UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

REALIZZAZIONE DIGITALE DI UN SISTEMA MULTIEFFETTO PER CHITARRA ELETTRICA ATTRAVERSO DSP

RELATORE: prof. Buso Simone

LAUREANDO: Fasolato Tomas
1216266

ANNO ACCADEMICO: 2022/2023

28 settembre 2023

Indice

Introduzione **1**

1 Descrizione Hardware **2**
1.1 Scheda Nucleo . . . 2
1.2 Input Stage . . . 3
1.3 Output Stage . . . 7
1.4 Power Stage . . . 8
1.5 Realizzazione PCB . . . 9

2 Descrizione Codice **11**
2.1 Schema a blocchi del codice . . . 11
2.2 Configurazione delle periferiche . . . 13
2.2.1 ADC 1 . . . 13
2.2.2 ADC 2 . . . 15
2.2.3 DAC . . . 16
2.2.4 Digital IN/OUT . . . 17
2.2.5 Pinout Scheda Nucleo . . . 17
2.3 Descrizione Effetti . . . 19
2.3.1 Distortion . . . 19
2.3.2 Tremolo . . . 20
2.3.3 Chorus . . . 22
2.3.4 Generazione Sinusoide . . . 23

3 Test **25**
3.1 Setup di misura . . . 25
3.1.1 Oscilloscopio . . . 25
3.1.2 Generatore di funzioni . . . 26
3.2 Riproduzione onda originale . . . 27
3.3 Test effetti . . . 27
3.3.1 Test effetto Distortion . . . 27
3.3.2 Test effetto Tremolo . . . 28
3.3.3 Test effetto Chorus . . . 28

Conclusione **30**

Bibliografia **31**

Introduzione

In questo elaborato sono raccolti i passaggi chiave per il progetto e la realizzazione di un sistema multieffetto per chitarra elettrica, sotto forma di pedale così da permetterne l'utilizzo mentre si suona lo strumento. Per la realizzazione si è scelto di utilizzare il microcontrollore *STM32* montato nella scheda NUCLEO "STM32F334R8", studiata nel corso di elettronica industriale durante il terzo anno.

Il suono della chitarra viene convertito attraverso i pickup in un segnale di tensione. Il pedale, utilizzando un circuito di condizionamento, adatta il segnale ai range di ingresso del microcontrollore che attraverso il convertitore interno digitalizza il segnale. A questo punto gli effetti vengono implementati tramite Digital Signal Processing (DSP), l'innesco e il disinnesco degli stessi avviene tramite appositi switch, mentre il controllo dei parametri è realizzato da appositi potenziometri collegati al secondo convertitore della scheda Nucleo. Infine il DAC interno si occupa di riconvertire il segnale elaborato per fornirlo in ingresso al circuito di condizionamento in Output, utilizzato per adattare il segnale prima che entri nell'amplificatore o nel resto della catena del suono.

L'obiettivo finale, oltre a prendere dimestichezza con l'utilizzo di un microcontrollore nella realizzazione pratica di un progetto completo, è muovere i primi passi nel modo del Digital Signal Processing realizzando una catena di acquisizione elaborazione e trasmissione di un segnale. Si è infine scelto di realizzare un PCB contenente i circuiti di condizionamento da collegare alla scheda Nucleo così da rendere i collegamenti più sicuri e meno affetti da disturbi rispetto all'implementazione su breadboard.

1 Descrizione Hardware

Definiti il concept generale del progetto e gli obiettivi da perseguire si è potuto passare alla ricerca dei componenti e circuiti utili alla sua realizzazione.

1.1 Scheda Nucleo

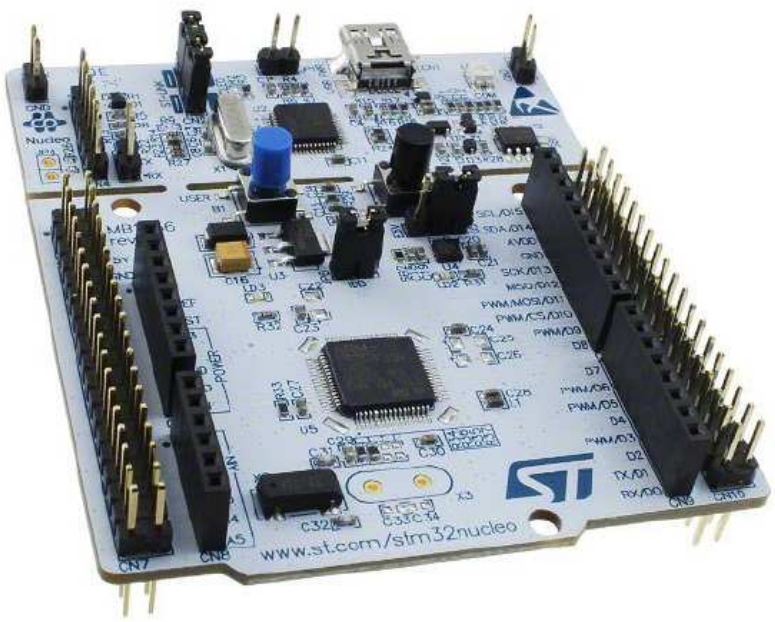

Figura 1.1: Scheda Nucleo F334R8

La scheda scelta per sviluppare questo progetto è la NUCLEO F334R8 (denominata semplicemente scheda Nucleo) di casa STMicroelectronics, una board programmabile che monta il microcontrollore STM32F334R8; appartenente alla famiglia STM32F3 è basato su un core ARM Cortex M4 le cui caratteristiche principali sono elencate di seguito.

Core	Cortex M4	
Aritmetica	32 bit	con unità floating point
Frequenza di clock	72	MHz
Memoria Flash	fino a 64	kByte
Memoria SRAM	fino a 12	kByte
Pin I/O	fino a 51	

La scheda è divisa in due parti, eventualmente separabili, una principale in cui è alloggiato il microcontrollore e tutte le sue periferiche e una secondaria in cui è presente *ST-LINK/V2-1 debugger/programmer*. La parte secondaria ha la funzione di interfacciare la board al PC, com'è facilmente intuibile dalla presenza del connettore miniB-USB, al suo interno è presente anche il debugger che permette (grazie all'IDE proprietario) di verificare il codice eseguendolo a blocchi separati. La scheda ha la possibilità di essere

alimentata in due modi, scelti attraverso la posizione di un jumper specifico[1]. Il primo modo è fornendo una tensione a un pin specifico della scheda: da 7 a 12 V al pin VIN, 5 V al pin E5V e 3.3 V al pin +3V3. Il secondo modo è attraverso il connettore miniB-USB collegando la scheda al pc o ad un alimentatore USB. I punti di forza di questa board sono sicuramente le sue periferiche, e il gran numero di pin disponibili, che la rendono un sistema completo ed economico per lo sviluppo di semplici progetti/prototipi come questo. Di seguito vengono citate le più rilevanti:

- 2 ADC a 12 bit, un totale di 21 ingressi e un ampiezza di ingresso consentita fre 0 e 3.6 V;
- 2 DAC a 12 bit con un totale di 3 uscite analogiche e possibilità di configurare un buffer in uscita
- DMA a 7 canali (Direct Memory Acces) configurabili anche per l'utilizzo con ADC e DAC
- fino a 16 timer, fra i quali uno ad alta risoluzione (217 ps) HRTIM a 16 bit con 10 canali indipendenti
- fino a 51 pin configurabili come I/O (non contemporaneamente dato che alcuni sono condivisi con il resto di periferiche)
- 2 led già connessi a dei pin digitali, (molto utili per il debug del codice)
- 2 pulsanti anch'essi connessi ad ingressi digitali e già pronti per essere utilizzati nel codice.

La scheda Nucleo fornisce attraverso un pin predisposto una tensione pari a 3.3V, utilizzata per il collegamento di switch e potenziometri.

1.2 Input Stage

Il circuito di interfacciamento fra chitarra e microcontrollore è necessario per due motivi principali: in primo luogo adatta l'ampiezza del segnale proveniente dallo strumento per renderla compatibile con i parametri di ingresso dell'ADC montato sulla scheda Nucleo. Inoltre limita la banda del segnale di ingresso alla frequenza di Nyquist così da evitare aliasing e disturbi dovuti a componenti ad alta frequenza che non contengono informazione utile.

La chitarra attraverso i suoi pickup trasforma l'oscillazione delle corde metalliche in un segnale di tensione. La banda di questo segnale rispecchia la banda di frequenze raggiunte dallo strumento, ovverosia dalla nota più bassa che ha una frequenza di circa 80Hz alla nota più alta che arriva a toccare i 1100Hz circa. Il segnale ha media nulla e un'ampiezza che arriva a toccare 800mV di picco, sia positivi che negativi.

[1]Per una descrizione più approfondita sul connettore di alimentazione e sulla sua configurazione si rimanda il lettore allo user manual della board che contiene informazioni più dettagliate a riguardo

L'ADC della scheda Nucleo accetta valori di tensione in ingresso che vanno da 0V a 3.6V e viene fatto lavorare ad una frequenza di campionamento di circa 57Khz. Questi parametri rendono necessaria l'aggiunta di una componente continua e di un fattore di scala al nostro segnale di partenza. Doverosa è anche l'aggiunta di un filtraggio pari al più alla frequenza di Nyquist ovvero 28Khz.

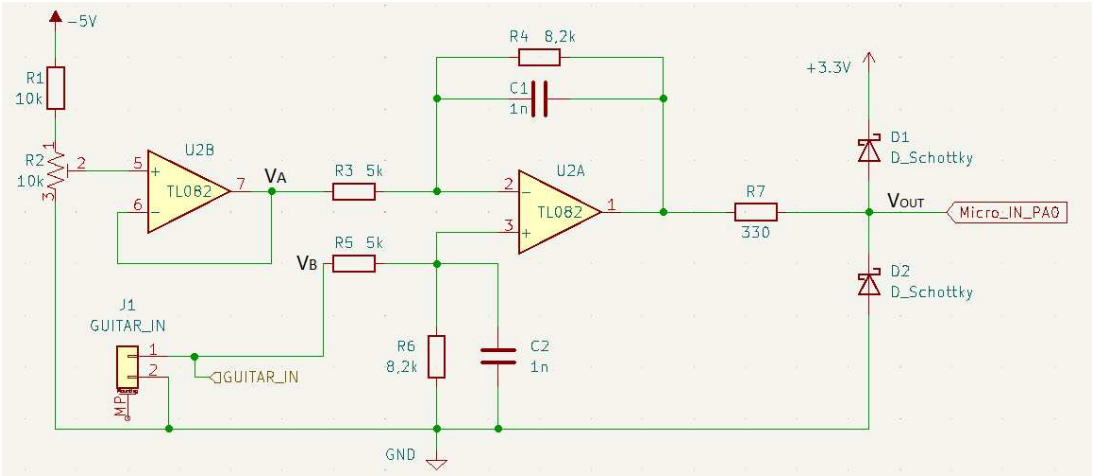

Figura 1.2: Schema elettrico INPUT Stage

Per realizzare le suddette trasformazioni al segnale di ingresso è stato scelto di utilizzare il circuito in Figura 1.2 avente la seguente funzione di trasferimento.

$$V_{OUT} = (V_B - V_A) \cdot \frac{R4}{R3} \cdot \frac{1}{1 + sCR4}$$

Segnale	Descrizione
V_A	Segnale di offset prelevato dall'alimentazione negativa (dato il collegamento al morsetto invertente)
V_B	Segnale di ingresso proveniente dalla chitarra
V_{OUT}	Segnale di uscita del circuito, collegato all'ingresso del convertitore

Per una maggiore sicurezza è stata considerata un ampiezza massima del segnale proveniente dalla chitarra pari a $\pm$ 1V, inoltre sono stati inseriti i due diodi Schottky all'ingresso del microcontrollore a scopo di protezione. Se per qualsiasi ragione il segnale in ingresso alla scheda superasse la soglia dei 3.3V [2] o scendesse sotto gli 0V uno dei due diodi entrerebbe subito in conduzione, limitando il segnale ad una delle due soglie e salvaguardando l'ingresso della scheda da possibili danni.

