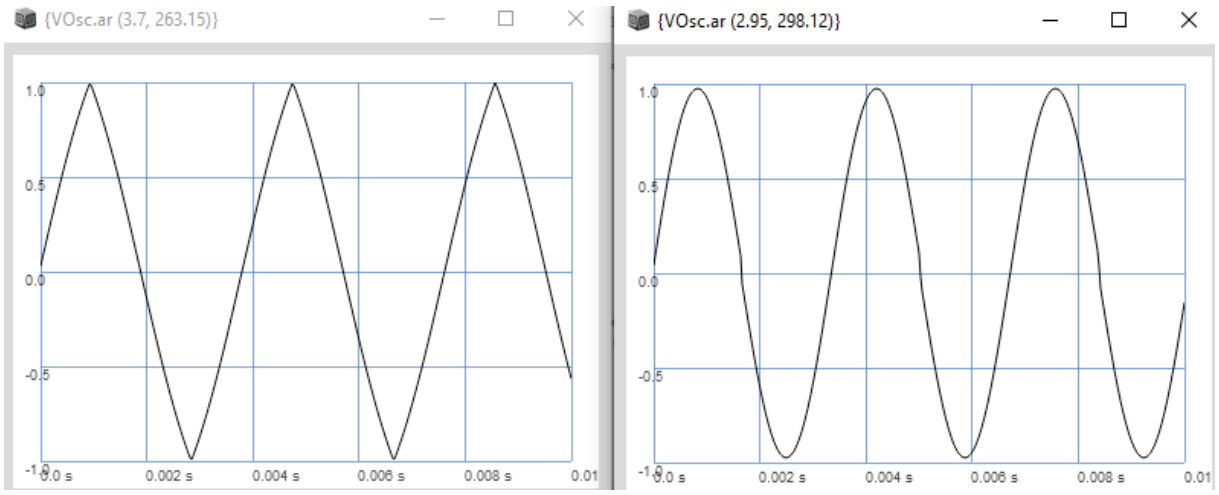

Figura 6.8: forme d'onda degli oscillatori utilizzati per il risultato ottenuto, a sinistra "third" a destra "fourth.

Sintesi granulare

La sintesi granulare è una tecnica che permette di produrre suoni difficilmente ottenibili attraverso altri metodi. Spesso viene paragonata alla tecnica pittorica del "pointillisme", la somma di frammenti sonori di dimensione microscopica viene percepita come un suono continuo. [8]

I frammenti sonori sono detti grani, ognuno di questi può avere caratteristiche diverse dal precedenti, ad esempio durata e ampiezza. I grani successivi sono sovrapposti, per questo motivo ad ognuno è applicato un inviluppo (di solito una Gaussiana) per evitare che si creino artefatti sonori o discontinuità del segnale.

La sorgente sonora dalla quale vengono estratti i grani può essere:

- sintetica: segnali continui generati da oscillatori che vengono finestrati e scomposti in segnali impulsivi.
- registrata: un segnale audio registrato. In questo caso l'operazione si chiama "granulazione", piccole porzioni del segnale vengono scomposte e successivamente sovrapposte per creare nuove timbriche.

Nel corso di questo progetto sono stati utilizzati file audio registrati per la generazione dei grani. Per permettere la granulazione in tempo reale, occorre prima di tutto caricare il file audio in un buffer. Il metodo readChannel permette di leggere solo uno dei due canali (il sinistro) in caso di file stereo in modo da avere un segnale mono, che è necessario per la sintesi granulare su supercollider.

```
~grain=Buffer.readChannel(s,"C:/…/audio.wav", channels:[0]);
```

La UGen chiamata "GrainBuf" riceve un segnale audio memorizzato in un buffer e lo usa come sorgente per la sintesi granulare.

```
GrainBuf.ar(numChannels: 1, trigger: 0, dur: 1, sndbuf, rate: 1, pos: 0, interp: 2, pan: 0, envbufnum: -1, maxGrains: 512, mul: 1, add: 0)
```

Gli argomenti del metodo "ar" applicato alla UGen GrainBuf sono:

- numChannels: il numero di canali in output, di default è 1.
- trigger: di solito è una UGen che definisce ogni quanto viene riprodotto un nuovo grano. Ad esempio utilizzando la UGen "Impulse" che genera impulsi con una certa frequenza si definiscono quanti grani al secondo vengono letti.
- dur: la durata dei grani in secondi.
- sndbuf: il buffer da utilizzare per la generazione dei grani, deve contenere un segnale mono.
- rate: la velocità di riproduzione del campione sonoro. Questo parametro è quello che viene controllato dalla velocità del veicolo, infatti aumentando la velocità di riproduzione il tono (pitch) del segnale generato risulta più acuto.
- pos: la posizione iniziale di riproduzione del grano, 0 è l'inizio e 1 la fine del file.
- interp: il metodo di interpolazione utilizzato per i grani riprodotti con velocità di riproduzione maggiore di 1.

- pan: determina dove posizionare l'output in caso di uscita stereo (se numChannels è uguale a 2).
- envbufnum: l'inviluppo applicato ai vari grani. Il default è -1 e consiste in una funzione di Hann.
- maxGrains: il numero massimo di grani sovrapposti.
- mul: fattore moltiplicato al segnale generato in uscita;
- add: per scalare il segnale in uscita aggiungendo un valore fisso.

È stato creato un nuovo SynthDef per la sintesi granulare, con le seguenti caratteristiche:

- l'uscita è mono, il numero di canali specificato è 1.
- come trigger viene utilizzata un impulso con frequenza "speedrate" che è possibile valorizzare in fase di istanziazione del Synth e modificare in tempo reale.
- la durata e la posizione iniziale del grano sono valorizzate in fase di istanziazione del Synth. Ad entrambi i valori è stata sommata una UGen "LFNoise1" che produce con frequenza 100 Hz valori all'interno di un range esponenziale specificato. Questo serve ad introdurre un fattore di casualità che risulta necessario specialmente per la posizione iniziale di riproduzione del grano. Infatti se lo startPos rimane fisso si creano degli artefatti sonori e viene percepito un fischio ad alta frequenza che risulta fastidioso. Sommando un valore casuale, anche infinitesimale, si fa in modo che la posizione iniziale del grano sia sempre diversa, e in questo modo non vengono generati artefatti. Il fattore di casualità è stato aggiunto anche alla durata, così che ogni grano abbia una durata leggermente diversa.
- il segnale è filtrato con un filtro passa banda con frequenza centrale 2500 Hz e rapporto tra larghezza di banda e frequenza centrale 1.5.
- l'argomento "volume" permette di pilotare il livello del segnale in fase di istanziazione e in tempo reale. L'argomento "molti" viene utilizzato come volume di tutte le componenti del suono in modo da aver la possibilità di alzare il livello complessivo senza alterare il bilanciamento tra le varie componenti.
- l'argomento "playbackrate" determina la velocità di riproduzione del campione e quindi la sua intonazione. Questo argomento varia con la velocità del veicolo.
- l'interpolazione è cubica (valore 4), ha costo computazionale più elevato ma offre la migliore resa quando il "playbackrate" viene modificato dinamicamente.

```
SynthDef.new (\granularspeed ,{

  arg out=0, bufnum=0, startPos=0, speedrate=100, duration=0.5, playbackrate=1, volume= 0.8, molti=1;

  Out.ar(out, molti*volume*BPF.ar(GrainBuf.ar (1, Impulse.kr(speedrate), duration + LFNoise1.kr(100).exprange(0.01,0.02), bufnum, playbackrate, startPos + LFNoise1.kr(100).exprange(0.01,0.02), 4), freq: 2500, rq: 1.5)!2);

}).add;
```

All’interno dell’OSCDef “ignitionlistener” il Synth per la sintesi granulare viene istanziato con i valori corrispondenti al veicolo fermo (velocità nulla):

```
OSCdef ('ignitionlistener', {

~grainsynth=Synth.new(\granularspeed, [\speedrate, 0.linlin(0, 20, 100, 200), \bufnum, ~grain.bufnum, \playbackrate, 0.linexp(5, 20, 0.6, 2), \startPos, kmh.linlin(0, 20, 0.8, 0.7), \volume, kmh.linlin(0, 20, 0.55, 0.45), \duration, kmh.linlin(0, 20, 0.15, 0.12),\molti, molti]);

}, "/ignition");
```

Nell’OSCDef “speedlistener” i valori dei parametri vengono variati dinamicamente a seconda della velocità del veicolo, in questo esempio:

- lo speedrate, ovvero il numero di grani al secondo, cresce da 100 fino a 200 Hz alla velocità massima considerata di 20 km/h.
- la duration varia da 0.15 a 0.12 millisecondi. I grani diventano più corti con l’aumentare della velocità (ed aumentando lo speedrate ne vengono letti più in un secondo). La posizione iniziale dei grani cambia con l’aumento della velocità, questi piccoli cambiamenti continui contribuiscono a rendere il suono più dinamico e vivo.
- la velocità di riproduzione cresce con l’aumentare della velocità, questo risulta in un aumento dell’intonazione (pitch) del suono generato.
- il volume cresce con l’aumentare della velocità del veicolo.

```
OSCdef ('speedlistener', {

    arg msg;

    ~grainsynth.set(\speedrate, msg[1].linlin(0, 20, 100, 200));

    ~grainsynth.set(\duration, msg[1].linlin(0, 20, 0.15, 0.12));

    ~grainsynth.set(\playbackrate, msg[1].linlin(0, 20, 0.6, 2));

    ~grainsynth.set(\startPos, msg[1].linlin(0, 20, 0.8, 0.7));

    ~grainsynth.set(\volume, msg[1].linexp(0, 20, 0.55, 0.45));

}, "/speedEngine");
```

Nell’OSCdef “shutdownlistener” che si attiva allo spegnimento del motore del veicolo elettrico, il suono viene interrotto e il synth “grainsynth” viene liberato mediante il metodo “free”.

```
OSCdef ('shutdownlistener', {

    ~grainsynth.free;

}, "/shutdown");
```

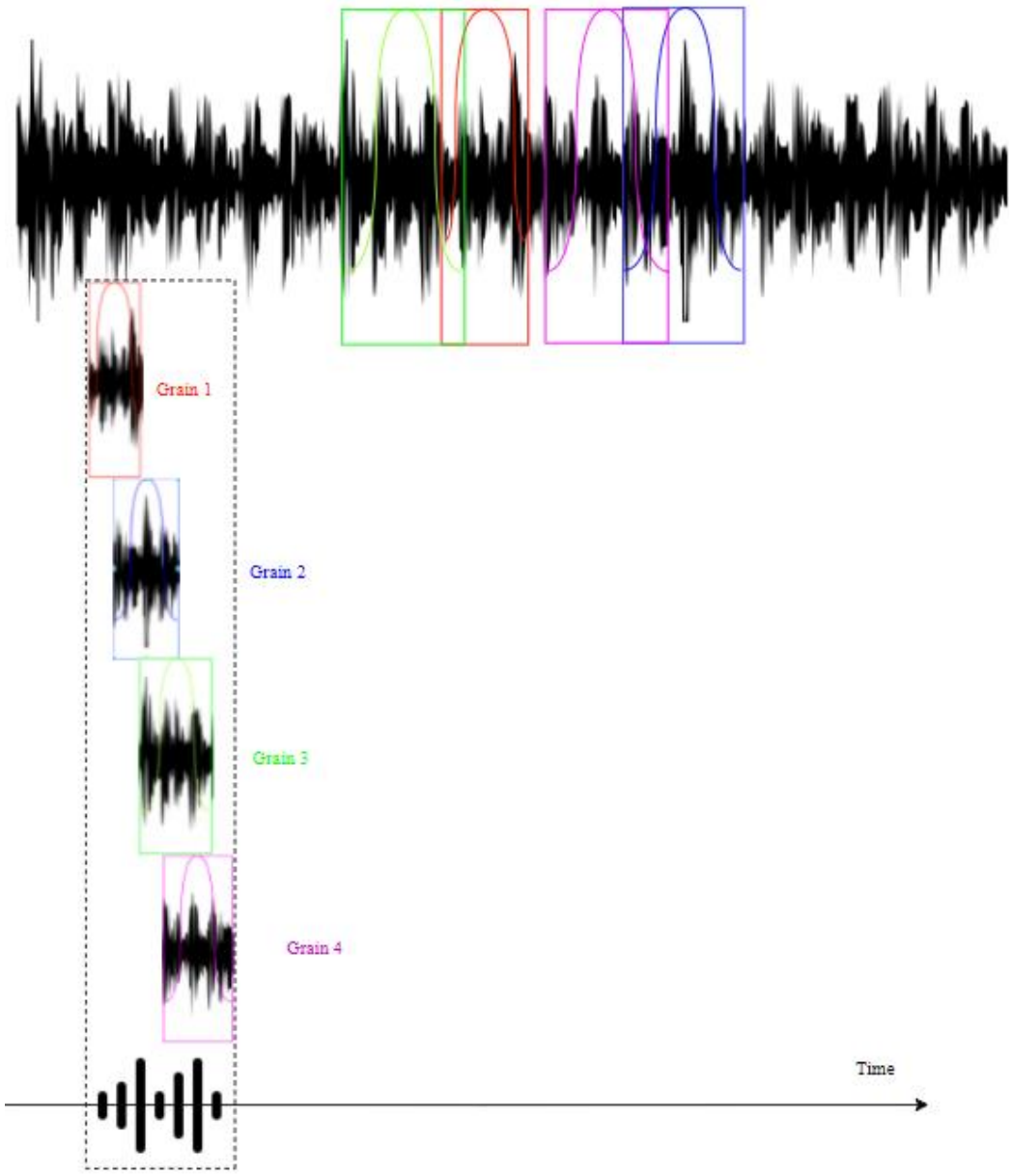

Figura 6.9: sintesi granulare, in alto la forma d'onda del segnale audio registrato, i grani vengono selezionati a seconda dei parametri "Posizione" e "Durata", viene poi applicato un inviluppo e infine sono sovrapposti.

I file audio usati come sorgente per estrarre i grani sono stati creati ad hoc utilizzando il plugin "Xpand!2" sviluppato da Air Music Technology. Questo plugin è composto da quattro parti (canali MIDI) in parallelo, ad ognuna di queste è possibile assegnare un diverso suono selezionato da una vasta libreria di timbri.

È stato utilizzato anche il software Ableton per generare il file ".wav". Inizialmente utilizzando il plugin "Xpand!2" all'interno di Ableton è stata registrata una traccia MIDI utilizzando uno o più dei suoni presenti nella libreria. La traccia MIDI risultante è stata "congelata" in modo da trasformarla in traccia audio e poter esportare un file in formato ".wav". Da questi file audio generati sono stati ottenuti risultati diversi utilizzando la sintesi granulare attraverso il SynthDef "granularspeed".

Sono stati utilizzati 7 suoni con caratteristiche diverse:

- S1 è un violino, un suono caldo con attacco morbido. La traccia MIDI consiste in una singola nota suonata in modo continuo.
- S2 è un pad (tono generato da un sintetizzatore spesso usato per creare un atmosfera o armonia di sottofondo) caratterizzato da un suono brillante, ricco di frequenze medio-alte. La traccia MIDI in questo caso consiste in una dissonanza, un DO e un SI suonati contemporaneamente (un bicordo, ovvero un accordo formato da due note).
- S3 è un coro femminile: in questo caso il coro intona un accordo consonante (DO SOL DO).
- S4 è un coro femminile dissonante: lo stesso coro usato precedentemente ma con un accordo dissonante (DO FA# SI)
- S5 è un drone: suono molto discontinuo e con poco contenuto armonico.
- S6 è la combinazione di due pad: il primo aggressivo e dinamico, il secondo più caldo suggerisce un'atmosfera sognante. La traccia MIDI registrata consiste di un suono consonante (FA LA)
- S7 è un organo combinato con un pad aggressivo dal timbro metallico. Il suono registrato è dissonante (DO FA#).

6.3 Strato "potenza"

Lo strato sonoro dedicato alla coppia motrice serve ad introdurre la percezione uditiva della potenza del veicolo. In questo progetto questo strato è controllato dalla posizione del pedale acceleratore (in quanto più facile da simulare grazie al sensore ToF), di solito viene utilizzata la coppia motrice. Questa approssimazione nasce dalla considerazione intuitiva che, a parità di velocità, se il pedale dell'acceleratore è più premuto il livello di potenza dell'automobile risulta maggiore in quanto proporzionale ai giri motore.

Per introdurre la percezione di potenza sono possibili due approcci:

- variare dinamicamente alcuni parametri della sintesi granulare precedentemente descritta. Sono stati effettuati tentativi in tal senso ma non è stato raggiunto un risultato soddisfacente.
- utilizzare un nuovo Synth con parametri controllati dalla posizione del pedale, questo nuovo segnale viene bilanciato con quelli precedenti. In questo progetto si è scelto di percorrere questa strada.

```
SynthDef.new (\granularpedal ,{

  arg out=0, bufnum=0, startPos=0.6, speedrate=400, duration=0.5, playbackrate=0.16, volume=0.08, molti=1;

      Out.ar(out, molti*volume*BPF.ar(GrainBuf.ar (1, Impulse.ar(speedrate), duration + LFNoise1.kr(100).exprange(0.001,0.005) , bufnum, playbackrate, startPos + LFNoise1.kr(100).exprange(0.1,0.2), 2), freq: 1000, rq: 1.4)!2);

}).add;
```

Per la generazione del segnale pilotato dalla posizione del pedale, si è utilizzata nuovamente la tecnica della sintesi granulare. È stato creato un SynthDef chiamato “granularpedal”.

Un Synth viene istanziato all’accensione del veicolo, nell’OSCDef “ignitionlistener”, a volume nullo.

Nell’OSCdef “gaspedallistener” viene cambiato dinamicamente il volume. Inizialmente parte ad un livello molto basso, a pedale completamente premuto invece il livello di questa componente è molto più alto, all’inizio è appena percepibile e diventa gradualmente più riconoscibile.

```
OSCdef ('gaspedallistener', {
    arg msg;
        if (msg[1]>10){
        ~grainpedal.set(\volume, LinLin.kr(msg, 10, 100, 0.0005, 0.13));
        }
}, "/gaspedalpos");
```

In questo caso il segnale viene generato solo oltre un certo livello. Nel prototipo il valore della posizione del pedale è compreso tra 0 e 100, la componente sonora legata alla potenza viene generata solo a partire da una certa soglia (10).

Nell’OSCdef “shutdownlistener” che si attiva allo spegnimento del motore del veicolo elettrico, il suono viene interrotto e il synth “grainpedal” viene liberato mediante il metodo “free”.

La sorgente sonora utilizzata è il suono di una matita sul foglio, registrata col microfono di un cellulare.

