UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

DISSENY D'UN EQUALITZADOR DE SISTEMES DE P.A. (PUBLIC ADRESS) AUTOMÀTIC

Grau: Grau en Enginyeria de Sistemes Audiovisuals

Data d'entrega: 10 de gener de 2019

Estudiant: Cristian Torrente Guerrero

Director: Néstor Berbel Artal

AGRAÏMENTS

Als meus pares, a la meva germana i a la meva parella per tenir tanta paciència amb mi i per donar-me suport en els moments que més m'ha costat tirar endavant aquest projecte i en tots els projectes de la vida.

Agrair sobretot al meu director de projecte Néstor Berbel, perquè sense ell aquest projecte no hagués sigut possible, ha sigut una peça clau d'aquest projecte. Gràcies per tenir tanta paciència amb mi, per buscar qualsevol espai de temps en els meus horaris ajustats i sobretot gràcies per tot el que he après amb tu.

INDEX

INDEX DE FIGURES

INDEX DE TAULES.

INDEX D'EQUACIONS.

RESUM

En aquest projecte es vol dissenyar un equalitzador de Public Adress (P.A.) amb la particularitat de què sigui totalment automatitzat. Per fer una petita introducció, un equalitzador de Public Adress (P.A.) serveix per poder equalitzar tots els tipus de sistemes d'altaveus. Aquest procés es fa, ja que depenent del lloc on col·loquis els altaveus, poden tenir un rendiment que no seria l'esperat, podent provocar un augment d'un o diversos rangs freqüencials que no és el desitjat.

Em vaig decidir a fer aquest projecte a l'haver treballat com a tècnic de so i veure que el sistema per equalitzar el sistema de P.A. és un procés molt lent i que et treu temps per poder fer la mescla del directe que realment és l'important.

Per la realització d'aquest projecte farem servir la placa de desenvolupament Teensy 3.6 que es programa amb l'entorn Arduino. En aquesta placa de desenvolupament és on realitzarem la lectura de dades de les entrades de línia i de l'entrada de micròfon amb alimentació phantom (+48V), la transformada de Fourier per poder obtenir els valors de freqüències (20 – 20000 Hz) i les amplituds (dB), l'equalització de les bandes de freqüències i l'enviament del senyal equalitzat cap a les sortides de línia. Per la visualització de les dades s'utilitzarà una pantalla LCD de 4.3", on es veurà un petit analitzador d'espectres amb 7 bandes de freqüències.

Per la part de hardware s'ha hagut de dissenyar dues entrades de línia per poder adaptar el voltatge de 1,27V fins a 3,3V, una entrada per micròfon amb alimentació phantom (+48V) i dues sortides de línia per adaptar el voltatge de 3,3V fins a 1,27V. Per poder portar a terme tot això s'ha hagut de dissenyar una placa de circuit imprès (PCB) on també anirà connectat la placa de desenvolupament Teensy 3.6 i la pantalla LCD.

ABSTRACT

In this project, we want to design a Public Adress equalizer with the particularity of being fully automated. To make a small introduction, a Public Adress equalizer is used to equalize all types of speaker systems. This process is done because, depending on the place where the speakers are placed, they may have a non-expected performance, and may cause an increase in one or more frequency ranges that would not be desired.

I decided to do this project after working as a sound technician and seeing that the system to equalize the P.A. system becomes a very slow process and it takes time to make the mix of the live performance, which is what really matters.

For the realization of this project we will use the Teensy 3.6 development board that is programmed with the Arduino environment. In this development board we will read the data of the line inputs and the microphone input with phantom power (+ 48V), the Fourier transform to be able to obtain the values of frequencies (20 - 20000 Hz) and the amplitudes (dB), the equalization of the frequency bands and the transmission of the equalized signal to the line outputs. For the visualization of the data a 4.3 "LCD screen will be used, in which we will see a small spectrum analyzer with 7 bands of frequencies.

For the hardware part, two line inputs to be able to adapt the voltage from 1.27V to 3.3V, a microphone input with phantom power (+ 48V) and two line outputs to adapt the voltage from 3,3V to 1,27V have been designed. In order to carry this out, a printed circuit board (PCB) has been designed, where the Teensy 3.6 development board and the LCD screen will also be connected.

1. INTRODUCCIÓ

Avui en dia el món de l'àudio professional hi ha moltes maneres per poder equalitzar un sistema de "Public Adress", ja siguin equalitzadors gràfics connectats en sèrie a la taula de mescles, sistemes de configuració d'altaveus digitals (crossovers), ...

El procés d'equalització tradicional es basa en llençar un soroll rosa pels altaveus, captar la resposta de la sala amb un micròfon de mesura connectat a un analitzador d'espectre, sigui en format rack o una interfície de so connectada a un ordinador amb algun software que et permeti visualitzar l'espectre freqüencial ("Smaart Live") i un equalitzador gràfic de X bandes de freqüències per poder amplificar o atenuar la banda que presenta alguna anomalia per la resposta que té la nostra sala.

S'utilitza el soroll rosa perquè és el soroll que està caracteritzat per una densitat espectral inversament proporcional a la freqüència. Quan aquest soroll es visualitza en un analitzador amb filtres d'octava, es veu que totes les bandes d'octava tenen el mateix nivell sonor. Això passa perquè els filtres d'octava, terç d'octava... són filtres proporcionals, i per tant, cada cop que baixem una octava, dupliquem l'amplada de banda i per aquest motiu el soroll rosa decreix 3 dB per octava, just la proporció en què augmenta l'amplada de banda, el doble. D'aquesta manera visualitzem el soroll rosa com un soroll de nivell constant en totes les bandes d'octava.

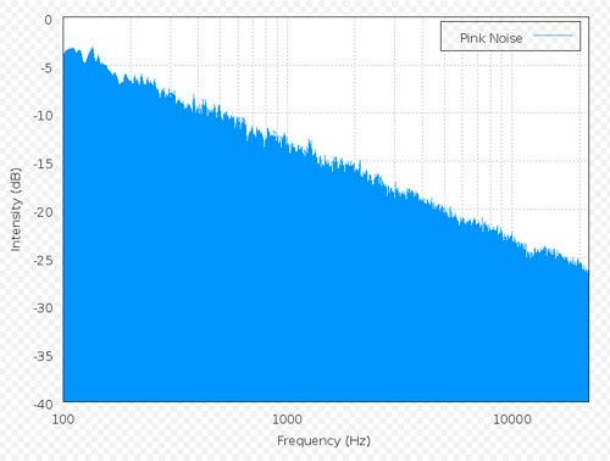


Figura 1 – Espectre freqüencial del soroll rosa

El micròfon de mesura és un micròfon omnidireccional i de resposta en freqüència plana, aquest tipus de micròfons són de condensador i necessiten una alimentació suplementaria de 48 V, també coneguda com a alimentació phantom.

Figura 2 – Micròfon de mesura

Un analitzador d'espectres és un equip de mesura electrònica que permet visualitzar en una pantalla els components espectrals en un espectre de freqüències dels senyals en l'entrada. En altres paraules ens permet veure la freqüència i la mida d'una ona electromagnètica. En el nostre cas ens interessa visualitzar la freqüència i l'amplitud del soroll rosa captat pel micròfon de mesura. Com he esmentat abans també existeix software que ens fan d'analitzador d'espectres com el "Smaart Live".

Figura 3 – Analitzador d'espectres

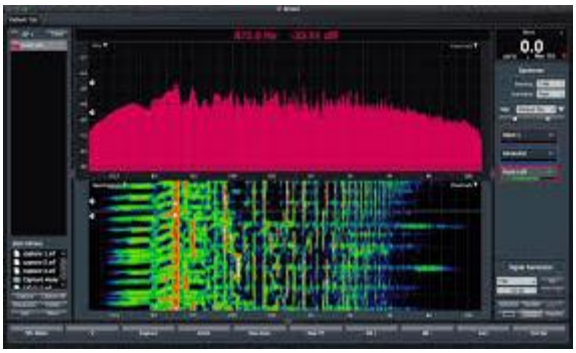


Figura 4 – Smaart Live.

L'equalitzador gràfic és un dispositiu que processa senyals d'àudio i ens permet dividir aquest senyal en diferents bandes de freqüència, podent alterar el guany de cada banda de forma independent. Es compon de filtres passa banda.

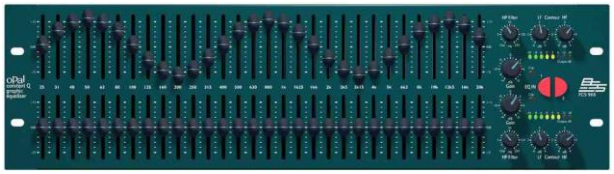


Figura 5 – Equalitzador gràfic

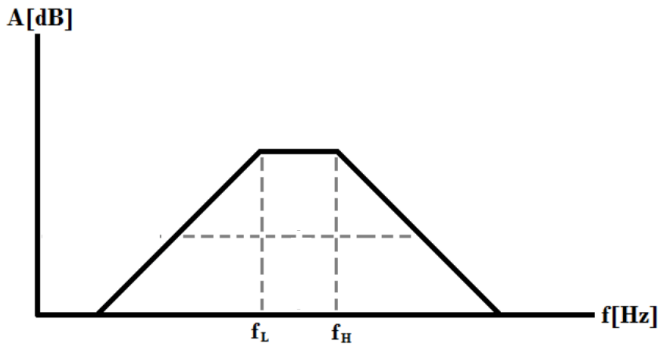


Figura 6 – Filtre passa banda

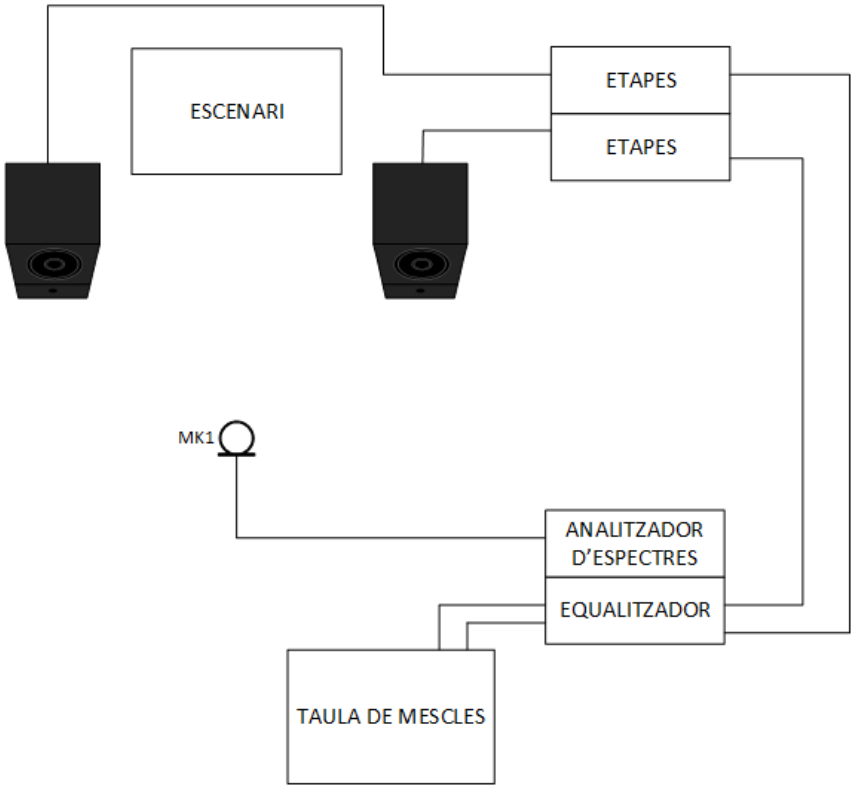


Figura 7 – Sistema d'equalització de P.A. tradicional

L'objectiu principal d'aquest projecte és implementar un equalitzador de "Public Adress" que faci el procés d'equalització totalment automàtic. Per poder fer això dividirem el projecte en dos grans parts diferencials:

- La part de hardware
- La part de software

La part de hardware estarà bàsicament constituïda per les etapes d'entrada (micròfon i línia), les etapes de sortida, la pantalla on es mostrarà la informació a l'usuari i el microprocessador central que farà el control de les etapes d'entrada, de sortida i de la pantalla.

2. DISSENY DEL HARDWARE

En aquest apartat s'explica les diferents parts del disseny del hardware per poder fer possible l'equalitzador automàtic. S'analitzaran els esquemes de cada etapa, l'alimentació necessària i el hardware necessari per poder connectar la pantalla LCD.

Les especificacions a complir pel disseny del hardware es mostren a la següent taula.

Característica	**Valor**
Tensió de línia	1,736 V
Tensió entrada/sortida Teensy	3,3 V
Alimentació	±5 V
Alimentació phantom	48 V
Alimentació LCD	3,3 V

Taula 1 – Especificacions hardware

El disseny del hardware d'aquest projecte es divideix en:

- Etapa d'entrada de línia
- Etapa d'entrada del micròfon
- Etapa de sortida de línia
- Pantalla LCD

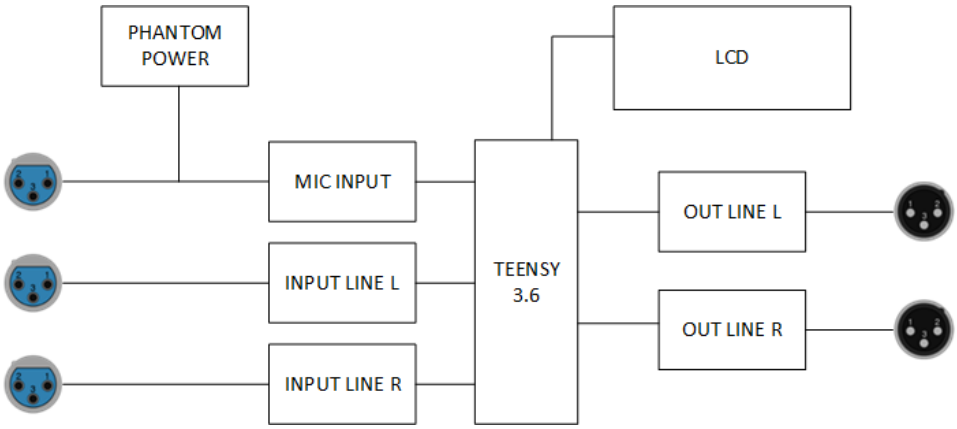


Figura 8 – Esquema del hardware

2.1. ETAPA D'ENTRADA DE LÍNIA

En aquesta part explicarem el funcionament de les dues entrades de línia que tenim al nostre sistema. En ser un sistema estèreo necessitarem dos canals de línia, l'esquerre i el dret.

Hem optat per posar línies balancejades a les entrades de línia de l'equalitzador. Una línia balancejada és aquella que porta dos senyals, una que porta el senyal en fase, anomenada viu, i l'altre desfasada 180 graus, anomenada contrafase. Aquests dos conductors van recoberts d'una malla connectada a terra. Les línies balancejades són utilitzades per tenir menys interferències electromagnètiques. Per aquest motiu son molt utilitzades a l'hora de fer tirades llargues de cables i sobretot per connectar micròfons.

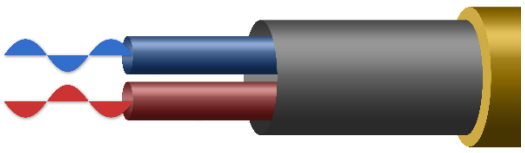


Figura 9 – Línia balancejada

El connector més comú utilitzat per fer les línies balancejades és el connector XLR o també conegut com a *Canon*. Del connector XLR (Figura 10 i Figura 11) surten tres conductors: positiu, negatiu i massa. Aquests aniran connectats de la següent manera:

- Pin 1: Massa
- Pin 2: Positiu
- Pin 3: Negatiu

Figura 10 - XLR

Figura 11 - XLR

El senyal de línia en l'àmbit professional té un nivell de +4 dBU, que si ho passem a voltatge lineal, obtenim un voltatge de pic de 1,736 V. Aquest serà el voltatge que obtindrem a l'entrada de línia.

Un cop ja tenim els senyals balancejats, les haurem de passar a una sola senyal, ja que utilitzarem un conversor analògic-digital ("Analog to Digital Converter" o ADC) per llegir cada un dels nostres canals d'entrada de línia. Per tant aquí arribem a la primera part important de l'etapa d'entrada de línia.