Dimensionamento componenti:

Ipotesi:

$$R = R4 = R6, \quad C = C1 = C2 \quad R3 = R5$$

[2]Valore di tensione fornito da un pin della scheda Nucleo. Scelto come livello di riferimento a scapito dei 5V perché minore dei 3.6V massimi accettati come ingresso al convertitore A/D

Scelgo di ottenere una frequenza di taglio del filtro pari a 20 KHz, circa venti volte maggiore della frequenza massima generata dalla chitarra.

Scelgo C pari a 1 nF, calcolo R

$$F_c = \frac{1}{2\pi RC} \Rightarrow R = \frac{1}{2\pi F_C \cdot C} = \frac{1}{2\pi \cdot 20 \cdot 10^3 \cdot 10^{-9}} = 7957\,\Omega$$

Il valore della serie E12 più vicino a quello ottenuto è 8.2 KΩ, il che porta ad avere un effettiva frequenza di taglio pari a :

$$F_{Creal} = \frac{1}{2\pi 8,2 \cdot 10^{-6}} = 19.409 KHz$$

Calcolo di V_A, utilizzata per alzare il valore del segnale affinché non assuma mai valori negativi.

$$V_B - V_A \geq 0,\, V_{Bmin} = -1\,V \Rightarrow V_a \leq V_{Bmin} = -1\,V \Rightarrow V_a \leq -1\,V$$

Avendo già preso un sufficiente margine di sicurezza nella scelta del valore di soglia inferiore di V_B (-1 V) è stata posta $V_A = -1\,V$. Una volta montato il circuito sarà possibile regolare precisamente V_A al valore corretto con l'aiuto di un multimetro agendo sul trimmer R2.

Calcolo di R3, responsabile del guadagno in continua del filtro ($\frac{R4}{R3}$) quindi trascurando il termine ($\frac{1}{sC \cdot R}$).

$$V_{OUT} = (V_{Bmax} - V_A) \cdot \frac{R4}{R3} \leq 3.3\,V,\, V_{Bmax} = 1\,V \;\Rightarrow 2 \cdot \frac{R4}{R3} \leq 3.3 \Rightarrow$$

$$\Rightarrow R3 \geq \frac{2 \cdot R4}{3.3}\,\Omega = \frac{2 \cdot 8.2}{3.3}\,K\Omega = 4.96\,K\Omega \Rightarrow R3 \geq 4.96\,K\Omega$$

Per R3 è stato scelto un valore di $5K\Omega$ ottenuto collegando in parallelo due resistori dal valore 10 $K\Omega$ contenuti nella serie E12, per semplicità grafica negli schemi è indicata solo la resistenza risultante.

Di seguito in Figura 1.3 e 1.4 vengono riportati i diagrammi di Bode (Modulo) del circuito di ingresso, Simulato in alto e reale in basso.

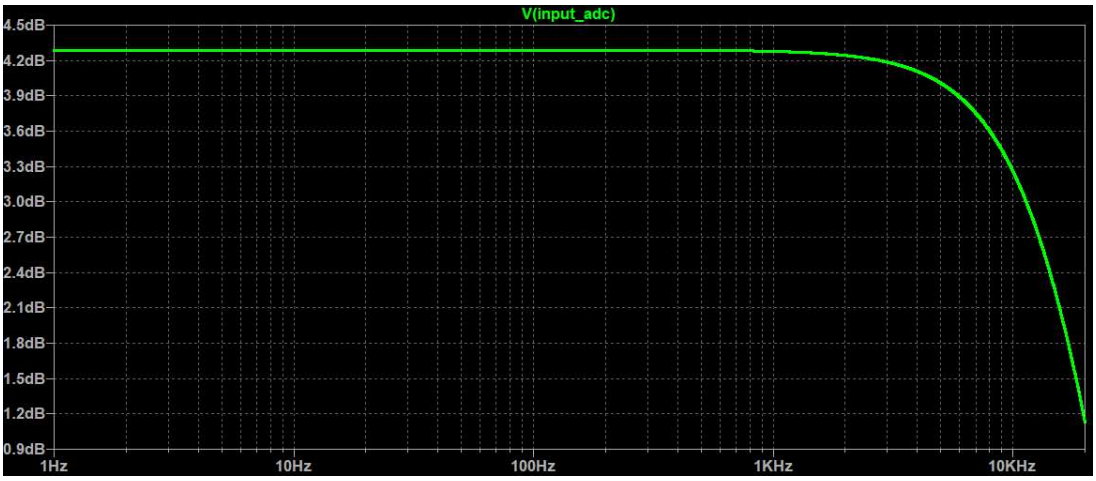

Figura 1.3: Diagramma di Bode del modulo dell'INPUT stage, SIMULAZIONE

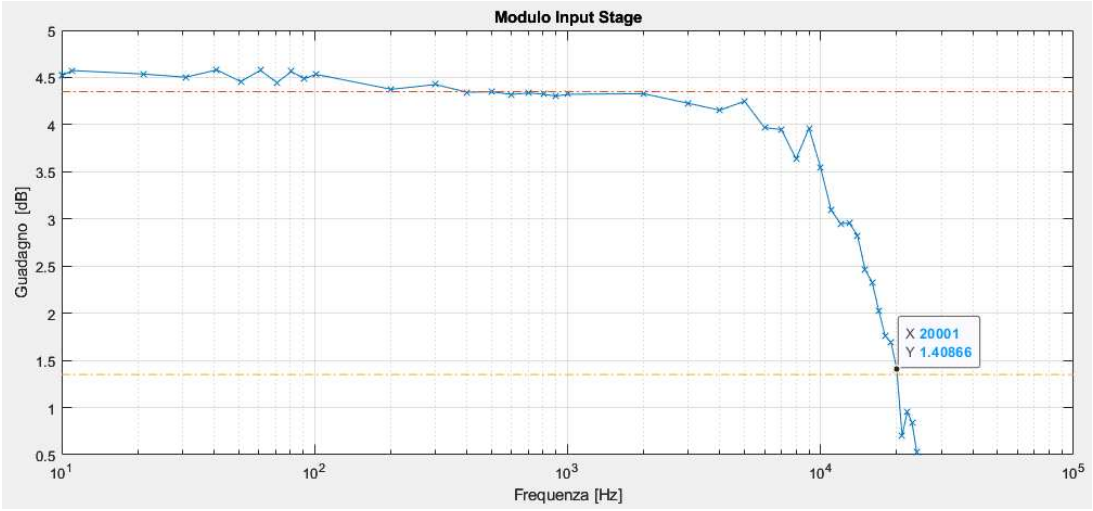

Figura 1.4: Diagramma di Bode del modulo dell'INPUT stage, Real

Come si può vedere dal secondo grafico il circuito rispecchia abbastanza fedelmente il comportamento in frequenza della simulazione, la linea giallo scuro è posta a -3dB dal valore di guadagno in banda ed evidenzia una frequenza di taglio di circa 20Khz, fedele al valore di progetto.

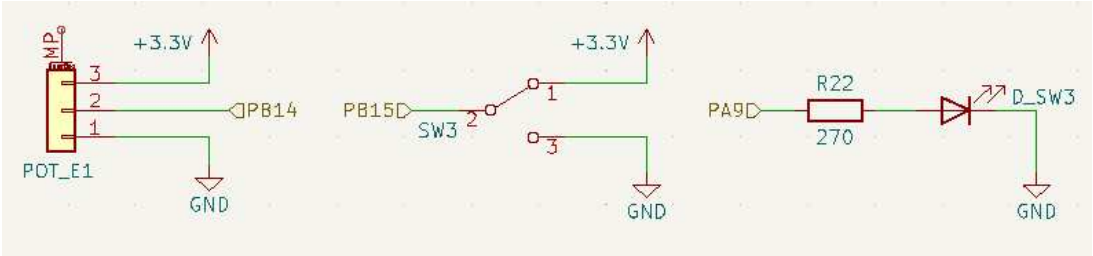

Figura 1.5: Collegamento di un Potenziometro(sinistra), di uno switch(centro) e di un LED (destra)

Collegamento Switch Potenziometri e LED Per l'interazione e il controllo del pedale si rendono necessari switch, potenziometri e Led di indicazione, in Figura 1.5 si

possono vedere come questi tre tipi di componenti sono stati collegati ai pin del microcontrollore, utilizzando per il livello digitale high la tensione di 3.3V fornita direttamente dalla scheda Nucleo.

1.3 Output Stage

Il circuito di uscita ha la funzione di ricostruire il segnale proveniente dal DAC della scheda Nucleo e riadattarlo in modo da poter essere reinserito nella catena del suono che va da strumento ad amplificatore. Per far ciò si rendono necessari due circuiti di filtro a cascata: un passa alto e un passa basso, che di fatto realizzano un filtro passabanda a due stadi. Il primo ha la funzione di eliminare la componente continua dal segnale per tornare ad avere un segnale a media nulla, utilizzabile dagli amplificatori. Mentre il secondo ha la funzione di ricostruire il segnale proveniente dal DAC tagliando tutte le componenti in frequenza che non contengono informazioni ovvero tutte le frequenze che vanno dai circa 1100Hz (nota più alta della chitarra) a salire.

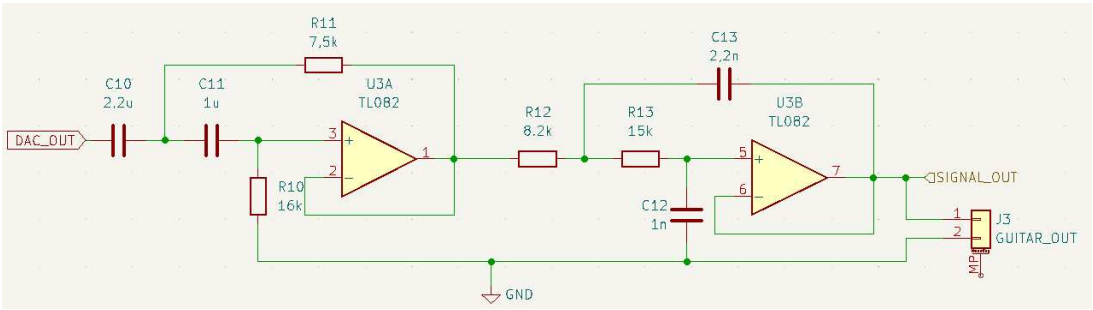

Figura 1.6: Schema elettrico OUTPUT Stage

Per velocizzare la realizzazione dei filtri di uscita e per facilitare l'operazione di dimensionamento si è scelto di utilizzare il filter design tool messo a disposizione online gratuitamente dall'azienda Texas Instruments. I parametri impostati sono stati *Frequenza di taglio superiore = 10KHz* (circa 10 volte maggiore alla frequenza massima contenente informazione) e *Frequenza di taglio inferiore = 10Hz* (circa 10 volte minore della frequenza minima contenente informazione). Ulteriore condizione di cui tener conto è l'obbligo di scegliere una frequenza di taglio superiore minore o uguale alla frequenza di Nyquist dettata dal convertitore. Scegliendo 10KHz come Freq. di taglio superiore tale condizione è rispettata (Fsample = 57KHz, fNy = 28 KHz) Funzione molto interessante del tool è quella di mettere a disposizione dell'utente la scelta della serie dei componenti reali posseduti, rendendo così visibile la differenza fra i valori di progetto e i valori reali.

Di seguito vengono riportati i diagrammi di Bode (Modulo) del circuito di uscita, ottenuti da simulazione in alto e reali in basso.