Tramite il software “Audition” della Adobe, il file è stato manipolato in frequenza, a partire dallo spettrogramma è stata selezionata una parte del segnale. Lo spettrogramma è una rappresentazione grafica dell’intensità di un suono in funzione del tempo (asse x) e frequenza (asse y), a ciascun punto del grafico è assegnata una tonalità che ne rappresenta l’intensità sonora in un dato istante di tempo e ad una data frequenza. In *Figura 6.10* lo spettrogramma del segnale risultante. [15]

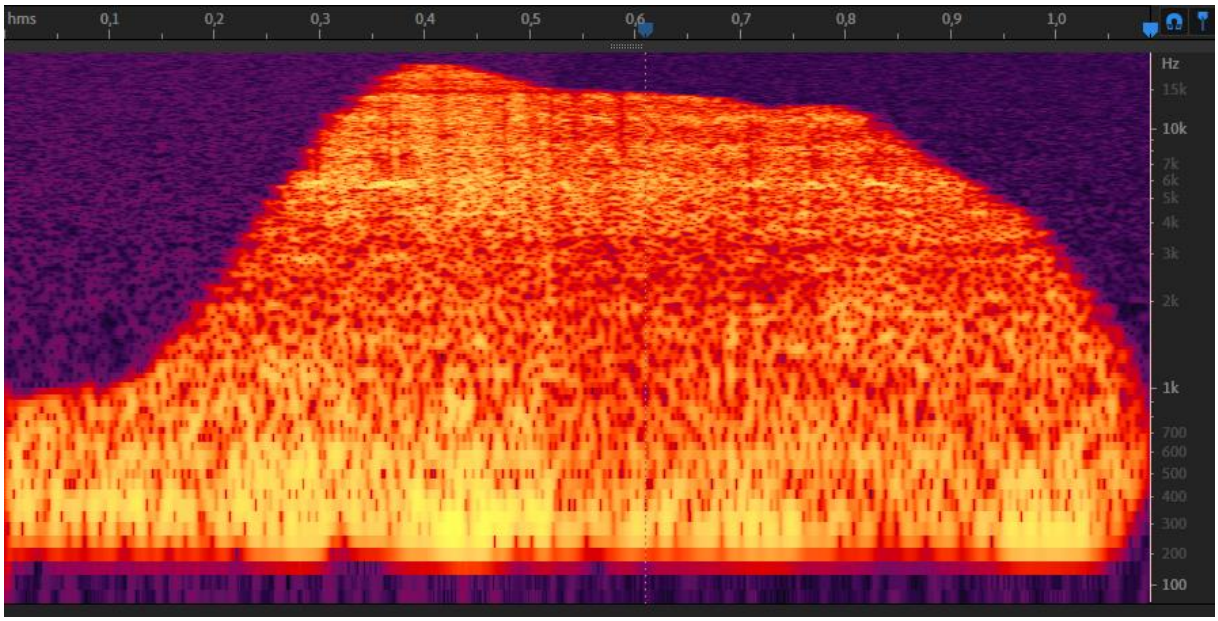

Figura 6.10: spettrogramma del file audio utilizzato come sorgente per lo strato "potenza"

6.4 Strato di "realismo aggiuntivo"

Il terzo strato sonoro è costituito da una componente fissa a frequenze medio-basse, la cui funzione è svolgere da collante tra i vari strati e aggiungere maggior realisticità al suono complessivo.

Per ottenere questo risultato un suono registrato viene riprodotto in loop. La UGen "PlayBuf" consente la riproduzione in loop di un file audio memorizzato in un buffer. Un nuovo SynthDef "scoppio" è stato creato.

```
SynthDef.new (\scoppio ,{
    arg out=0, bufnum=0, rate=1, startPos=0, loop=1, volume=0.8, molti=1;
    var signal;
    signal = PlayBuf.ar(1, bufnum: bufnum, rate:rate, trigger: 1.0, startPos: startPos, loop: loop);
    Out.ar(out, molti*volume*signal!2);
}).add;
```

Nell'OSCdef "ignitionlistener" viene istanziato un Synth "scoppiosynth" al quale viene passato un buffer e alcuni parametri quali il rate (velocità di riproduzione del campione), la posizione iniziale (0 è inizio file), il volume e il parametro "loop", se è 1 vuol dire che il file deve essere riprodotto in loop.

```
~scoppiosynth=Synth.new(\scoppio, [\rate, 0.linlin(0, 20, 1.1, 1.15), \bufnum, ~scoppio.bufnum, \startPos, 0, \loop, 1, \volume, 0.linlin(0, 20, 0.08, 0.07), \molti, molti]);
```

Nell'OSCdef "speedlistener" vengono modificati il rate e il volume in modo dinamico, aumentando la velocità del veicolo aumenta leggermente anche la velocità di riproduzione (e quindi il tono o pitch percepito) e diminuisce il volume. Quando il veicolo è fermo, questo suono riprodotto in loop (che rappresenta il rumore dello scoppio del motore), è ad un livello alto e risulta facilmente distinguibile, man mano che il veicolo avanza le altre componenti legate a velocità e potenza diventano predominanti.

```
OSCdef ('speedlistener', {
        arg msg;
        ~scoppiosynth.set(\rate, msg[1].linlin(0, 20, 1.1, 1.15), \volume, msg[1].linexp(0, 20, 0.08, 0.06));
}, "/speedEngine");
```

Nell'OSCdef "shutdownlistener" che si attiva allo spegnimento del motore del veicolo elettrico, il suono viene interrotto e il synth "scoppiosynth" viene liberato mediante il metodo "free".

La sorgente sonora utilizzata è stata la registrazione dello scoppio di un motore. Questo segnale è stato manipolato mediante dei plugin tramite l'utilizzo del software Ableton, l'obiettivo era quello di trasformare il suono in uno più continuo ed etereo, armonico, che fosse percettivamente più "elettrico".

Per raggiungere questo risultato sono stati utilizzati i seguenti plugin:

- risuonatore: il suono del motore a scoppio è un rumore, non ha un'intonazione chiara. Tramite questo plugin il segnale diventa armonico e percettivamente sembra prodotto da una sorgente "elettrica".
- filtro passa banda: per selezionare solo una certa banda di frequenze, indicativamente tra i 100 e i 1000 Hz.
- phaser: effetto elettronico che si basa sulla somma del segnale originale e di una sua copia sfasata. [16] Questo effetto contribuisce a rendere il suono più etereo.
- compressore: effetto che comprime il segnale quando oltrepassa una certa soglia in modo da diminuirne la dinamica e renderlo più uniforme.
- fltro passa banda: un altro filtro passa banda utilizzato per eliminare alcune alte frequenze introdotte dal Phaser.

In *Figura 6.11* lo spettrogramma (generato con Audition) del segnale generato, come si può notare il segnale è concentrato tra i 100 e 1000 Hz.

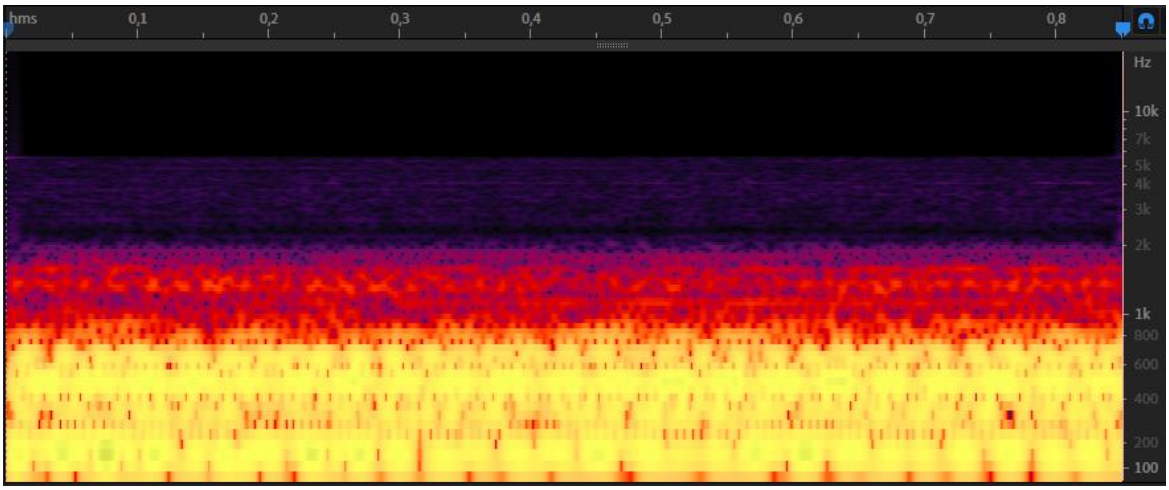

Figura 6.11: spettrogramma del segnale riprodotto in loop

6.5 Attenuatore

Come specificato dalla normativa europea, l'utente deve avere a disposizione un metodo per attenuare il livello sonoro quando il veicolo è in movimento. In questo progetto è stata supposta la presenza di un bottone che quando premuto mandi un segnale CAN. Questo messaggio viene intercettato dallo script Python su Raspberry che di conseguenza invia un messaggio OSC al server supercollider, che si occupa di attenuare il suono delle varie componenti.

Su supercollider è stato definito un OSCdef "attenuationlistener". Il valore della variabile "molti" è un volume comune a tutte le componenti sonore, in questo modo viene attenuato il volume mantenendo il bilanciamento tra le parti.

```
OSCdef ('attenuationlistener', {
        arg msg;
        var attenuator=0.5;
        if (msg[1]==0){
                attenuator=1;
        };
        ~first.set(\molti, attenuator);
        ~second.set(\molti,attenuator);
        ~third.set(\molti, attenuator);
        ~fourth.set(\molti, attenuator);
        ~scoppiosynth.set(\molti, attenuator);
        ~grainsynth.set(\molti, attenuator);
        ~grainpedal.set(\molti, attenuator);
```

Il messaggio OSC è associato ad un valore, quando il valore ricevuto è 0 vuol dire che il suono era già stato attenuato in precedenza e deve tornare al livello normale.

6.6 Retromarcia

In questo progetto non è stato progettato un suono differente per la retromarcia, viene utilizzato lo stesso prodotto in avanzamento. L'unica differenza è che a veicolo fermo il suono viene generato ad un livello maggiore se la retromarcia è inserita, come osservato in uno dei modelli elettrici in commercio. Un veicolo che esce da un parcheggio in retromarcia è più pericoloso per pedoni o altri veicoli, quindi produce un segnale più facilmente rilevabile.

Il messaggio OSC che indica la retromarcia è associato ad un valore che ne rappresenta lo stato, se il valore è 0 vuol dire che la retromarcia è stata disinserita e il segnale deve tornare al livello originario.

```
OSCdef ('reverselistener', {
        arg msg;
        var mul=1.5;
        if (msg[1]==0){
                mul=1;
        };
        ~first.set(\molti, mul);
        ~second.set(\molti,mul);
        ~third.set(\molti, mul);
        ~fourth.set(\molti, mul);
        ~scoppiosynth.set(\molti, mul);
        ~grainsynth.set(\molti, mul);
        ~grainpedal.set(\molti, mul);
}, "/reversegear");
```

Oltre i 20 km/h il suono prodotto dall'AVAS non è più obbligatorio in quanto è sufficiente il rumore emesso dal rotolamento degli pneumatici, le varie componenti vengono gradualmente attenuate oltre questa velocità all'interno dell'OSCdef "speedlistener".

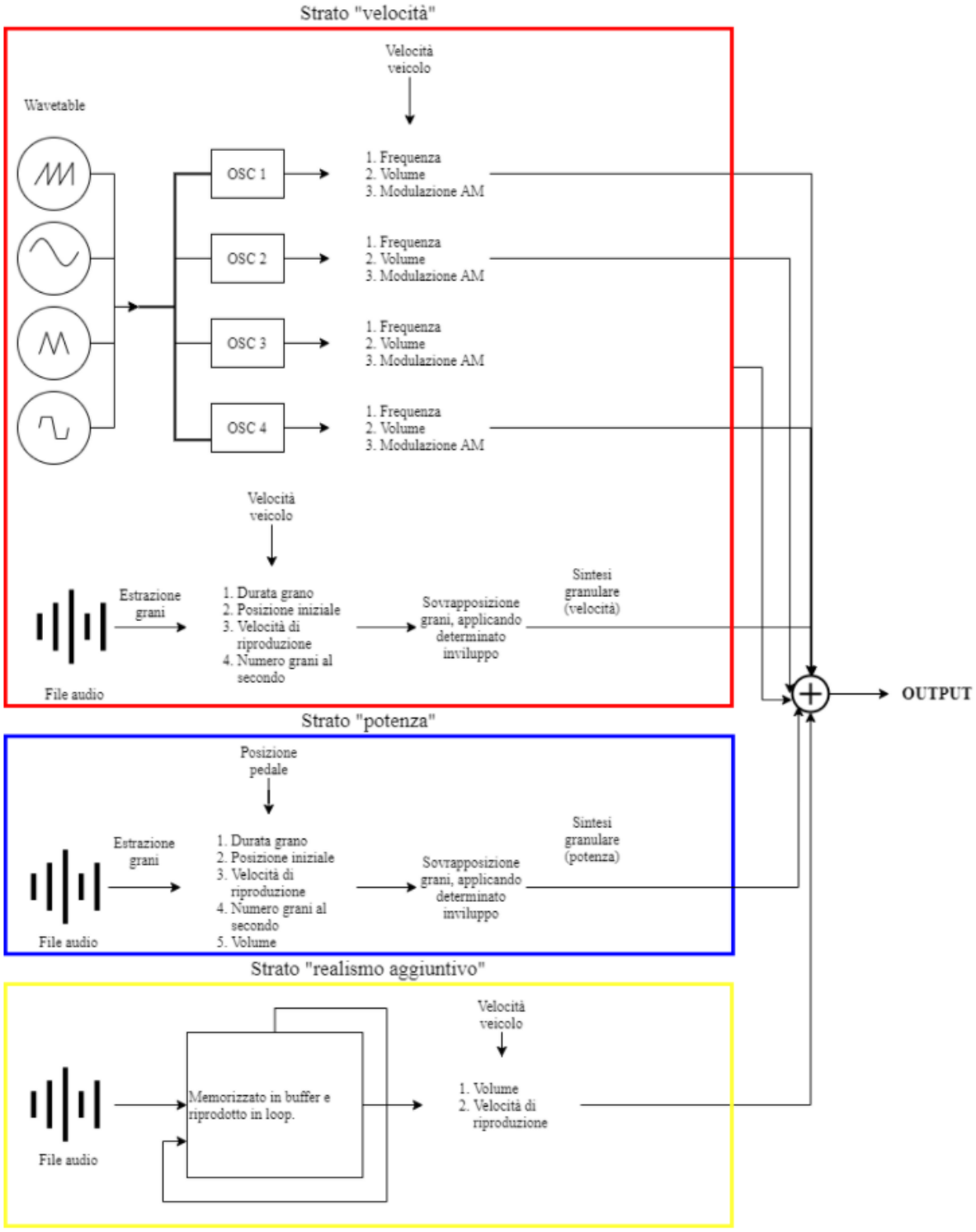

Figura 6.12: componenti del suono dell'AVAS

7 Prototipo con Raspberry PI

Una volta progettato il suono dell'AVAS, è stato costruito un prototipo in grado di leggere i dati dalla rete del veicolo e modificare le caratteristiche del suono in tempo reale.

I componenti utilizzati sono:

- scheda di controllo Raspberry PI 4, con sistema operativo Ubuntu. Uno script python utilizza la libreria "cantools" per leggere e decodificare i messaggi CAN (Controller Area Network) e che comunica col server supercollider tramite protocollo OSC. Lo script supercollider gira sulla stessa scheda.
- CAN bus shield: dispositivo che permette l'interfacciamento di una scheda Raspberry con la rete di un veicolo elettrico, in modo da poter leggere i messaggi CAN via porta seriale.

7.1 Controller Area Network (CAN)

Il Controller Area Network è uno standard seriale per bus di campo, di tipo multicast. È utilizzato soprattutto in ambiente automotive, per collegare diverse unità di controllo elettronico anche in ambienti molto disturbati dalla presenza di onde elettromagnetiche. Essendo un bus multicast, è molto semplice aggiungere o rimuovere dispositivi, in quanto non è necessario modificare il sistema. [17]

In una rete CAN i messaggi (o frame) sono di quattro tipi:

- Data frame: trasporta i dati trasmessi da un nodo ad un altro.
- Remote frame: trasmesso da un nodo per ottenere un Data Frame con un determinato identificatore
- Error frame: trasmesso da un nodo che rileva un errore. La rete CAN è una rete molto affidabile in quanto ci sono cinque diverse tecniche per la rilevazione di errori e un procedimento di auto-diagnosi: quando viene trovato un errore, il nodo trasmette un error frame e il messaggio viene ritrasmesso.
- Overload frame: se un nodo è occupato invia un overload frame per ritardare la trasmissione di un pacchetto successivo.

La struttura di un data frame CAN è riportata in *Tabella 7.1*.

Nome del campo	**Numero di bit**
Start of frame (inizio del messaggio)	1
Identificatore	11
Richiesta remota di trasmissione (RTR)	1
Bit aggiuntivo di identificazione	1
Bit riservato (r0)	1
Codice lunghezza dati (DLC)	4

Campo dati	0-8 byte
CRC (controllo parità)	15
Delimitatore CRC	1
ACK	1
Delimitatore ACK	1
End of frame (fine del messaggio)	7

Tabella 7.1: struttura di un data frame CAN [17]

Il protocollo di comunicazione è stato standardizzato negli strati di basso livello (scambio dati o data link, fisico e rete). Allo strato applicativo ogni progettista può apportare modifiche. Ogni casa automobilistica ha un database di segnali, di solito in formato "dbc". Questo database viene utilizzato per decodificare i messaggi CAN ricevuti.

Nel corso del progetto è stato utilizzato un dbc di una famosa casa automobilistica. I segnali utilizzati sono:

- ignition: indica lo stato della chiave d'accensione, a seconda dell'angolo d'inserimento si sa lo stato del veicolo.
- reversestatus: lo stato della retromarcia, se ha valore 1 è inserita.
- speed: la velocità istantanea del veicolo.
- gaspedalposition: la posizione del pedale dell'acceleratore.

7.2 Can bus shield

Il can bus shield è un dispositivo che permette di ricevere i messaggi dalla rete del veicolo in una scheda esterna, in questo caso il Raspberry PI 4.

In questo progetto è stato usata la scheda "USB2CAN" di Innomaker, un dispositivo che permette il collegamento diretto tramite USB, è inoltre compatibile con Raspberry PI 4. La scheda non ha bisogno di alimentazione esterna, si carica direttamente tramite USB. La velocità di trasmissione dei messaggi CAN può essere programmata da un minimo di 20 Kbps ad un massimo di 1Mbps. [18]

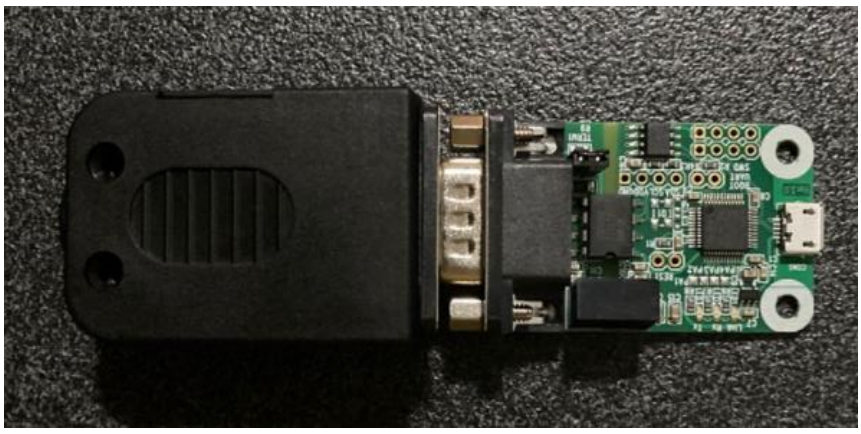

Figura 7.1: scheda USB2CAN di Innomaker

Solitamente vengono utilizzati due shield, uno funge da trasmettitore e un altro da ricevitore, ma è stato possibile lavorare anche con una sola scheda utilizzandola in loopback, in questo modo la stessa scheda funge sia da trasmettitore che ricevitore.

Can-utils

Per verificare la comunicazione di questo modulo viene utilizzato "can-utils", uno strumento ottimizzato per sistemi Linux che permette la gestione dei messaggi CAN. Direttamente dal terminale grazie a questa interfaccia è possibile inizializzare la scheda shield e inviare e ricevere messaggi CAN direttamente da terminale.

I comandi utilizzati da terminale sono i seguenti:

- candump: utilizzato per ricevere messaggi CAN.
- cangen: comando che invia frame CAN con una certa frequenza. È possibile specificare ID del frame.
- cansend: utilizzato per generare un singolo data frame, oltre all'ID è possibile specificare la lunghezza del pacchetto e i dati.

Cantools

La libreria "cantools" di python permette di utilizzare i comandi di "can-utils" in uno script python, inoltre è possibile decodificare e codificare frame CAN ricevuti. Per la codifica viene utilizzato un database in formato "dbc".