2. 1. 1. AMPLIFICADOR D'INSTRUMENTACIÓ INA217

Per convertir el senyal diferencial que es rep del connector XLR a un senyal "single ended" o senyal referencial a massa, hem optat per utilitzar un amplificador d'instrumentació. L'amplificador diferencial rep el senyal diferencial i el converteix a un senyal "single ended", i a més, pot aplicar un cert guany, controlat mitjançant un potenciòmetre. L'amplificador escollit és el *INA217* [1] , el qual és un amplificador d'instrumentació de baix soroll i de baixa distorsió, són coses que interessen molt en el món de l'àudio.

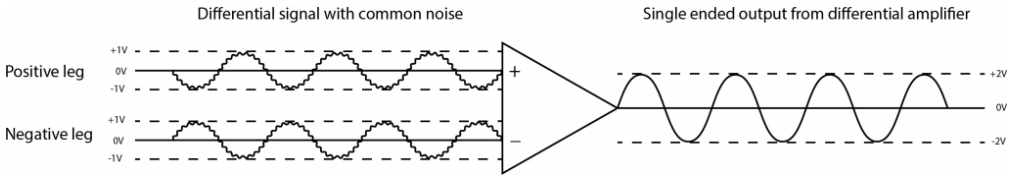


Figura 12 – Senyal diferencial a Single Ended

En la següent imatge podrem veure un esquema simplificat de l'amplificador d'instrumentació:

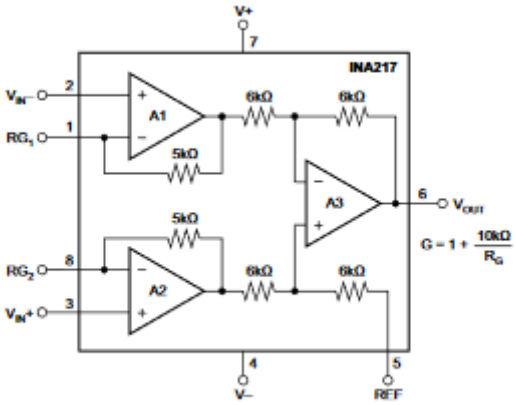


Figura 13 – INA217

Com podem veure a la imatge anterior, el INA217 esta compost per dos circuits amb amplificadors no inversors i amb un amplificador restador.

L'amplificador no inversor, com el seu nom ben ve indica, serveix per no modificar la polaritat del senyal d'entrada a l'amplificador. En aquest cas aquests dos circuits estan pensats per poder aplicar el guany de la resistència variable que podem posar als pins 1 i 8 del INA217.

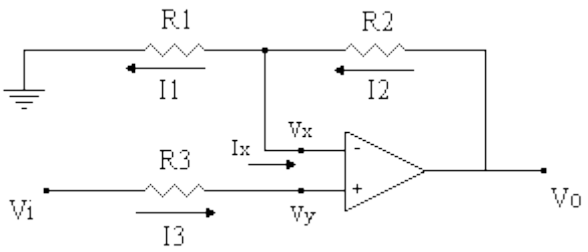


Figura 14 – Amplificador no inversor

$$\frac{Vo}{Vi} = \frac{R2}{R1} + 1$$

Equació 1 – Guany amplificador no inversor

En la formula anterior podem veure com calcular el guany d'un amplificador no inversor, i com podem observar aquest guany mai serà menor que 1.

Un cop obtingut els dos senyals sense canviar la polaritat i amb el guany desitjat passarem a l'altre amplificador operacional, que ens farà de restador, per poder obtenir un sol senyal i en fase.

Aquest amplificador utilitza ambdues entrades invertides amb un guany de 1, per produir una sortida igual a la diferència entre les entrades.

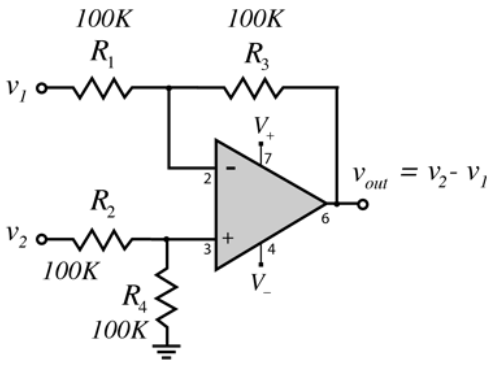


Figura 15 – Amplificador restador

Per tant a la sortida del INA217 tindrem un sol senyal amb la resta dels dos senyals d'entrada que venien del XLR.

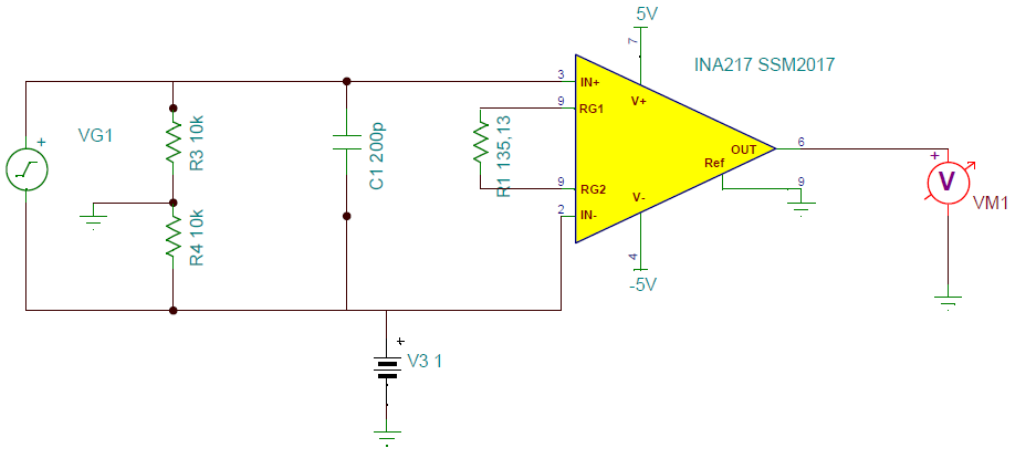


Figura 16 – Esquema etapa INA217

Aquest és l'esquema amb el qual vam realitzar la simulació del INA217, aplicant a l'entrada un senyal sinusoidal de 10 mV d'amplitud, a la freqüència de 1 kHz i aquest és el resultat obtingut. L'amplificador d'instrumentació haurà d'estar alimentat amb una tensió de ±5V.

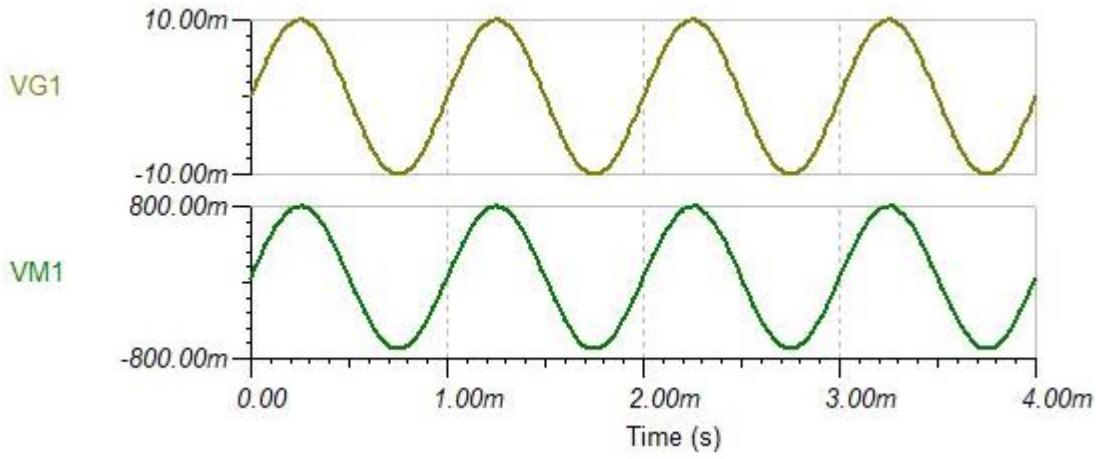


Figura 17 – Simulació INA217

Com podem veure en la simulació feta amb el software Tina-Ti, tenim un senyal d'entrada de 10 mV d'amplitud i a la sortida aconseguim un senyal de 800 mV d'amplitud, per tant estem tenint una amplificació per 8. Per tant ja tenim el disseny del preamplificador.

Ara el següent pas serà adaptar els voltatges per a la nostra placa de desenvolupament Teensy 3.6, que té una tensió màxima d'entrada de 3,3V.

2. 1. 2. ETAPA DE CONDICIONAMENT.

En aquesta etapa el que volem aconseguir és adaptar la sortida de l'amplificador d'instrumentació al nivell de tensions que accepta el conversor analògic-digital del microprocessador. Tal com s'indica més endavant, el microprocessador utilitzat es el Teensy 3.6, que té un ARM Cortex M3. El conversor analògic-digital accepta voltatges de 0 V a 3,3 V, mentre que la sortida de l'amplificador d'instrumentació pot variar de -5 V a +5 V, corresponent a l'alimentació del dispositiu.

Aquesta etapa de condicionament haurà de sumar un senyal continu al senyal altern provinent de l'amplificador d'instrumentació, perquè el conversor analògic-digital vegi un senyal de tensió entre 0 V i 3,3 V.

Per fer aquesta etapa s'implementa un circuit sumador. Primer de tot es va optar per utilitzar 2 TL971, però a l'hora de fer el llistat de components vam veure que hi havia el TL972 [2] que són 2 TL971 en el mateix encapsulat. D'aquesta manera només havíem d'alimentar un sol amplificador operacional.

El disseny que es va pensar és el següent:

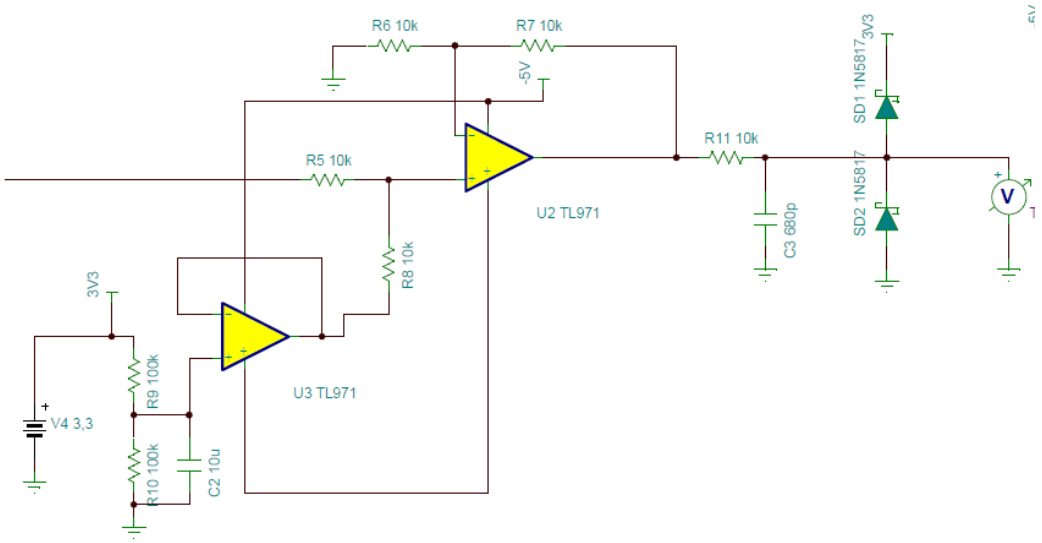


Figura 18 – Etapa d'acondicionament

En la figura anterior podem veure dos amplificadors operacionals, un amb un amplificador seguidor i l'altre amb un amplificador sumador.

Com podem veure abans de l'amplificador seguidor tenim un divisor de tensió, aquest divisor de tensió està dissenyat perquè ens passi d'un senyal de 3,3 V a un senyal de 1,65 V, volem aquest senyal per tenir una tensió continua a l'entrada del TL972. Per aquest motiu es fa un amplificador seguidor, per tenir el mateix valor de voltatge a l'entrada que a la sortida.

L'altre circuit que tenim és un circuit sumador. Aquest circuit sumador ens adaptarà el senyal entre 0 V i els 3,3 V

Després de l'amplificador operacional, trobem un filtre per treure la continua i quedar-nos amb el senyal que ens interessa, el senyal d'àudio. Per tant la freqüència de tall del filtre haurà de ser:

$$f_t = \frac{1}{2\pi RC}$$

Equació 2 – Freqüència de tall del filtre

També posarem dos díodes per protegir la nostra placa de desenvolupament, ja que si tenim més de 3,3 V a l'entrada, podem tenir problemes.

L'amplificador operacional TL972 també estarà alimentat a ±5V per utilitzar la mateixa alimentació a tots els circuits.

La simulació d'aquest circuit és la següent:

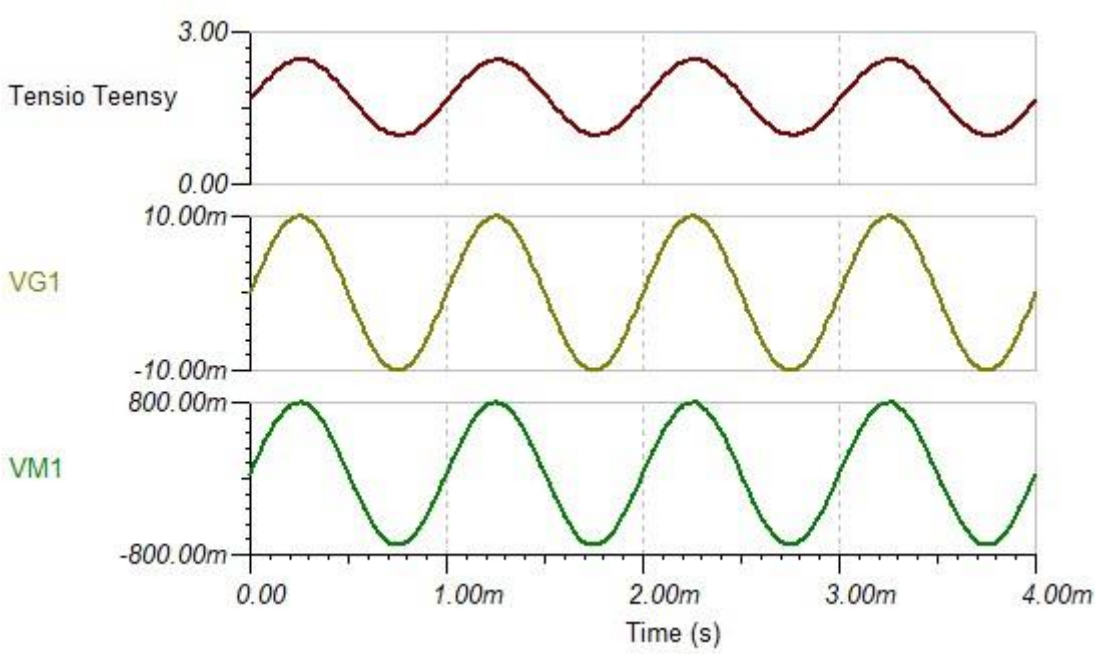


Figura 19 – Simulació etapa d'acondicionament

Com podem veure obtenim un senyal a la sortida (Tensió Teensy) de 3,3 V de pic a pic, per tant les nostres dues etapes estan funcionant correctament.

Un cop tenim aquest senyal de 3,3 V ja el podem enviar cap a un dels ADC's del nostre microprocessador.

Per tant el disseny final de l'etapa d'entrada de línia és el següent:

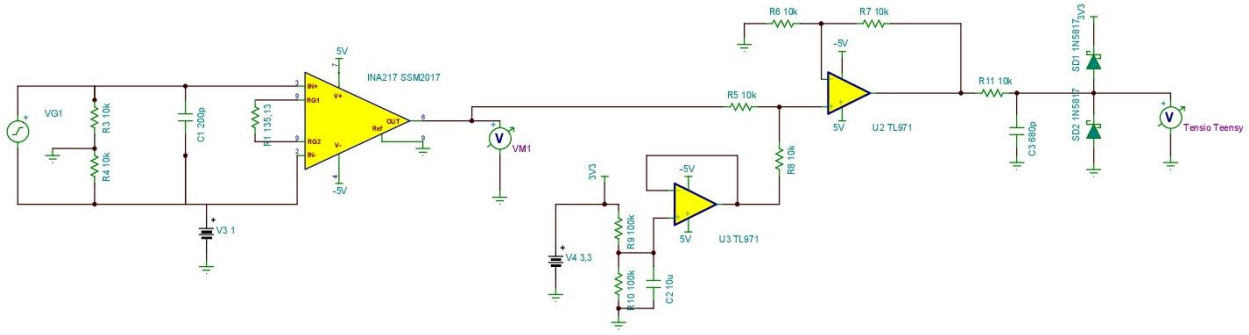


Figura 20 – Disseny final etapa d'acondicionament

2.2. ETAPA D'ENTRADA DEL MICRÒFON

En aquest apartat explicarem el disseny de l'etapa d'entrada del micròfon, una de les principals característiques d'aquesta etapa és que necessitem una alimentació extra per poder alimentar el micròfon de mesura, ja que aquests tipus de micròfons són de condensador i necessiten una alimentació de 48 V, com ja s'ha explicat a l'apartat 1.