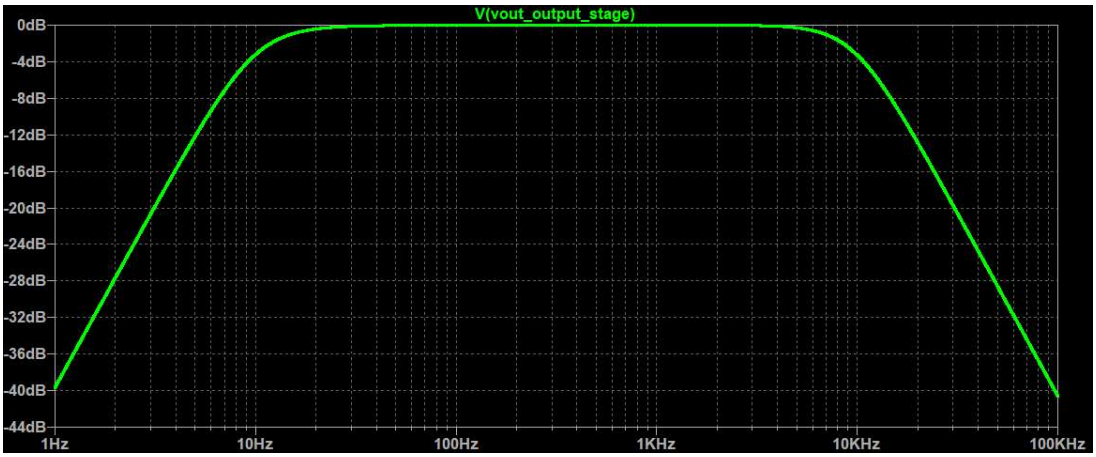

Figura 1.7: Diagramma di Bode del modulo dell'OUTPUT stage, SIMULAZIONE

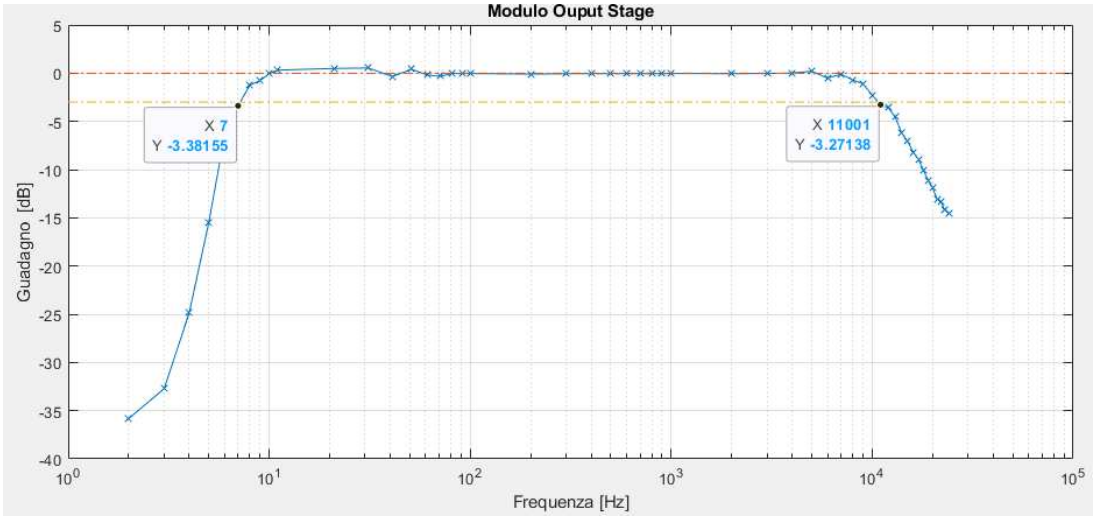

Figura 1.8: Diagramma di Bode del modulo dell'OUTPUT stage, SIMULAZIONE

come si può notare dalle immagini il comportamento in frequenza della simulazione viene rispecchiato nel diagramma reale, la linea color giallo scuro è stata posta a -3dB dalla linea rossa che rappresenta il Guadagno in banda passante. Si denota una frequenza di taglio inferiore pari a circa 7Hz e una frequenza di taglio superiore di circa 11 kHz, valori che si discostano leggermente da quelli di progetto, ma comunque accettabili per lo scopo di questo filtro.

Per la realizzazione di questi circuiti la scelta dell'amplificatore operazionale è ricaduta sul **TL082** prodotto da Texas Instruments. La possibilità di gestire un'alimentazione duale, la sua banda passante di 3 MHz, una THD dello 0.003% lo rendono ideale per filtrare i semplici segnali audio di questo progetto.

1.4 Power Stage

Doveroso è un breve accenno all'insieme di elementi che compongono il circuito di alimentazione. Essendo questo un progetto che utilizza segnali audio, il circuito di alimentazione deve consentire ai vari circuiti di gestire tali segnali. In particolare agli amplificatori operazionali coinvolti nei circuiti di Input e di Output è necessario fornire una tensione di

alimentazione duale così da renderli in grado di gestire segnali a media nulla. Fra le varie opzioni disponibili per generare una tensione duale, quella adottata in questo progetto è stata scelta principalmente per la disponibilità immediata dei componenti utilizzati.

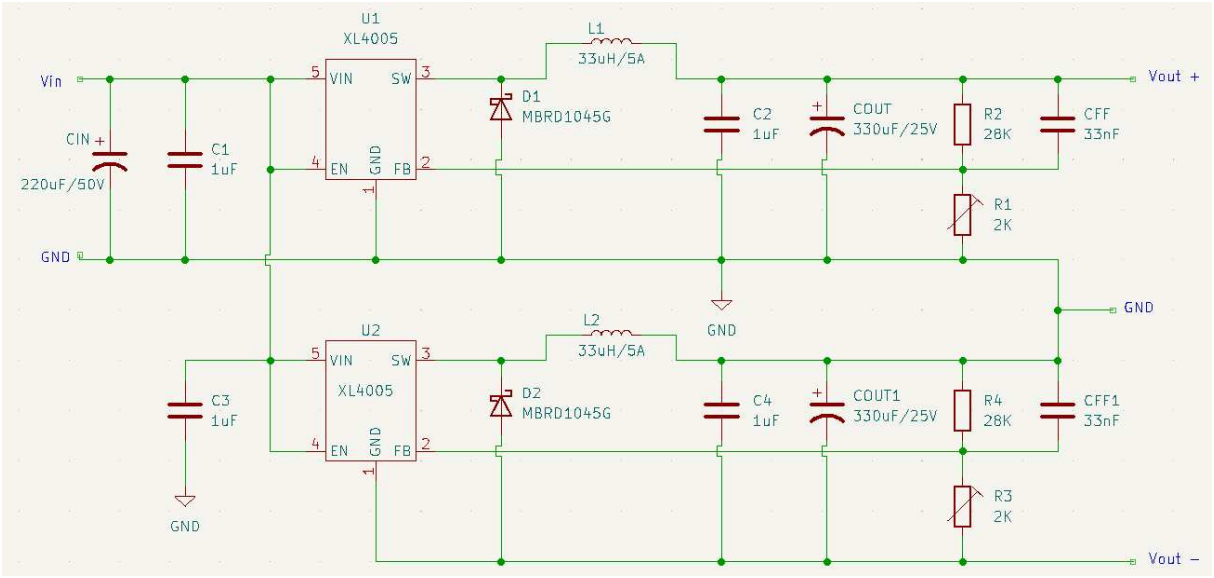

Figura 1.9: Schema elettrico POWER Stage

La tensione di ingresso al circuito è fornita da un comune alimentatore a 12 volt, tipico dei pedali per chitarra. Il sistema è stato realizzato connettendo in serie come da schema elettrico due convertitori Buck (DC-DC step down). Componente cardine di questi convertitori è l'integrato XL4005 prodotto da XLSEMI che ha le seguenti caratteristiche principali:

- Tensione di ingresso da 5 V a 32 V;
- Tensione in uscita regolabile da 0.8V a 30 V;
- Corrente Massima di 5 A;
- Circuito interno di protezione da cortocircuiti.

Si è così ottenuto un circuito in grado di fornire tensioni positive e negative regolabili attraverso due trimmer. La tensione di uscita è stata regolata a $\pm$ 5 V, così da riuscire ad alimentare sia gli amplificatori operazionali sia la scheda Nucleo.

1.5 Realizzazione PCB

Per la realizzazione del PCB è stata scelta la suite KiCad, che mette a disposizione un ambiente di sviluppo CAD open source in grado di gestire sia la parte di schemi elettrici sia la parte riguardante la progettazione dello stampato. E' stata realizzata una board in grado di connettersi direttamente alla scheda Nucleo utilizzando i connettori Morpho già presenti di serie, con questo approccio si crea una connessione stabile fra le due schede, evitando il più possibile problemi e malfunzionamenti dovuti a falsi contatti o a contatti laschi che si possono verificare specie utilizzando delle bradboard. Di seguito si può vedere il risultato della progettazione CAD ovvero il footprint della scheda.

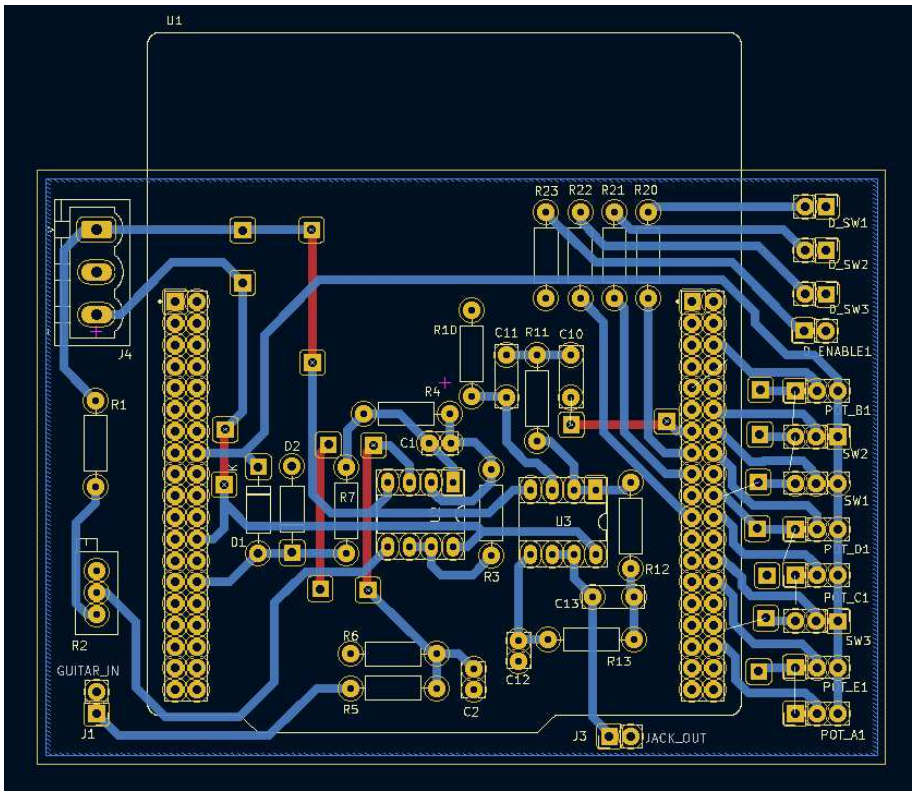

Figura 1.10: Footprint PCB

2 Descrizione Codice

Dopo aver trovato l'hardware su cui sviluppare il progetto si è potuto passare alla programmazione della scheda Nucleo. La scelta dell'ambiente di sviluppo è ricaduta su un software di casa Keil, in particolare il microVision 5, già studiato e utilizzato durante le attività di laboratorio del corso di elettronica industriale. Una fra le funzioni più utili integrata in questo IDE è quella di debug del codice. Come si vede in Figura 2.1 è possibile visualizzare il valore in tempo reale sia delle variabili sia dei registri di memoria della scheda, in cui il codice viene eseguito passo passo. Questa funzione insieme ai led già integrati nella scheda e collegati alle uscite digitali (e quindi controllabili da righe di codice) rendono il processo di scrittura e verifica del codice più semplice.