Di seguito un esempio di decodifica di un segnale CAN su python, utilizzando la libreria "cantools":

```
db = cantools.database.load_file('*.dbc')

messaggio = db.decode_message(messaggio.id, messaggio.data)
```

Una volta decodificati i messaggi, tramite un serie di condizionali "if" viene comparato l'id del messaggio con quello dei segnali che interessano (quindi velocità, posizione del pedale, accensione o spegnimento). Se l'ID del messaggio CAN ricevuto è uguale ad uno degli ID di questi messaggi, viene inviato un messaggio tramite OSC al server supercollider che modifica il suono opportunamente.

In *Figura 7.2* lo schema del prototipo che legge i messaggi del veicolo e genera il suono. Il Raspberry è collegato ad un altoparlante che si occupa di irradiare il suono.

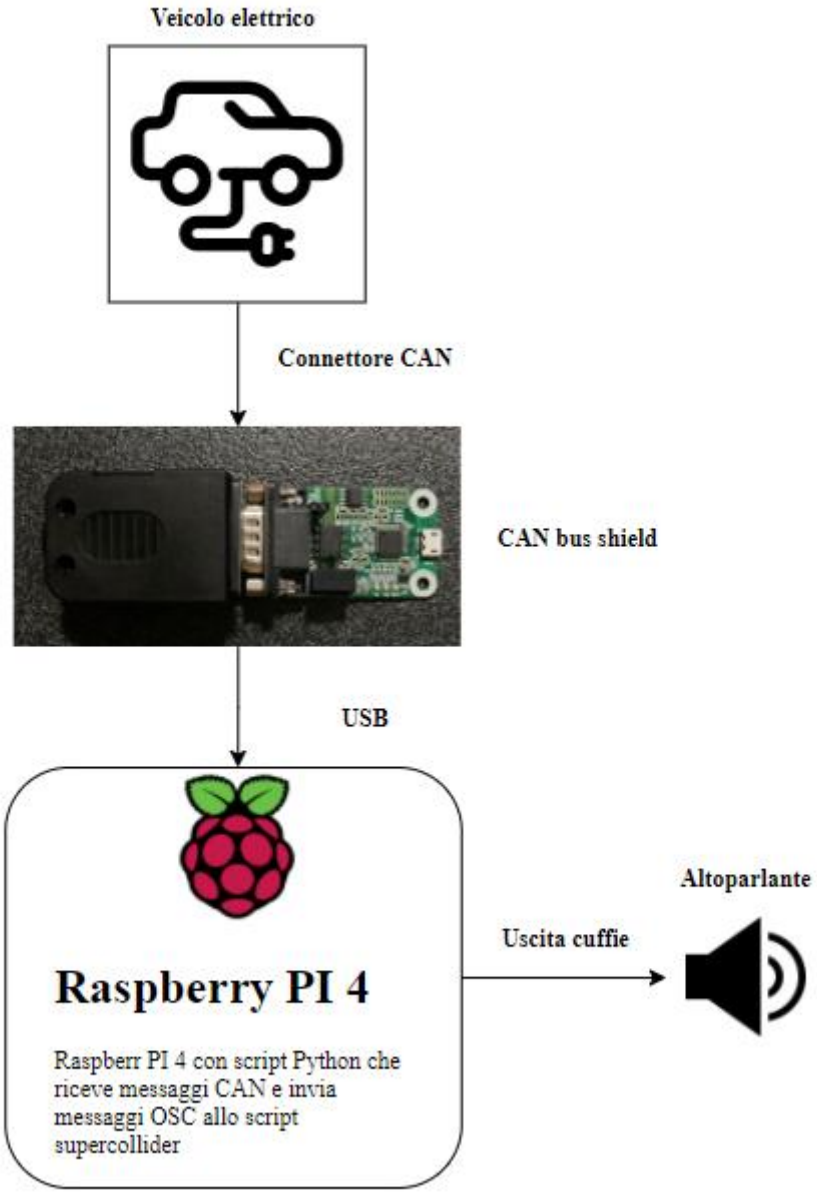

Figura 7.2: prototipo che legge dati dal veicolo e genera il suono

8 Verifica in camera anecoica

L'ultima fase di questo progetto è stata la verifica che i suoni progettati rispettassero le specifiche stabilite dalla normativa europea.

Nella Regolamentazione n.138 della UNECE vengono elencate le metodologie e condizioni di misurazione necessarie:

- per effettuare la misurazione occorre usare un fonometro (o mezzo equivalente) che soddisfi i requisiti della classe 1. Il sistema deve essere verificato tramite un calibratore acustico.
- le misure vanno effettuate usando la curva di ponderazione "A" in frequenza.
- le prove possono essere effettuate all'aperto o al chiuso. In caso di prova all'aperto, il luogo dev'essere piano e libero nel raggio di 50 m dal centro della pista di oggetti capaci di riflettere. In prossimità dei microfoni non devono esserci ostacoli che possano influenzare il campo acustico. La prova può anche essere effettuata in camera semianecoica o anecoica al chiuso.
- la prova all'aperto viene effettuata con veicolo in movimento, mentre all'interno il veicolo è fermo e vengono simulati i segnali provenienti dalla rete del veicolo stessa. In questo progetto la prova verrà effettuata in un ambiente chiuso.
- utilizzare due fonometri, uno a sinistra l'altro a destra della sorgente, posizionati come in *Figura 8.1*.

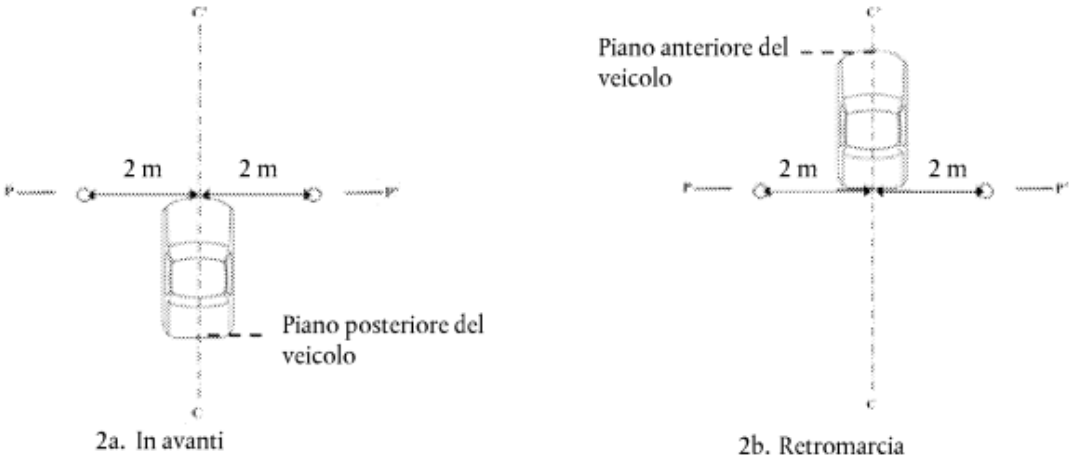

Figura 8.1: posizioni di misurazione per i veicoli in movimento all'interno e da fermi. [6]

- occorre misurare il rumore di fondo per almeno dieci secondi e riportare lo spettro in frequenza in terzi d'ottava corrispondente al livello massimo rilevato tra le due posizioni di registrazione.
- calcolare la differenza tra valore massimo e minimo del livello di pressione sonora ponderato del rumore di fondo. A seconda di questo valore, può essere necessaria una correzione del livello di pressione sonora ponderato in base alla curva A del veicolo.

Correzione per il rumore di fondo		
Intervallo dal valore massimo al valore minimo del livello di pressione sonora ponderato A del rumore di fondo rappresentativo per un determinato periodo di tempo ΔLbgn, p-p in dB(A)	Livello di pressione sonora del risultato della prova j-th meno il livello del rumore di fondo ΔL = Ltest,j – Lbgn in dB(A)	Correzione in dB(A) Lcorr
-	ΔL>=10	Nessuna correzione necessaria
<= 2	8<=ΔL<10	0,5
	6<=ΔL<8	1
	4,5<=ΔL<6	1.5
	3<=ΔL<4,5	2,5
	ΔL<3	Non può essere riportata alcuna misurazione valida

Tabella 8.1: correzione per il livello del rumore di fondo quando si misura il livello di pressione sonora ponderato A del veicolo [6]

- per l'analisi in terzi di ottava, il livello di rumore di fondo nelle bande di interesse deve essere inferiore di almeno 6 dB (A) rispetto alla misurazione dell'AVAS nella stessa banda di interesse.
- i microfoni devono essere posizionati all'altezza di 1,2 m sopra il livello del suono. La direzione di riferimento della capsula microfonica è quella orizzontale, parallela al terreno.
- il veicolo deve avere batteria carica, se ci sono più modalità operative deve essere selezionata la modalità che produce le emissioni sonore più basse per stabilire se il livello sia al di sopra del livello minimo (*Tabella 2.2*). Gli pneumatici utilizzati devono essere quelli selezionati dal costruttore del veicolo, devono essere gonfiati alla pressione raccomandata dal costruttore stesso. Se il veicolo è dotato di motore a combustione interna, questo dev'essere spento.

La verifica consiste in due procedure:

- prova per il livello sonoro del veicolo: per ognuno dei suoni vengono effettuate prove a velocità costante a 10 km/h e 20 km/h (in caso di veicolo fermo vanno simulate queste condizioni). Su entrambi i lati del veicolo vanno effettuate almeno quattro misurazioni da 5 secondi per ogni condizione di prova. Va inoltre effettuata una prova del suono in retromarcia alla velocità costante di 6 km/h.
- prova per la variazione in frequenza del veicolo: il microfono deve essere posizionato di fronte alla sorgente sonora, alla stessa altezza circa e ad 1 metro di distanza. L'emissione sonora deve essere misurata alle velocità simulate da 5 km/h a 20 km/h in

gradini di 5 km/h. Per ognuna di queste condizioni di prova il rumore deve essere misurato quattro volte per almeno 5 secondi.

8.1 Condizioni di verifica e strumentazione utilizzata

Il test è stato effettuato nella camera anecoica del Politecnico di Torino.

La camera anecoica è un ambiente strutturato in modo da ridurre il più possibile la riflessione acustica delle pareti, in modo da simulare condizioni di spazio aperto di dimensione infinita senza oggetti riflettenti. [19]

Figura 8.2: parete della camera anecoica del Politecnico di Torino

Per effettuare le misurazioni è stato utilizzato il Bruel & Kjaer type 2250, un fonometro di classe 1 a canale singolo.

Il microfono di questo fonometro è il modello 4950 con le specifiche riportate in *Tabella 8.2*.

Tensione di polarizzazione (prepolarizzato)		0 V
Sensibilità a 250 Hz		50 mV/Pa +-2dB
Risposta in frequenza a campo libero (250 Hz)		+-2B, da 6.5 a 16000 Hz
Limite inferiore in frequenza		Da 1 a 5 Hz
Capacità condensatore		12.5 pF
Limite superiore del range dinamico (distorsione 3%)		>142 dB, SPL (RMS)
Massimo livello di pressione sonora		>160 dB
Frequenza di risonanza del diaframma		12.5 kHz
Diametro	con griglia	13.2 mm
	senza griglia	12.7 mm
Altezza	con griglia	14.9 mm
	senza griglia	14.0 mm

Tabella 8.2: principali caratteristiche del microfono Bruel & Kjaer tipo 4950 [20]

L'analizzatore portatile è composto da 11 pulsanti e un display touch 240 x 320, in bianco e nero. L'utente tramite i pulsanti può controllare la misurazione e visualizzare i risultati sullo schermo.

Prima di iniziare la serie di misurazioni, il microfono è stato calibrato utilizzando il calibratore Larson Davis CAL200, che produce output ad un livello di 94 o 114 dB ad una frequenza di 1kHz.

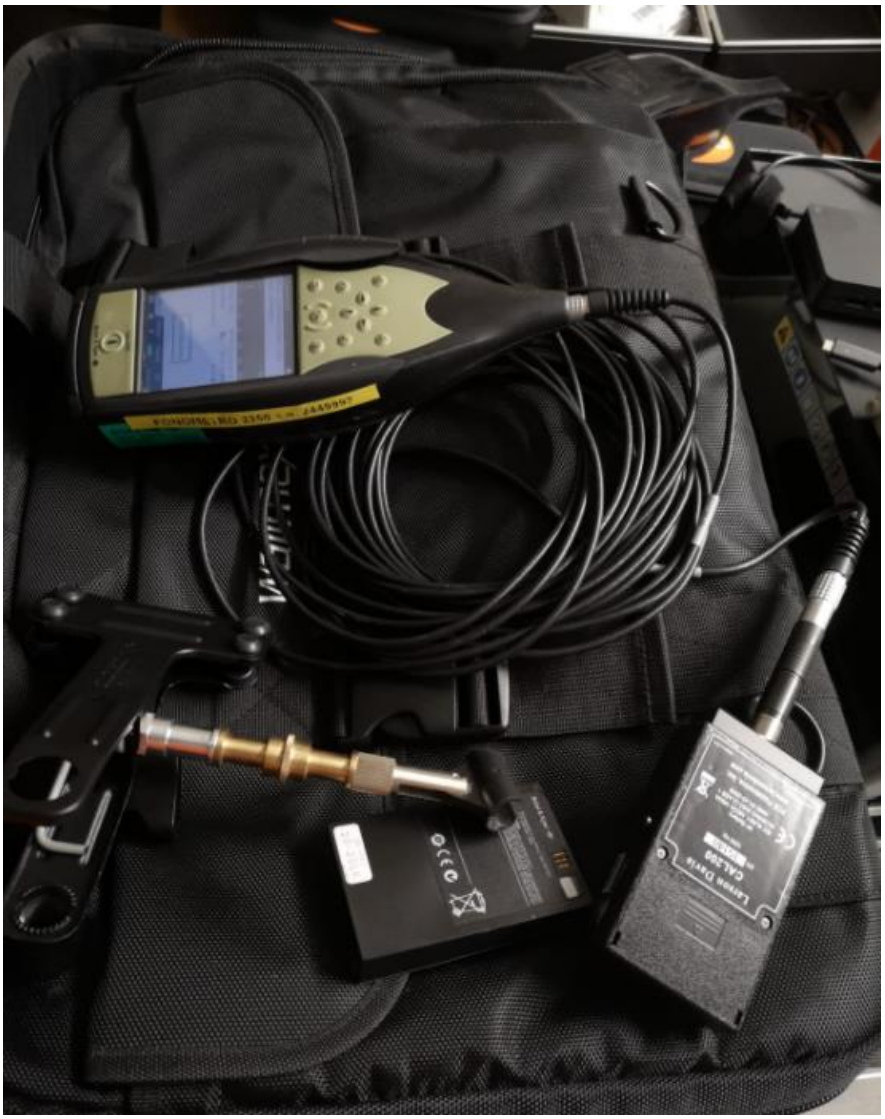

Figura 8.3: calibrazione del fonometro tramite Larson Davis CAL2000

Una delle limitazioni della verifica effettuata è l'impossibilità di utilizzare un veicolo elettrico come sorgente sonora da registrare. La normativa stabilisce infatti che, anche in condizioni di spazio chiuso e veicolo fermo, occorre utilizzare il mezzo elettrico con il suo altoparlante integrato come sorgente per il suono dell'AVAS.

Non avendo disponibile un veicolo da utilizzare, né essendo possibile il suo trasporto in camera anecoica, è stato utilizzante un altoparlante fornito dal Politecnico di Torino.

Il suono è stato generato dallo script supercollider sulla scheda Raspberry PI 4, collegata all'altoparlante tramite l'uscita cuffie. Variando opportunamente alcuni parametri dello script, si sono simulate le condizioni di prova necessarie e si è generato e registrato il suono dell'AVAS.

L'altoparlante utilizzato è il Genelec 8030A a due vie, con risposta in frequenza lineare tra i 58 Hz e i 20 kHz, più che sufficiente per garantire una risposta fedele del segnale generato che si concentra principalmente tra i 200 e i 3000 Hz. [21]

Figura 8.4: Genelec 8030A [21]

8.2 Prima procedura di prova

La prima procedura di prova consiste nella valutazione del livello sonoro prodotto. Come precedentemente descritto, occorre valutare il livello su entrambi i lati del veicolo con due microfoni. Avendo a disposizione un solo fonometro, la procedura è stata divisa in due serie di misurazioni, la prima da sinistra e la seconda da destra rispetto alla sorgente sonora, ovvero l'altoparlante Genelec 8030A.

La prima serie di misurazioni è stata effettuata orientando l'altoparlante verso una delle pareti laterali, il microfono è stato posizionato come in *Figura 8.5*. Queste misurazioni rappresentano il livello rilevato alla sinistra della sorgente sonora.

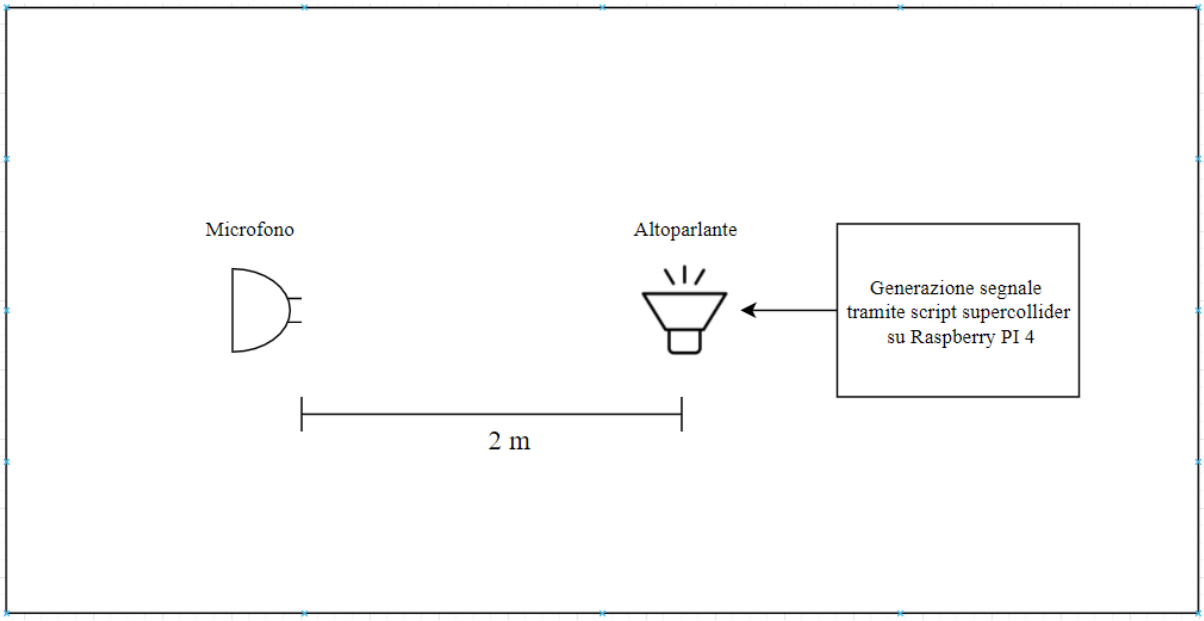

Figura 8.5: schema della disposizione degli strumenti per la prima serie di misurazioni

Figura 8.6: disposizione degli strumenti per la prima serie di misurazioni

Il microfono è stato posizionato all'altezza di 1.2 metri, come specificato dalla Regolamentazione n.138 della UNECE. L'altoparlante è stato posto alla stessa altezza.

Per la seconda serie di misurazioni, l'altoparlante è stato orientato verso la parete opposta, in modo da ottenere il livello alla destra rispetto alla sorgente sonora.