L'alimentació phantom o alimentació fantasma, és una forma de proporcionar alimentació (corrent continua) als dispositius d'àudio que ho necessiten. L'estàndard defineix 3 tipus d'alimentació, 48 V, 24 V i 9 V. En el nostre cas farem servir la de 48 V i per tant haurem d'utilitzar resistències de 6,8 kΩ per poder distribuir el voltatge esmentat abans. En el cable que s'utilitza per aquest tipus d'alimentació, el senyal d'àudio viatge pels dos fils interns i una malla exterior que s'utilitza per protegir de la interferència electromagnètica (cable balancejat), explicat en l'apartat 2.1. La corrent continua, s'aplica entre ambdós fils interiors (positiu i negatiu) i la malla exterior, de manera que en absència de senyal d'àudio, els dos fils estiguin al mateix potencial respecte a la malla i per tant no existeixi diferencia entre ells.

En aquest cas, per l'entrada del micròfon també utilitzarem els connectors XLR, ja que en els micròfons és el cable més utilitzat i ens permetrà enviar l'alimentació phantom cap al micròfon de mesura.

Per fer aquesta etapa d'entrada del micròfon seguirem el mateix disseny que l'etapa d'entrada de línia però afegint l'etapa d'alimentació phantom necessària pel micròfon de mesura.

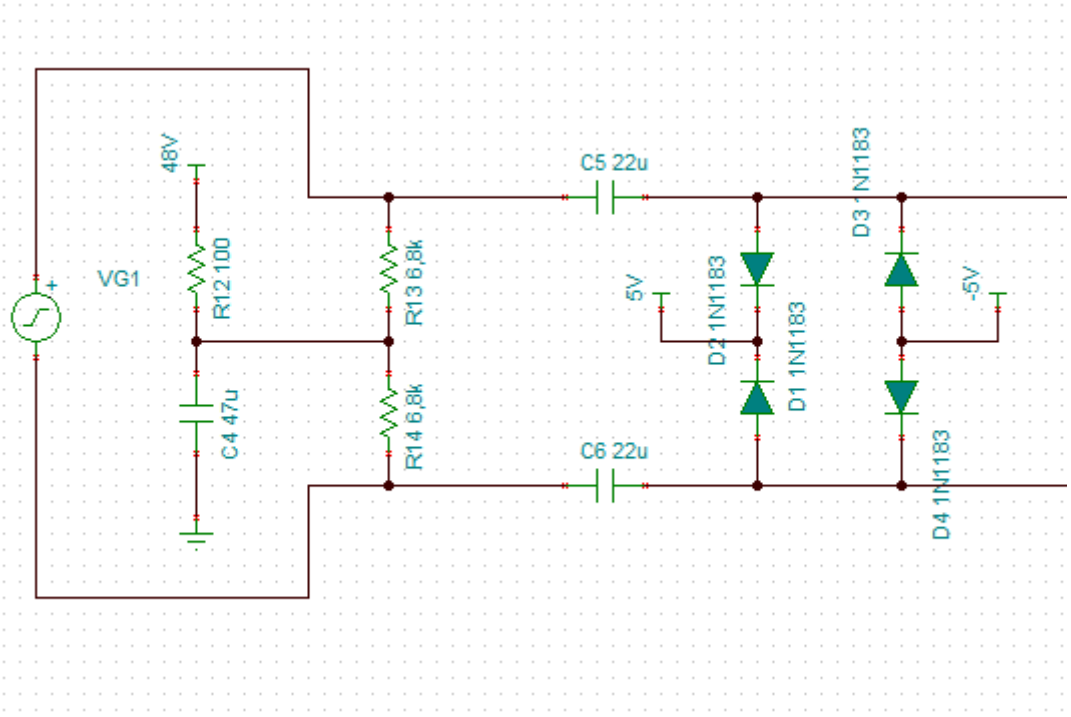


Figura 21 – Etapa d'alimentació phantom

Aquesta és l'etapa d'alimentació phantom dissenyada, com podem veure utilitzem les resistències esmentades anteriorment de 6,8 kΩ per poder subministrar els 48 V cap al micròfon. També s'han inclòs uns díodes per poder protegir l'etapa següent.

Un cop passat els díodes ja tornem a l'etapa d'entrada de línia esmentada a l'apartat 2.1. Per tant el disseny final de l'etapa d'entrada del micròfon és el següent:

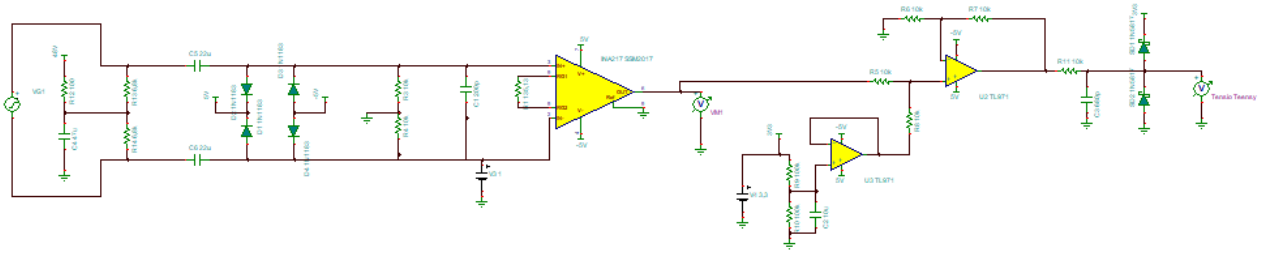


Figura 22 – Etapa d'entrada de micròfon

2.3. ETAPA DE SORTIDA DE LÍNIA

En aquest apartat explicarem el disseny i funcionament de l'etapa de sortida de línia dissenyada per aquest projecte. Per fer-ho s'ha utilitzat el mateix amplificador operacional que a les entrades, el TL972, però en aquest cas per una finalitat diferent que a les entrades de línia. En l'esquema realitzat per veure la simulació fem servir el TL971 però en el disseny final s'ha utilitzat el TL972 com ja s'ha explicat anteriorment.

2. 3. 1. DISSENY DEL FILTRE PASSA BANDA

Un cop rebem el senyal del DAC (Digital to Analog Convert) del Teensy, primer de tot haurem de filtrar la tensió continua del senyal modulat, ja que si no la filtrem, obtindrem un senyal amb petits salts que ens modificarà el senyal d'àudio. Això és degut a passar el senyal de digital a analògic, en teoria la sortida del DAC serà 0 volts quan a l'entrada binaria siguin tots zeros, però a la pràctica hi haurà un voltatge de sortida petit produït per l'error de balanç de l'amplificador del DAC. Per tant haurem de filtrar aquest petit voltatge. Per fer-ho s'ha dissenyat el següent filtre:

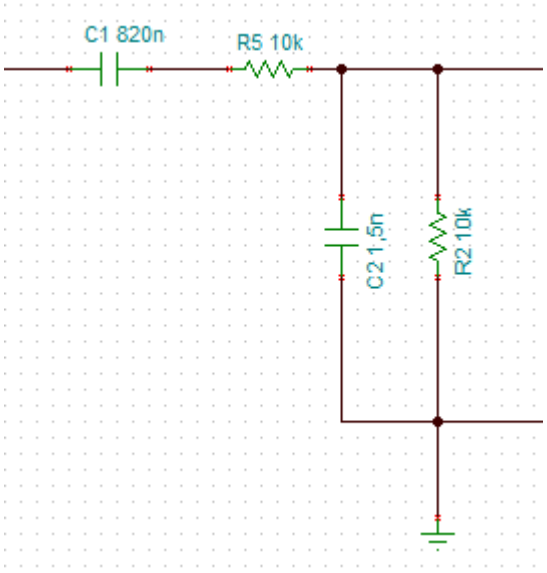


Figura 23 – Filtre a la sortida del DAC

Aquest filtre és del tipus passa banda, per tant s'han hagut d'escollir dues freqüències de tall per poder escollir els components per fer els filtres. Com que l'espectre d'àudio audible va des dels 20 Hz fins als 20 kHz, la primera freqüència de tall serà de 10 Hz i la segona serà de 24 kHz.

Per el càlcul de C1, utilitzarem la següent formula:

$$f_L = \frac{1}{2\pi C_1 (R + R)}$$

$$C_1 = \frac{1}{2\pi f_L (R + R)}$$

Equació 3 – Càlcul de C1

Amb una freqüència de 10 Hz i amb una resistència de 10 kΩ, obtindrem un valor de C1 de 820 nF.

Per el càlcul de C2, utilitzarem la següent formula:

$$f_H = \frac{1}{2\pi C_2 \left(\frac{R}{2}\right)}$$

$$C_2 = \frac{1}{2\pi f_H (\frac{R}{2})}$$

Equació 4 – Càlcul de C2

Amb una freqüència de 24 kHz i amb una resistència de 24 kΩ, obtindrem un calor de C2 de 1,5 nF.

2. 3. 2. AMPLIFICADOR INVERSOR /AMPLIFICADOR NO IVERSOR

Com hem vist a l'apartat anterior 2.3.1, només rebem un senyal procedent del DAC amb la informació de l'àudio, però com hem estat parlant anteriorment, utilitzarem connectors XLR per fer les sortides, per tant necessitarem el senyal en fase i en contrafase.

Per poder realitzar-ho s'ha optat per fer dos amplificadors, un amplificador inversor, que ens donarà el senyal en contrafase i un amplificador no inversor, que ens donarà el senyal en fase.

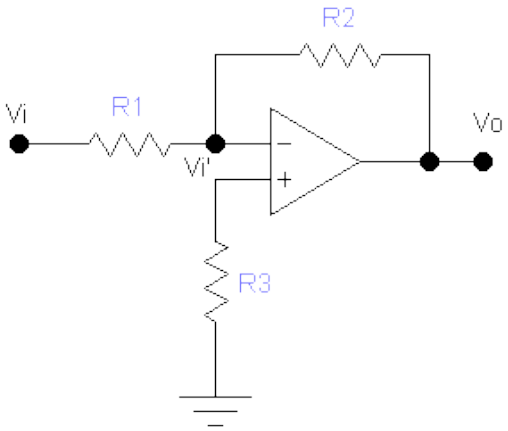


Figura 24 – Amplificador inversor

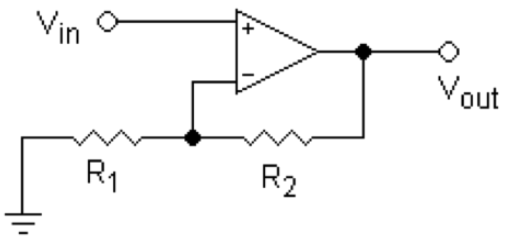


Figura 25 – Amplificador no inversor

A l'ajuntar els dos amplificadors (inversor i no inversor) queda de la següent manera:

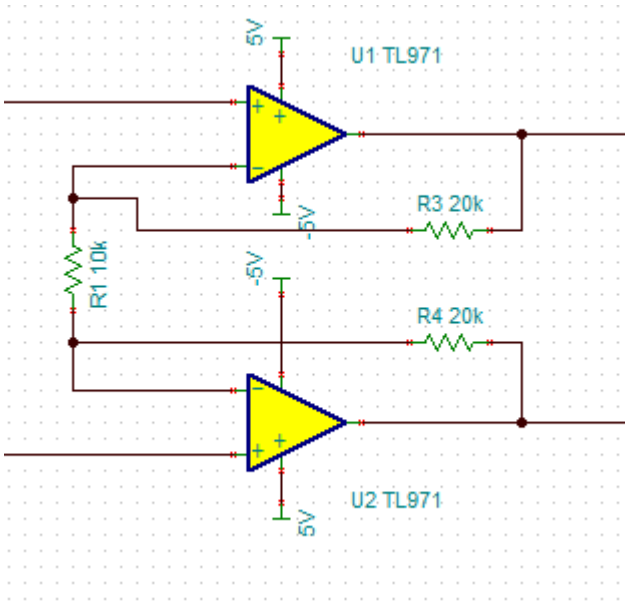


Figura 26 – Inversor més no inversor

Per tant el disseny final de l'etapa de sortida de línia, ajuntant el filtre passa banda i l'amplificador inversor/no inversor, quedaria de la següent manera:

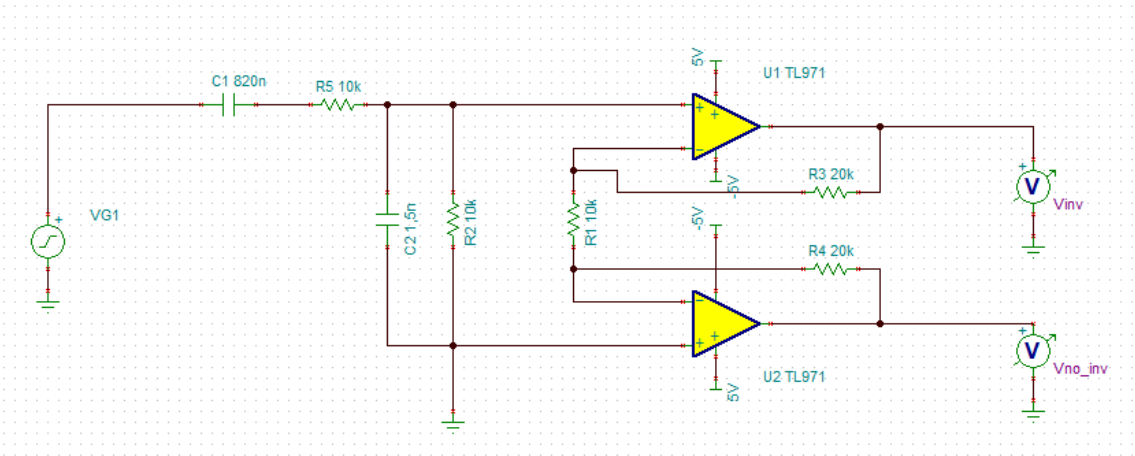


Figura 27 – Disseny final etapa de sortida de línia

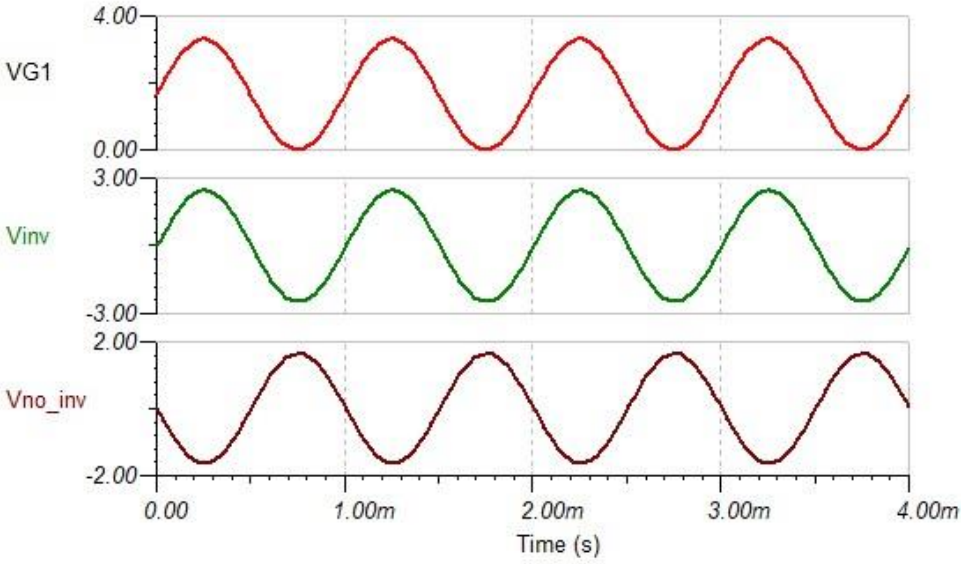


Figura 28 – Simulació etapa de sortida de línia

En la simulació podem comprovar com a la sortida de l'etapa obtindrem dos senyals, una en fase i l'altre en contrafase, per tant ja podrem connectar a la línia balancejada. També obtenim un voltatge de 1,736 V que és el que volíem per la sortida de línia.