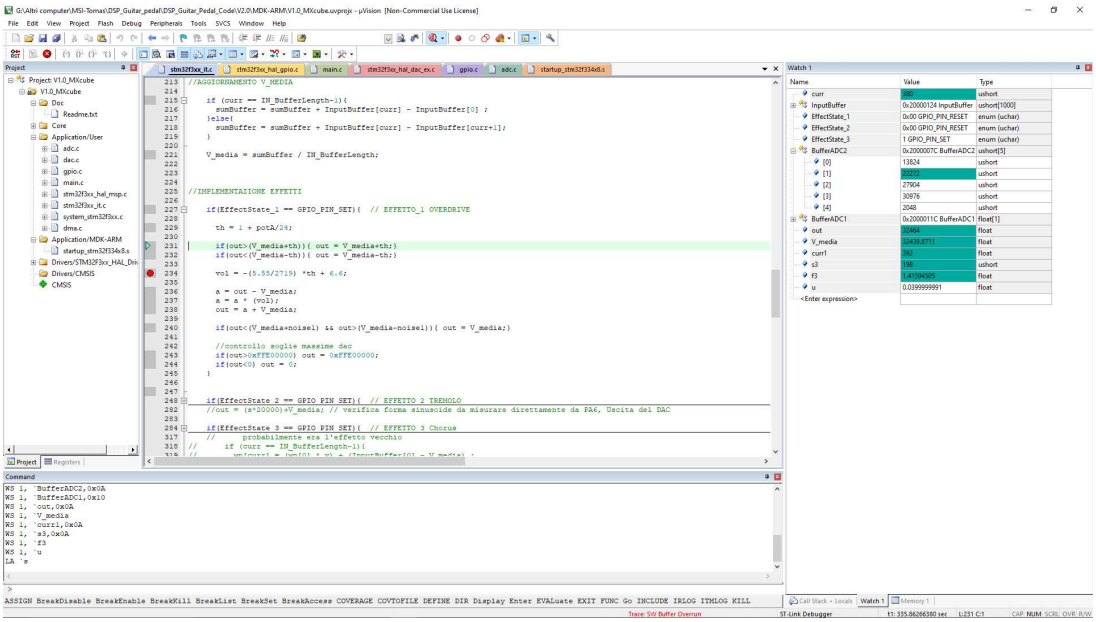

Figura 2.1: Finestra dell'IDE microvision5 in modalità debug, sulla destra si può notare una watchwindow in cui sono indicati i valori di alcune variabili

2.1 Schema a blocchi del codice

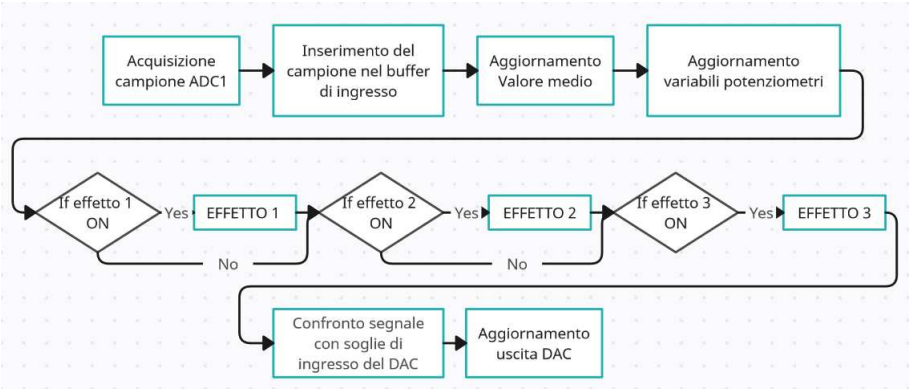

Figura 2.2: Schema a blocchi del codice dell'ISR relativa all'ADC1

Come stile di stesura del codice si è scelto il classico approccio ad interrupt: quando una periferica ha terminato l'esecuzione delle sue operazioni genera un interrupt mandando in esecuzione una routine di servizio specifica scritta dall'utente (ISR, Interrupt Service Routine). La periferica principale di questa applicazione è senz'altro l'ADC1, in quanto è lui il responsabile dell'acquisizione dei campioni dal segnale di ingresso ed è la sua frequenza di campionamento che detterà la velocità dell'intero sistema. Dato il ruolo centrale dell'ADC1 la parte più rilevante di codice è contenuta nella sua ISR. Lo schema a blocchi visibile in figura 2.2 mostra appunto la catena di acquisizione ed elaborazione del segnale. Una volta che il convertitore ha terminato l'acquisizione di un campione questo viene inserito in un buffer di ingresso e il campione più vecchio viene eliminato, realizzando di fatto un buffer circolare così da avere a disposizione uno storico di campioni per realizzare le varie elaborazioni[1]. Una volta aggiornato il buffer di ingresso si passa al calcolo della componente media del segnale, codice riportato in Figura 2.3. Questa operazione permette, se necessario, di riportare il segnale a media nulla o comunque di conoscere con precisione la componente continua del segnale.

```
//AGGIORNAMENTO V_MEDIA

    if (curr == IN_BufferLength-1){
      sumBuffer = sumBuffer + InputBuffer[curr] - InputBuffer[0] ;
    }else{
      sumBuffer = sumBuffer + InputBuffer[curr] - InputBuffer[curr+1];
    }

    V_media = sumBuffer / IN_BufferLength;
```

Figura 2.3: Porzione di codice in cui si calcola il valor medio del segnale di ingresso

Proseguendo nella lettura dello schema a blocchi si vede una serie di strutture condizionali a cascata. Questo è il punto in cui sono implementati i vari effetti. La struttura if controlla lo stato di una variabile che rispecchia lo stato dello switch (relativo all'effetto in considerazione) e in base a questa attiva l'elaborazione del segnale oppure no, per poi passare all'effetto successivo. Infine dopo aver subito le varie elaborazioni descritte il campione viene confrontato con i valori di soglia massimi e minimi permessi dal DAC e viene inviato in uscita.

Avendo inserito la parte principale del codice all'interno dell'ISR, al main rimane lo scopo di inizializzare tutte le periferiche e di controllare lo stato degli switch per la prima assegnazione delle variabili di stato degli effetti, come si vede in Figura 2.4, prima di entrare nel loop infinito.

[1]In particolare si fa riferimento ad alcuni effetti ritardanti, che uniscono la componente attuale del segnale con dei campioni più vecchi, un esempio è l'effetto Chorus descritto nella sezione 2.3.3

```c
int main(void)
{
  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC2_Init();
  MX_ADC1_Init();
  MX_DAC2_Init();

  //Prima assegnazione variabili effetto_____________________________________
  EffectState_1 = HAL_GPIO_ReadPin(GPIOA, PIN_EF1);
  HAL_GPIO_WritePin(GPIOA, PIN_LED_EF1, EffectState_1);

  EffectState_2 = HAL_GPIO_ReadPin(GPIOA, PIN_EF2);
  HAL_GPIO_WritePin(GPIOB, PIN_LED_EF2, EffectState_2);

  EffectState_3 = HAL_GPIO_ReadPin(GPIOB, PIN_EF3);
  HAL_GPIO_WritePin(GPIOA, PIN_LED_EF3, EffectState_3);
  //        ____________________________________________________________

  while (1)
  {}

}
```

Figura 2.4: Porzione di codice che raffigura la funzione main

2.2 Configurazione delle periferiche

Un processo fondamentale per arrivare a un codice funzionante è quello di configurazione delle periferiche. Documento fondamentale per portare a termine la configurazione è il Reference Manual del microcontrollore ([8]) fornito dal produttore, al suo interno contiene tutte le informazioni dettagliate e le istruzioni per configurare il chip. La stessa casa produttrice mette a disposizione anche un software di configurazione automatica con interfaccia grafica, meno ricco di informazioni rispetto al Reference manual, ma decisamente più user frendly. Il software, *STM32CubeMX*, è in grado di generare già un progetto compilabile e funzionante per l'IDE microVision così da rendere la configurazione più veloce e soprattutto meno affetta da errori di sintassi. Tuttavia, uno studio del Reference manual è stato necessario per modificare direttamente dal codice alcune funzioni di configurazione e per capire a fondo il funzionamento di altre, come ad esempio il funzionamento dei convertitori analogico digitali accoppiati al DMA. Inoltre, il software ha solo funzione di configurazione delle periferiche, infatti non inserisce i comandi start e di stop delle stesse, lasciati all'utente.

2.2.1 ADC 1

Questo convertitore analogico digitale, come detto in precedenza, è la periferica principale su cui si basa il funzionamento del progetto. Dovendo acquisire un segnale audio si è scelto di far lavorare il convertitore con il maggior numero di bit disponibili ovvero 12. Dovendo utilizzare il pedale mentre si suona lo strumento, la latenza del segnale fra ingresso e uscita dovrà essere minima; fondamentale è quindi la scelta della modalità di funzionamento del convertitore affinché venga garantita sia una frequenza di campiona-

mento adeguata al segnale in ingresso [2] sia un ritardo minimo fra acquisizione dell'ingresso e attivazione dell'uscita. Date le premesse è stato scelto di far funzionare il convertitore in *"CONTINUOUS CONVERSION MODE"*, come si vede in figura 2.5, in questa modalità (una volta fatto partire alzando il segnale "ADSTART") il convertitore inizia a convertire una sequenza di canali (in questo caso solo uno); una volta finita la conversione questa riparte nuovamente mentre viene attivato un flag EOS(end of sequence) il quale attiverà un interrupt che permetterà l'esecuzione dell'ISR dedicata.

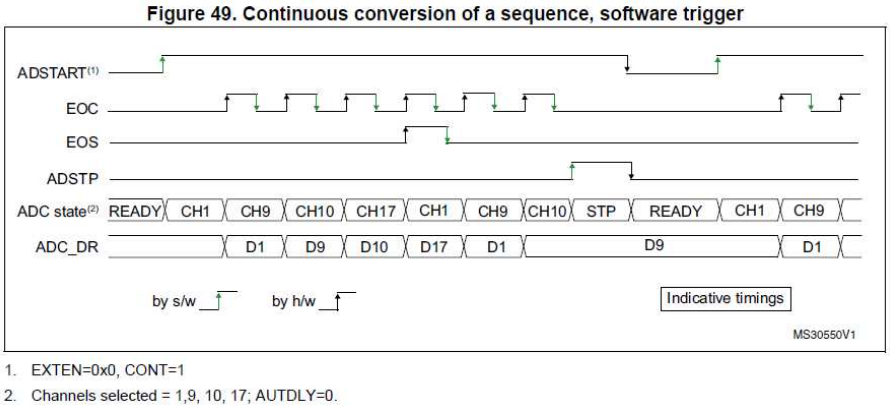

Figura 2.5: Grafico che mostra la scansione temporale di una conversione di una sequenza di canali

Data la modalità di conversione continua, per controllare la frequenza di campionamento del segnale bisogna agire sul tempo di conversione di un canale; ciò è possibile attraverso due registri, uno che sceglie il numero di bit del convertitore (già impostato a 12) e il secondo sceglie quanti cicli di clock attendere fra una conversione e la successiva. Questo secondo registro offre la possibilità di impostare un numero limitato di valori, nel nostro caso è stato scelto il valore massimo ovvero *"601.5"*. Un altro parametro su cui si può agire è la frequenza di lavoro del convertitore (F_{ADCCLK}), modificabile attraverso l'impostazione di un prescaler che di fatto divide la frequenza di clock del microcontrollore; nel nostro caso il prescaler è impostato a 2, il che porta ad avere una F_{ADCCLK} pari a 36MHz. Avendo tutti i parametri necessari si può calcolare il tempo di conversione come segue:

$$T_{ADC} = T_{SMPL} + T_{SAR} = [601.5_{|min} + 12.5_{|12bit}] \cdot T_{ADCCLK}$$

$$T_{ADC} = 17.05 \cdot 10^{-6} \Rightarrow F_{ADC} = 58.631KHz$$

Essendoci solo un canale da convertire F_{ADC} può essere considerata anche come frequenza di campionamento, dato che terminata una conversione ne inizia subito una successiva.

[2] una frequenza di campionamento tipica delle schede audio di fascia media per pc è di 48 KHz

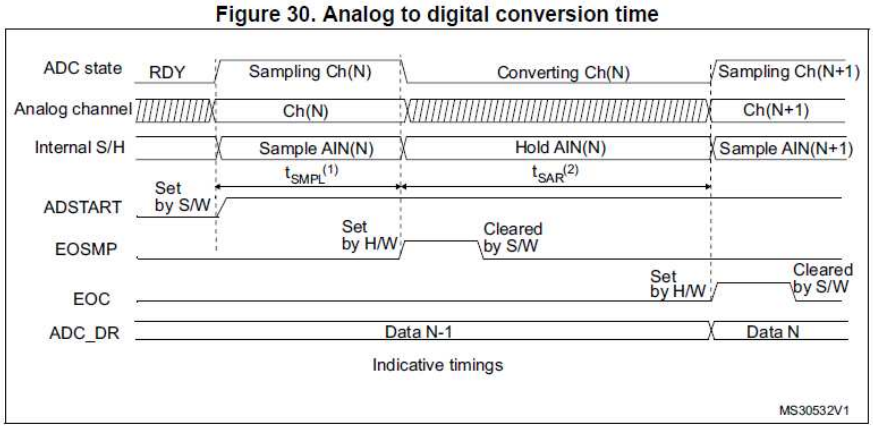

Figura 2.6: Grafico che mostra il tempo impiegato per convertire un singolo canale

La Figura 2.6 mostra nel dettaglio il tempo di conversione di un canale, in particolare si può notare che il tempo fra il set del segnale ADSTART e quello del set di EOC (end of conversion) coincide con quanto riportato sopra nel calcolo di T_{ADC}.