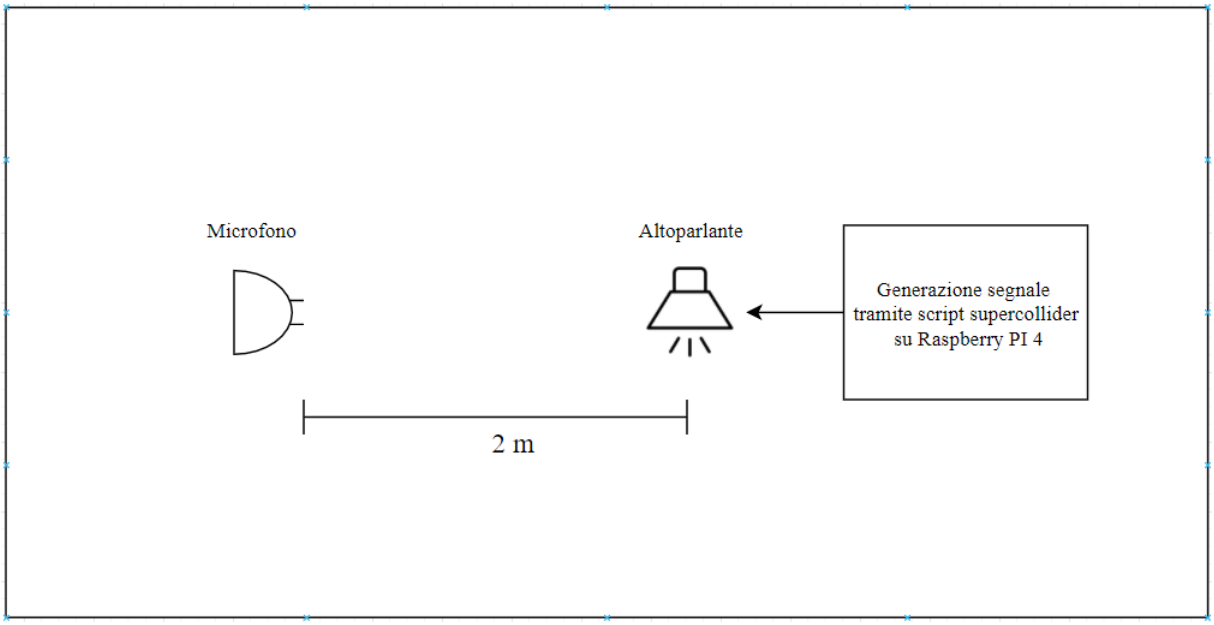

Figura 8.7: schema della disposizione degli strumenti per la seconda serie di misurazioni

Per ognuna delle due serie di misurazioni sono stati analizzati 7 diversi suoni, a partire dai 7 diversi file audio utilizzati per la sintesi granulare.

Per ogni suono sono state individuate 5 condizioni di prova:

- velocità costante simulata di 10 km/h, suono attenuato dal conducente del veicolo mediante apposito pulsante. Misurando il livello attenuato si può concludere che se supera il livello minimo sonoro specificato dalla *Tabella 2.1*, allora anche il livello in condizioni normali è sufficiente.
- velocità costante simulata di 10 km/h, suono dell'AVAS senza attenuazione. Non avendo modo di simulare in modo realistico la componente relativa alla posizione del pedale dell'acceleratore, è stata simulata una situazione con pedale premuto fino in fondo che corrisponde al livello sonoro più alto raggiungibile. Essendo questo il rumore massimo possibile generabile a 10 km/h, occorre verificare che sia inferiore al livello specificato dalla normativa, ovvero 75 dB (A).
- velocità costante simulata di 20 km/h, suono attenuato. Condizione che serve a verificare che a 20 km/h il livello sonoro minimo sia rispettato in tutte le condizioni.
- velocità costante simulata di 20 km/h, suono non attenuato ma pedale premuto fino in fondo, in modo da verificare che sia rispettato il limite massimo del livello sonoro.
- prova del segnale di retromarcia alla velocità costante di 6 km/h, in questo caso la componente legata alla posizione del pedale dell'acceleratore non è presente, né la possibilità di attenuazione del segnale.

Ognuna di queste condizioni dovrebbe essere misurata per 4 volte per almeno 5 secondi, facendo poi la media dei livelli ottenuti per tutte le bande.

Per semplicità, sono stati ripresi almeno 20 secondi per ciascuna delle 5 condizioni di prova e tramite un software della Bruel & Kjaer è stato calcolato il livello medio complessivo e in bande di terzi d'ottava. L'operazione di utilizzare 4 campioni da 5 secondi e farne la media è stata considerata equivalente al fare la media del campione di almeno 20 secondi utilizzato.

In ogni misurazione sono state effettuate 7 registrazioni, una per ogni suono, consistenti dalle cinque condizioni di prova (di almeno 20 secondi) intervallate da brevi silenzi.

Tramite il software della Bruel & Kjaer sono stati estratti dati per ognuno dei campioni di 20 secondi, quindi avendo 14 registrazioni (7 per il livello a sinistra della sorgente e 7 per quello a destra) con 5 campioni l'una sono estratti 70 file di dati. Sono state inoltre state effettuate due misurazioni aggiuntive per il rumore di fondo, almeno 10 secondi per posizione, in totale quindi per questa prima procedura sono stati estratti 72 file di dati.

I dati sono stati estratti in un foglio di calcolo Excel, dove la prima colonna indica la frequenza centrale della banda in terzi d'ottava, la seconda indica invece il valore medio estratto per il campione selezionato. La terza colonna indica invece il livello minimo. In fondo al file è presente il livello complessivo sonoro, sia con pesatura A che C.

1			
2			
3	Intervallo temp. spett	05/10/2021 10:10:38 - 10:10:50	
4	Frequenza [Hz]	LZeq [dB]	LZSmin [dB]
5	12,50	39,01168498	33,92
6	16	46,59115802	43,61
7	20	44,35329399	39,76
8	25	40,90071363	35,51
9	31,50	28,24693534	26,57
10	40	24,42765924	22,59
11	50	26,62551068	24,04
12	63	17,70025672	15,07
13	80	17,1577633	15,13
14	100	12,06505286	10,02
15	125	5,463158139	4,23
16	160	1,716772803	-0,01
17	200	0,73457216	-0,6
18	250	0,964099662	-0,64
19	315	0,288195521	-0,82
20	400	-0,031369325	-0,8
21	500	0,327993998	-0,25
22	630	1,442928539	0,88
23	800	2,823079133	2,23
24	1000	2,081531278	1,6
25	1250	2,665677934	2,25
26	1600	3,792588537	3,43
27	2000	7,565121585	7,33
28	2500	5,555684272	5,09
29	3150	6,432412271	6,15
30	4000	7,811045391	7,48
31	5000	8,156575244	7,96
32	6300	8,856709598	8,64
33	8000	9,778378079	9,62
34	10000	10,76710179	10,63
35	12500	11,60567778	11,46
36	16000	12,92356073	12,73
37	20000	15,59273583	15,48
38	A	19,08103394	—
39	C	42,73348386	—

Figura 8.8: esempio di file dati, rumore di fondo (da sinistra)

8.3 Seconda procedura di prova

La seconda procedura di prova serve per misurare la variazione in frequenza del suono prodotto. L'altoparlante in questo caso è stato rivolto nella direzione del microfono, non è necessario misurare i livelli a sinistra o destra del veicolo, è sufficiente il suono diretto ad un metro di distanza.

Rispetto alla configurazione della prima procedura è stato cambiato l'orientamento dell'altoparlante e il microfono è stato avvicinato alla sorgente.

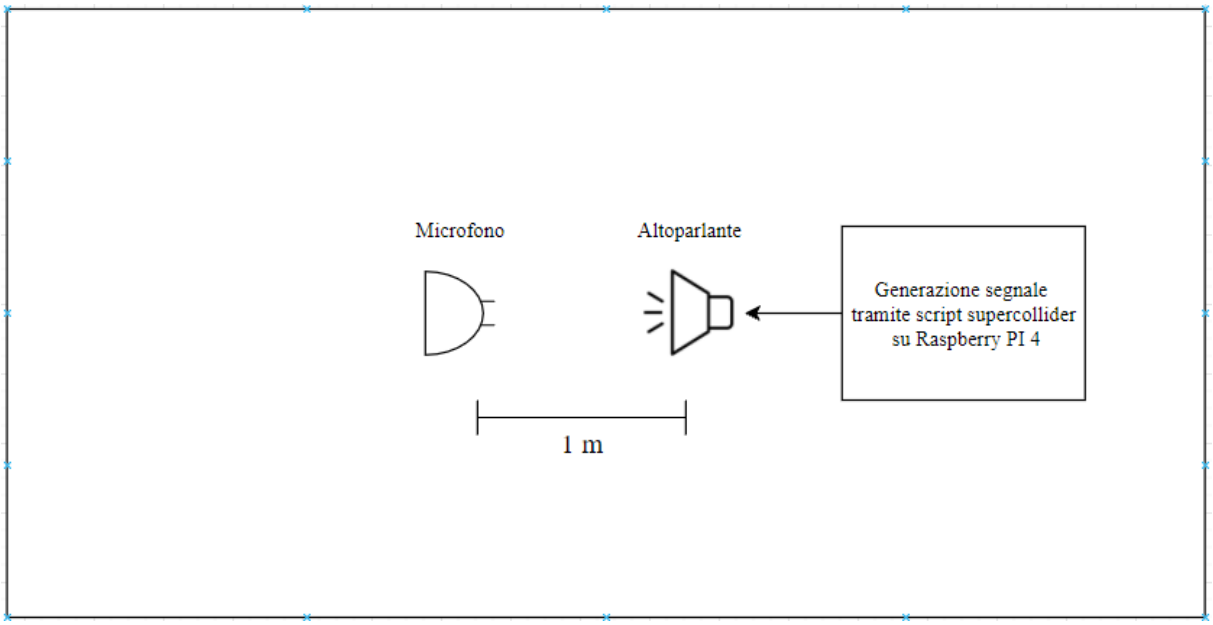

Figura 8.9: schema della disposizione degli strumenti per la seconda serie di misurazioni

Figura 8.10: disposizione degli strumenti per la seconda procedura di prova

Per ogni suono sono stati misurate 4 condizioni di prova:

- velocità costante simulata di 5 km/h.
- velocità costante simulata di 10 km/h.
- velocità costante simulata di 15 km/h.
- velocità costante simulata di 20 km/h.

In questa prova è importante verificare la variazione in frequenza del suono, per cui la componente relativa alla posizione del pedale dell'acceleratore non è stata considerata.

Per ognuno dei 7 segnali è stata effettuata una registrazione, ognuna di queste contenente almeno 20 secondi per ciascuna delle condizioni di prova, intervallate da brevi silenzi. Tramite il software della Bruel & Kjaer sono stati generati 28 file di dati (7 segnali per 4 condizioni di prova), anche in questo caso si è fatta la media dei campioni di almeno 20 secondi invece che la media tra 4 valori estratti da 5 secondi di segnale. Per questa procedura verranno analizzati anche i file audio in formato ".wav".

8.4 Analisi dei dati

Il primo dato da analizzare riguarda la misura del rumore ambientale, del quale sono stati misurati due campioni da 10 secondi l'uno nel corso della prima procedura di prova, uno da sinistra e l'altro da destra. I valori riportati sono stati approssimati alla prima cifra decimale, come specificato nella normativa europea.

Livello medio del rumore ambientale in dB (A)	
Sinistra	19.1
Destra	19.3

Tabella 8.3: livello complessivo del rumore ambientale

In questo caso la differenza tra valore massimo e minimo del volume è di soli 0.2 dB (A). Come suggerito dalla *Tabella 8.1* i vari segnali registrati dovranno essere corretti solo se la differenza tra il risultato di una prova e il rumore di fondo è minore di 10 dB (A), altrimenti non ci sarà bisogno di alcuna correzione.

È inoltre necessario riportare il livello del rumore in terzi di ottava nella registrazione con livello più alto, in questo caso quella di destra.

Il livello in terzi di ottava generato tramite il software Bruel & Kjaer è in dB, non è pesato. Per questo è stata applicata la pesatura successivamente, andando a sommare ad ogni banda il livello della curva di pesatura A riportato in *Tabella 8.4*. I livelli della curva di pesatura A in terzi di ottava sono stati trovati in rete. [22] Il livello in dB (A) ottenuto è stato approssimato alla prima cifra decimale, come specificato dalla Regolamentazione n.138 della UNECE.

Frequenza (Hz)	**Livello medio in dB**	**Curva di pesatura A (dB)**	**Livello medio in dB (A)**
12,50	37,46101	-63,4	-25,9
16	42,09423	-56,7	-14,6
20	40,85473	-50,5	-9,6
25	37,08134	-44,7	-7,6
31,50	26,34935	-39,4	-13,1

40	25,40065	-34,6	-9,2
50	30,3269	-30,2	0,1
63	21,3143	-26,2	-4,9
80	20,46069	-22,5	-2,0
100	15,06388	-19,1	-4,0
125	10,79742	-16,1	-5,3
160	4,030406	-13,4	-9,4
200	1,495292	-10,9	-9,4
250	2,70749	-8,6	-5,9
315	1,828798	-6,6	-4,8
400	0,620761	-4,8	-4,2
500	0,985485	-3,2	-2,2
630	2,077015	-1,9	0,2
800	3,179507	-0,8	2,4
1000	2,662044	0	2,7
1250	3,241732	0,6	3,9
1600	4,078311	1	5,1
2000	6,811299	1,2	8,0
2500	5,717004	1,3	7,0
3150	6,963524	1,2	8,2
4000	8,530578	1	9,5
5000	8,229221	0,5	8,7
6300	8,962751	-0,1	8,9
8000	9,932609	-1,1	8,8
10000	10,87185	-2,5	8,4
12500	11,77495	-4,3	7,4
16000	13,03014	-6,6	6,4
20000	15,47365	-9,3	6,2

Tabella 8.4: livello del rumore ambientale in terzi di ottava, misurato da destra

Il livello in terzi di ottava del rumore in dB (A) è da tenere conto, infatti il livello nelle bande di interesse dei segnali misurati dev'essere almeno 6 dB in più del livello del rumore ambientale nella stessa banda.

8.4.1 S1

Prima procedura di prova

Inizialmente viene valutato il livello complessivo del suono in tutte le condizioni di prova. Il valore massimo è 75 dB (A) mentre il minimo è 50 dB (A) a 10 km/h e 56 dB (A) a 20 km/h (*Tabella 2.1)*. In retromarcia il valore minimo da rispettare è 47 dB (A).

Per le condizioni di prova con suono attenuato, viene preso in considerazione il livello minimo tra la misurazione di sinistra e quella di destra. Per le condizioni con pedale premuto fino in fondo, viene invece preso il livello massimo tra le due misurazioni.

I risultati in *Tabella 8.5* indicano che il livello complessivo del suono S1 è adeguato in ogni condizione di prova.

Condizione di prova	Livello complessivo in dB (A)	Livello di pressione sonora del risultato della prova j-th meno il livello del rumore di fondo ΔL in dB (A)	Condizione da rispettare
10 km/h attenuato	55,8	>10, nessuna correzione necessaria	>=50 dB (A) OK
10 km/h non attenuato con pedale premuto fino in fondo	73,8	>10, nessuna correzione necessaria	<=75 dB (A) OK
20 km/h attenuato	58,6	>10, nessuna correzione necessaria	>=56 dB (A) OK
20 km/h non attenuato con pedale premuto fino in fondo	74,2	>10, nessuna correzione necessaria	<=75 dB (A) OK
6 km/h in retromarcia	61,8	>10, nessuna correzione necessaria	>=47 dB (A) OK

Tabella 8.5: livello complessivo S1

Occorre verificare che almeno due bande in terzi di ottava (almeno una delle quali inferiore ai 1600 Hz) soddisfino i livelli minimi riportati in *Tabella 2.2*.

I valori estratti dei livelli in terze di ottave erano in dB, prima del confronto è stata quindi applicata la curva di pesatura A. Le condizioni di prova utilizzate sono quelle con suono attenuato a 10 km/h e 20 km/h, la prova in retromarcia non ha bisogno dell'analisi in terzi di ottava. In *Tabella 8.6* sono riportati i livelli minimi tra la registrazione di destra e quella di sinistra. Il livello minimo è rispettato in più bande, ma ne sono state riportate solo due per ogni condizione, quelle che bastano per validare il suono S1.

Condizione di prova	Frequenza (Hz)	Livello minimo da rispettare in dB (A)	Livello di pressione sonora in dB	Livello di pressione sonora in dB (A)
10 km/h attenuato	250	43	55,8	47,2 OK
	315	44	54,5	47,9 OK
20 km/h attenuato	250	48	58,4	49,8 OK
	315	49	57,4	50,8 OK

Tabella 8.6: bande in terze d'ottava che rispettano il livello minimo per S1

Il livello nelle bande di interesse dei segnali misurati è almeno 6 dB in più del livello del rumore ambientale nella stessa banda (*Tabella 8.4*).

Seconda procedura di prova

Per la seconda procedura di prova, è stato disegnato tramite Excel l'andamento in frequenza a partire dai dati estratti dalla misurazione effettuata ad un metro dalla sorgente.

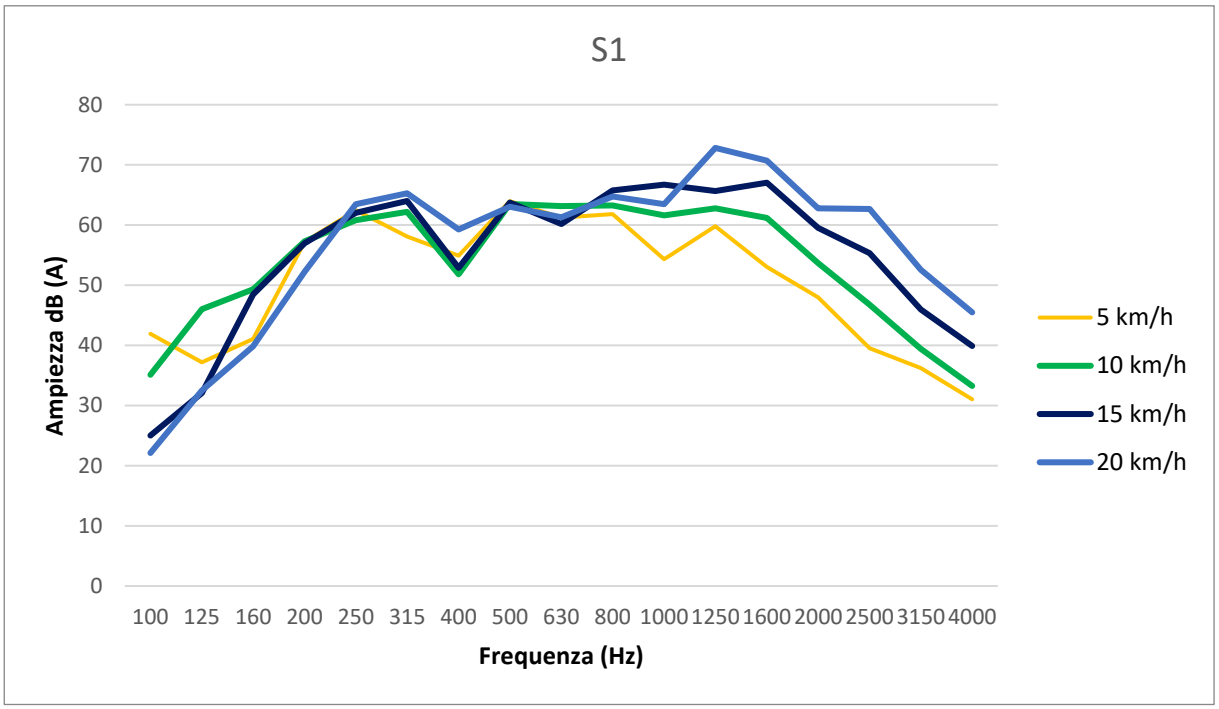

Figura 8.11: andamento in frequenza di S1 nelle quattro velocità di prova

Dal grafico ottenuto si nota abbastanza chiaramente un andamento in frequenza crescente a seconda della velocità, specialmente intorno alla frequenza di 1 kHz.

Per individuare con maggior precisione la variazione di un tono nella banda di frequenza selezionata precedentemente, sono stati analizzati i file audio ".wav". Tramite il software "Audition" di Adobe è stato generato lo spettro in frequenza del segnale.

In fase di analisi dei dati, è stata riscontrata difficoltà nell'individuare i toni in frequenza che variano al variare della velocità, questo perché i suoni sono composti da diversi strati. Ad esempio i quattro oscillatori in parallelo hanno frequenza che cambia con la velocità in modo lineare, mentre le frequenze che interessano sono dovute alla sintesi granulare in cui la velocità di riproduzione dei grani varia esponenzialmente al variare della velocità.