2.4. PLACA DE DESENVOLUPAMENT TEENSY 3.6

En aquest apartat s'explica perquè es va escollir la placa de desenvolupament Teensy 3.6 i les diferents alternatives que hi havia abans d'escollir aquesta placa. Veurem que hi ha altres microcontroladors o altres plaques que són més potents, com DSP, però també veurem que per fer un prototip, la placa que ha sigut escollida és de les més potents en relació a qualitat preu.

Abans de començar amb aquest projecte es van mirar altres plaques de desenvolupament més conegudes com són Arudino o Raspberry Pi. També hi havia la possibilitat de poder escollir entre algun DSP o FPGA, però aquestes opcions eren les de cost més elevat i per tant ja es van descartar.

La primera placa de desenvolupament en la que vam pensar va ser l'Arduino MEGA, però ja vam veure directament que segurament necessitaríem alguna placa més potent per poder realitzar les operacions ràpidament i que no tingués problemes de saturació el microcontrolador.

Després es va optar per l'opció de l'Arduino DUE, és similar a l'Arduino MEGA però més potent. Les principals diferencies entre els dos, eren la velocitat del rellotge i les entrades/sortides que tenien cadascú.

Per últim, vam trobar la placa de desenvolupament Teensy 3.6 i en veure les especificacions ja vam veure que aquesta seria la placa ideal, tant per cost, com per qualitat. Per tant aquesta va ser la placa escollida.

Les principals característiques de la placa de desenvolupament Teensy 3.6 són les següents [3]:

- 180 MHz ARM Cortex-M4 with Floating Point Unit
- 1MB de memòria Flash, 256KB de memòria RAM, 4KB de memòria EEPROM
- Chip microcontrolador MK66FX1M0VMD18

- Port USB d'alta velocitat (480 Mbit/seg)
- 2 CAN Bus Ports
- 32 General Purpose DMA Channels
- 22 PWM Outputs
- 4 I2C Ports
- 11 Touch Sensing Inputs
- 62 I/O Pins
- 25 entrades analògiques amb 2 ADC's amb 13 bits de resolució
- 2 sortides analògiques (DAC's) amb 12 bits de resolució
- Micro SD port

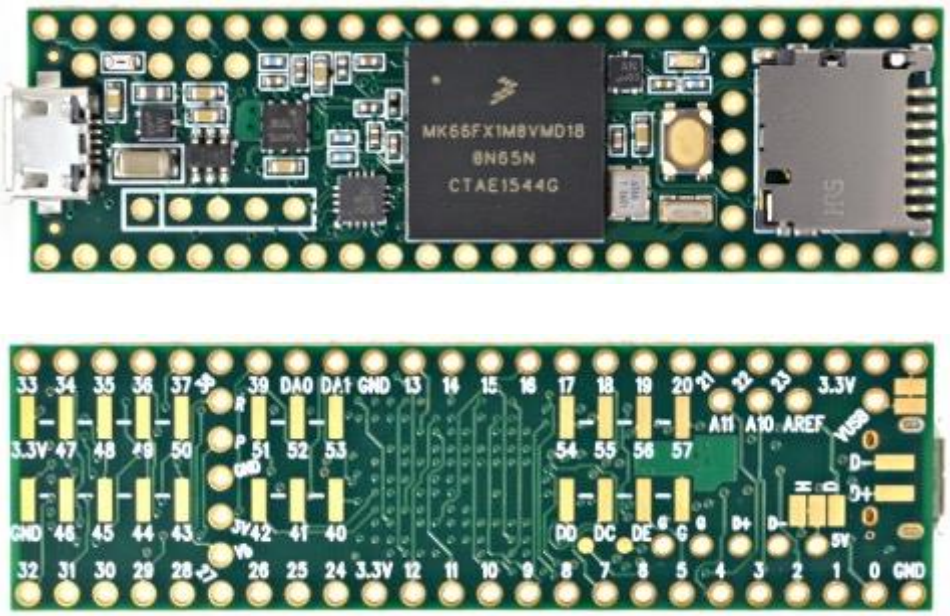


Figura 29 – Teensy 3.6

2. 4. 1. MICROCONTROLADOR MK66FX1M0VMD18

Un microcontrolador és un circuit integrat programable, capaç d'executar les ordres enregistrades en la seva memòria. Està compost de diversos blocs funcionals, els quals compleixen una tasca especifica. Un microcontrolador inclou les tres principals unitats funcionals d'una computadora:

- Unitat central de processament
- Memòria
- Perifèrics d'entrada/sortida.

Figura 30 – Exemple microcontrolador

Els microcontroladors van ser dissenyats per a aplicacions per a sistemes encastats, en contraposició als microprocessadors utilitzats en els ordinadors personals per a aplicacions d'ús general. Gràcies a la seva mida i cost reduït, fan que siguin ideals per al control digital de molts dispositius.

El microcontrolador que porta integrat el Teensy 3.6 és el MK66FX1M0VMD18 [4]. Aquest microcontrolador porta integrat dins seu un processador ARM Cortex-M4F de 180 MHz de velocitat de rellotge. La configuració de memòries és la següent:

- Memòria Flash: 1,2 MB
- Memòria SRAM: 256 KB
- Memòria EEPROM: 4 KB

2. 4. 2. CONNEXIONS UTILITZADES DEL TEENSY

En aquest apartat veurem totes les entrades i sortides necessàries per fer la connexió de les diferents etapes que tenim, etapes d'entrada de línia, etapes de sortida, etapa d'entrada de micròfon i la pantalla LCD.

Connexió	Pin Teensy
ADC0	14
ADC1	15
ADC2	16
DAC0	21
DAC1	22
3,3V	3V3
Pantalla LCD	Veure apartat 2.5

Taula 2 – Connexions Teensy 3.6

2.5. PANTALLA LCD

La pantalla LCD ens servirà per poder mostrar en tot moment les dades que la nostra placa de desenvolupament (Teensy 3.6) està rebent de les entrades de línia i de l'entrada del micròfon. La pantalla escollida és una Newhaven Display de 4,3".

Les característiques principals de la pantalla són les següents:

- Pantalla de 4.3 "
- Panell tàctil resistiu
- Alimentació de 3,3V

Com podem veure podrem alimentar directament amb el Teensy 3.6, ja que té sortides de 3,3V integrades a la placa.

En la següent taula veurem les connexions entre la pantalla i la placa de desenvolupament.

Pin Newhaven Display	Pin Teensy 3.6	Símbol
1	3V3	3V3
2	GND	GND
3	13	SCK
4	12	MISO
5	11	MOSI
6	10	CS
7	6	INT
8	5	NoPD
9	-	AUDIO_L
10	-	N.C.
11	-	GPIO1
12	-	GPIO2

13	-	GPIO3
14	-	GPIO4
15	-	N.C
16	-	N.C.
17	3V3	3V3
18	-	N.C.
19	GND	GND
20	GND	GND

Taula 3 – Connexions Teensy/pantalla

Els pins GPIO al principi es van tenir en compte i es van incloure en el disseny final però no s'utilitzen, ja que són innecessaris.

Aquesta informació ha sigut extreta del manual de la pantalla [5] [6].

2.6. DISSENY FINAL

En aquest apartat, veurem els esquemes de totes les etapes (entrades de línia, entrada de micròfon i sortides), la connexió de la pantalla LCD, els circuits d'alimentació amb els condensadors de desacomplament necessaris i la connexió de tots els elements a la placa de desenvolupament Teensy 3.6.

Un cop mostrat tots els exemples, veurem el disseny final de la placa de circuit imprès. La placa la vam portar a fresar a l'empresa AISLER, envies el disseny de la PCB per internet, amb els fitxers corresponents per poder dur a terme la fabricació de la placa.

2. 6. 1. ALTIUM DESIGNER

Per la realització del disseny de tot el hardware necessari per a aquest projecte s'ha utilitzat el software Altium Designer. L'Altium Designer és un software de disseny de circuits electrònics i de plaques de circuit imprès. Hi ha molts softwares d'aquest tipus, com poden ser ORCAD, Eagle, Kicad, EasyEda...

Per la realització del disseny s'han de seguir uns certs passos. Primer de tot haurem de crear l'esquema del circuit amb tots els components necessaris i les connexions que han de tenir. Hem de posar les alimentacions necessàries, els GND corresponents... Per posar els components que necessitem l'Altium Designer conté certes llibreries on podem trobar el que necessitem. En el nostre cas hi havia certs components que a les llibreries incloses al software no els trobàvem, llavors vam haver de buscar les llibreries corresponents i instal·lar-les al Altium.

Les llibreries les vam trobar a la pàgina web *https://www.snapeda.com* , on podem trobar tant el símbol de l'element que busquem, com el footprint necessari per fer la PCB.

Un cop fet el disseny dels esquemes del circuit, haurem d'assignar els footprints necessaris per a la realització de la placa de circuit imprès, per així poder col·locar els components sabent l'espai que ocuparia cadascú a la placa PCB.

Un cop assignats els footprints realitzarem la PCB, per fer aquest procés, haurem de definir una mida de placa, que el podrem canviar més endavant, i les característiques que ha de tenir. Un cop creada, haurem de col·locar tots els components sobre la placa, seguint el criteri que més ens convingui, per exemple, els condensadors de desacomplament el mes a prop de l'alimentació possible... Quan ja estiguin tots els components col·locats, haurem de fer les pistes necessàries per transportar els senyals. Depenent de quin tipus de senyal hagin de portar les pistes, les farem d'una mida o d'una altra. Les mides de les pistes que s'han utilitzat per a aquest projecte són:

- Pistes d'alimentació: 1 mm
- Pistes de senyals: 0.3 mm

2. 6. 2. ESQUEMES FINALS

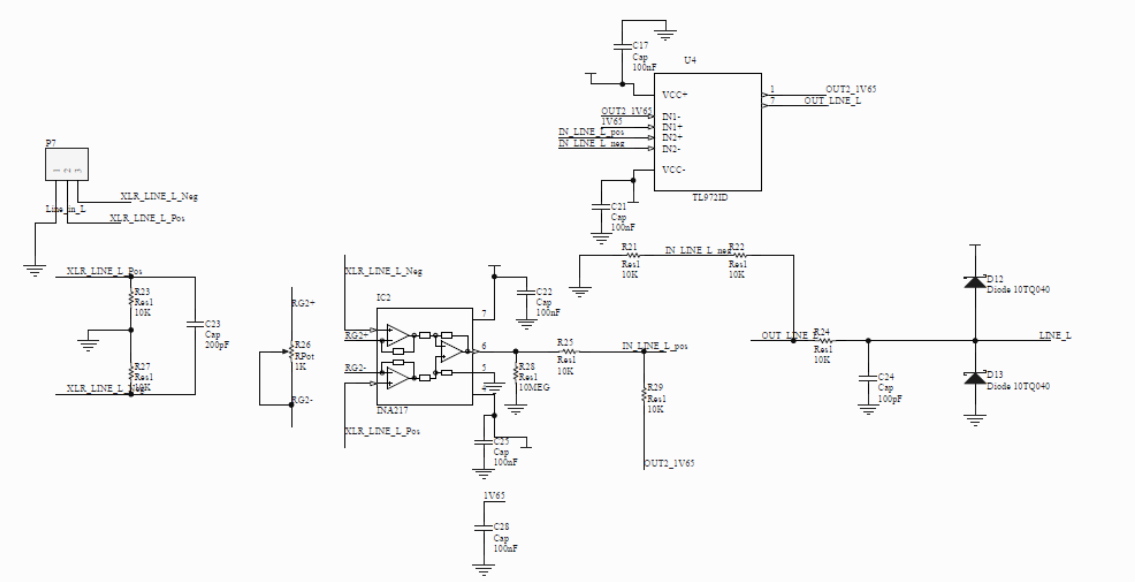


Figura 31 – Etapa d'entrada de línia L

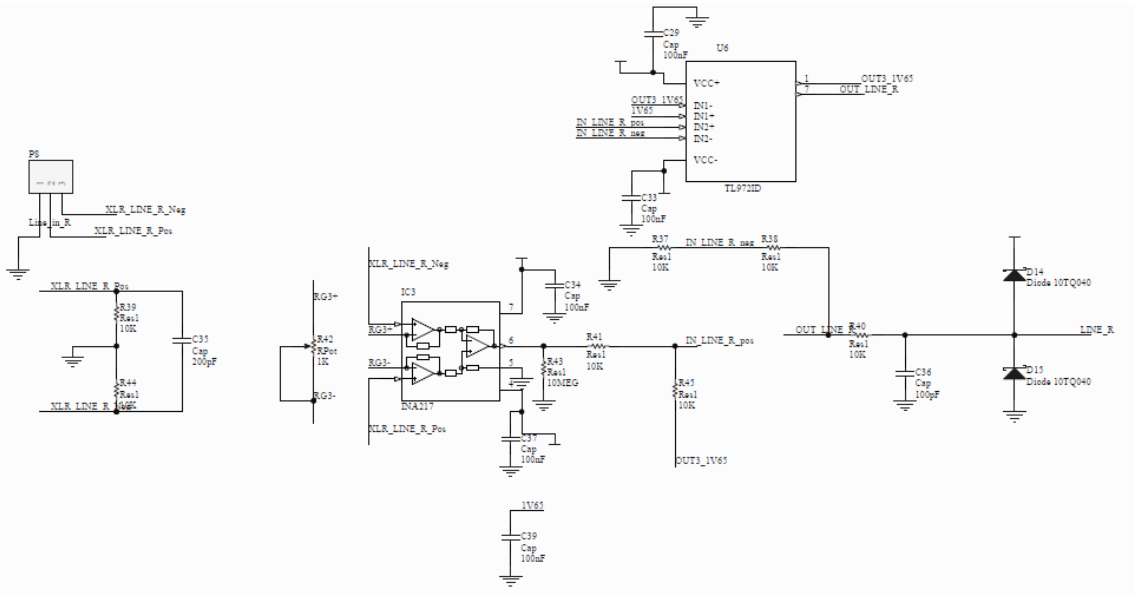


Figura 32 – Etapa d'entrada de línia R

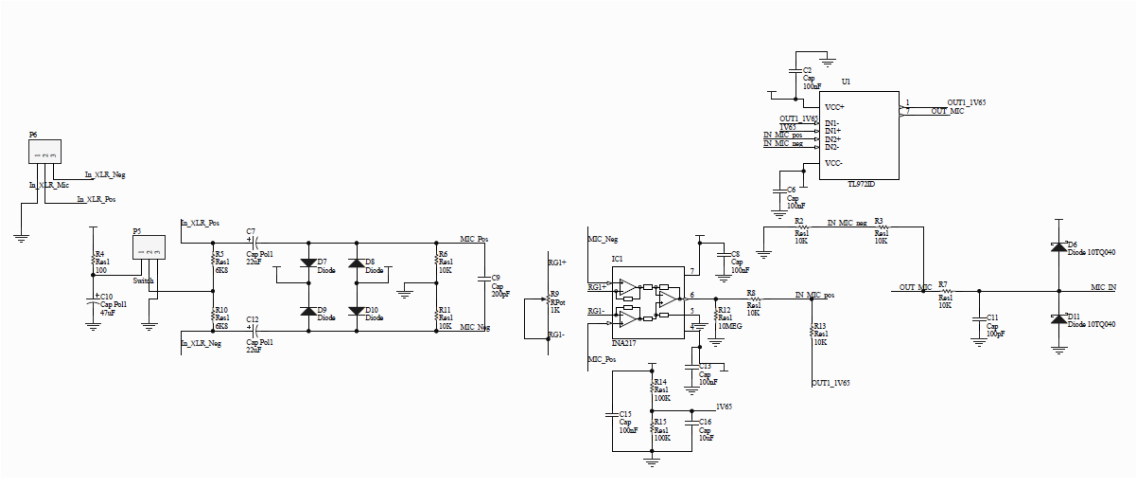


Figura 33 – Etapa d'entrada de micròfon

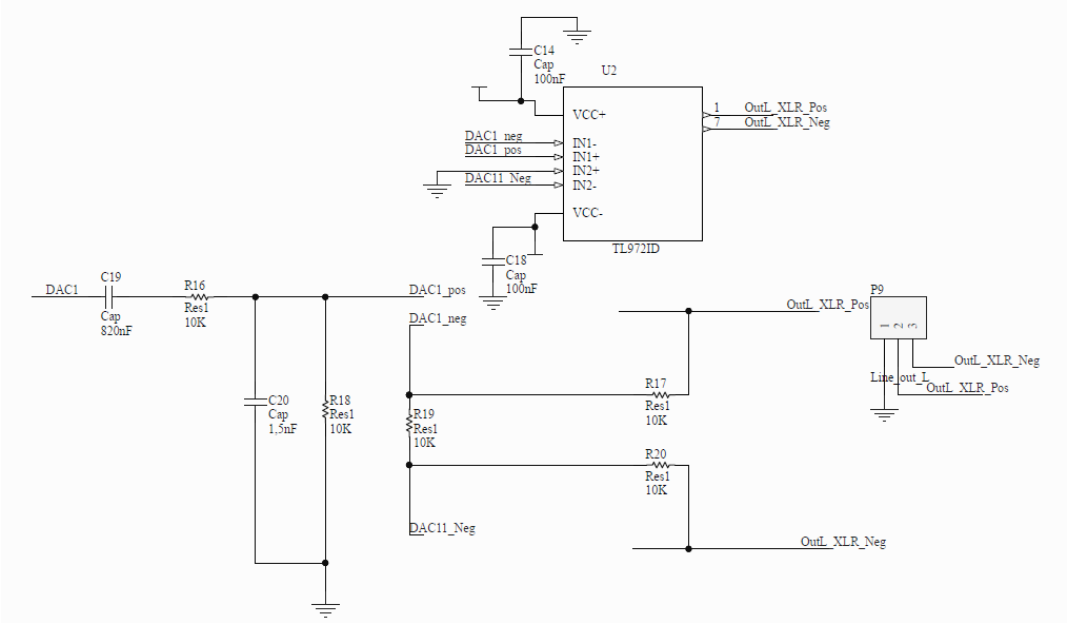


Figura 34 – Etapa de sortida de línia L

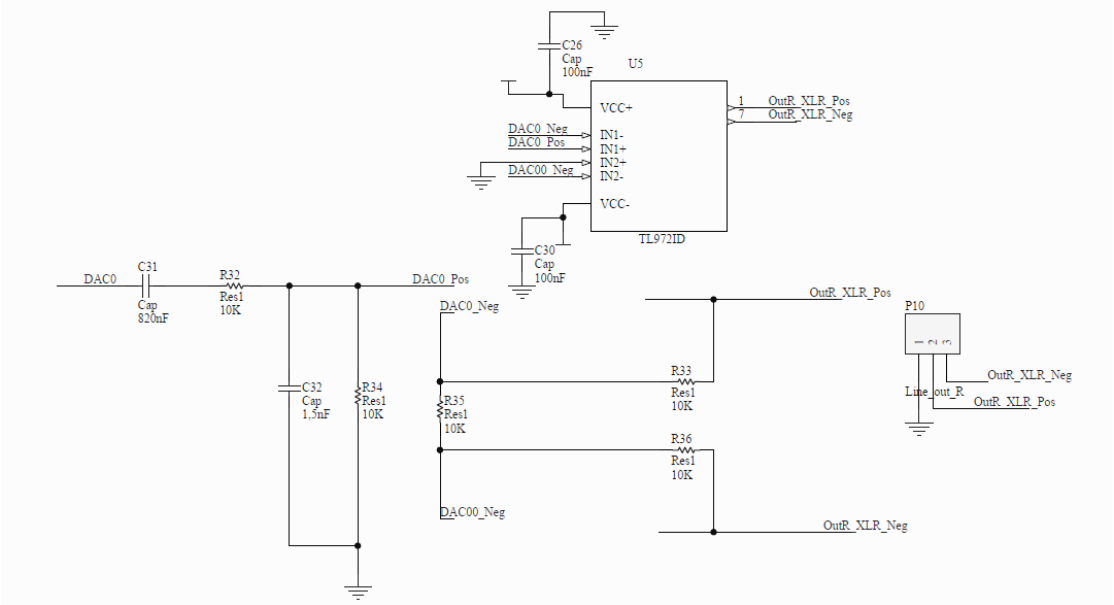


Figura 35 – Etapa de sortida de línia R

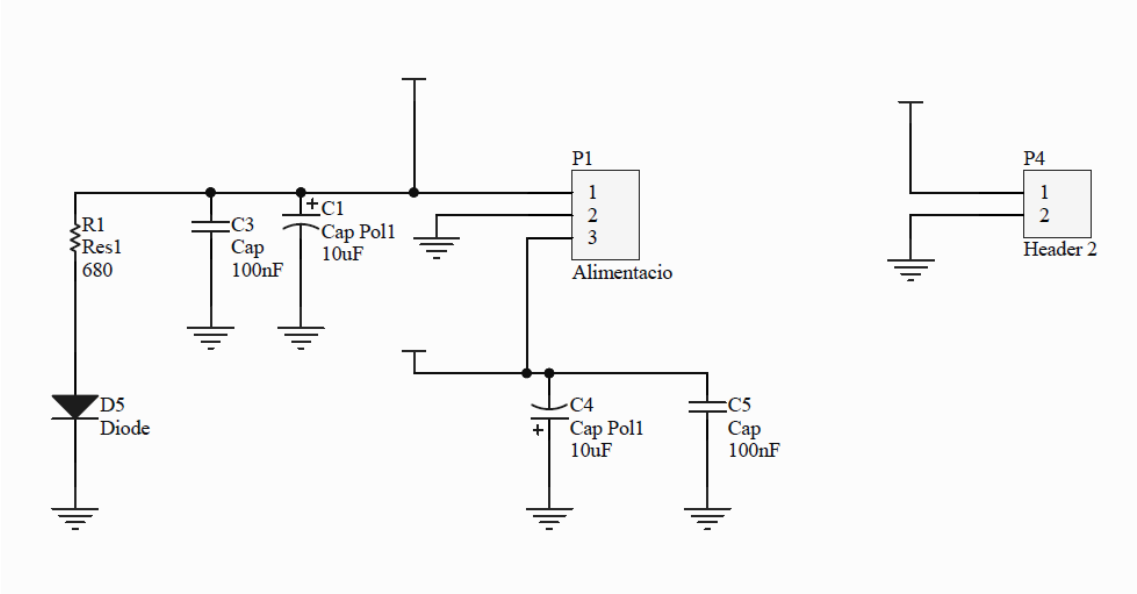


Figura 36 – Alimentació

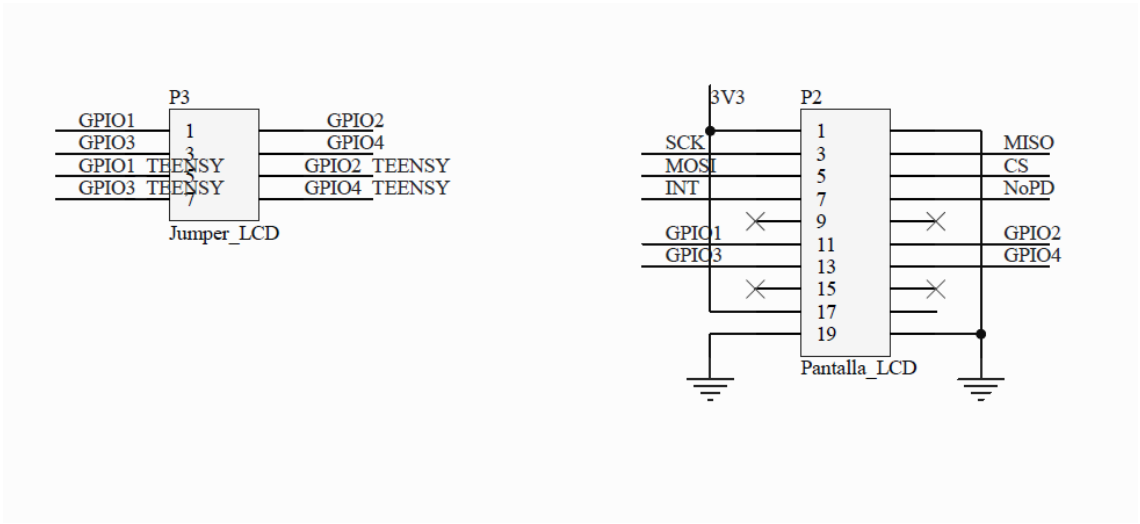


Figura 37 – Pantalla

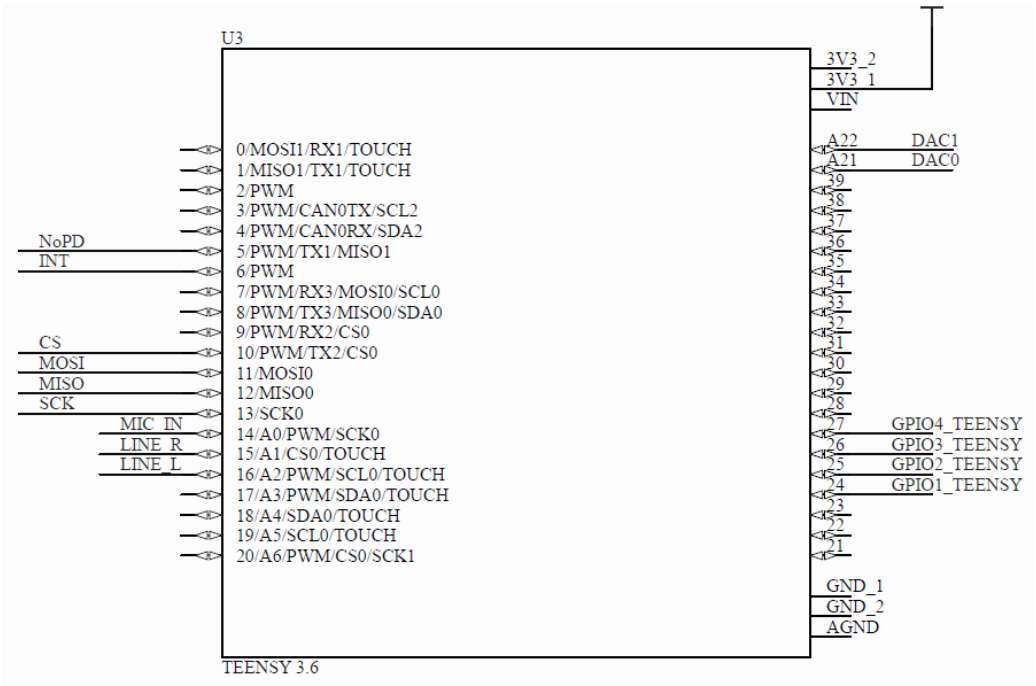


Figura 38 – Connexionat Teensy 3.6

2. 6. 3. PLACA DE CIRCUIT IMPRÈS

Per fer aquesta placa de circuit imprès es va seguir una estratègia per poder col·locar tots els components correctament i que la placa no fos molt gran. El primer component que es va col·locar va ser el Teensy 3.6, ja que és el més gran de tot. Un cop col·locat el Teensy, ja haurem de començar a dissenyar la col·locació de les etapes d'entrades i sortides, i de l'alimentació necessària per poder alimentar tota la placa.

Es va dissenyar una de les entrades de línia com a referència, per saber quin espai ocuparia. Un cop dissenyada, es va intentar comprimir al màxim perquè ocupes el mínim espai possible però que les pistes tinguessin suficient espai per passar. Un cop finalitzat el disseny de l'etapa d'entrada de línia, es realitzaran dos més iguals, corresponents a l'altra entrada de línia i a l'entrada de micròfon. A la de micròfon se li va afegir l'etapa d'alimentació phantom just a sobre de l'etapa.

Tots els condensadors electrolítics estan col·locats a prop dels connectors d'entrada, tant de micròfon, com els d'alimentació.

Després es va realitzar el disseny de l'etapa de sortida, seguint el mateix procés que amb les etapes d'entrada. Un cop dissenyada una de les etapes, es va fer el mateix per l'altra etapa de sortida.

Per últim es va col·locar el connector de la pantalla, un connector de 20 pins.

Un cop tots els components han estat col·locats a la placa, haurem de realitzar totes les pistes per comunicar tots els components. Molt important, no poden haver-hi pistes amb angles de 90º.

La mida de les pistes se seguirà el comentat a l'apartat 2.6.1.

Aquest és el disseny final de la placa de circuit imprès de la capa TOP i la capa BOTTOM.

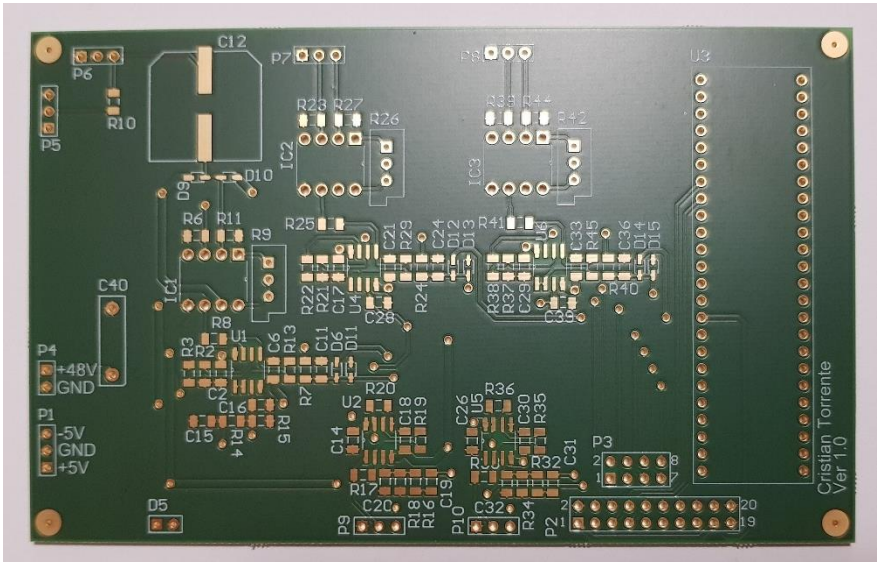


Figura 39 – Top Layer

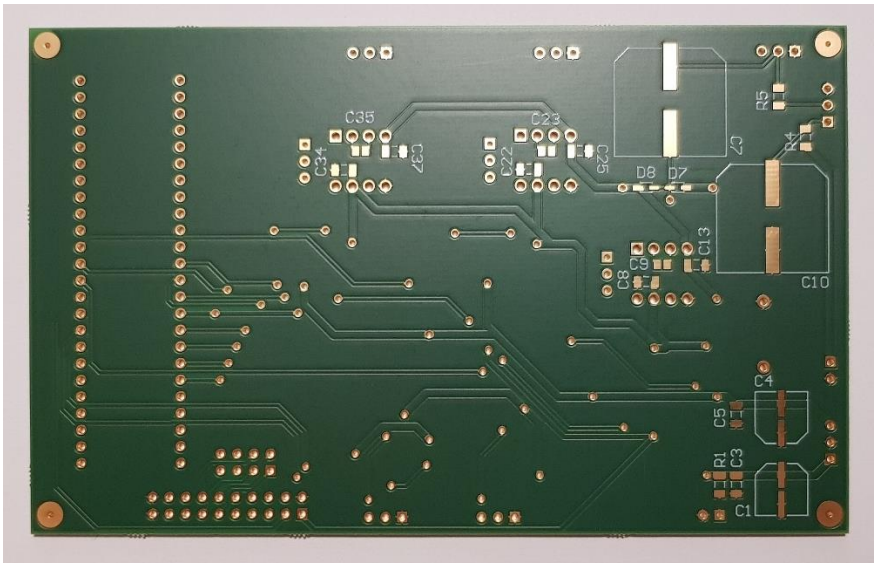


Figura 40 – Bottom Layer

3. DISSENY DEL SOFTWARE

En aquest apartat explicarem el software dissenyat per fer possible el procés d'equalització automàtica del que consta aquest projecte. Per fer aquesta programació utilitzarem l'entorn de programació Arduino. A continuació veurem un esquema a mode d'introducció per centrar-nos en els blocs principals de programació:

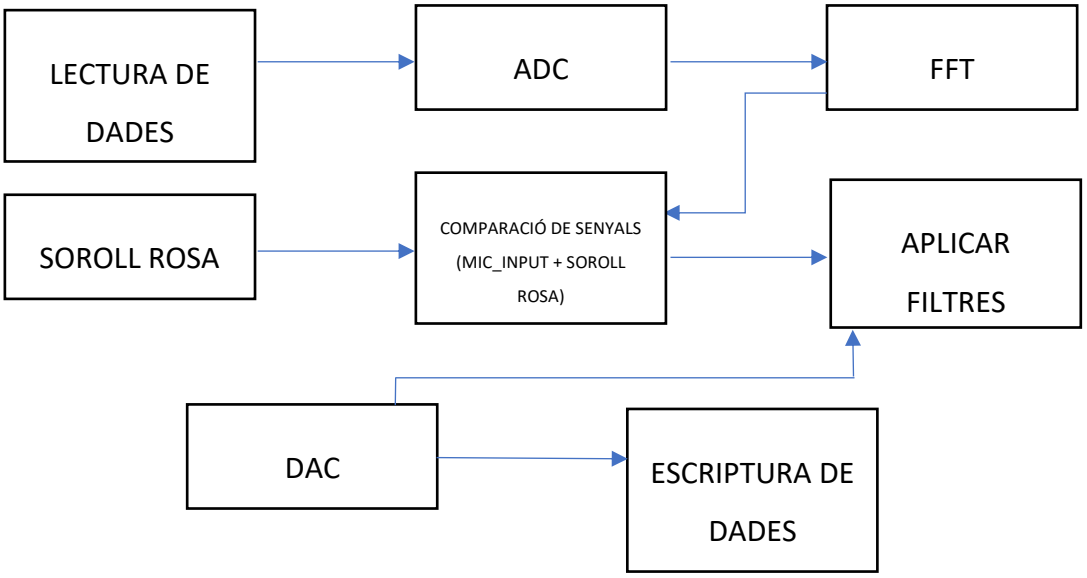


Figura 41 – Blocs de programació

En aquest esquema podem veure a mode de resum el procés que ha de passar un senyal d'àudio per ser equalitzat a nivell de programació.