L'onda in figura 2.7 è stata ottenuta alzando e abbassando un pin digitale all'interno dell'ISR dell'ADC1, così da evidenziare il tempo di conversione (distanza fra un fronte di salita e il successivo) e quindi la frequenza di campionamento che, come si può vedere, è simile a quella calcolata attraverso la formula.

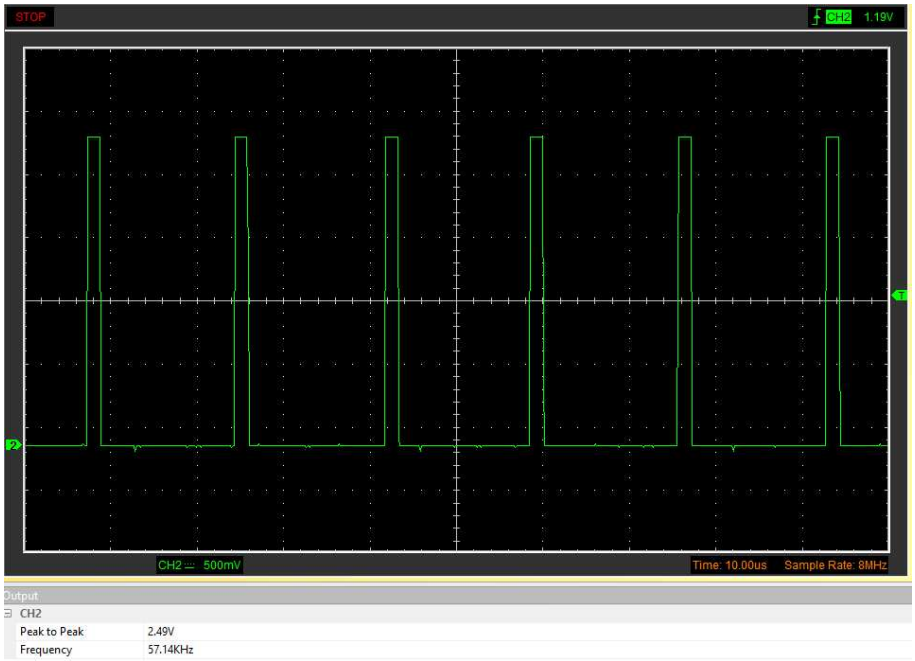

Figura 2.7: Test della frequenza di campionamento dell'ADC1

2.2.2 ADC 2

Il secondo convertitore analogico digitale ha lo scopo di convertire i 5 segnali provenienti dai potenziometri, responsabili del controllo degli effetti del pedale. Data la minore precisione richiesta si è deciso di far lavorare l'ADC2 a 8 bit. Lo scopo dei potenziometri

è quello di "servire" la catena di effetti fornendo, nella maniera più semplice possibile, il valore a cui sono impostati senza impiegare troppe risorse e senza istruzioni di start e stop che tolgano tempo alla parte principale di programma. Per rispettare le premesse è stato scelto di utilizzare l'approccio a DMA (Direct Memory Access conrtoller) con cui si può integrare l'ADC. Il DMA è una porzione di hardware che si occupa del trasferimento di dati da periferiche a memoria o da memoria a memoria senza l'utilizzo della CPU così da lasciarla libera per altre porzioni di codice. l'STM32f334R8 è fornito di un DMA a 7 canali, per la nostra applicazione, 5 di questi sono stati assegnati ai canali dell'ADC2; così, una volta avviata la periferica all'accensione del microcontrollore, i valori dei potenziometri saranno automaticamente disponibili e aggiornati in un vettore, accessibile dal codice e quindi utilizzabile nelle elaborazioni degli effetti. Essendo il DMA perfettamente integrato con le altre periferiche la sua configurazione ha richiesto poche righe di codice oltre a quelle della sua inizializzazione generate automaticamente da STMCube32.

In Figura 2.9 è evidenziata la riga di codice che abilita la DMA request per l'ADC2, mentre in Figura 2.8 si vede il comando che fa partire l'adc e definisce il vettore in cui saranno salvati i dati.

```
HAL_ADCEx_Calibration_Start(&hadc2, ADC_SINGLE_ENDED);
HAL_ADC_Start_DMA(&hadc2,(uint32_t*)BufferADC2, BufferADC2Length);
```

Figura 2.8: Comando con cui si avvia l'ADC2 e si imposta il vettore in cui il DMA salverà i dati

```
hadc2.Instance = ADC2;
hadc2.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
hadc2.Init.Resolution = ADC_RESOLUTION_8B;
hadc2.Init.ScanConvMode = ADC_SCAN_ENABLE;
hadc2.Init.ContinuousConvMode = ENABLE;
hadc2.Init.DiscontinuousConvMode = DISABLE;
hadc2.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
hadc2.Init.ExternalTrigConv = ADC_SOFTWARE_START;
hadc2.Init.DataAlign = ADC_DATAALIGN_LEFT;
hadc2.Init.NbrOfConversion = 5;
hadc2.Init.DMAContinuousRequests = ENABLE;
hadc2.Init.EOCSelection = ADC_EOC_SEQ_CONV;
hadc2.Init.LowPowerAutoWait = DISABLE;
hadc2.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
```

Figura 2.9: Porzione di codice in cui viene configurato l'ADC2, in particolare sono evidenziate rispettivamente la riga in cui si imposta il prescaler a 2, la riga in cui vengono impostati 8 bit di risoluzione e la riga in cui si abilita la DMA request

2.2.3 DAC

Una volta finite le elaborazioni dei campioni il compito di fornirli al circuito di uscita spetta al convertitore digitale analogico (DAC). In questo caso è stato scelto di far lavorare il DAC con il maggior numero di bit a disposizione (12 bit) così da ottenere una migliore precisione nel segnale di uscita. Il controllo dell'uscita del DAC avviene nell'ISR dell'ADC1, una volta finite le elaborazioni degli ingressi, questo porta il dac a lavorare circa alla frequenza dell'ADC1 descritta sopra. In figura 2.10 si vede il comando con cui si invia l'uscita al DAC, si possono notare il numero di bit impostati e l'allineamento del dato.

```
//AGGIORNAMENTO USCITA DAC
HAL_DAC_SetValue(&hdac2,DAC2_CHANNEL_1,DAC_ALIGN_12B_L, out);
```

Figura 2.10: Riga di codice in cui si imposta l'uscita del dac

2.2.4 Digital IN/OUT

Come descritto in precedenza sono presenti nel progetto 3 switch che hanno la funzione di attivare e disattivare gli effetti. Per la loro programmazione si è scelto nuovamente un approccio ad interrupt e non a polling, così da non dover impiegare periodicamente risorse per verificarne lo stato. Il segnale di interrupt viene generato quando lo switch cambia di stato, a quel punto il programma esgue l'ISR dedicata in cui si esegue la verifica dello stato dello switch e vengono aggiornati di conseguenza la variabile di stato e il pin collegato al LED indicatore. In Figura 2.11 si vede a scopo la porzione di ISR dedicata al controllo dell'effetto 1, mentre in Figura 2.12 è mostrata la porzione di codice in cui vengono configurati i pin GPIO come interrupt.

```
// ISR EFFECT_1 CHECK
if(__HAL_GPIO_EXTI_GET_FLAG(PIN_EF1)||__HAL_GPIO_EXTI_GET_IT(PIN_EF1))
{

  EffectState_1 = HAL_GPIO_ReadPin(GPIOA, PIN_EF1);
  HAL_GPIO_WritePin(GPIOA, PIN_LED_EF1, EffectState_1);

  __HAL_GPIO_EXTI_CLEAR_FLAG(PIN_EF1);
  __HAL_GPIO_EXTI_CLEAR_IT(PIN_EF1);
  HAL_GPIO_EXTI_Callback(PIN_EF1);
}
```

Figura 2.11: Porzione di codice in cui viene controllato lo stato di uno switch, in questo caso quello relativo all'effetto 1

```
/*Configure GPIO pin : PB15 */
GPIO_InitStruct.Pin = PIN_EF3;
GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pins : PA11 PA12 */
GPIO_InitStruct.Pin = PIN_EF1|PIN_EF2;
GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

Figura 2.12: Porzione di codice in cui vengono configurati come interrupt i pin collegati agli switch

2.2.5 Pinout Scheda Nucleo

In Figura 2.13 è indicata la distribuzione finale dei pin della scheda Nucleo, descritti successivamente da una tabella

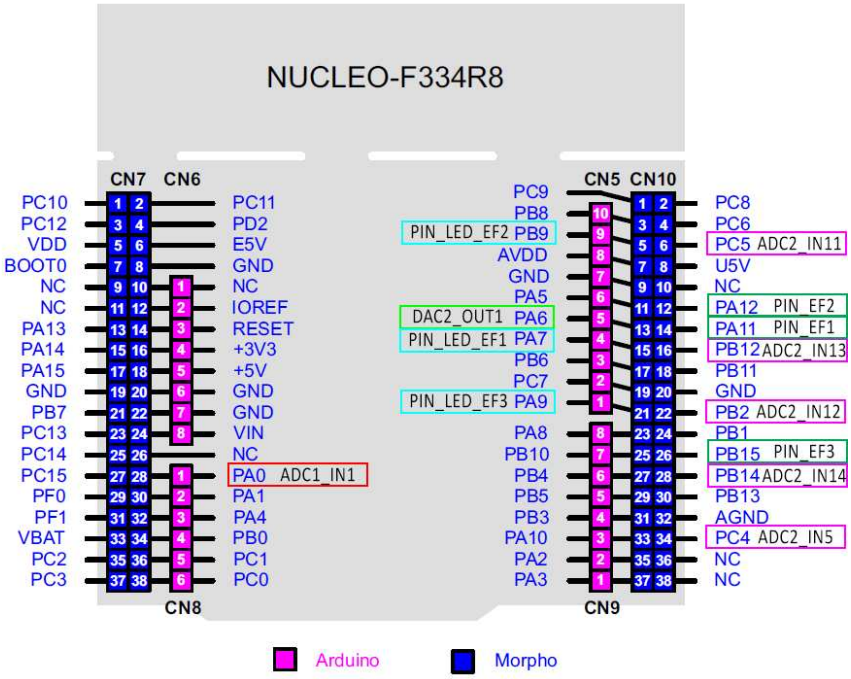

Figura 2.13: Pinout della scheda Nucleo

Segnale	Pin Scheda Nucleo	Descrizione
ADC_1_IN1	PA0	Segnale in INPUT all'ADC principale proveniente dalla catena del suono
ADC_2_IN5	PC4	Ingresso all'ADC secondario proveniente dal Potenziometro A
ADC_2_IN11	PC5	Ingresso all'ADC secondario proveniente dal Potenziometro B
ADC_2_IN12	PB2	Ingresso all'ADC secondario proveniente dal Potenziometro C
ADC_2_IN13	PB12	Ingresso all'ADC secondario proveniente dal Potenziometro D
ADC_2_IN14	PB14	Ingresso all'ADC secondario proveniente dal Potenziometro E
DAC_2_OUT1	PA6	Segnale in OUTPUT che dal DAC va ai filtri dell'output stage
PIN_EF1	PA11	Pin che attiva un interrupt al cambio di stato per abilitare/disabilitare l'effetto 1
PIN_EF2	PA12	Pin che attiva un interrupt al cambio di stato per abilitare/disabilitare l'effetto 2
PIN_EF3	PB15	Pin che attiva un interrupt al cambio di stato per abilitare/disabilitare l'effetto 3
PIN_LED_EF1	PA7	Pin configurato come Digital Output e collegato ad un LED che indica lo stato dell'effetto 1
PIN_LED_EF2	PB9	Pin configurato come Digital Output e collegato ad un LED che indica lo stato dell'effetto 2
PIN_LED_EF3	PA9	Pin configurato come Digital Output e collegato ad un LED che indica lo stato dell'effetto 3

2.3 Descrizione Effetti

2.3.1 Distortion

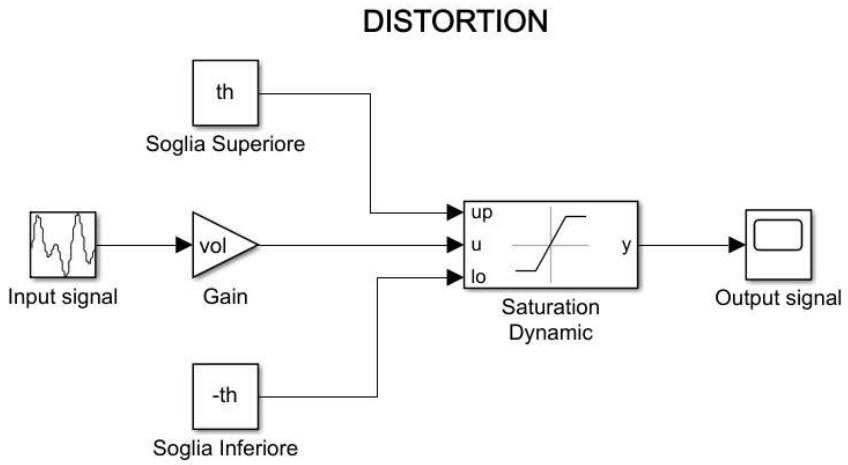

Figura 2.14: Schema a blocchi effetto Distortion

L'effetto Distortion (o clipper) è uno degli effetti per chitarra elettrica più famosi in assoluto. La sua facilità di realizzazione, sia a livello circuitale nella versione analogica, sia a livello di complessità di codice nella versione corrente, lo rende ideale per muovere i primi passi nel mondo degli effetti per chitarra. L'obiettivo di questo effetto è quello di creare un suono più "graffiato" rispetto all'originale, in particolare per realizzarlo viene creata una finestratura nella parte centrale del segnale attraverso due livelli di soglia simmetrici che di fatto realizzano un clipping, ovvero non viene permesso al segnale di superare ne la soglia superiore ne quella inferiore. Successivamente, attraverso un guadagno, viene amplificata la zona centrale rimanente. Il risultato nello spettro del segnale è la comparsa di armoniche a frequenze superiori, responsabili dell'effetto finale di distorsione. I parametri che controllano questo effetto sono due:

- Il livello di soglia;
- Guadagno del segnale di ingresso

Essendo le soglie simmetriche rispetto al valor medio del segnale è necessario un solo valore per controllarle entrambe. Il guadagno del segnale in ingresso all'effetto è necessario in quanto abbassando le soglie il volume del segnale in uscita sarebbe troppo basso. Per il controllo dell'effetto l'utente ha a disposizione un solo potenziometro dato che i due parametri sono legati via software da relazioni matematiche, abbassando le soglie viene alzato automaticamente il guadagno per far rientrare il segnale all'interno del range migliore.