Lo spettro in frequenza del suono dell'AVAS composto dai vari strati non offre abbastanza indizi per riuscire a distinguere le frequenze che cambiano con la velocità.

Per individuare la frequenza che varia esponenzialmente sono stati registrate le transizioni del segnale (solo lo strato della sintesi granulare) dalla velocità di partenza di 5 km/h alle altre velocità di prova. In questo modo guardando lo spettro in frequenza del segnale nel tempo, risulta facile distinguere la frequenza crescente. Come si nota nello spettrogramma in *Figura 8.12* la frequenza sale da circa 200 a circa 800 Hz.

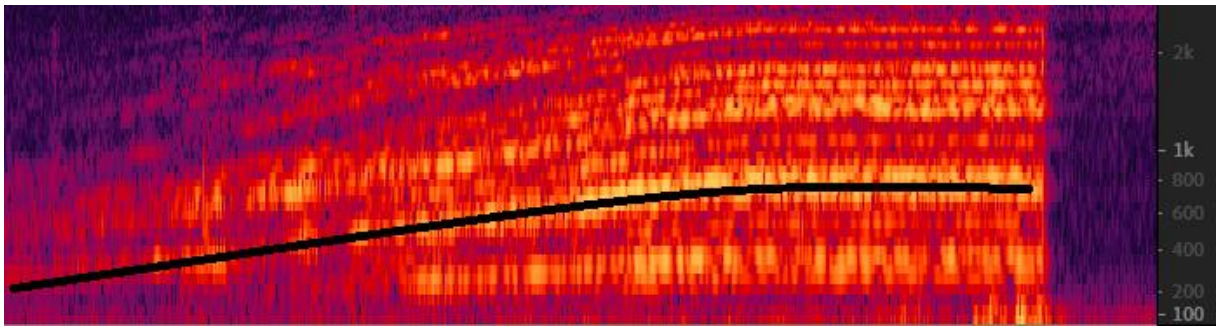

Figura 8.12: spettrogramma della transizione del segnale da 5 a 20 km/h

Analizzando lo spettro in frequenza di questo segnale si sono individuate le frequenze precise ad ogni velocità di prova, riportate in *Tabella 8.7*.

Osservando lo spettro del suono complessivo dell'AVAS (*Figura 8.13*), si possono notare dei picchi (massimi relativi cerchiati in bianco) in corrispondenza delle 4 frequenze rilevate (il picco a 237 Hz si osserva nella curva rossa, corrispondente allo spettro del segnale a 5 km/h e così via). Lo spettro è stato generato utilizzando una finestra di Hann (come richiesto dalla normativa) con dimensione di trasformazione FFT pari a 8192 campioni.

Per la rappresentazione è stata utilizzata una scala logaritmica in frequenza sull'asse "x". Per questo motivo, i picchi da ricercare nel grafico saranno equidistanti (avendo andamento esponenziale al variare della velocità).

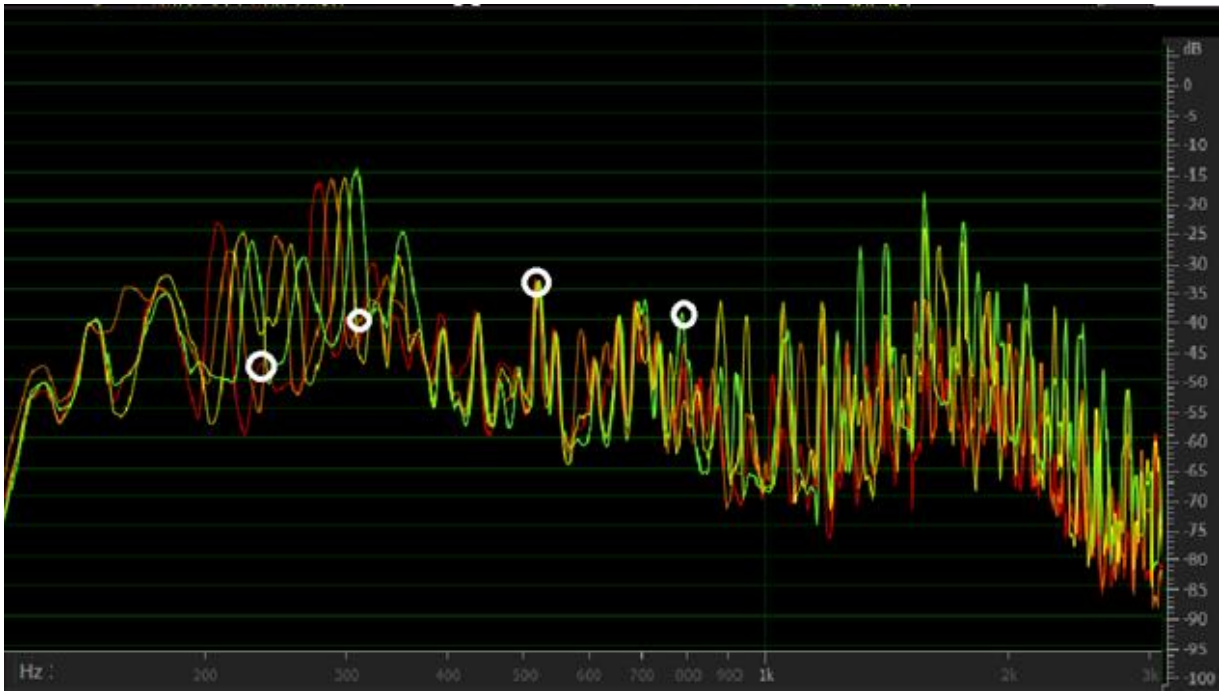

Figura 8.13: spettro in frequenza del segnale S1 a 5 (rossa), 10 (arancione), 15 (gialla) e 20 (verde) km/h

Tramite la formula specificata nella normativa europea, si calcola la percentuale di variazione in frequenza, che deve variare in proporzione della velocità in ciascun singolo rapporto di una media di almeno lo 0.8 % per 1 km/h.

Nella formula, "fspeed" è la frequenza ad una data velocità "v", "fref" è la frequenza alla velocità di riferimento "vref" di 5 km/h.

$$\Delta f = \frac{\frac{fspeed - fref}{v - vref}}{fref} \; x \; 100$$

		Risultati della prova alle velocità target			
		5 km/h (riferimento)	10 km/h	15 km/h	20 km/h
Frequenza	Hz	237	316	516	787
Percentuale minima (per garantire rapporto di almeno 0.8 % per 1 km/h)	%	n.d.	4%	8%	12%
Variazione di frequenza	%	n.d.	6.7 % OK	11.7 % OK	15.5 % OK

Tabella 8.7: risultati della prova di variazione in frequenza per il suono S1

8.4.2 S2

Le metodologie utilizzate per gli altri suoni sono le stesse elencate precedentemente per S1. Di seguito vengono riportati i risultati conseguiti.

Prima procedura di prova

Condizione di prova	**Livello complessivo in dB (A)**	**Livello di pressione sonora del risultato della prova j-th meno il livello del rumore di fondo ΔL in dB (A)**	**Condizione da rispettare**
10 km/h attenuato	64,7	>10, nessuna correzione necessaria	>=50 dB (A) OK
10 km/h non attenuato con pedale premuto fino in fondo	72,1	>10, nessuna correzione necessaria	<=75 dB (A) OK
20 km/h attenuato	64,6	>10, nessuna correzione necessaria	>=56 dB (A) OK
20 km/h non attenuato con pedale premuto fino in fondo	71,8	>10, nessuna correzione necessaria	<=75 dB (A) OK
6 km/h in retromarcia	70,4	>10, nessuna correzione necessaria	>=47 dB (A) OK

Tabella 8.8: livello complessivo S2

Condizione di prova	Frequenza (Hz)	Livello minimo da rispettare in dB (A)	Livello di pressione sonora in dB (valore minimo tra le due registrazione SX e DX)	Livello di pressione sonora in dB (A)
10 km/h attenuato	250	43	65,9	57,3 OK
	500	45	59,4	56,2 OK
20 km/h attenuato	250	48	66,4	57,8 OK
	500	50	56,8	53,6 OK

Tabella 8.9: bande in terze d'ottava che rispettano il livello minimo per S2

Seconda procedura di prova

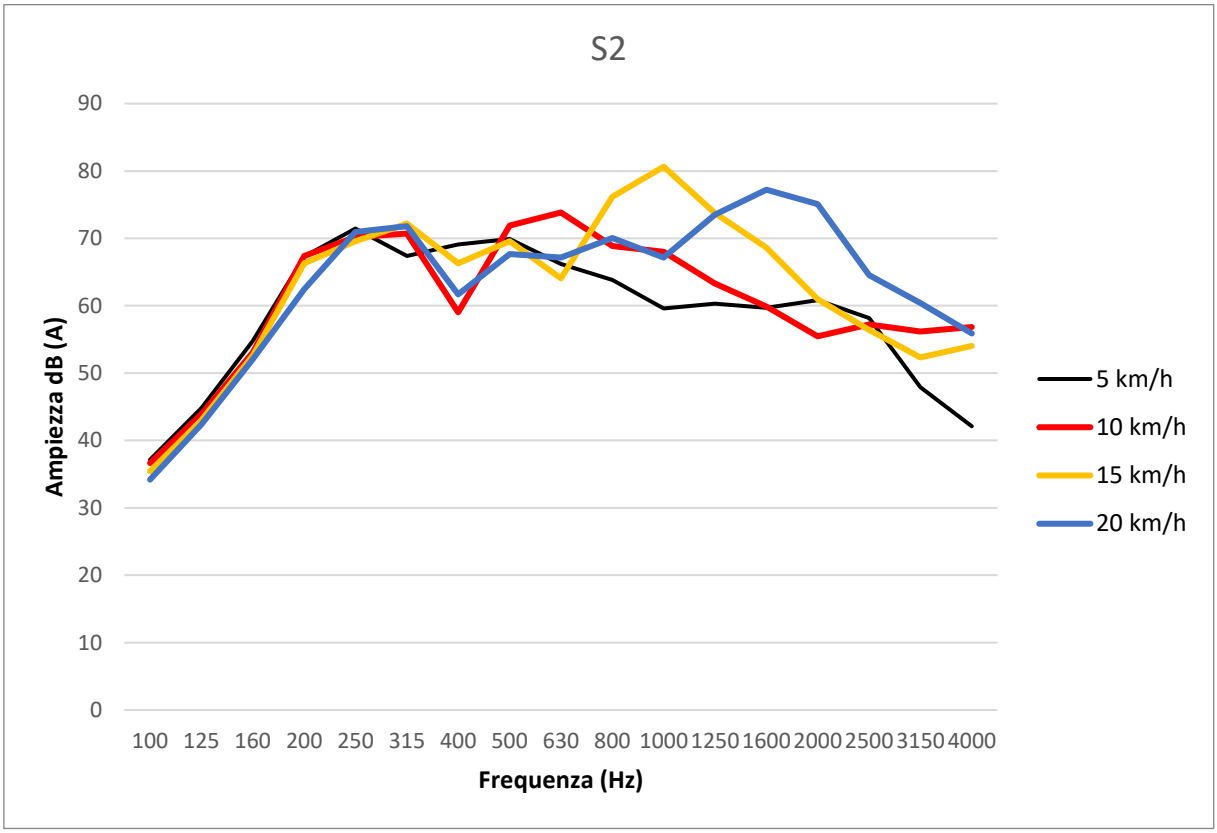

Figura 8.14: : andamento in frequenza di S2 nelle quattro velocità di prova

Tramite il software "Audition" è stato generato lo spettro in frequenza utilizzando una finestra di Hann e una dimensione di trasformazione FFT pari a 4096 campioni. In questo caso la frequenza che varia in modo esponenziale al variare della velocità è evidente dallo spettro

ottenuto, i picchi sono ben riconoscibili ed equidistanti tra loro (dato che la frequenza è rappresentata su una scala logaritmica).

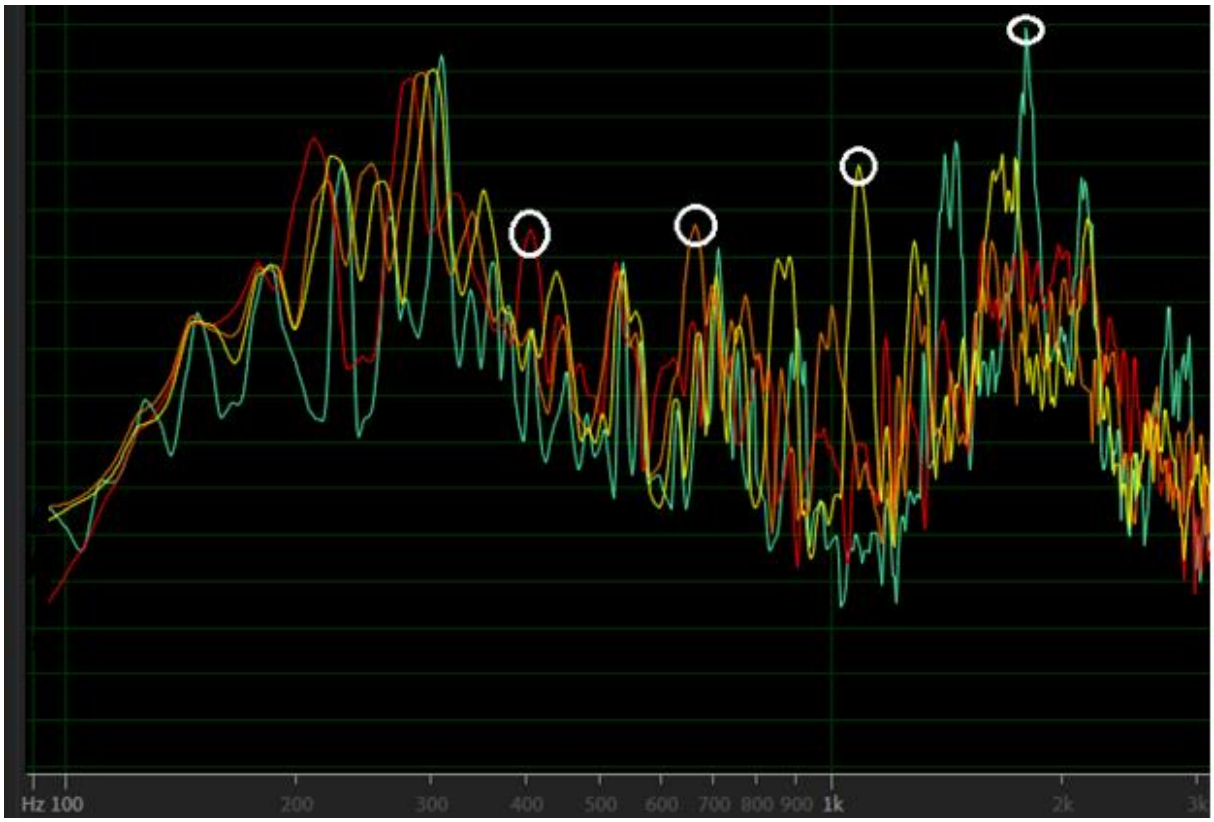

Figura 8.15: spettro in frequenza del segnale S2 a 5 (rossa), 10 (arancione), 15 (gialla) e 20 (azzurra) km/h

		Risultati della prova alle velocità target			
		5 km/h (riferimento)	10 km/h	15 km/h	20 km/h
Frequenza	Hz	398	656	1090	1787
Percentuale minima (per garantire rapporto di almeno 0.8 % per 1 km/h)	%	n.d.	4%	8%	12%
Variazione di frequenza	%	n.d.	13.0 % OK	17.4 % OK	23.3 % OK

Tabella 8.10: risultati della prova di variazione in frequenza per il suono S2

8.4.3 S3

Prima procedura di prova

Condizione di prova	Livello complessivo in dB (A)	Livello di pressione sonora del risultato della prova j-th meno il livello del rumore di fondo ΔL in dB (A)	Condizione da rispettare
10 km/h attenuato	66,1	>10, nessuna correzione necessaria	>=50 dB (A) OK
10 km/h non attenuato con pedale premuto fino in fondo	72,8	>10, nessuna correzione necessaria	<=75 dB (A) OK
20 km/h attenuato	66,2	>10, nessuna correzione necessaria	>=56 dB (A) OK
20 km/h non attenuato con pedale premuto fino in fondo	72,6	>10, nessuna correzione necessaria	<=75 dB (A) OK
6 km/h in retromarcia	72,1	>10, nessuna correzione necessaria	>=47 dB (A) OK

Tabella 8.11: livello complessivo S3

Condizione di prova	Frequenza (Hz)	Livello minimo da rispettare in dB (A)	Livello di pressione sonora in dB (valore minimo tra le due registrazione SX e DX)	Livello di pressione sonora in dB (A)
10 km/h attenuato	250	43	66,1	57,5 OK
	800	46	59,2	58,5 OK
20 km/h attenuato	250	48	66,8	58,2 OK
	800	51	55,8	55,0 OK