Primer de tot haurem de fer la lectura dels senyals de les diferents etapes d'entrades que tenim connectades a la nostra placa de desenvolupament Teensy 3.6, per tant haurem de fer una lectura de les 3 entrades que tenim al nostre sistema, les dues de línia mes la de micròfon. Un cop tenim la lectura de les dades haurem de passar-les al món digital, per això fem servir un ADC (Analog to Digital Convert), això ens permetrà fer la Fast Fourier Transform (FFT) de les 3 entrades que tenim, ja que ens permetrà veure l'espectre freqüencial de cada senyal i l'amplitud a cada banda de freqüència. Un cop tenim feta la FFT, haurem de generar un soroll rosa, ja que com hem dit a la introducció, el senyal rosa

és aquell que conté el mateix nivell a totes les bandes de freqüències, per tant per veure la resposta de la sala és el més ideal. Un cop generat el soroll rosa, el compararem amb la FFT que hem aplicat a l'etapa d'entrada del micròfon, per així poder veure la diferència entre un soroll rosa totalment pla (generat) i el soroll rosa que capta el micròfon, així es veuran les diferències entre un i l'altre i es podran corregir les deficiències aplicant l'equalització necessària. Un cop feta la comparació, haurem d'aplicar els diferents filtres per amplificar o atenuar el senyal d'entrada de línia, que és el que ens interessa. Un cop feta l'equalització, el senyal de línia l'haurem de tornar a passar al món analògic, ja que ara mateix està en el món digital, per tant haurem d'aplicar un DAC (Digital to Analog Convert). Un cop convertit el senyal en analògic l'únic que faltarà és escriure aquest senyal a les sortides corresponents perquè vagin cap a les etapes de sortides del nostre equalitzador.

També haurem de fer la programació de la pantalla LCD. En aquest cas farem un analitzador d'espectres de 7 bandes de freqüències, (hem agafat l'exemple de gameduino 2), amb aquest espectre obtindrem l'amplitud de cada banda de freqüència de manera visual i així podrem anar monitorant en tot moment com és el senyal d'entrada del micròfon.

3.1. CONFIGURACIÓ DEL ADC

En aquest apartat s'explicarà la configuració del ADC, que ens permetrà fer la lectura de dades de les etapes d'entrada de línia i micròfon, i la transformació al món digital per poder treballar amb aquests senyals dins del nostre microcontrolador. Per fer aquest procés haurem de configurar alguns paràmetres del ADC.

Primer de tot haurem de mirar les característiques del microcontrolador, per poder veure quina resolució màxima pot tenir el nostre ADC. Com podem veure al datasheet del microcontrolador MK66FX1M0VMD18 [4] la resolució màxima és de 16 bits, per tant és la màxima resolució que podrem configurar als ADC per poder fer la transformació al món digital.

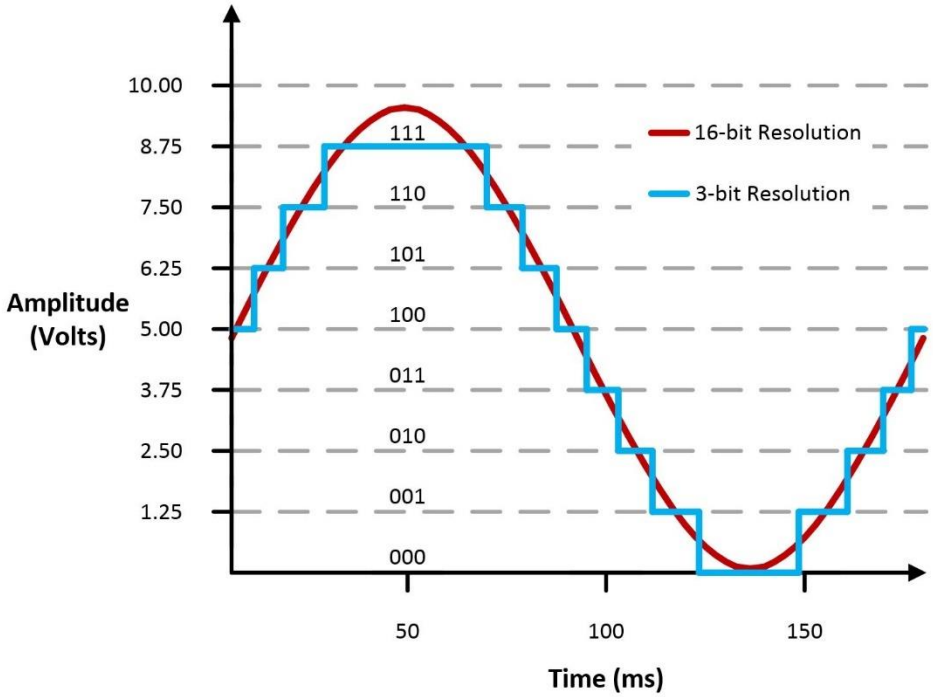


Figura 42 – Bit resolution

En aquesta imatge podem veure dos ADC's, un que treballa amb 3 bits de resolució i un altre amb 16 bits. Com podem veure a més resolució, més fidel serà el senyal digital amb el senyal analògic. En el nostre cas configurarem els ADC's a una resolució de 12 bits, ja que si configurem els ADC's a 16 bits, els bits 14, 15 i 16 es posen a 0 per poder omplir els valors fins a arribar a 16.

3. 1. 1. CONFIGURACIÓ DEL ADC DE LES ENTRADES DE LÍNIA

Primer de tot haurem de crear les variables necessàries per guardar els valors de lectura del ADC, per tant crearem dos variables tipo *integer:*

```
int read_ADC_L, read_ADC_R;
```

També haurem de configurar quins són els pins d'entrada, per tant haurem de declarar els pins A1 i A2 (corresponents als ADC's necessaris) i dir que són entrades:

```
const int readPin = A1;
const int readPin = A2;
void setup(){
     pinMode(readPin, INPUT);
     pinMode(readPin2, INPUT);
...}
```

En el setup del programa haurem de declarar els paràmetres de la resolució de lectura del analogRead que com ja hem dit anteriorment serà de 12 bits [7]:

```
void setup(){
...
     analogReadResolution(12);
...
}
```

Un cop definides les variables necessàries i la configuració requerida del ADC, haurem de fer la lectura de les dades en els pins corresponents:

```
void lectura_ADC_DAC(){
...
     read_ADC_L = analogRead(readPin);
     read_ADC_R = analogRead(readPin2);
...
}
```

Un cop tenim definits aquets paràmetres ja tenim l'ADC configurat i ja ens permetrà llegir les dades que tenim els pins A1 i A2 del Teensy, que corresponen a les dues entrades de línia.

3. 1. 2. CONFIGURACIÓ DEL ADC DE L'ENTRADA DE MICRÒFON

Per fer la configuració del ADC per l'entrada de micròfon hem utilitzat la llibreria *Teensy Audio Library* [8]. Com que a l'entrada del micròfon és a la que aplicarem la Fast Fourier Transform, utilitzarem la mateixa llibreria per fer la lectura de dades d'aquesta entrada.

Per configurar l'ADC per l'entrada de micròfon haurem de declarar l'entrada d'àudio necessària, en aquest cas el pin del Teensy que ens interessa és el A0.

```
AudioInputAnalog          adc0(A0);
```

Per tant amb aquest comando ja estem declarant el pin A0 com a entrada i ja li estem aplicant el ADC per tenir les dades en digital.

3.2. CONFIGURACIÓ DEL DAC

En aquest apartat s'explicarà la configuració del DAC, que ens permetrà l'escriptura de dades a les etapes de sortida del nostre equalitzador, el DAC serveix per transformar senyals digitals a analògics, és el contrari que el ADC.

En aquest cas també mirarem el datasheet del microcontrolador per veure les característiques que presenta el DAC i llavors farem la configuració. La resolució màxima del DAC és de 12 bits, per tant no podrem pujar la resolució a més bits. Aquesta serà la resolució màxima que aplicarem al nostre conversor digital analògic.

Primer de tot configurarem quins són els pins de sortida del Teensy, en aquest cas hem escollit els pins A21 i A22, ja que són els dos únics pins de la placa de desenvolupament que inclouen els DAC.

```
const int writePin_L = A21;
const int writePin_R = A22;
```

En el setup del programa haurem de crear la configuració necessària del DAC, en aquest cas només haurem de dir quina resolució ha de tenir el conversor. Com ja hem dit a l'inici d'aquest apartat la resolució del DAC serà de 12 bits:

```
void setup(){
      ...
      analogWriteResolution(12);
      ...
}
```

Per últim, haurem d'escriure les dades cap a les variables creades anteriorment, que són el *writePin_L* i el *writePin_R*:

```
void lectura_ADC_DAC(){
      analogWrite(writePin_L, read_ADC_L);
      analogWrite(writePin_R, read_ADC_R);
}
```

Amb el comando analogWrite, escriurem les dades rebudes de la variable read_ADC_x cap a les variables writePin_x.

3.3. CONFIGURACIÓ DEL TIMER

En aquest apartat explicarem la configuració del Timer i el motiu pel qual utilitzarem un. Utilitzarem el Timer per fer lectures del ADC i per generar el soroll rosa, ja que així el nostre sistema no està sempre actiu.

Per configurar el Timer primer de tot haurem d'inicialitzar-lo, llavors haurem de crear un objecte de IntervalTimer:

```
IntervalTimer ADC_DAC_Timer;
```

El paràmetre més important que ha de tenir el Timer és el temps d'activació i de parada, per tant li haurem de passar un temps perquè arranqui cada x segons.

En el nostre cas farem el càlcul a partir de la freqüència de mostreig. Definim una freqüència de mostreig de 50000 Hz i a partir d'aquí calculem el període.

```
const double samplingFrequency = 50000;

unsigned int sampling_period_us;

...

void setup(){

     ...

     Sampling_period_us=round(1000000*(1.0/samplingFrequency)
     );

     ...

}
```

Un cop calculat el període de mostratge, ja li podem passar al Timer, cada període de mostratge s'activarà el Timer en la rutina que nosaltres vulguem. Per tant:

```
ADC_DAC_Timer.begin(lectura_ADC_DAC, sampling_period_us);
```

Per activar la interrupció o per parar-la, ho haurem d'indicar amb dos comandos:

- `noInterrupts(); -> Desactiva la interrupció`
- `interrupts(); -> Activa la interrupció`
-

3.4. TRANSFORMADA RÀPIDA DE FOURIER (FFT).

En aquest apartat s'explicarà el perquè s'utilitza la transformada ràpida de Fourier i quin és el codi que executa aquest càlcul.

La transformada rapida de Fourier ("Fast Fourier Transform" o FFT), és un algorisme que permet calcular la transformada de Fourier discreta i la seva inversa. Les aplicacions més comunes de la FFT són a tractament d'imatges, tractament d'àudio, reducció de soroll als senyals, anàlisis en freqüència de qualsevol senyal discreta...

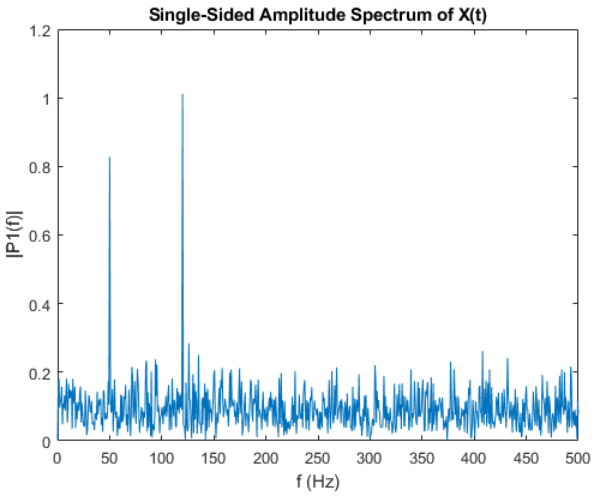


Figura 43 – Exemple FFT

En el nostre cas volem aplicar la transformada rapida de Fourier, per poder veure l'espectre freqüencial que té la nostra entrada de micròfon al rebre el soroll rosa que prové dels altaveus de la sala i per tant per poder veure quin espectre freqüencial té la sala a la qual aplicarem el nostre equalitzador. Per tant només haurem d'aplicar la FFT a l'etapa d'entrada del micròfon, que en aquest cas va a parar al pin d'entrada 14 del Teensy 3.6. Aquest serà el nostre ADC0.

Per fer la FFT, farem servir la llibreria Teensy Audio Library [8], la qual ens permet fer la lectura del ADC, com ja s'ha explicat en l'apartat 3.1.2, i aplicar la FFT a aquest senyal d'entrada. La limitació que té aquesta llibreria es que només et permet aplicar 1024 punts a la FFT, quan en el nostre cas seria interessant poder aplicar més punts.

Per tant el que haurem de fer primer de tot és declarar les llibreries necessàries per poder implementar el codi correctament:

```
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
```

Un cop declarades les llibreries, haurem de configurar els paràmetres de la FFT, per tant primer de tot haurem de crear els components d'àudio necessaris per poder implementar-la.

```
AudioAnalyzeFFT1024      myFFT;
AudioConnection          patchCord1(adc0, myFFT);
```

Amb el comando AudioAnalyzeFFT1024, declaro l'objecte FFT de 1024 punts amb el nom que vulguem, en aquest cas myFFT. El comando AudioConnection [9] ens servirà per enviar el senyal llegit del ADC0 (micròfon), cap a l'objecte myFFT.

També haurem de fer la configuració d'alguns paràmetres necessaris pel bon funcionament de la llibreria, haurem de configurar la memòria que requereix les connexions d'àudio i per la part de la FFT haurem de configurar quin tipus de finestra volem que faci.

```
void setup(){
     AudioMemory(12);
     myFFT.windowFunction(AudioWindowHanning1024);
}
```

Un cop declarades les variables, objectes i configuracions necessàries, passarem a la rutina per fer la lectura de dades de la FFT. En aquest cas farem un bucle per saber quan l'objecte myFFT està disponible, quan sigui així, començarà la FFT a llegir les dades que entren del ADC0 i les guardarà a una variable tipus float.

```
if(myFFT.available()){
     for (i=0; i<100; i++){
          N = myFFT.read(i);
}}
```

Per acabar es mostraran les dades per el serial monitor del software d'Arduino i aquestes dades seran les que ens interessarà mostrar per la nostra pantalla LCD.

3.5. PROGRAMACIÓ PANTALLA

En aquest apartat s'explicarà quin disseny s'ha fet per la pantalla tàctil. En la pantalla tàctil el que volem veure és el senyal del micròfon de mesura, per així poder veure en tot moment la resposta freqüencial de la sala. Com que podrem veure a temps real la resposta freqüencial de la sala, podrem veure també com està actuant l'equalitzador que ha estat dissenyat per aquest projecte.

Abans de començar amb la programació de la pantalla, es van estar provant vàries llibreries que estan dedicades a la programació de pantalles LCD de diferents característiques i polsades. Es va provar la llibreria Gameduino 2 [10] [11], ja que és una de les llibreries que són compatibles amb el Teensy i amb les pantalles FT_81x, que en el nostre cas és la FT_812. Es va carregar un projecte d'exemple per veure si la pantalla funcionava correctament, es va escriure una frase a la pantalla amb un fons de color.