```
if(EffectState_1 == GPIO_PIN_SET){  // EFFETTO_1 Distortion

  th = 1 + potA/24;

  if(out>(V_media+th)){ out = V_media+th;}
  if(out<(V_media-th)){ out = V_media-th;}

  vol = -(5.55/2719) *th + 6.6;

  a = out - V_media;
  a = a * (vol);
  out = a + V_media;

  if(out<(V_media+noisel) && out>(V_media-noisel)){ out = V_media;}

  //controllo soglie massime dac
  if(out>0xFFE00000) out = 0xFFE00000;
  if(out<0) out = 0;
}
```

Figura 2.15: Codice Effetto Distortion

In Figura 2.15 si può vedere lo scorcio di codice che realizza l'effetto Distortion descritto sopra. In particolare, si può notare che è stata aggiunta un ulteriore finestratura più ristretta attorno al valor medio così da eliminare il più possibile i rumori di fondo. Essendo molto ridotta in ampiezza non crea effetti rilevanti per quanto concerne l'introduzione di armoniche indesiderate. La riduzione del rumore di fondo mediante finestratura attorno al valor medio e il controllo del valore da inserire nel DAC con le soglie massime consentite (ultime due righe di codice visibili) sono stati inseriti in tutti gli effetti e nella porzione di codice principale.

2.3.2 Tremolo

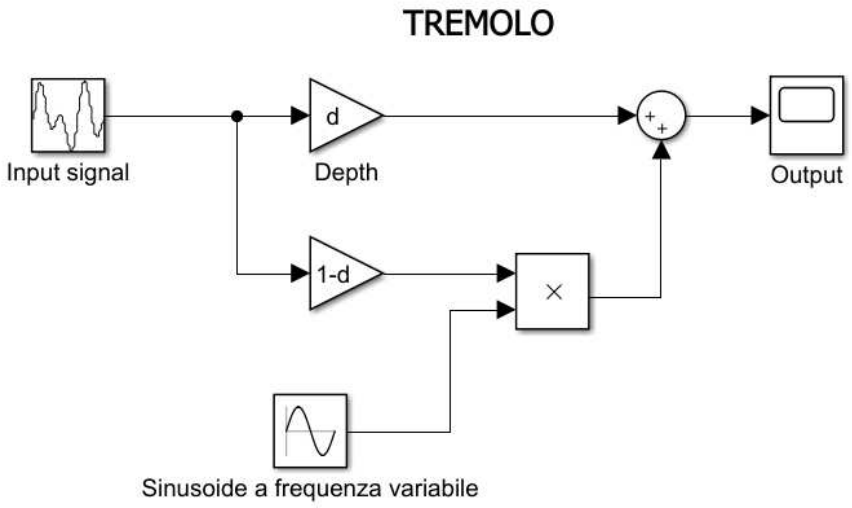

Figura 2.16: Schema a blocchi effetto Tremolo

Funzione di trasferimento Tremolo y = uscita, x[n] = campione corrente

$$y[n] = (x[n] \cdot (d-1)) + (x[n] \cdot d \cdot sen(f))$$

L'effetto Tremolo per chitarra elettrica è un altro degli effetti base, nonché uno dei primi storicamente realizzati proprio perché è possibile realizzarlo anche in modo "manuale", alzando e abbassando il potenziometro del volume della chitarra. Il tremolo infatti nella sua realizzazione più semplice, quella scelta per questo pedale, modula il volume di uscita con frequenza e intensità regolate dall'utente. Come segnale modulante è stata scelta una sinusoide a frequenza variabile generata dalla funzione *sen* descritta in seguito. Questo effetto è controllato da due parametri, uno che determina la frequenza della sinusoide e un secondo che aumenta o diminuisce l'intensità della modulazione. Questi due parametri sono controllati dai potenziometri B e C, in particolare per modulazione più intensa si intende quando la componente moltiplicata alla sinusoide (rappresentata dal ramo più basso nello schema a blocchi in Figura 2.16) prevale sulla componente originale ottenendo così un effetto molto più marcato.

```
if(EffectState_2 == GPIO_PIN_SET){  // EFFETTO_2 TREMOLO

  //Viene sottratto il valor medio per riportare il segnale a media nulla
  // così da facilitare l'interazione del segnale con la funzione sen(f)
  xn = out - V_media ;

  d = potB * 0.0039; // VALORE MASSIMO potB = 255 ->  0<d<0.99

  // controllo della frequenza di tremolo calcolando l'incremento fra un campione e il successivo
  // min(T=1500ms, freq = 0.67Hz)
  // max(T=150ms, freq = 6.67Hz)

  delta = 0.0001+ potC * 0.039 * 0.0001;

  //calcolo dell'angolo in radianti a cui generare la sinusoide
  f = f + delta;
  if (f > 6.282){f = f-6.282;}

  s = sen(f);

  k = (xn *(d-1))+ (xn * (d) * s); //calcolo della funzione di uscita derivata dallo schema a blocchi

  out = k + V_media; // Aggiunta del valor medio tolto in precedenza

  //Finestratura per la riduzione del rumore
  if(out<(V_media+noise2) && out>(V_media-noise2)){ out = V_media;}

  //controllo soglie massime dac
  if(out>0xFFE00000) out = 0xFFE00000;
  if(out<0) out = 0;
}
```

Figura 2.17: Codice Effetto Tremolo

In Figura 2.17 si vede il codice che implementa il tremolo, in particolare si possono notare i valori limite dei parametri di controllo, una frequenza di tremolo compresa fra 0.67 e 6.67 Hz e un'intensità che va da 0 a 1 (zero: tremolo non presente, uno: componente originale non presente).

2.3.3 Chorus

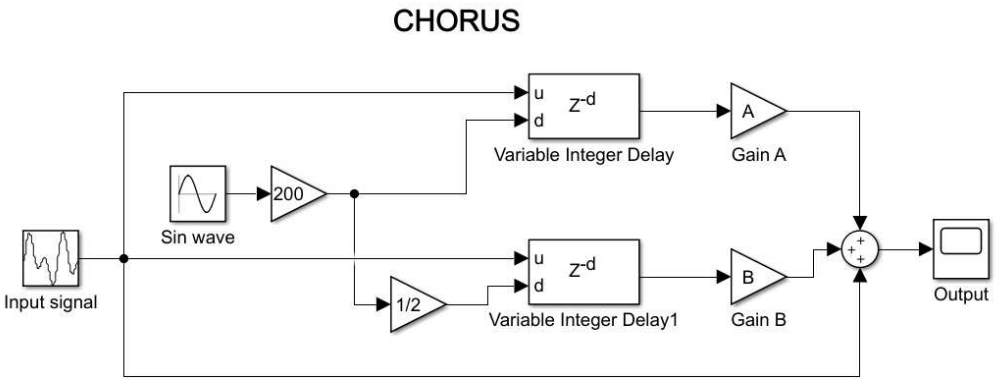

Figura 2.18: Schema a blocchi effetto Chorus

Funzione di trasferimento Chorus y = uscita, x[n] = campione corrente

$$y[n] = x[n] + (A \cdot x[n - (200 \cdot sen(f))]) + (B \cdot x[n - (100 \cdot sen(f))])$$

L'effetto Chorus (tradotto letteralmente coro) si ottiene quando suoni simili vengono riprodotti da sorgenti diverse, con una lieve differenza di timing, tono o volume. Il risultato è un suono più ricco che crea l'illusione di più strumenti che suonano contemporaneamente. Come mostrato in Figura 2.18 l'elemento cardine su cui si basa questo effetto è il blocco delay, responsabile dell'introduzione di un ritardo al segnale in ingresso ottenuto accedendo allo storico dei campioni contenuti nell'InputBuffer. In particolare il blocco delay ha due ingressi, il primo u è dedicato al segnale che si vuole ritardare, nel nostro caso il suono della chitarra, mentre il secondo d indica il numero del campione precedente a cui accedere [3]. Il parametro d dei due blocchi delay è scelto da programma attraverso le elaborazioni di una sinusoide generata dal codice stesso. Un'ulteriore accortezza è quella di limitare i valori del parametro d così da non eccedere il numero di campioni contenuti nel buffer di ingresso. L'effetto è formato da tre rami: uno principale, che porta in uscita il segnale di ingresso senza elaborazioni, e altri due che portano in uscita il segnale di ingresso elaborato dai blocchi delay. Questi due rami secondari contengono anche i parametri di controllo accessibili all'utente, ovvero i due guadagni GainA e GainB, controllati dai potenziometri D ed E, responsabili dell'intensità con cui vengono introdotte le componenti in ritardo.

[3] Ad esempio se d = 150, il campione in uscita dal blocco delay sarà il centocinquantesimo campione precedentemente salvato nell'Inputbuffer

```
if(EffectState_3 == GPIO_PIN_SET){  // EFFETTO_3 Chorus

  v = potD * 0.005; //v MAX = 1.275;
  u = potE * 0.005; //v MAX = 1.275;

  curr1 = curr;

  f3 = f3 + d3;
  if (f3 > 3.14){f3 = f3-3.14;}

  s3 = sen(f3) * 200;
  if(s3<0) s3 = -1 * s3;

  curr1 = curr1 - s3;

  if(curr1<0){curr1 = IN_BufferLength + curr1;}

  q1 = curr1;

  //calcolo della funzione di uscita derivata dallo schema a blocchi
  out = out + ( u * (InputBuffer[q1] - V_media))+ ( v * (InputBuffer[q1/2] - V_media));

  // il valor medio viene tolto dai due contributi secondari perche già presente nel segnalem "out"

  //Finestratura per la riduzione del rumore
  if(out<(V_media+noise3) && out>(V_media-noise3)){ out = V_media;}