Tabella 8.12: bande in terze d'ottava che rispettano il livello minimo per S3

Seconda procedura di prova

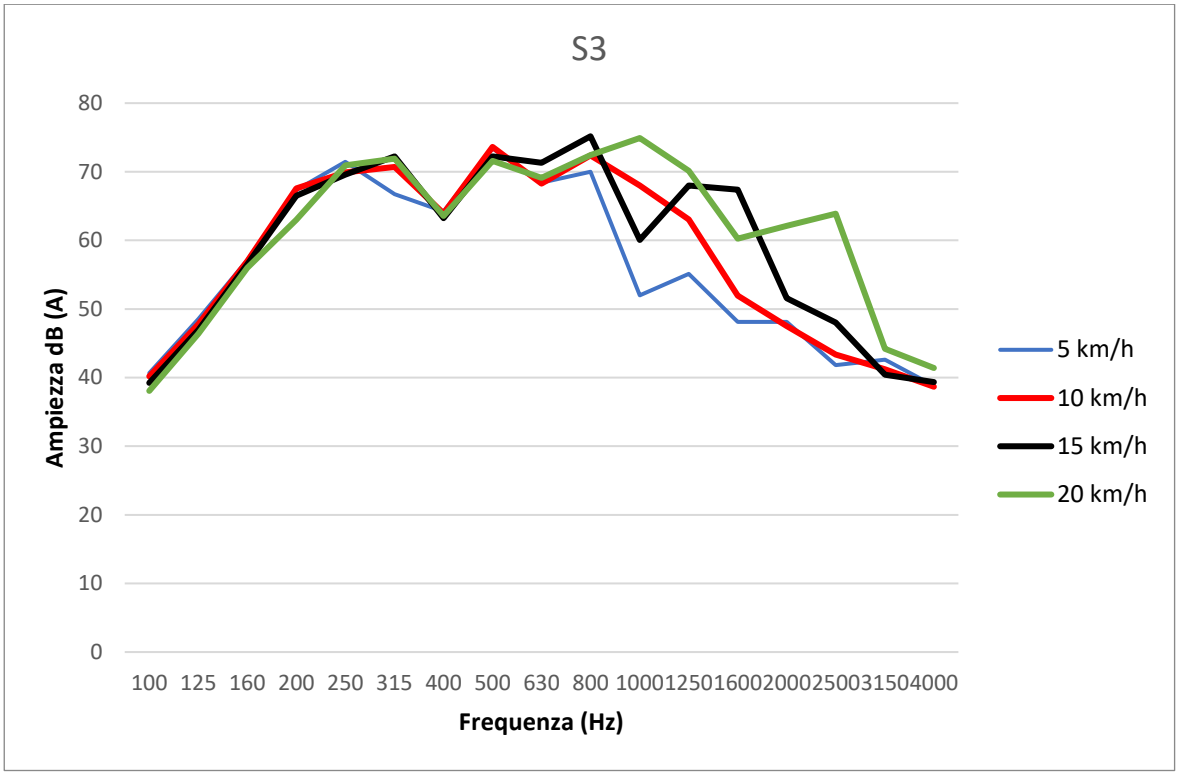

Figura 8.16: andamento in frequenza di S3 nelle quattro velocità di prova

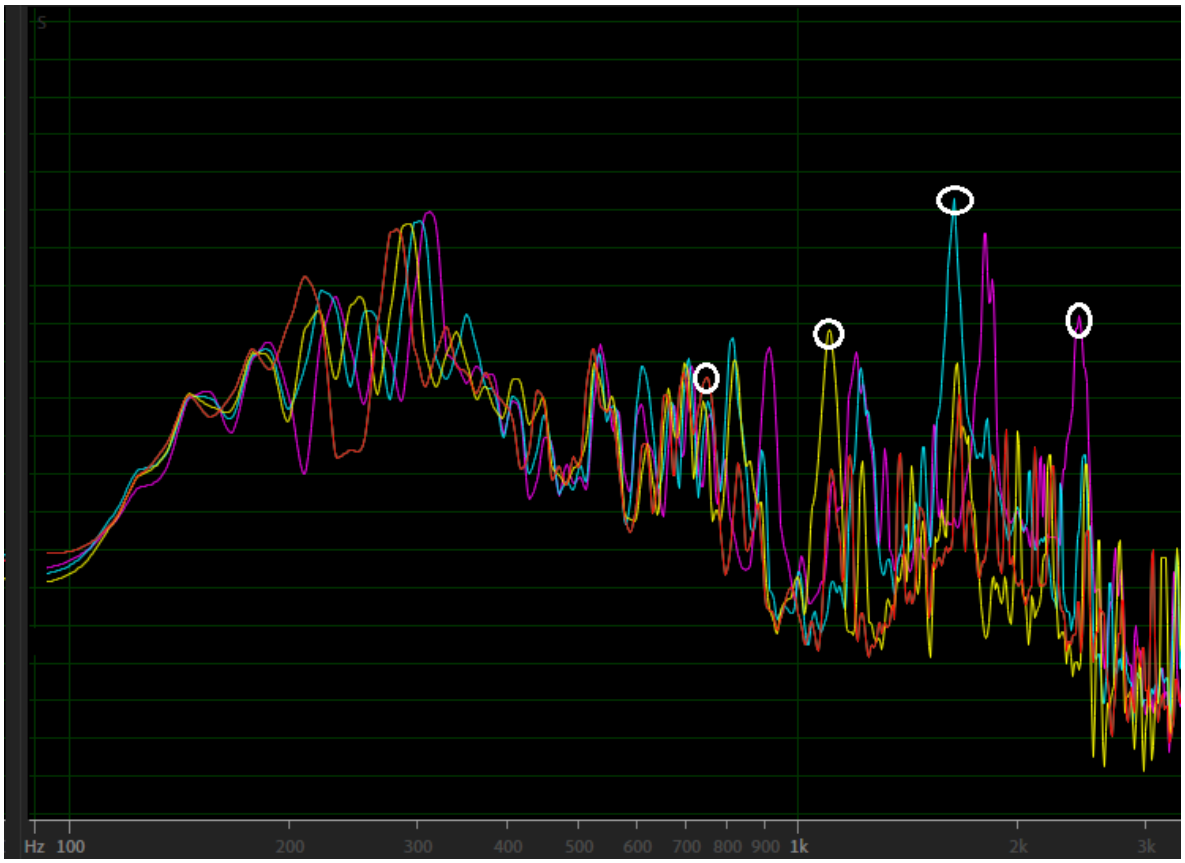

Figura 8.17: spettro in frequenza del segnale S3 a 5 (rossa), 10 (gialla), 15 (azzurra) e 20 (viola) km/h

Il tono che varia esponenzialmente in frequenza in questo caso è abbastanza facile da notare dallo spettro generato con Audition.

		Risultati della prova alle velocità target			
		5 km/h (riferimento)	10 km/h	15 km/h	20 km/h
Frequenza	Hz	738	1102	1629	2426
Percentuale minima (per garantire rapporto di almeno 0.8 % per 1 km/h)	%	n.d.	4%	8%	12%
Variazione di frequenza	%	n.d.	9.9% OK	12.0% OK	15.2% OK

Tabella 8.13: risultati della prova di variazione in frequenza per il suono S3

8.4.4 S4

Prima procedura di prova

Condizione di prova	**Livello complessivo in dB (A)**	**Livello di pressione sonora del risultato della prova j-th meno il livello del rumore di fondo ΔL in dB (A)**	**Condizione da rispettare**
10 km/h attenuato	70,8	>10, nessuna correzione necessaria	>=50 dB (A) OK
10 km/h non attenuato con pedale premuto fino in fondo	75,2	>10, nessuna correzione necessaria	<=75 dB (A) NOT OK
20 km/h attenuato	72,7	>10, nessuna correzione necessaria	>=56 dB (A) OK
20 km/h non attenuato con pedale premuto fino in fondo	77,0	>10, nessuna correzione necessaria	<=75 dB (A) NOT OK
6 km/h in retromarcia	73,6	>10, nessuna correzione necessaria	>=47 dB (A) OK

Tabella 8.14: livello complessivo S4

Condizione di prova	Frequenza (Hz)	Livello minimo da rispettare in dB (A)	Livello di pressione sonora in dB (valore minimo tra le due registrazione SX e DX)	Livello di pressione sonora in dB (A)
10 km/h attenuato	500	45	66,8	63,6 OK
	800	46	66,8	66,0 OK
20 km/h attenuato	500	50	66,0	62,8 OK
	800	51	63,4	62,6 OK

Tabella 8.15: bande in terze d'ottava che rispettano il livello minimo per S4

S4 sfora i livelli massimi in due delle condizioni analizzate, i livelli di questo suono devono essere abbassati e una nuova misurazione dopo gli aggiustamenti sarebbe necessaria.

Seconda procedura di prova

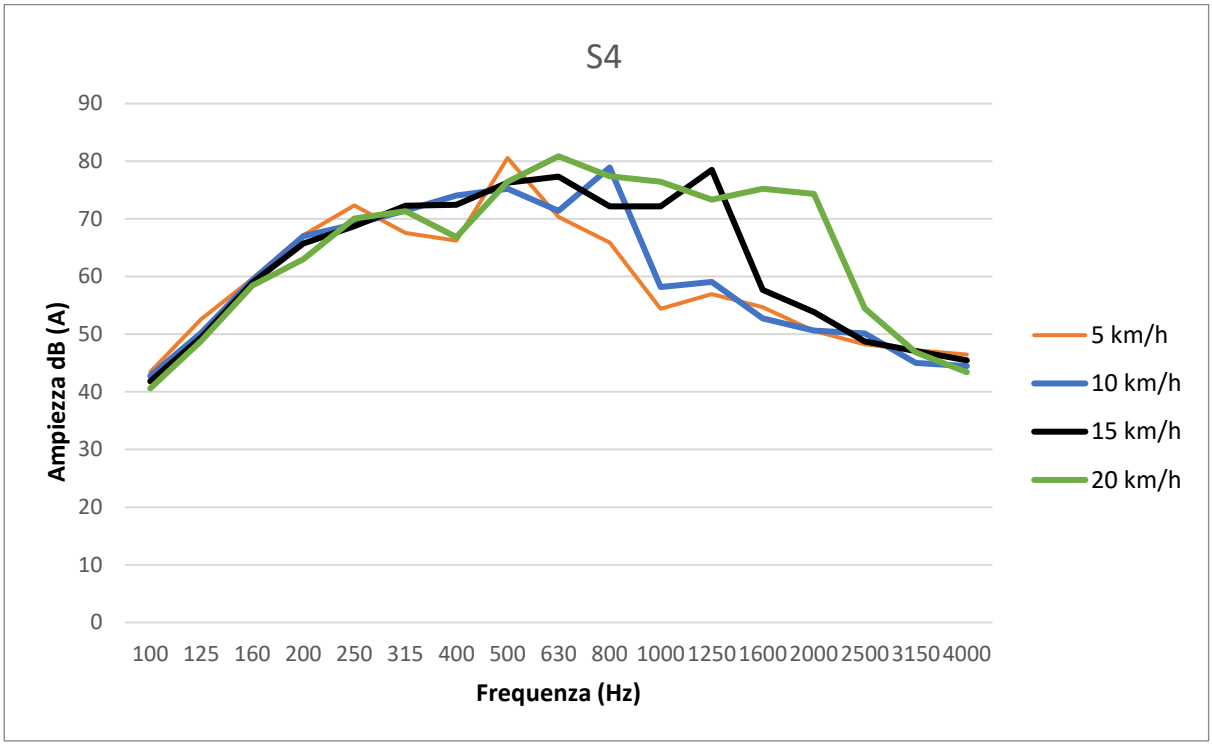

Figura 8.18:: andamento in frequenza di S4 nelle quattro velocità di prova

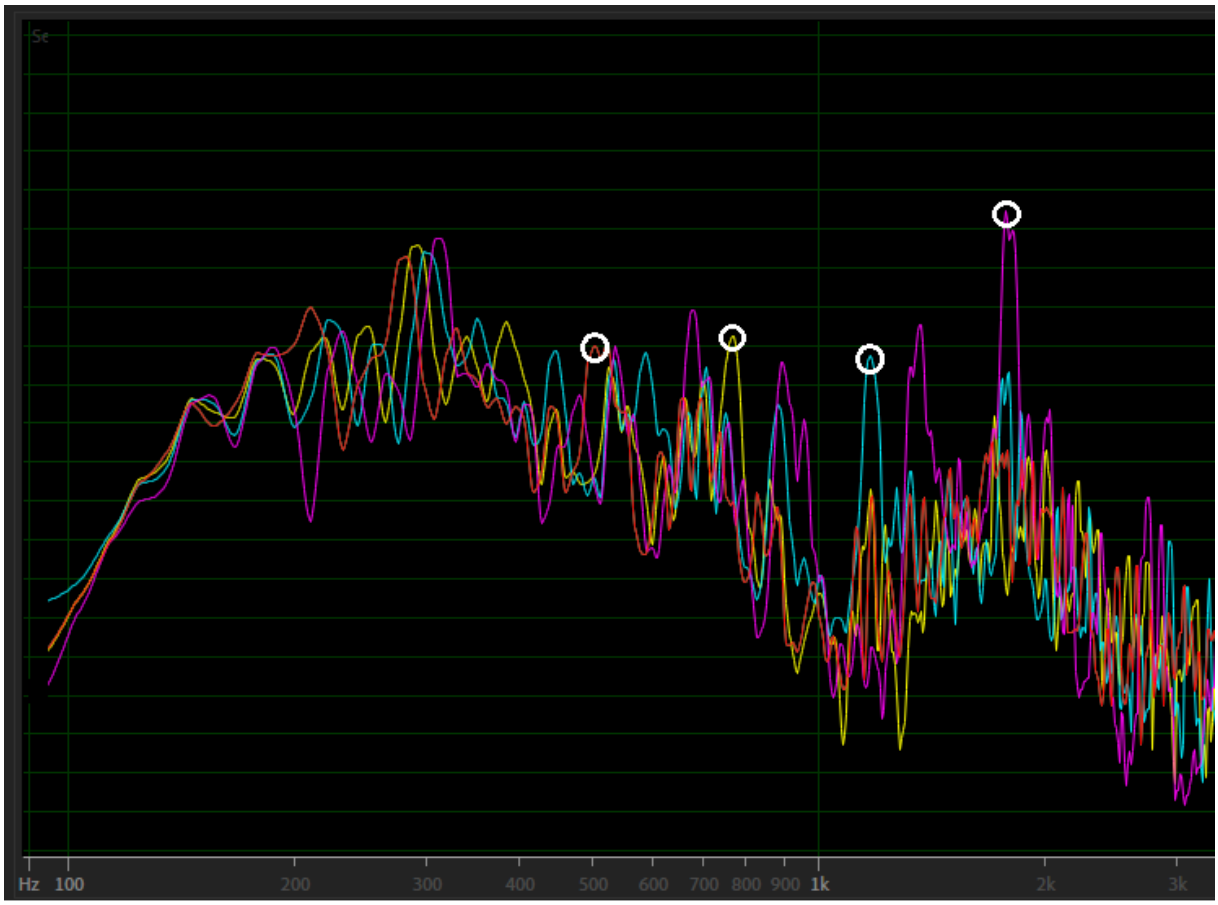

Figura 8.19: spettro in frequenza del segnale S4 a 5 (rossa), 10 (gialla), 15 (azzurra) e 20 (viola) km/h

		Risultati della prova alle velocità target			
		5 km/h (riferimento)	10 km/h	15 km/h	20 km/h
Frequenza	Hz	495	764	1163	1766
Percentuale minima (per garantire aumento di almeno 0.8 % per 1 km/h)	%	n.d.	4%	8%	12%
Variazione di frequenza	%	n.d.	10.9% OK	13.5% OK	17.1% OK

Tabella 8.16: Tabella 8.13: risultati della prova di variazione in frequenza per il suono S4

8.4.5 S5

Prima procedura di prova

Condizione di prova	Livello complessivo in dB (A)	Livello di pressione sonora del risultato della prova j-th meno il livello del rumore di fondo ΔL in dB (A)	Condizione da rispettare
10 km/h attenuato	67,2	>10, nessuna correzione necessaria	>=50 dB (A) OK
10 km/h non attenuato con pedale premuto fino in fondo	73,4	>10, nessuna correzione necessaria	<=75 dB (A) OK
20 km/h attenuato	69,0	>10, nessuna correzione necessaria	>=56 dB (A) OK
20 km/h non attenuato con pedale premuto fino in fondo	74,5	>10, nessuna correzione necessaria	<=75 dB (A) OK
6 km/h in retromarcia	71,1	>10, nessuna correzione necessaria	>=47 dB (A) OK

Tabella 8.17:livello complessivo S5

Condizione di prova	Frequenza (Hz)	Livello minimo da rispettare in dB (A)	Livello di pressione sonora in dB (valore minimo tra le due registrazione SX e DX)	Livello di pressione sonora in dB (A)
10 km/h attenuato	315	44	66,5	59,9 OK
	500	45	65,1	61,9 OK
20 km/h attenuato	315	44	67,0	60,4 OK
	500	50	63,0	59,8 OK

Tabella 8.18: bande in terze d'ottava che rispettano il livello minimo per S5

Seconda procedura di prova

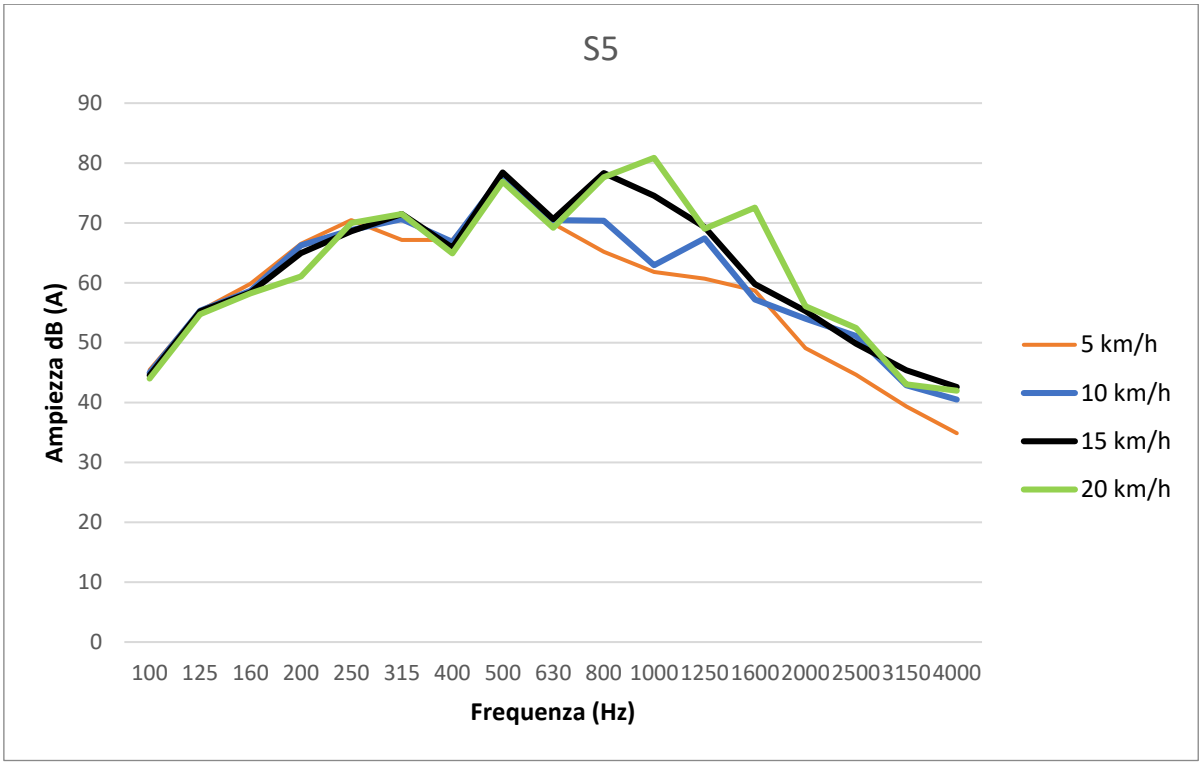

Figura 8.20: andamento in frequenza di S5 nelle quattro velocità di prova

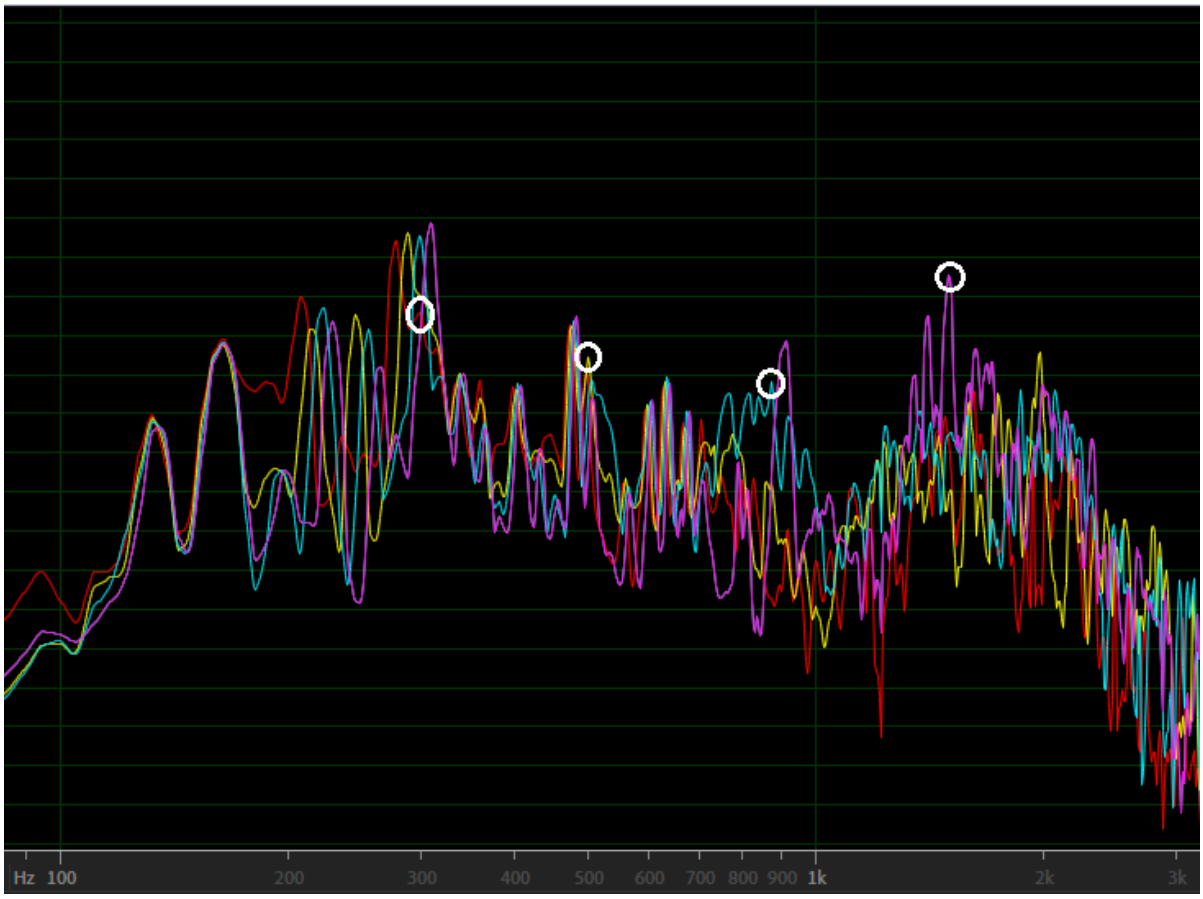

Figura 8.21: spettro in frequenza del segnale S5 a 5 (rossa), 10 (gialla), 15 (azzurra) e 20 (viola) km/h

In questo caso come per S1 la frequenza che varia esponenzialmente con la velocità non era facile da individuare dallo spettro in frequenza. Per ottenere il risultato si è applicato lo stesso metodo utilizzato per S1, registrando le transizioni tra le diverse velocità di prova, in seguito sono stati individuati i picchi nello spettro in frequenza in *Figura 8.21*.