Al veure que tot funcionava correctament, ja es va començar a mirar exemples de programació d'analitzadors d'espectres i es va trobar un a la llibreria Gameduino 3 [12], que és amb el que ens hem basat per fer la programació de la nostra pantalla. L'exemple que trobem a la llibreria Gameduino 3, és un analitzador d'espectres de 7 bandes de freqüències [13].

En el nostre cas farem servir el mateix analitzador d'espectre de 7 bandes, l'ideal seria tenir moltes més bandes de freqüències, per poder veure l'espectre freqüencial amb més detall, però amb una pantalla de 4,3 polsades, no podem incloure moltes bandes de freqüències.

Per fer la programació, primer de tot, es va revisar tota la programació que hi havia a l'exemple i es va anar descartant el que no necessitàvem, ja que aquest analitzador d'exemple, inclòs a la llibreria, feia la lectura de dades directament d'una entrada analògica d'un circuit de preamplificació.

En el nostre cas, el senyal que ens interessa mostrar per pantalla és el senyal d'entrada del micròfon, per poder veure la resposta de la sala al soroll rosa, per tant haurem d'assignar com a entrada el pin 14 del Teensy, es a dir el ADC0.

Les bandes de freqüències que es mostraran a la pantalla són: 63, 160, 400, 1000, 2500, 6250, 16000 Hz. Aquestes bandes de freqüències es mostren en format barra, on les barres representen el nivell d'amplitud del senyal a la banda de freqüència indicada. En la pantalla també podrem visualitzar un gràfic on veurem la forma d'ona del senyal d'entrada a temps real.

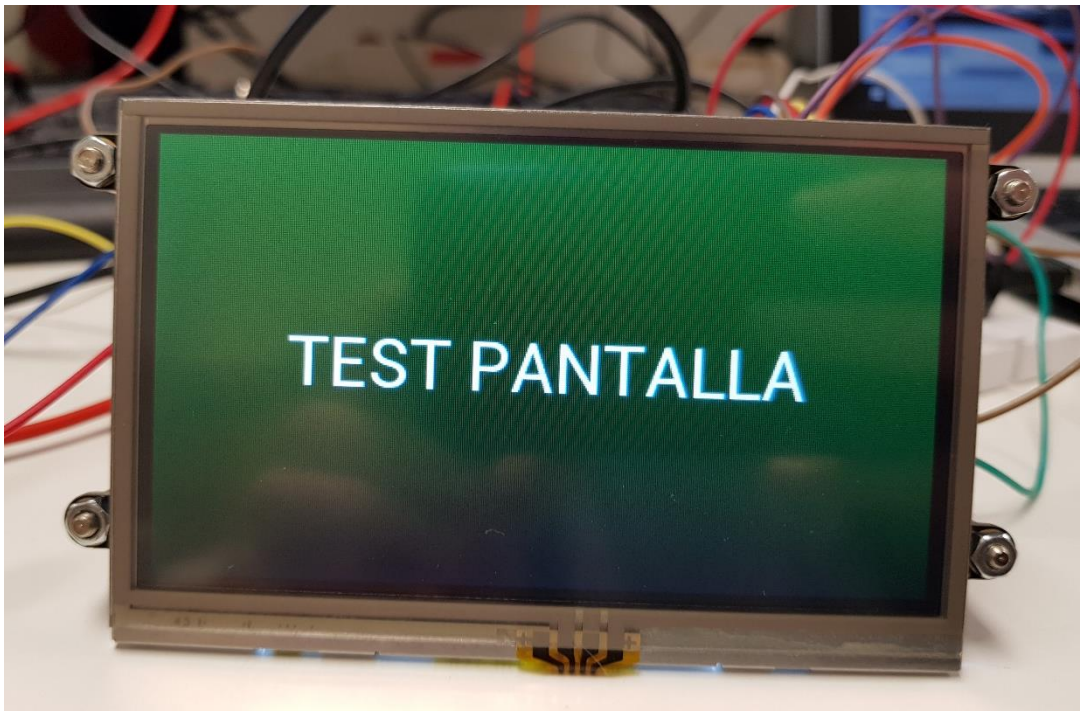


Figura 44 – Test pantalla

3.6. ESQUEMA DE PROGRAMACIÓ

En aquest apartat podem veure l'esquema que ha de seguir la programació del microcontrolador perquè tot funcioni correctament. Com es pot veure en l'esquema, el micròfon sempre estarà captant dades i si el nombre de mostres és igual a 1024, començarà a mostrar les dades per pantalla.

També es podrà activar l'equalització en qualsevol moment, si és que si, es començarà a emetre el soroll rosa i si hi ha 1024 mostres, es farà la FFT del micròfon i seguidament es comparà el senyal del micròfon amb el senyal de referència (soroll rosa). Un cop feta la comparació, s'equalitzarà els senyals del ADC1 i ADC2.

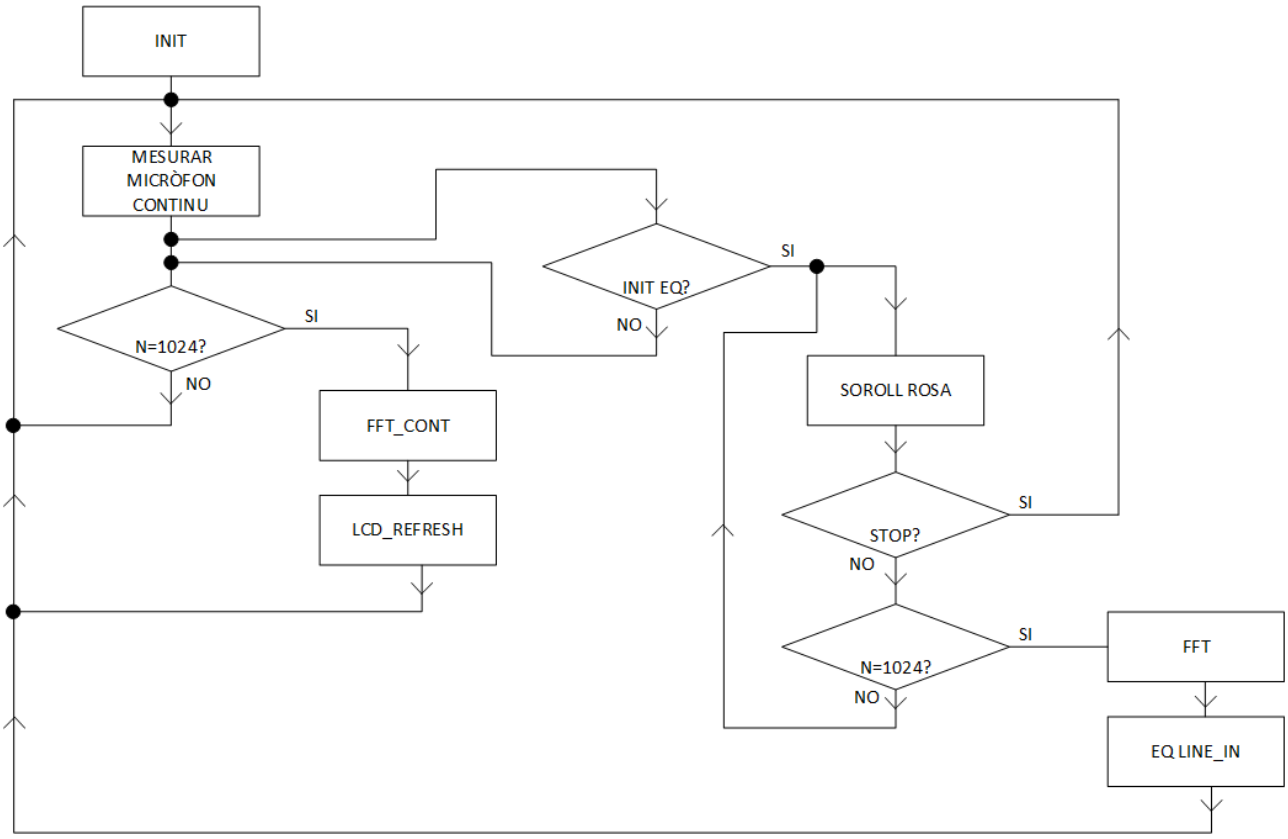


Figura 45 – Esquema de programació

4. PRESSUPOST

En aquest apartat veurem tots els components, amb el seu preu corresponent, que s'han adquirit per fer possible aquest prototip. Al final de la taula veurem el total del cost del prototip.

Component	Preu unitari	Quantitat	Subtotal
TL972ID	0,744	10	7,44 €
R 680 ohm	0,0139	10	0,14 €
R 10K	0,184	40	7,36 €
R 100 ohm	0,187	10	1,87 €
R 6K8 (1%)	0,0178	10	0,18 €
R 10 MEG	0,0196	10	0,20 €
R 100K	0,0166	10	0,17 €
C 100nF	0,221	30	6,63 €
C 200 pF	0,132	10	1,32 €
C 47uF	0,701	1	0,70 €
C 10uF	0,36	5	1,80 €
C 820nF	0,282	5	1,41 €
C 1,5nF	0,156	10	1,56 €

C 100pF	0,151	10	1,51 €
C 10 uF	0,329	5	1,65 €
C 22 uF	0,366	2	0,73 €
Potenciòmetre 1K	2,79	3	8,37 €
INA217	6,54	5	32,70 €
PANTALLA	41,01	1	41,01 €
DIODES	0,165	12	1,98 €
XLR FEMALE	1,97	3	5,91 €
XLR MALE	1,43	2	2,86 €
Teensy 3,6	44,29	1	44,29 €
PCB	20	3	60,00 €
TOTAL			**231,78 €**

5. RESULTATS EXPERIMENTALS

En aquest apartat del projecte veurem els resultats obtinguts de totes les parts realitzades en aquest projecte, veurem el prototip muntat, com funcionen les etapes d'entrada i les de sortida i el funcionament de la pantalla LCD.

Com es pot comprovar no es parla de resultats en relació amb l'equalització automàtica, això és degut a que per falta de temps no s'han pogut realitzar totes les parts desitjades en aquest projecte. Més endavant, a les conclusions, ja desenvoluparem el perquè.

5.1. HARDWARE

Començarem amb els resultats obtinguts de tot el hardware. Per realitzar les proves al hardware del prototip, es van utilitzar:

- Font d'alimentació: per alimentar el circuit a ±5V
- Generador de funcions: per introduir senyals sinusoidals a les diferents etapes d'entrada del prototip
- Analitzador d'espectres: per poder visualitzar tant els senyals d'entrada, com els de sortida.

Abans de mostrar els resultats, mostrarem el prototip un cop soldat a la placa de circuit imprès.

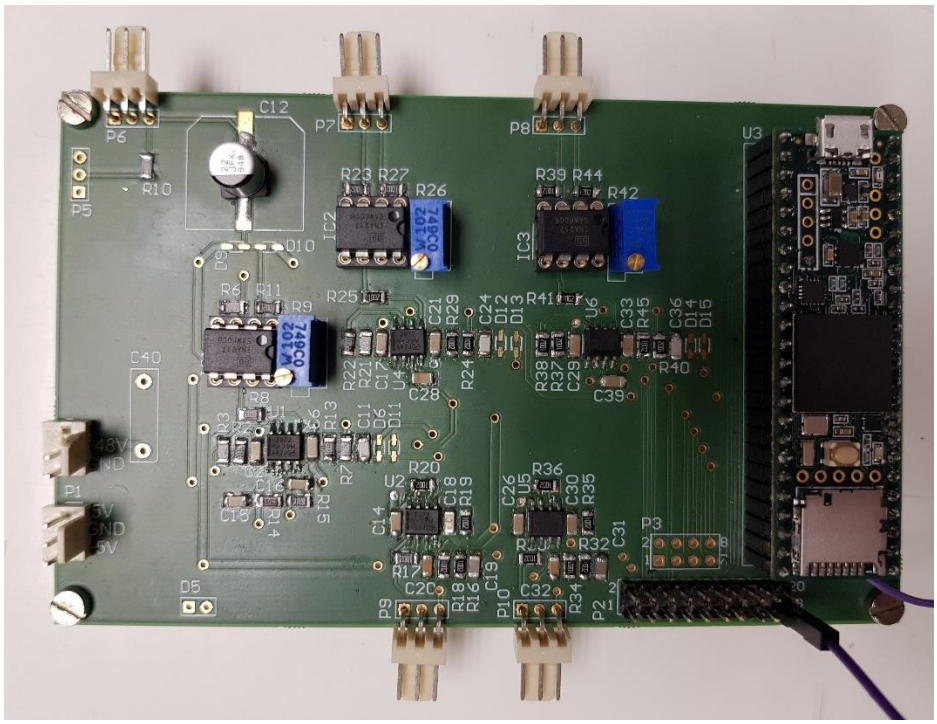


Figura 46 – Prototip cara top

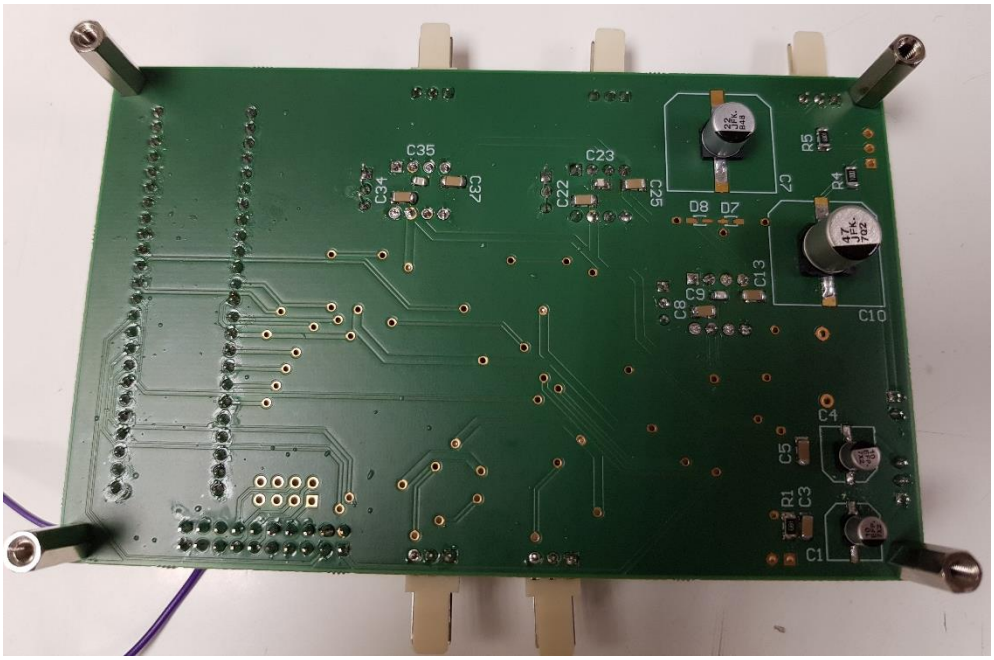


Figura 47 – Prototip cara bottom

La primera prova que vam fer, va ser quan s'estava soldant la placa. Cada cop que teníem una etapa nova soldada, alimentàvem el circuit per veure com es comportava el consum de la placa, al veure que el consum era molt baix (aprox. 0,01 A), vam seguir soldant fins a tenir tota la placa preparada.

Es pot veure que els footprints dels condensadors electrolítics no coincideixen amb les mides del component. Per fer el footprint vam mirar el datasheet del fabricant per veure les mides que tenia el component i quan van arribar els components ja vam veure que les mides no coincidien, però a l'hora de soldar no hi ha agut cap problema, ja que el footprint que es va dissenyar era més gran que les mides del component real.

Per poder mostrar els resultats, es van realitzar captures de l'analitzador d'espectres de diferents situacions, també es mostrarà una taula amb les diferents etapes d'entrada i de sortida, introduint un senyal sinusoidal a vàries freqüències, per veure el comportament de cadascuna d'aquestes etapes.