  //controllo soglie massime dac
  if(out>0xFFE00000) out = 0xFFE00000;
  if(out<0) out = 0;

}
```

Figura 2.19: Codice Effetto Chorus

2.3.4 Generazione Sinusoide

Come descritto nella sezione precedente, gli effetti 2 e 3 necessitano di una funzione in grado di calcolare il seno di un numero. Essendo la sinusoide una funzione periodica il problema si può risolvere in due modi: il primo prevede l'utilizzo di una lookup table in cui vengono inseriti tutti i valori della funzione a distanza determinata; il secondo approccio prevede l'utilizzo dello sviluppo in serie di Taylor della funzione seno per calcolare il risultato richiesto. Come si vede in Figura 2.20 è stato scelto il secondo approccio con una serie di Taylor approssimata fino al quinto ordine. Motivo principale di questa scelta è il notevole risparmio di memoria ottenuto senza inserire una lookup table nel codice e la semplicità computazionale del calcolo.

La funzione può ricevere in ingresso valori in radianti compresi fra 0 e 2π, tuttavia, come si vede dalla catena di if presente nel codice, per rispettare le ipotesi dell'approssimazione con serie di Taylor [4], il calcolo viene effettuato con valori di x fra 0 e $\pi/2$ (1.571) per poi utilizzare delle trasformazioni lineari per arrivare al risultato finale così da ottenere un errore minore.

[4]L'approssimazione in serie di Taylor della funzione seno è valida solo per valori in ingresso vicini a 0

```
float sen(float t){ // Riceve in INPUT valori di t da 0 a 2pigreco in radianti

  float val = 0;
  float sign = 0;
  float x = 0;

  if(t<1.571){
    x = t;
    sign = 1;
  }else{
    if(t<3.141){
      x = 3.141-t;
      sign = 1;
    }else{
      if(t<4.712){
        x = t - 3.141;
        sign = -1;
      }else{
        x = 6.282-t;
        sign = -1;
      }
    }
  }
  float terzo = x*x*x*0.166667; // -> x^3/6
  float quinto = x*x*x*x*x*0.008333; // -> x^5/120

  val = x - terzo + quinto;