		Risultati della prova alle velocità target			
		5 km/h (riferimento)	10 km/h	15 km/h	20 km/h
Frequenza	Hz	296	495	867	1502
Percentuale minima (per garantire rapporto di almeno 0.8 % per 1 km/h)	%	n.d.	4%	8%	12%
Variazione di frequenza	%	n.d.	13.4% OK	19.3% OK	27.1% OK

Tabella 8.19: risultati della prova di variazione in frequenza per il suono S5

8.4.6 S6

Prima procedura di prova

Condizione di prova	**Livello complessivo in dB (A)**	**Livello di pressione sonora del risultato della prova j-th meno il livello del rumore di fondo ΔL in dB (A)**	**Condizione da rispettare**
10 km/h attenuato	67,6	>10, nessuna correzione necessaria	>=50 dB (A) OK
10 km/h non attenuato con pedale premuto fino in fondo	73,6	>10, nessuna correzione necessaria	<=75 dB (A) OK
20 km/h attenuato	69,8	>10, nessuna correzione necessaria	>=56 dB (A) OK
20 km/h non attenuato con pedale premuto fino in fondo	75,0	>10, nessuna correzione necessaria	<=75 dB (A) OK
6 km/h in retromarcia	71,7	>10, nessuna correzione necessaria	>=47 dB (A) OK

Tabella 8.20: livello complessivo S6

Condizione di prova	Frequenza (Hz)	Livello minimo da rispettare in dB (A)	Livello di pressione sonora in dB (valore minimo tra le due registrazione SX e DX)	Livello di pressione sonora in dB (A)
10 km/h attenuato	400	45	60,3	55,5 OK
	1000	46	60,3	60,3 OK
20 km/h attenuato	400	50	60,5	55,7 OK
	1000	51	53,7	53,7 OK

Tabella 8.21: bande in terze d'ottava che rispettano il livello minimo per S6

Seconda procedura di prova

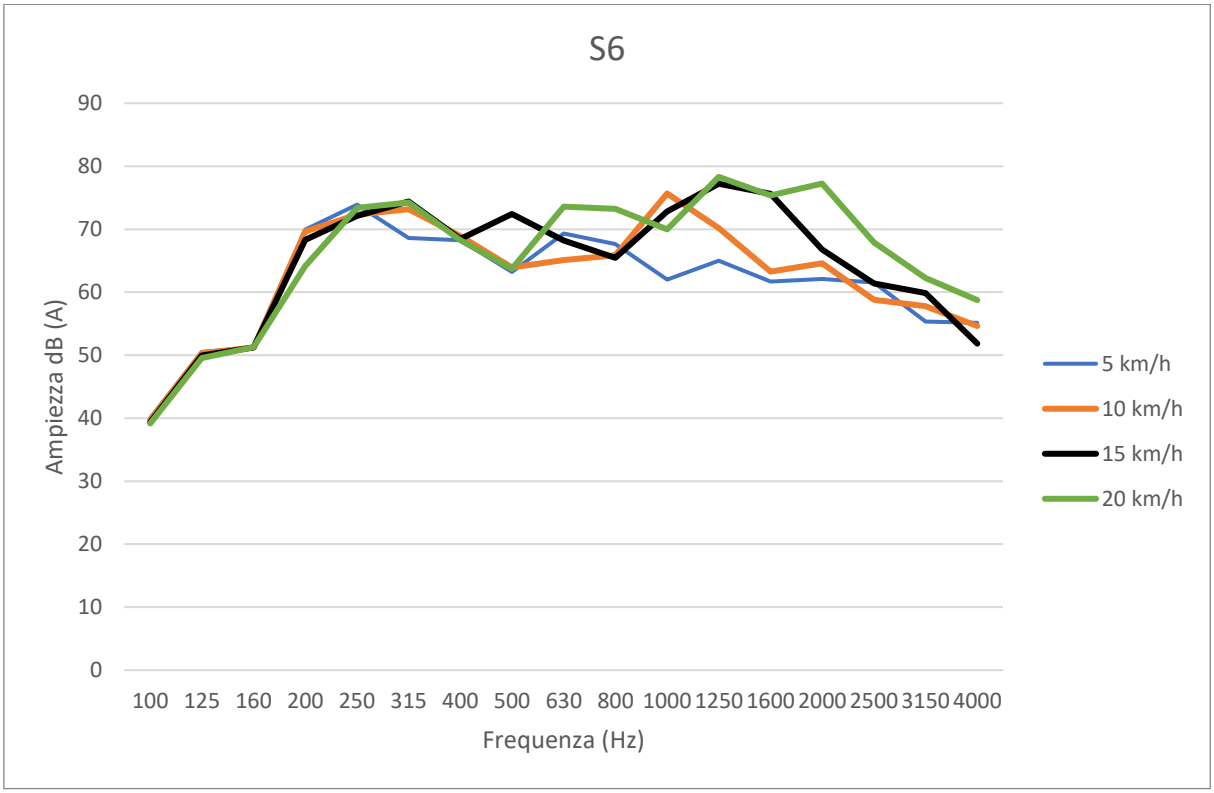

Figura 8.22: andamento in frequenza di S6 nelle quattro velocità di prova

Per S6 risulta difficile individuare la frequenza che interessa dallo spettro in frequenza, come per S1 e S5 si è utilizzata la registrazione delle transizioni tra le varie velocità di prova.

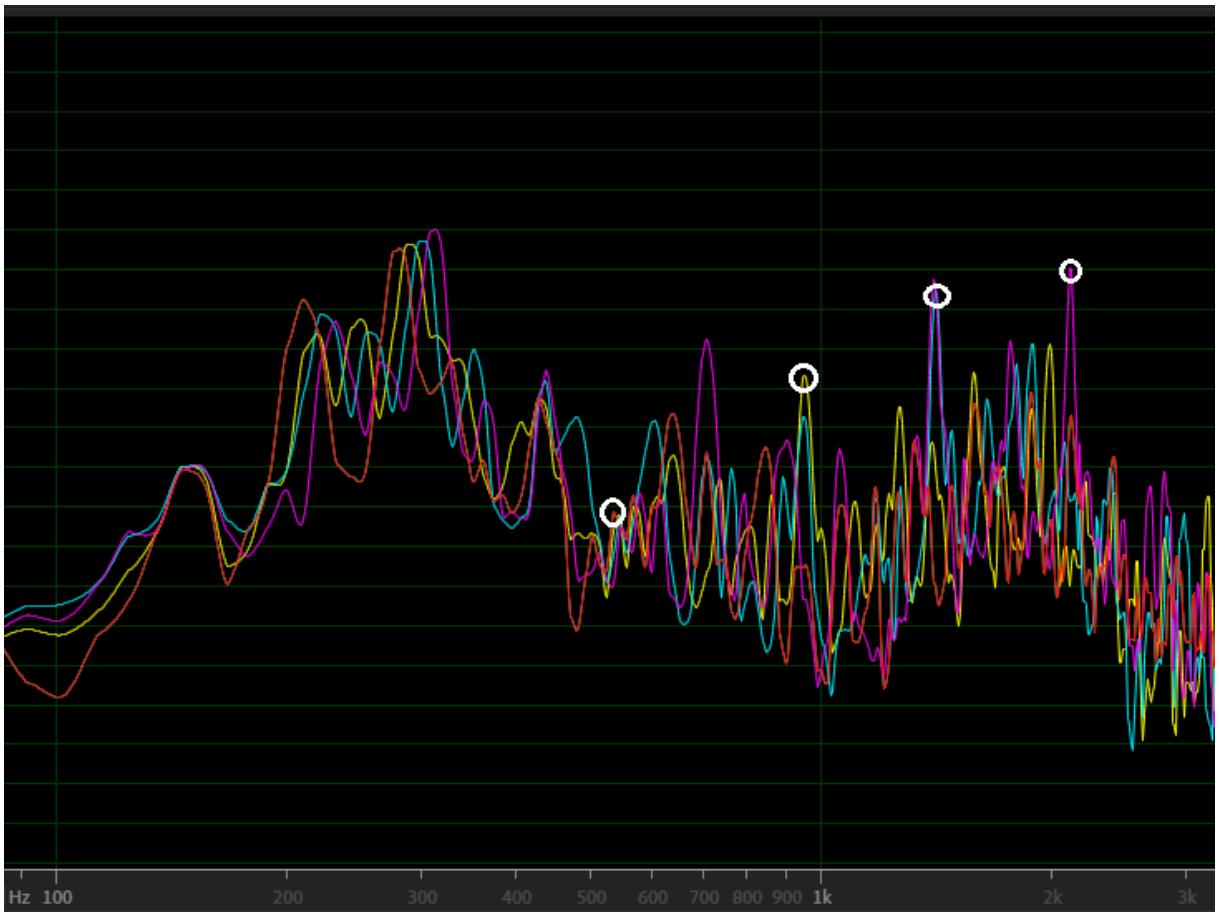

Figura 8.23: spettro in frequenza del segnale S6 a 5 (rossa), 10 (gialla), 15 (azzurra) e 20 (viola) km/h

		Risultati della prova alle velocità target			
		5 km/h (riferimento)	10 km/h	15 km/h	20 km/h
Frequenza	Hz	527	938	1395	2109
Percentuale minima (per garantire rapporto di almeno 0.8 % per 1 km/h)	%	n.d.	4%	8%	12%
Variazione di frequenza	%	n.d.	15.6% OK	16.5% OK	20.0% OK

Tabella 8.22: risultati della prova di variazione in frequenza per il suono S6

8.4.7 S7

Prima procedura di prova

Condizione di prova	Livello complessivo in dB (A) (valore minimo tra le due registrazione SX e DX)	Livello di pressione sonora del risultato della prova j-th meno il livello del rumore di fondo ΔL in dB (A)	Condizione da rispettare
10 km/h attenuato	66,4	>10, nessuna correzione necessaria	>=50 dB (A) OK
10 km/h non attenuato con pedale premuto fino in fondo	72,5	>10, nessuna correzione necessaria	<=75 dB (A) OK
20 km/h attenuato	67,2	>10, nessuna correzione necessaria	>=56 dB (A) OK
20 km/h non attenuato con pedale premuto fino in fondo	72,7	>10, nessuna correzione necessaria	<=75 dB (A) OK
6 km/h in retromarcia	72,2	>10, nessuna correzione necessaria	>=47 dB (A) OK

Tabella 8.23:livello complessivo S7

Condizione di prova	Frequenza (Hz)	Livello minimo da rispettare in dB (A)	Livello di pressione sonora in dB (valore minimo tra le due registrazione SX e DX)	Livello di pressione sonora in dB (A)
10 km/h attenuato	250	43	66,6	58,0 OK
	630	46	61,8	60,0 OK
20 km/h attenuato	250	48	66,7	58,1 OK
	630	51	56,9	55,0 OK

Tabella 8.24: bande in terze d'ottava che rispettano il livello minimo per S7

Seconda procedura di prova

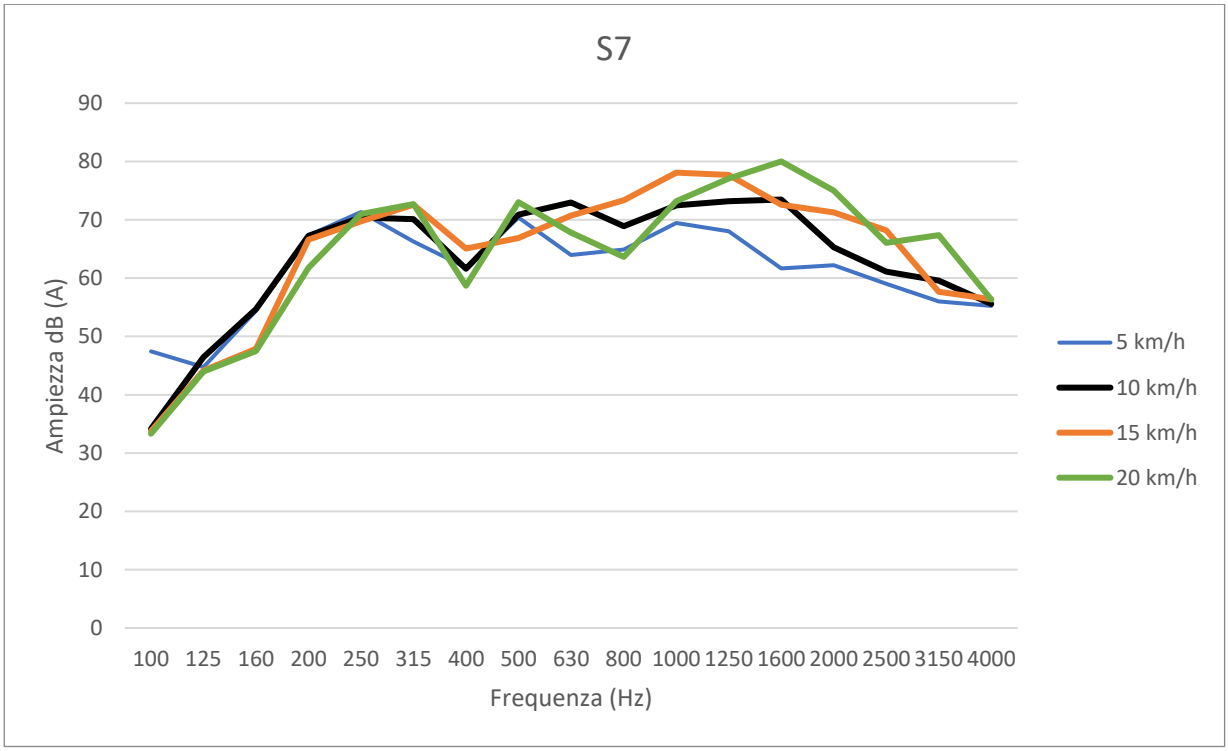

Figura 8.24: andamento in frequenza di S7 nelle quattro velocità di prova

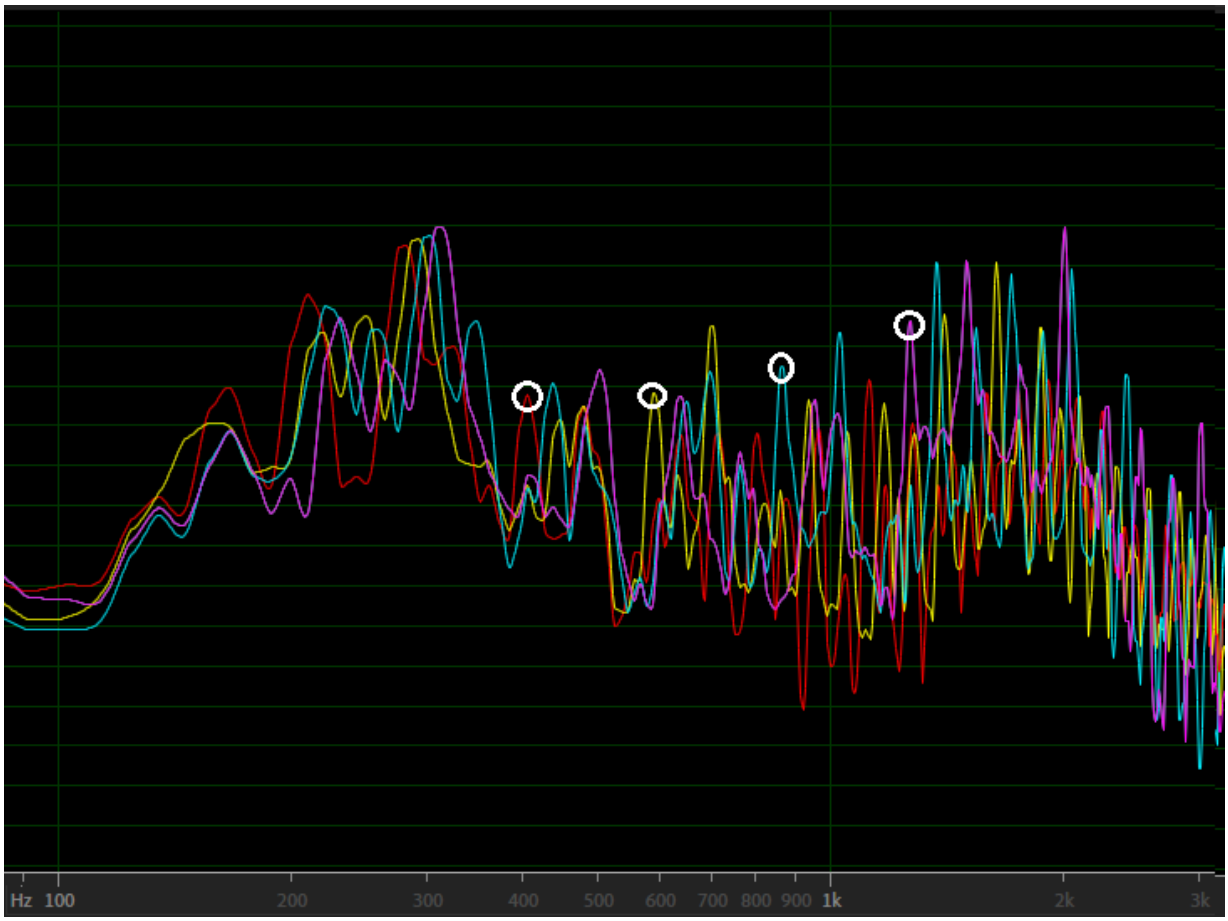

Figura 8.25: spettro in frequenza del segnale S7 a 5 (rossa), 10 (gialla), 15 (azzurra) e 20 (viola) km/h

		Risultati della prova alle velocità target			
		5 km/h (riferimento)	10 km/h	15 km/h	20 km/h
Frequenza	Hz	398	581	861	1260
Percentuale minima (per garantire rapporto di almeno 0.8 % per 1 km/h)	%	n.d.	4%	8%	12%
Variazione di frequenza	%	n.d.	9.2% OK	11.6% OK	14.4% OK

Tabella 8.25: risultati della prova di variazione in frequenza per il suono S7

9 Conclusioni

9.1 Risultati della verifica in camera anecoica

Suono	Conformità livelli	Conformità variazione in frequenza
S1	SI	NO
S2	SI	SI
S3	SI	SI
S4	NO	SI
S5	SI	NO
S6	SI	NO
S7	SI	SI

Tabella 9.1: risultati verifica in camera anecoica

Tra i sette suoni analizzati, solo 3 sono risultati conformi alla normativa: S2, S3 e S7.

I tre suoni S1, S5 e S6 sono stati considerati non conformi alla prova della variazione in frequenza. Infatti, nonostante sia effettivamente presente almeno un tono che cambia al variare della velocità rispettando la percentuale minima di variazione, la rilevazione di questa frequenza è risultata impossibile utilizzando solo le misurazioni effettuate durante la prova in camera anecoica, sono state necessarie registrazioni successive per individuare il tono. Per questi suoni andrebbe cambiato il bilanciamento tra i vari strati per rendere più evidente questa variazione.

Per quanto riguarda i livelli, l'unico suono che ha sforato i livelli massimi è S4. Essendo ben al di sopra dei livelli minimi, anche nelle bande in terzi di ottava, è sufficiente abbassare il livello complessivo del suono nelle varie condizioni di prova per ottenere un suono conforme alla normativa.