  return sign * val;
}
```

Figura 2.20: Porzione di codice in cui è implementata la funzione che genera una sinusoide a frequenza variabile utilizzata negli effetti 2 e 3

La Figura 2.21 mostra il confronto fra una sinusoide generata attraverso il codice (Verde) e una generata utilizzando il generatore di funzioni del PC (Gialla). Come si può vedere le due sinusoidi sono molto simili, si può quindi constatare che la funzione *sen* può essere utilizzata per gli effetti 2 e 3.

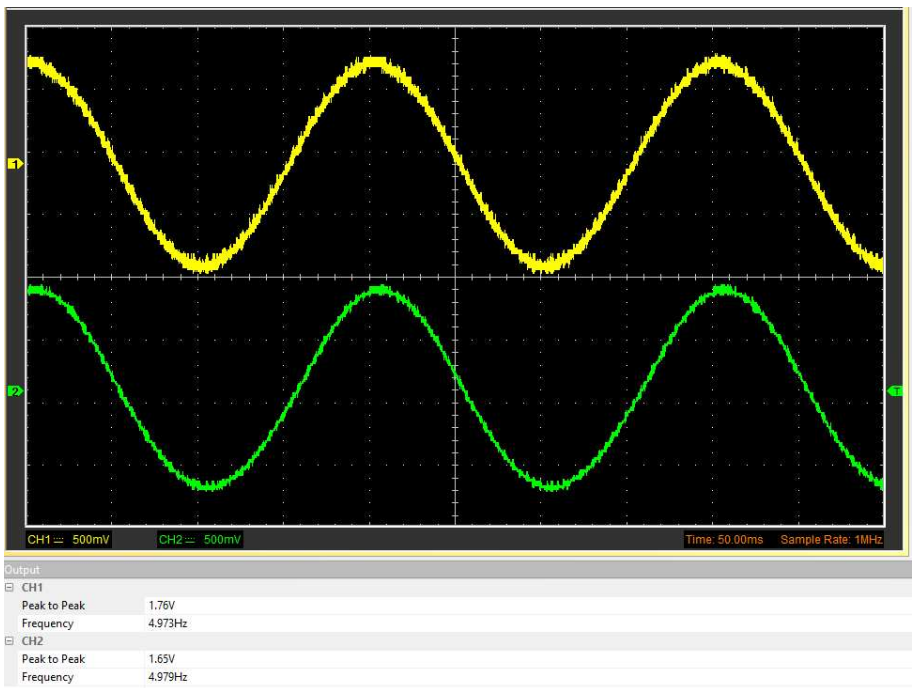

Figura 2.21: Sinusoide Generata dalla funzione sin del codice (Verde), Sinusoide di confronto generata dal generatore di funzioni (Gialla)

3 Test

3.1 Setup di misura

Per effettuare le misurazioni è stato allestito il classico setup generatore di funzioni in ingresso e oscilloscopio in uscita per verificare le forme d'onda

3.1.1 Oscilloscopio

Come oscilloscopio è stato utilizzato un Hantek 6022BE avente:

- 2 canali;
- Una Larghezza di banda di 20 MHz;
- Campionamento di 48MSample/s
- Alimentazione e interfaccia USB

Si tratta di un oscilloscopio per PC che può essere controllato sia nella maniera più tradizionale attraverso il software di casa Hantek sia utilizzando il software National Instruments LabView che ne permette la programmazione attraverso delle librerie dedicate fornite direttamente dal costruttore. L'utilizzo combinato di Oscilloscopio e Generatore di Funzioni del PC ha permesso di ottenere i diagrammi di Bode reali dei circuiti ad operazionali mostrati nel Capitolo 1, in modo automatico: senza cioè impostare manualmente tutte le frequenze e rilevare a mano le misure.

Figura 3.1: Hantek 6022BE, oscilloscopio USB per PC

3.1.2 Generatore di funzioni

Come generatore di funzioni è stata usata la scheda audio integrata a un PC, dato che la banda dei segnali gestita dal pedale combacia con la banda dei segnali di una comune scheda audio. La scheda audio in questione lavora con una frequenza di campionamento di 48kHz, che le permette di generare segnali fino a circa 23kHz, almeno venti volte maggiore delle frequenze raggiunte dalla chitarra. Il software che ha reso possibile questo tipo di utilizzo si chiama Visual Analyzer, la cui finestra principale è mostrata in Figura 3.2. Si tratta di un software opensource sviluppato da Alfredo Accattatis in collaborazione con L'università di Roma Tor Vergata che mette a disposizione un set di strumenti come un analizzatore di spettro, un oscilloscopio e un generatore di funzioni. La potenzialità di questa applicazione è che sfruttando solo la scheda audio del pc fornisce degli strumenti di misura, sicuramente non paragonabili a strumenti di laboratorio, ma sufficienti per avere un'idea qualitativa di cosa accade all'interno dei circuiti.

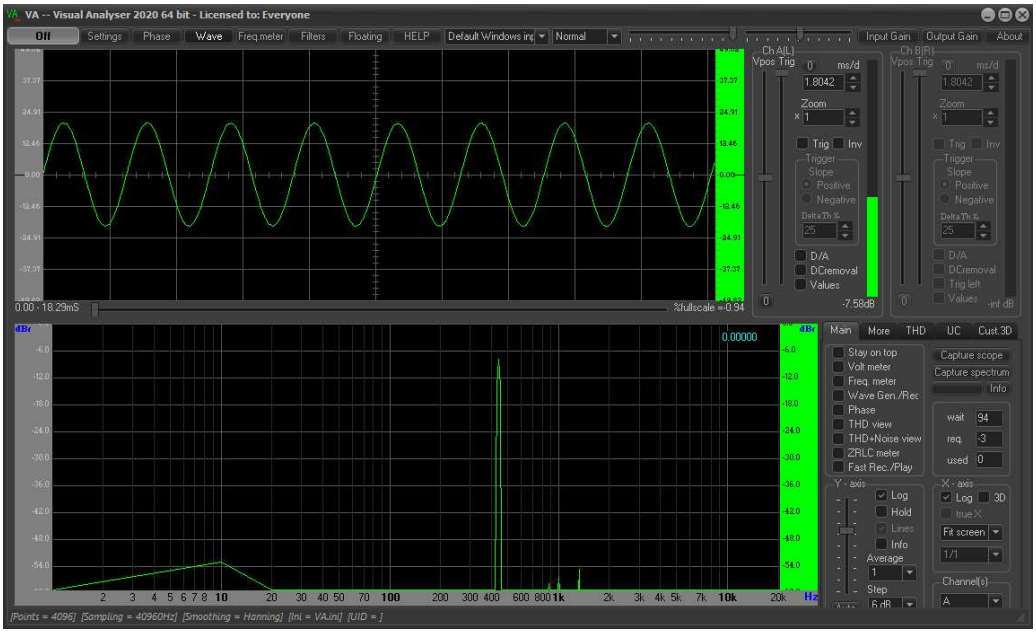

Figura 3.2: Schermata principale del software Visual Analyzer, in alto la parte in cui si visualizza il segnale di ingresso mentre in basso la finestra che ne mostra lo spettro in frequenza

Come segnale di test è stata utilizzata una sinusoide con ampiezza picco picco di 650 mV e frequenza di 440 Hz. Questa frequenza è stata scelta perché corrisponde alla nota di La, storicamente utilizzata per accordare la chitarra prima dell'avvento degli accordatori elettronici, oltre che essere circa al centro della banda passante.

3.2 Riproduzione onda originale

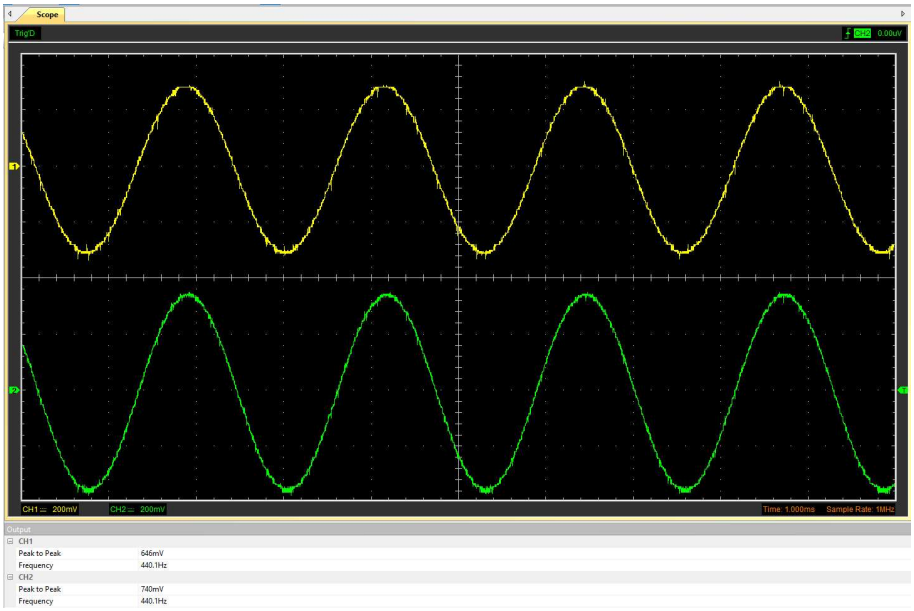

Figura 3.3: Nell'immagine viene mostrato il segnale di Test, in ingresso al pedale (CH1, Giallo) e la relativa uscita senza effetti (CH2, Verde)

In figura 3.3 è mostrato il segnale di test riprodotto dal pedale senza nessun effetto attivo, la forma d'onda in uscita riproduce fedelmente quella in ingresso a scapito di un piccolo guadagno, del tutto ininfluente agli scopi del progetto in quanto il segnale andrà comunque amplificato una volta uscito dal pedale.

3.3 Test effetti

3.3.1 Test effetto Distortion

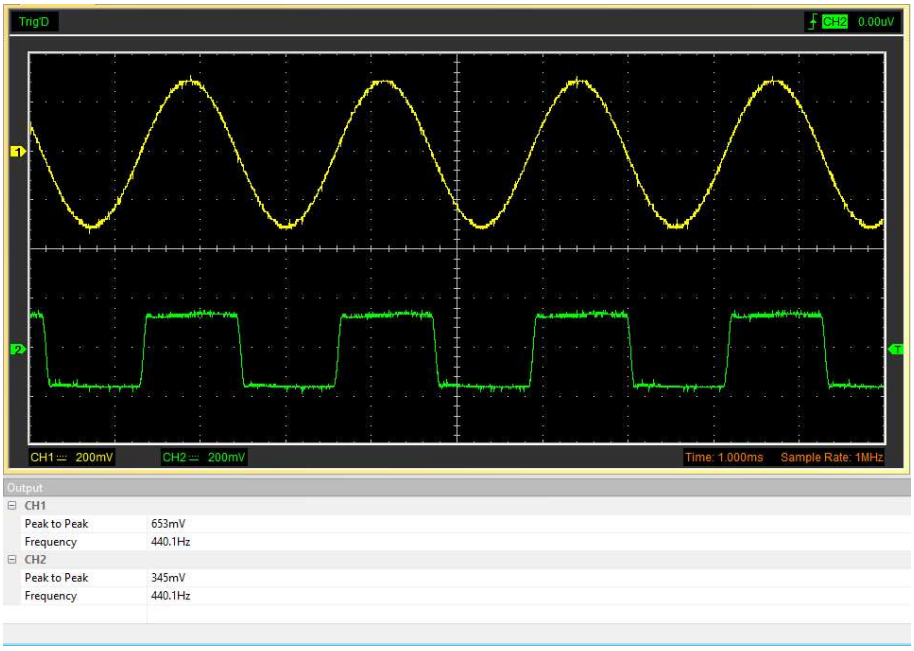

Figura 3.4: Segnale in ingresso del pedale (CH1, Giallo) e il relativo segnale in uscita (CH2, verde) con l'effetto 1 (Distortion) attivato

La Figura 3.4 mostra ingresso e uscita del pedale con l'effetto Distortion attivo, in particolare si può notare la forma d'onda verde (Uscita) ottenuta effettuando un clipping del segnale di ingresso (Giallo), infatti come si vede dalle misurazioni riportate in fondo alla figura la frequenza dei due segnali è rimasta invariata.

3.3.2 Test effetto Tremolo

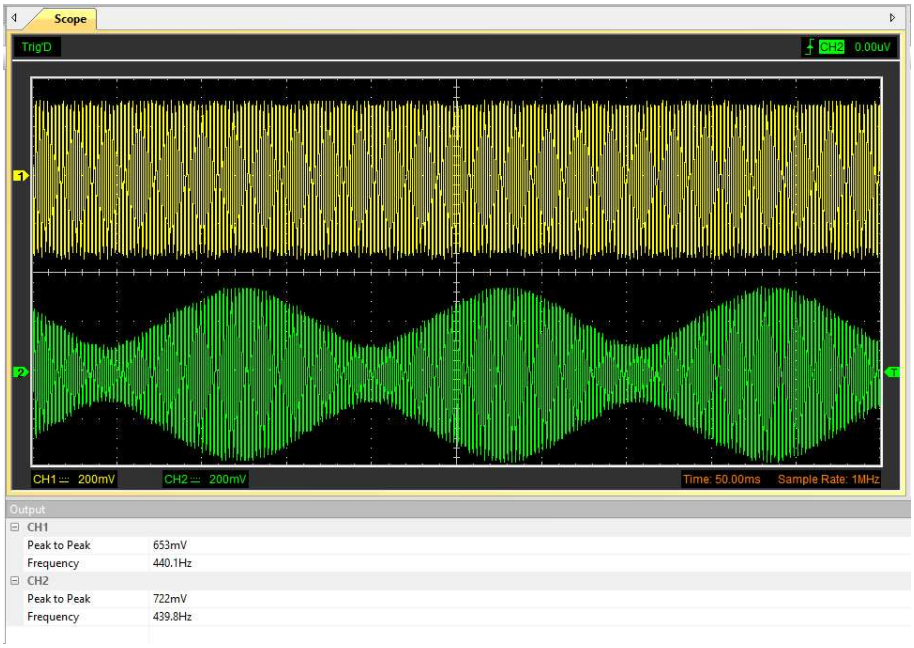

Figura 3.5: Segnale in ingresso del pedale (CH1, Giallo) e il relativo segnale in uscita (CH2, verde) con l'effetto 2 (tremolo) attivato

La Figura 3.5 mostra ingresso e uscita del pedale con l'effetto Tremolo attivo. Nel segnale di uscita si può vedere un andamento dell'ampiezza sinusoidale come la forma d'onda utilizzata nel codice per la modulazione del segnale. Come nell'effetto precedente dalle misurazioni si vede come la frequenza dei due segnali rimanga pressoché invariata.

3.3.3 Test effetto Chorus

Il test dell'effetto Chorus con una sinusoide genera in uscita una somma di sinusoidi a frequenza differente rendendo la forma d'onda difficilmente distinguibile, per questo in Figura 3.7 e 3.8 sono mostrati gli spettri in frequenza del segnale di ingresso e di uscita. Come si può vedere lo spettro dell'uscita è molto più ricco di componenti rispetto a quello di ingresso che ne presenta soltanto una. Questo indica un suono più profondo che è il risultato desiderato, tuttavia una volta collegata la chitarra al pedale e il pedale all'amplificatore questo effetto risulta molto più affetto da rumore di fondo rispetto agli altri due.

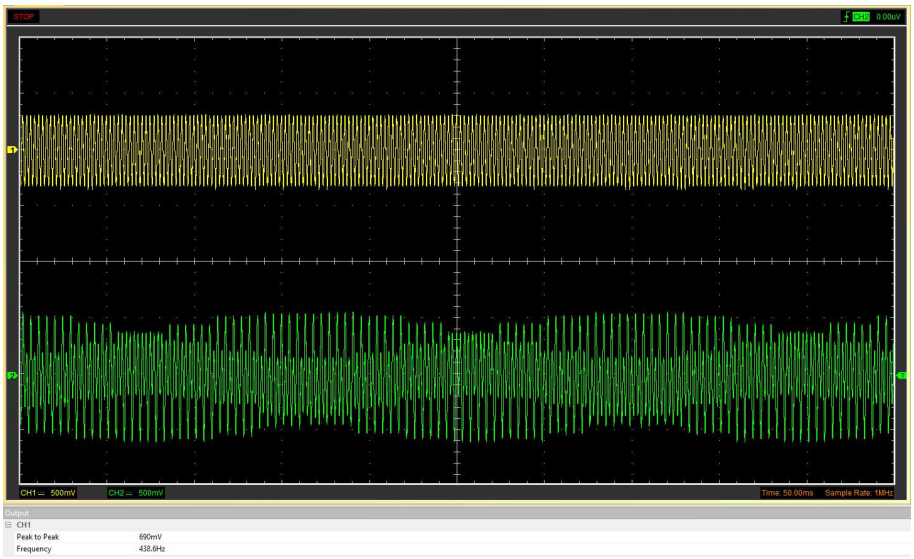

Figura 3.6: Segnale in ingresso del pedale (CH1, Giallo) e il relativo segnale in uscita (CH2, verde) con l'effetto 3 (chorus) attivato

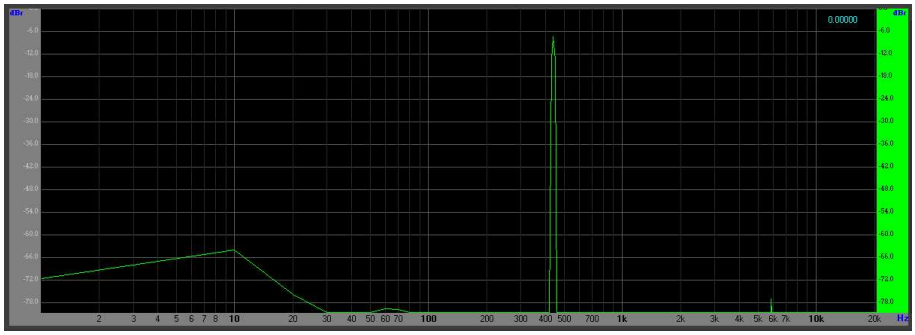

Figura 3.7: Spettro in frequenza del segnale in ingresso al pedale

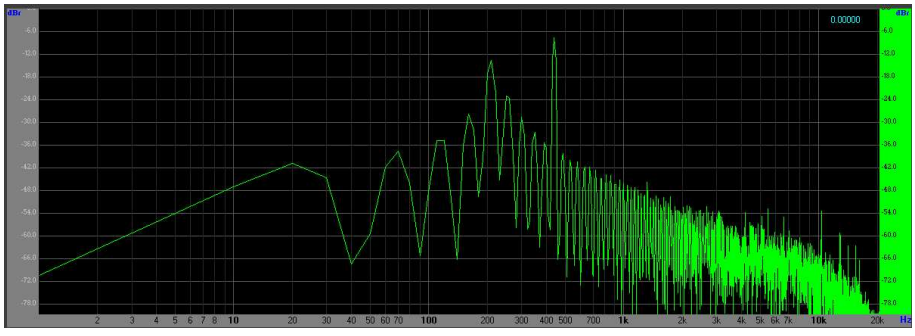

Figura 3.8: Spettro in frequenza del segnale in uscita al pedale con effetto 3 (Chorus) attivato

Conclusione

Nella realizzazione di questo progetto sono sicuramente migliorate le competenze di configurazione del microcontrollore STM32F334R8 sia utilizzando il software STM32Cube MX sia analizzando il listato prodotto da tale codice per effettuare modifiche in maniera meno invasiva. Inoltre la realizzazione completa del pedale, dalla progettazione dei circuiti al montaggio della PCB, ha reso possibile l'incremento delle abilità di problem solving e di visione d'insieme sia del progetto sia del processo realizzativo.

Il risultato ottenuto, come si può vedere dai test descritti nel capitolo 3, è complessivamente soddisfacente; tuttavia, nell'utilizzo pratico, mentre si suona lo strumento sono presenti dei lievi disturbi al segnale specie se si utilizza ad un volume elevato. Un modo per limitare questi effetti potrebbe essere quello di utilizzare un PCB multilayer inserendo dei piani di massa più estesi.

L'obiettivo iniziale di muovere i primi passi nel mondo del DSP è stato raggiunto realizzando tre effetti storici dei pedali per chitarra, inoltre è stata fondamentale l'attività di studio dei reference manual della scheda Nucleo, per trovare il modo migliore di configurare le periferiche e per conoscere la posizione dei pin della scheda Nucleo in modo da scegliere (dove possibile) il posizionamento più comodo per la PCB.

Bibliografia

[1] Alfredo Accattatis. *Visual Analyzer software*. URL: https://www.sillanumsoft.org/prod01.htm.

[2] Simone Buso. *Esercitazioni con il microcontrollore STM32F334R8 per il corso di Microcontrollori e DSP*. 2018.

[3] Simone Buso. *Introduzione alle applicazioni industriali di microcontrollori e DSP*. Esculapio, 2018. ISBN: 9788874889358.

[4] HANTEK. *Hantek 6022BE documentation*. URL: https://www.hantek.it/oscilloscopi-usb-per-pc/163-hantek-6022be-oscilloscopio-usb-2-canali-20mhz.html.

[5] Texas Instruments. *Filter design tool*. URL: https://webench.ti.com/filter-design-tool/.

[6] Texas Instruments. *TL082 Datasheet*. URL: https://www.ti.com/lit/ds/symlink/tl082.pdf?ts=1688993016865&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTL082%253Futm_source%253Dsupplyframe%2526utm_medium%253DSEP%2526utm_campaign%253Dnot_alldatasheet%2526DCM%253Dyes%2526dclid%253DCNeUuv3hg4ADFXVU5QodAp0CLQ.

[7] Matlab-corp. *Delay-Based Audio Effects*. URL: https://it.mathworks.com/help/audio/ug/delay-based-audio-effects.html. (Consultato Marzo 2023).

[8] ST microelectronics. *Documentazione Microcontrollore STM32F334R8*. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f334r8.html#documentation.

[9] ST microelectronics. *Documentazione scheda Nucleo F334R8*. URL: https://www.st.com/en/evaluation-tools/nucleo-f334r8.html#documentation.

[10] XLSEMI. *Datasheet XL4005*. URL: https://www.laskakit.cz/user/related_files/xl4005_datasheet.pdf.