In generale i suoni S5, S6 e S7 pur rispettando la normativa erano ad un livello elevato, anche questi sono risultati molto al di sopra dei livelli minimi quindi potrebbero essere attenuati e risultare comunque conformi.

Una considerazione aggiuntiva riguarda la presenza di toni che variano linearmente in frequenza rispetto alla velocità (i quattro oscillatori in serie e lo strato di "realismo aggiuntivo"), queste variazioni hanno reso difficile la lettura dei toni, sarebbe stato più opportuno rendere l'andamento in frequenza degli altri strati sonori più statico per facilitare la lettura dei dati (specialmente nei suoni S1, S5 e S6).

9.2 Considerazioni aggiuntive e sviluppi futuri

Durante questo progetto nella fase di sound design si è lavorato principalmente in cuffia, questa è una limitazione in quanto alcuni difetti possono essere difficili da rilevare. In una prima fase sarebbe stato necessario lavorare in una stanza trattata acusticamente, con degli altoparlanti aventi risposta in frequenza lineare nel range d'interesse. Utilizzando l'altoparlante della Genelec, si sono rilevate risonanze indesiderate in un paio di condizioni di prova (S4 e S6 a 20 km/h) e la saltuaria presenza di clip da correggere.

Sulla base dei risultati ottenuti, sarebbe necessario apportare delle modifiche ai suoni progettati, in particolare lo spettro in frequenza del segnale avrebbe dovuto essere più statico al variare della velocità, in modo da rendere più facilmente individuabile il tono che cambia al variare della velocità stessa.

Il prototipo che legge dati dal veicolo è stato testato generando messaggi CAN tramite l'interfaccia "can-utils". Il suono avrebbe dovuto essere provato su un veicolo elettrico utilizzando l'altoparlante installato nel veicolo, che ha una sua risposta in frequenza. Inoltre la scocca del veicolo funge da filtro acustico, effetto che non è stato considerato. Questa fase finale di design descritta nel seminario della Siemens viene chiamata "fine tuning", il sound designer raffina il suono generato tenendo conto del risultato finale a bordo del veicolo.

La verifica in camera anecoica (o in una delle altre condizioni specificate dalla normativa) andrebbe effettuata generando il suono dell'AVAS con il veicolo, nella prova realizzata non sono state considerate la risposta in frequenza dell'altoparlante montato sul veicolo (che non è lo stesso utilizzato in camera anecoica, non essendo disponibile) e l'effetto di filtro acustico della carrozzeria.

Appendice

1 Simulazione veicolo tramite sensore ToF

```
import serial
import argparse
import random
import time
import keyboard
from pythonosc import osc_message_builder
from pythonosc import udp_client
import socket
def send_osc(client, listener, value):
    client.send_message(listener, value)
if __name__ == '__main__':
    PORT = 'COM2'
    ser = serial.Serial(PORT, 115200, timeout=10)
    parser = argparse.ArgumentParser()
    parser.add_argument("--ip", default='127.0.0.1',
                        help="The ip of the OSC server")
    parser.add_argument("--port", type=int, default='57120',
                        help="The port of the OSC server")
    args = parser.parse_args()
    client = udp_client.SimpleUDPClient(args.ip, args.port)
    send_osc(client, "/ignition", 0.0)
    first=True
    attenuator=0
    reverse=0
    while (1==1):

        if(ser.in_waiting >= 4):

            serialString = ser.read(4)
            if (serialString[0]==171):
                throttle = serialString[1]
                speed = serialString[2]

                print("Throttle: ", throttle, " - Speed: ", speed)
                send_osc(client, "/speedEngine", speed)
                send_osc(client, "/gaspedalpos", throttle)
        if keyboard.read_key() == "a":
            if attenuator == 0:
                send_osc(client, "/attenuator", 1)
                attenuator=1
            else:
                send_osc(client, "/attenuator", 0)
                attenuator=0
```

```
        if keyboard.read_key() == "r":
            if reverse == 0:
                send_osc(client, "/reversegear", 1)
                reverse=1
            else:
                send_osc(client, "/reversegear", 0)
                reverse=0
        if keyboard.read_key() == "q":
            send_osc(client, "/shutdown", throttle)
            break
```

L'attenuazione e la retromarcia sono simulate cliccando rispettivamente "a" e "r" dalla tastiera. Tramite "q" viene simulato lo spegnimento del veicolo.

2 Script python per decodifica messaggi CAN e invio messaggi OSC

```
import argparse
import random
import time
from pythonosc import osc_message_builder
from pythonosc import udp_client
import socket
import can
import cantools
from pprint import pprint
from datetime import datetime
db = cantools.database.load_file('/*.dbc')
# funzione per inviare messaggi OSC
def send_osc(client, listener, value):
    client.send_message(listener, value)
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--
ip", default='127.0.0.1', help="The ip of the OSC server")
    parser.add_argument("--
port", type=int, default='57120', help="The port of the OSC server")
    args = parser.parse_args()
    client = udp_client.SimpleUDPClient(args.ip, args.port)
    #definizione  degli id dei messaggi e il nome del segnale che interess
ano, variano a seconda del DBC utilizzato
    ignition_msg_id=
    ignition_sig_name=
    ignition_sig_ON=
    ignition_sig_OFF=
    speed_msg_id=
    speed_sig_name=
```

```
    speed_calc=
    gaspedal_msg_id=
    gaspedal_sig_name=
    reverse_msg_id=
    reverse_sig_name=
    attenuator_msg_id=
    attenuator=0
    with can.interface.Bus(
        bustype="socketcan", channel="can0", bitrate=125000
    ) as bus:
    try:
        while True:

            message = bus.recv(1)
                if(message is None):
                    continue

                decoded = db.decode_message(message.arbitration_id, messag
e.data)
                items = decoded.items()

                if(message.arbitration_id == ignition_msg_id):
                    search_key = ignition_sig_name
                    res = [val for key, val in items if search_key in key]
                    if res[0] == ignition_sig_OFF:  #spegnimento
                        send_osc(client, "/shutdown", 1)
                        break
                    elif res[0] == ignition_sig_ON:  #accensione
                        send_osc(client, "/ignition", 0)
                elif(message.arbitration_id == gaspedal_msg_id):  #posizio
ne pedale
                    search_key= gaspedal_sig_name
                    res = [val for key, val in items if search_key in key]
                    gas_position = res[0]
                    send_osc(client, "/gaspedal", gas_position)
                elif(message.arbitration_id == speed_msg_id): #velocità
                    search_key = speed_sig_name
                    res = [val for key, val in items if search_key in key]
                    speed = res[0]
                    send_osc(client, "/speedEngine", speed*speed_calc)
                elif(message.arbitration_id == reverse_msg_id):
#retromarcia
                    search_key = reverse_sig_name
                    res = [val for key, val in items if search_key in key]
                    reverse = res[0]
                    send_osc(client, "/reversegear", reverse)
```

```
                elif(message.arbitration_id == attenuator_msg_id): #attenu
azione
                    if attenuator == 0:
                        send_osc(client, "/attenuator", 1)
                        attenuator=1
                    else:
                        send_osc(client, "/attenuator", 0)
                        attenuator=0
```

3 Script supercollider

```
(
var envtri, sig, envsq, envsaw, startfirst, startsecond, startthird, startfourth, endfirst, endsecond,
endthird, endfourth;
~sin = Signal.sineFill(1024,1!1,0!1).asWavetable;
envtri = Env([0,1,0,-1,0]);
sig=envtri.asSignal(1024);
~tri=sig.asWavetable;
envsq = Env([1,1,-1], curve: \step);
sig=envsq.asSignal(1024);
~square=sig.asWavetable;
envsaw = Env([0,1,-1,0],[1,0,1], curve: \linear);
sig=envsaw.asSignal(1024);
~saw=sig.asWavetable;
~triBuf=Buffer.loadCollection(s, ~tri);
~squareBuf=Buffer.loadCollection(s, ~square);
~sawBuf=Buffer.loadCollection(s, ~saw);
~sinBuf=Buffer.loadCollection(s, ~sin);
~tri2Buf=Buffer.loadCollection(s, ~tri);
~saw2Buf=Buffer.loadCollection(s, ~saw);
~sin2Buf=Buffer.loadCollection(s, ~sin);
~square2Buf=Buffer.loadCollection(s, ~square);

//caricamento file in buffer
~scoppio=Buffer.readChannel(s, "*golf electric 2.wav", channels:[0]);
~grain=Buffer.readChannel(s,"*/string.wav", channels:[0]);
~grainele=Buffer.readChannel(s,"*/medio alto.wav", channels:[0]);

// definizione dei range di frequenza dei quattro oscillatori
startfirst=198;
endfirst=225.33
startsecond=224;
endsecond=261;
startthird=263;
endthird=305;
startfourth= 308;
endfourth=357.6;
```

```
// all'accensione del veicolo tutti gli strati vengono inizializzati
OSCdef ('ignitionlistener', {
   arg msg;
   var buftri, first, second, third, fourth, fifth, bufsq, bufsaw, bufsin, buf2tri, buf2saw, buf2sin,
buf2sq, scoppiosynth, grainsynth, molti;
   buftri=~triBuf.bufnum;
   bufsq=~squareBuf.bufnum;
   bufsaw=~sawBuf.bufnum;
   bufsin=~sinBuf.bufnum;
   buf2tri=~tri2Buf.bufnum;
   buf2saw=~saw2Buf.bufnum;
   buf2sin=~sin2Buf.bufnum;
   buf2sq=~square2Buf.bufnum;
   molti=1;
   ~first = Synth.new(\blend, [\bufpos, bufsin + 0.01, \start, 0.linlin(0, 20, startfirst, endfirst),
\volume, 0.linlin(0, 20, 0.03, 0.025), \molti, molti, \fm, 0.linlin (0,20,20,30.67)]);
   ~second = Synth.new(\blend, [\bufpos, buftri + 0.04, \start, 0.linlin(0, 20, startsecond,
endsecond), \volume, 0.linlin(0, 20, 0.03, 0.025), \molti, molti, \fm, 0.linlin (0,20,20,30.67)]);
   ~third = Synth.new(\blend, [\bufpos, bufsin + 0.7, \start, 0.linlin(0, 20, startthird, endthird)
,\volume, 0.linlin(0, 20, 0.03, 0.025), \molti, molti, \fm, 0.linlin (0,20,20,30.67)]);
   ~fourth = Synth.new(\blend, [\bufpos, bufsaw + 0.95, \start, 0.linlin(0, 20, startfourth,
endfourth),  \volume, 0.linlin(0, 20, 0.007, 0.006), \molti, molti, \fm, 0.linlin (0,20,20,30.67)]);
   ~scoppiosynth=Synth.new(\scoppio, [\rate, 0.linlin(0, 20, 1.1, 1.2), \bufnum,
~scoppio.bufnum, \startPos, 0, \loop, 1, \volume, 0.linlin(0, 20, 0.08, 0.07), \molti, molti]);
   ~grainsynth=Synth.new(\granularspeed, [\speedrate, 0.linlin(0, 20, 100, 200), \bufnum,
~grain.bufnum, \playbackrate, 0.linexp(5, 20, 0.6, 2), \startPos, 0.linlin(0, 20, 0.8, 0.7), \volume,
0.linlin(0, 20, 0.55, 0.53), \duration, 0.linlin(0, 20, 0.15, 0.12)]);
   ~grainpedal=Synth.new(\granularpedal, [\bufnum, ~grainele.bufnum, \startPos,
~grainele.numFrames*0.5, \volume, 0]);
}, "/ignition");

// allo spegnimento del veicolo tutti gli strati vengono liberati tramite il metodo "free"
OSCdef ('shutdownlistener', {
   arg msg;
   ~first.free;
   ~second.free;
   ~third.free;
   ~fourth.free;
   ~fifth.free;
   ~scoppiosynth.free;
   ~grainsynth.free;
   ~grainpedal.free;
}, "/shutdown");
```

```
// quando varia la velocità del veicolo il suono viene modificato in tempo reale
OSCdef ('speedlistener', {
   arg msg;
   ~first.set(\start, msg[1].linlin(0, 20, startfirst, endfirst), \volume, msg[1].linlin(0, 20, 0.03,
0.025));
   ~second.set(\start, msg[1].linlin(0, 20, startsecond, endsecond), \volume, msg[1].linlin(0, 20,
0.03, 0.025));
   ~third.set(\start, msg[1].linlin(0, 20, startthird, endthird), \volume, msg[1].linlin(0, 20, 0.03,
0.025));
   ~fourth.set(\start, msg[1].linlin(0, 20, startfourth, endfourth), \volume, msg[1].linlin(0, 20,
0.007, 0.006));
   ~scoppiosynth.set(\rate, msg[1].linlin(0, 20, 1.1, 1.2), \volume, msg[1].linlin(0, 20, 0.08, 0.07),
\playbackrate, msg[1].linexp(5, 20, 0.6, 2));
   ~grainsynth.set(\speedrate, msg[1].linlin(0, 20, 100, 200));
   ~grainsynth.set(\startPos, msg[1].linlin(0, 20, 0.8, 0.7));
   ~grainsynth.set(\playbackrate, msg[1].linlin(0, 20, 0.6, 2));
   ~grainsynth.set(\volume, msg[1].linlin(0, 20, 0.55, 0.53));
   ~grainsynth.set(\duration, msg[1].linlin(0, 20, 0.15, 0.12));
   if (msg[1]>20){
         ~first.set(\volume, msg[1].linlin(20, 150, 0.025, 0.005));
         ~second.set(\volume, msg[1].linlin(20, 150, 0.025, 0.005));
         ~third.set(\volume, msg[1].linlin(20, 150, 0.025, 0.005));
         ~fourth.set(\volume, msg[1].linlin(20, 150, 0.006, 0.001));
         ~scoppiosynth.set(\volume, msg[1].linlin(20, 150, 0.07, 0.01));
         ~grainsynth.set(\volume, msg[1].linlin(20, 150, 0.53, 0.15));
   }
}, "/speedEngine");

// quando viene premuto l'apposito pulsante il suono viene attenuato
OSCdef ('attenuationlistener', {
   arg msg;
   var attenuator=0.5;
   if (msg[1]==0){
                  attenuator=1;
   };
   ~first.set(\molti, attenuator);
   ~second.set(\molti,attenuator);
   ~third.set(\molti, attenuator);
   ~fourth.set(\molti, attenuator);
   ~scoppiosynth.set(\molti, attenuator);
   ~grainsynth.set(\molti, attenuator);
   ~grainpedal.set(\molti, attenuator);
}, "/attenuator");
```

```
// quando viene premuto il pedale aumenta il volume dello strato “potenza”
OSCdef ('gaspedallistener', {
   arg msg;
   if (msg[1]>10){
         ~grainpedal.set(\volume, msg[1].linlin(10, 100, 0.0005, 0.13));
   }
}, "/gaspedalpos");

// quando viene inserita la retromarcia il suono viene enfatizzato
OSCdef ('reverselistener', {
   arg msg;
   var mul=1.5;
   if (msg[1]==0){
         mul=1;
   };
   ~first.set(\molti, mul);
   ~second.set(\molti,mul);
   ~third.set(\molti, mul);
   ~fourth.set(\molti, mul);
   ~scoppiosynth.set(\molti, mul);
   ~grainsynth.set(\molti, mul);
   ~grainpedal.set(\molti, mul);
}, "/reversegear");

// definizione dei SynthDef
SynthDef.new (\scoppio ,{
   arg out=0, bufnum=0, rate=1, startPos=0, loop=1, volume=0.8, molti=1;
   Out.ar(out, molti*volume*BPF.ar(PlayBuf.ar(1, bufnum: bufnum, rate:rate, trigger: 1.0,
startPos: startPos, loop: loop), freq: 1000, rq: 5 )!2);
}).add;

SynthDef.new (\granularspeed ,{
   arg out=0, bufnum=0, startPos=0, speedrate=100, duration=0.5, playbackrate=1, volume= 0.8,
molti=1;
   Out.ar(out, molti*volume*BPF.ar(GrainBuf.ar (1, Impulse.kr(speedrate), duration +
LFNoise1.kr(1000).exprange(0.01,0.02), bufnum, playbackrate, startPos +
LFNoise1.kr(1000).exprange(0.01,0.02), 4), freq: 2500, rq: 1.5)!2);
}).add;

SynthDef.new (\granularpedal ,{
   arg out=0, bufnum=0, startPos=0.6, speedrate=400, duration=0.1, playbackrate=0.16,
volume=0.08, molti=1;
         Out.ar(out, molti*volume*BPF.ar(GrainBuf.ar (1, Impulse.ar(speedrate), 0.2 +
LFNoise1.kr(100).exprange(0.001,0.005) , bufnum, 0.16, startPos +
LFNoise1.kr(100).exprange(0.1,0.2), 2), freq: 1000, rq: 0.3)!2);
}).add;
```

```
SynthDef.new (\blend, {
    arg out=0, bufpos=0, start= 180, volume=0.01, molti=1,fm;
    var modulator;
      modulator = SinOsc.ar(fm,0,0.1,1);
    Out.ar(out, molti*volume*BPF.ar(VOsc.ar (bufpos, start, mul:
modulator), freq: 1000, rq: 5 )!2);
    }).add;
)
```

Bibliografia

[1] Fleury, S., Jamet, É., Roussarie, V., Bosc, L., & Chamard, J.-C. (2016). *Effect of additional warning sounds on pedestrians' detection of electric vehicles: An ecological approach.*

[2] Parizet, E., Ellermeier, W., & Robart, R. (2014). *Auditory warnings for electric vehicles: detectability in normal-vision and visually impaired listeners*.

[3] Misdariis, N., & Pardo, L.-F. (2018). *The sound of silence of electric vehicles – Issues and answers*.

[4] Verrecas, B. *Creare il suono adatto per i veicoli elettrici con l'Active Sound Design (Webinar Siemens)*.

[5] *Regolamento (UE) N. 540/2014 del parlamento europeo e del consiglio*. (2014).

[6] *Regolamento n. 138 della Commissione economica per l'Europa delle Nazioni Unite (UNECE)*. (2016).

[7] *Supercollider*. (1996). Tratto da Supercollider: https://supercollider.github.io/

[8] Valle, A. (2015). *Introduzione a Supercollider*.

[9] Petiot, J.-F., Krinstensen, B. G., & Maier, A. M. (2013). *How Should an Electric Vehicle Sound? User and Expert Perception*.

[10] *Sensore di prossimità V6180*. Tratto da STMicroelectronics: https://www.st.com/content/st_com/en/about/media-center/pressitem.html/it/stmicroelectronics-proximity-sensor-solves-smartphone-hang-ups.html

[11] *Supercollider documentation*. Tratto da https://doc.sccode.org/ [12] Farnell, A. (2010). Designing Sound.

[13] *Modello di suono di un motore a quattro tempi*. Tratto da Wikibooks: https://en.wikibooks.org/wiki/Designing_Sound_in_SuperCollider/Cars

[14] *Igniter*. Tratto da Krotos Audio: https://www.krotosaudio.com/igniter/

[15] *Spettrogramma*. Tratto da Wikipedia: https://it.wikipedia.org/wiki/Spettrogramma

[16] *Phaser*. Tratto da Wikipedia: https://it.wikipedia.org/wiki/Phaser_(musica)

[17] *Controller Area Network*. Tratto da Wikipedia: https://it.wikipedia.org/wiki/Controller_Area_Network

[18] *USB2CAN user manual*. Tratto da Github: https://github.com/INNO-MKER/usb2can

[19] *Camera anecoica*. Tratto da Wikipedia: https://it.wikipedia.org/wiki/Camera_anecoica

[20] *Datasheet microfono tipo 4950*. Tratto da Bruel & Kjaer: https://www.bksv.com/-/media/literature/Product-Data/bp2180.ashx

[21] *8030A Studio Monitor*. Tratto da Genelec: https://www.genelec.com/previous-models/8030a

[22] *Le curve di ponderazione*. (s.d.). Tratto da Bioecotecnica: http://www.bioecotecnica.it/documenti%20acustica/5-curve%20di%20ponderazione.pdf