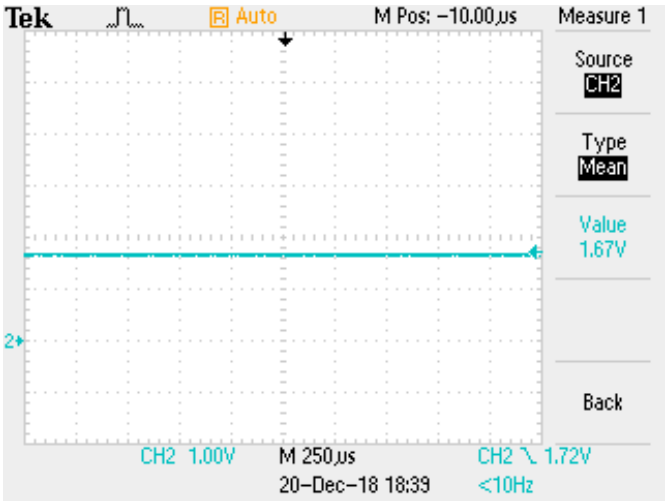


Figura 48 – Circuit que introdueix 1,67 V

En aquesta imatge podem observar com el circuit que ens dona els 1,67 V de continua, funciona correctament.

Ara ens centrarem en el canal dret del nostre sistema:

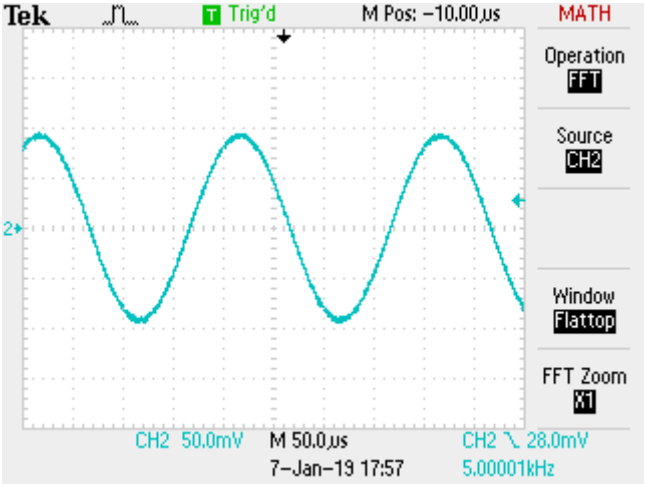


Figura 49 – Senyal d'entrada a l'etapa de línia dreta

En aquesta imatge podem veure el senyal d'entrada que introduirem a l'etapa d'entrada de línia dreta. Tenim un senyal de 5 kHz i amb 200 mV de pic a pic.

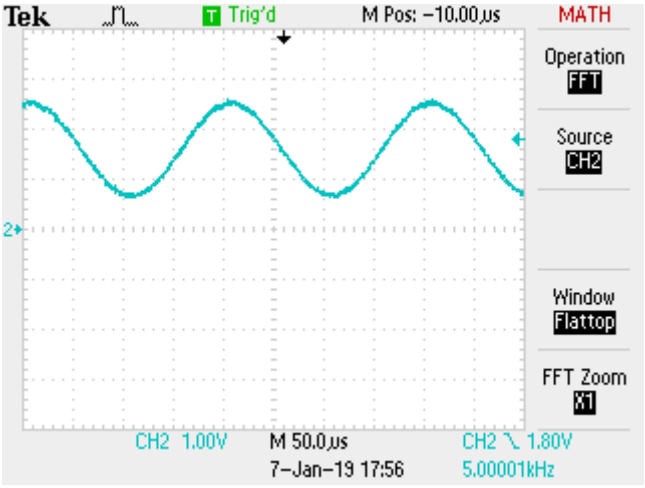


Figura 50 – Senyal de sortida de l'etapa analògica

Aquí podem observar la sortida de l'etapa analògica, abans de passar pel Teensy, es a dir abans de passar pels conversors analògics digitals. Obtenim un senyal de 5 kHz amb una tensió aproximada de pic a pic de 2 V. Com podem veure l'etapa analògica funciona correctament.

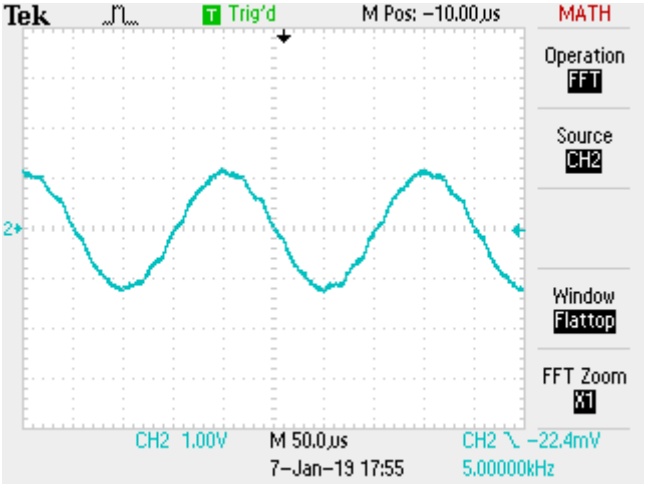


Figura 51 – Senyal a l'etapa de sortida dreta

En aquesta imatge podem observar el senyal de 5 kHz un com ha passat pel Teensy, es a dir, s'ha convertit del món analògic al digital i del digital a l'analògic. Com podem veure aquest senyal ja comença a tenir petits salts en la seva forma d'ona, això es degut per culpa de la resolució. En tenir una resolució baixa del DAC i quan introduïm senyals a partir de 5 kHz, comencem a tenir certs problemes amb el senyal de sortida. Si només ens fixem en la part del circuit analògic, deixant de banda els conversors, podem veure que funciona correctament, a la sortida obtindríem un senyal de 5 kHz amb una tensió de pic a pic aproximadament de 2 V.

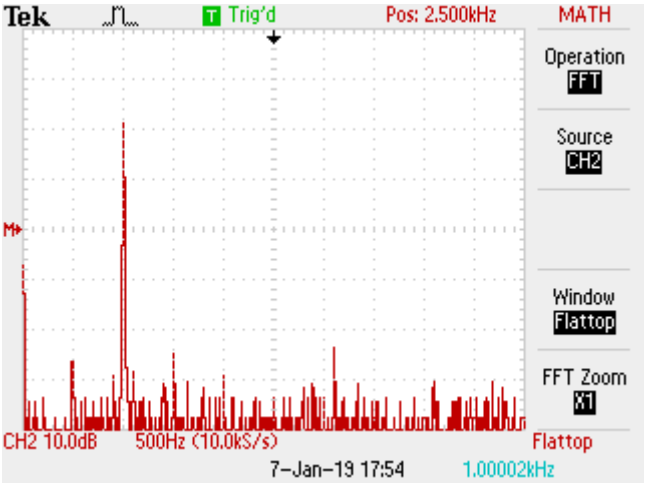


Figura 52 – FFT de l'etapa de sortida del canal dret

També es va realitzar la FFT del canal a una freqüència de 1 kHz i es pot veure que el resultat és molt satisfactori.

Com podem veure en les imatges anteriors, el canal dret del nostre prototip funciona correctament i com es pot esperar el canal esquerre i el de micròfon funcionen igual, ja que són el mateix disseny, exceptuant l'etapa de phantom del micròfon, a continuació es mostraran captures del canal esquerre del prototip.

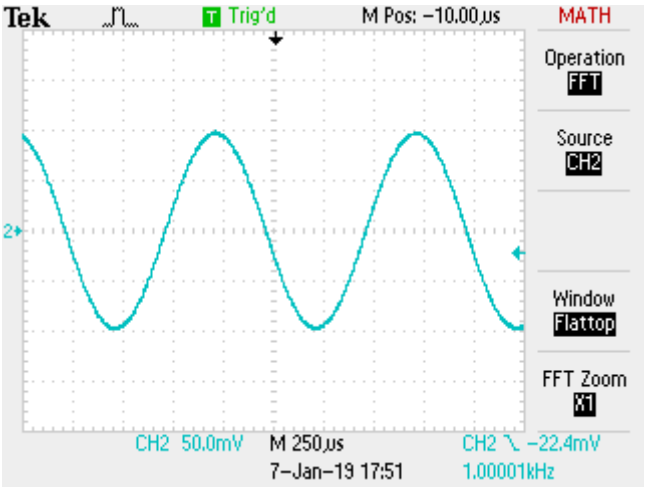


Figura 53 – Senyal d'entrada a l'etapa de línia esquerre

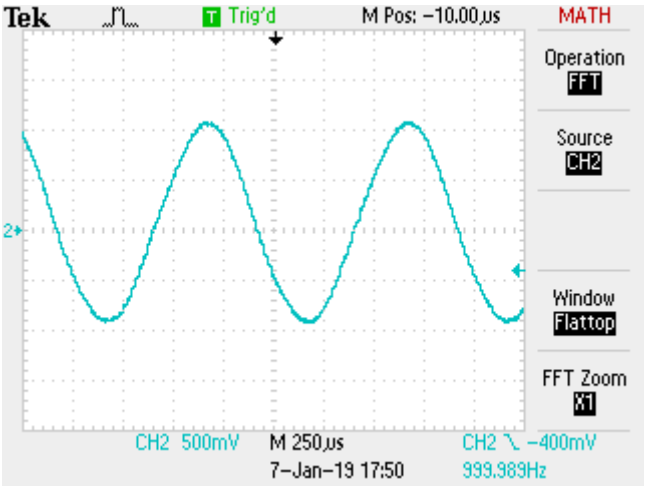


Figura 54 – Senyal de l'etapa de sortida esquerre

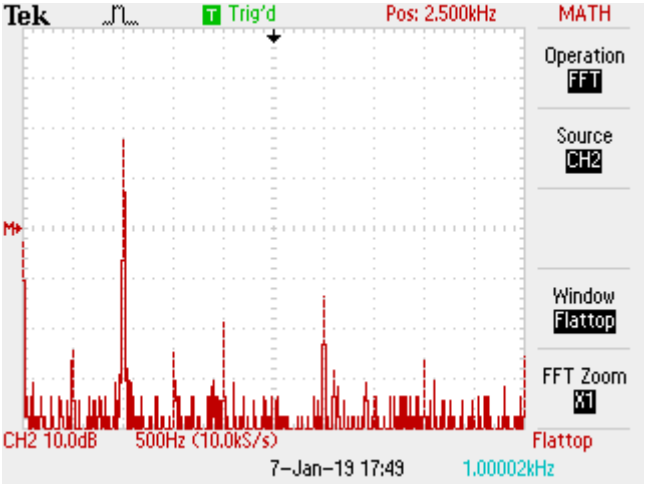


Figura 55 – FFT canal esquerre

Per veure els diferents voltatges de sortida de cadascuna de les etapes d'entrada i de sortida s'han fet unes taules per veure els valors d'amplitud a diversos valors de freqüència. També podrem veure el guany de cascuna de les bandes. Com veureu a continuació tots els valors són molt semblants.

Freqüència [Hz]	Vin [Vpp]	LEFT [Vpp]	Guany_LEFT	RIGHT [Vpp]	Guany_RIGHT	MIC [Vpp]	Guany_MIC
20	0,2	2,28	11,4	2,2	11	2,2	11
50	0,2	2,28	11,4	2,2	11	2,2	11
100	0,2	2,28	11,4	2,2	11	2,2	11
500	0,2	2,28	11,4	2,2	11	2,2	11
1000	0,2	2,28	11,4	2,2	11	2,2	11
5000	0,2	2,28	11,4	2,28	11,4	2,2	11
10000	0,2	2,24	11,2	2,28	11,4	2,26	11,3
20000	0,2	2,18	10,9	2,2	11	2,2	11

Taula 4 – Taula de resultats 1

Freqüència [Hz]	Vin [Vpp]	OUT_L [Vpp]	OUT_R [Vpp]	Guany
20	0,2	2,88	2,88	14,4
50	0,2	3,16	3,16	15,8
100	0,2	3,2	3,2	16
500	0,2	3,2	3,2	16
1000	0,2	3,2	3,2	16
5000	0,2	3,16	3,16	15,8
10000	0,2	2,8	2,8	14
20000	0,2			

Taula 5 – Taula de resultats 2

Com podem veure a la freqüència de 20 kHz no tenim valor de Vpp, això és perquè es distorsiona la sortida, per obtenir algun valor hauríem de pujar la freqüència de mostreig al doble, però això també comporta més potencia que necessitarem al microcontrolador.

Per tancar el capítol dels resultats en la part de hardware, dir que el consum que hem tingut de totes les etapes funcionant, amb el Teensy i la pantalla connectades, ha sigut de 0,08 A.

Figura 56 – Consum prototip

5.2. PANTALLA LCD

Com hem vist a l'apartat 3.5, s'ha agafat un exemple d'analitzador d'espectre de 7 bandes i s'ha adaptat al nostre sistema.

A continuació veurem el resultat:

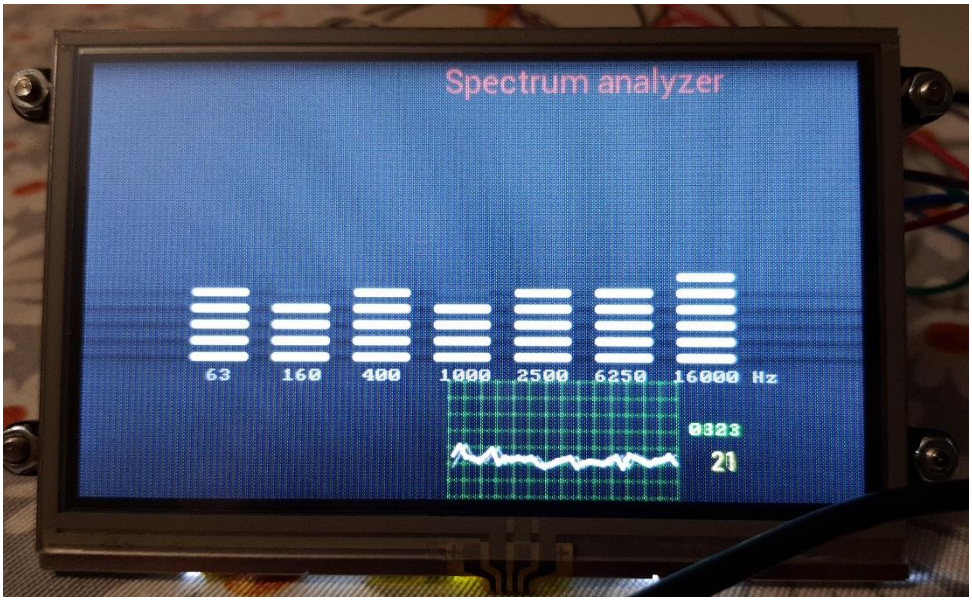


Figura 57 – Pantalla LCD

Com es pot veure en la imatge, tenim un analitzador d'espectre de 7 bandes de freqüències, amb un visor de forma d'ona just a sota. L'analitzador són 7 barres verticals que van pujant o baixant en funció de l'amplitud de cada banda de freqüència.

6. CONCLUSIONS

L'objectiu principal d'aquest projecte, fer l'equalitzador automàtic de P.A., no s'ha pogut assolir. Ha sigut un projecte molt ambiciós i per falta de temps no ha sigut possible acabar amb totes les idees expressades al principi d'aquesta memòria. Tot i això, el projecte podríem dir que està un 85 % finalitzat, ja que hi ha coses que estan mig implementades i que només faltaria l'últim retoc. Podríem dividir aquestes conclusions en dues parts diferencials, la part del hardware i la part del software.

La part del hardware sí que s'han complert els objectius establerts i tot ha funcionat correctament. S'ha de tenir en compte que el circuit es va provar directament en la placa de circuit imprès, sense haver provat cap etapa en una protoboard, només es va fer la simulació de cada etapa. Per tant es pot dir que el disseny del hardware ha assolit tots els objectius marcats al principi del projecte.

La part del software no ha acabat de complir tots els objectius establerts, ens hem trobat amb massa problemes de compatibilitat amb la placa de desenvolupament Teensy 3,6. Per exemple, la llibreria ArduinoFFT, és totalment compatible amb aquesta placa, però no acaba de funcionar bé. Amb la pantalla hem tingut problemes similars de compatibilitat. Com hem mostrat als resultats, la pantalla està programada i es veu que està funcionant, però de cop va deixar de funcionar. Hi ha algun problema amb la llibreria Gameduino 3, ja que en carregar un altre exemple d'una altra llibreria, com pot ser Gameduino 2, sí que funciona la pantalla. Per tant, ens hem trobat aquests problemes que ens han fet perdre temps en intentar arreglar-ho.

Estic satisfet amb el treball realitzat, però m'haguera agradat que el projecte hagués funcionat correctament, ja que és una idea que portava molt temps pensant i crec que és un aparell que pot ajudar molt en el món de l'àudio professional. Personalment estic satisfet pel fet de poder haver aplicat els coneixements après durant la meva vida d'estudi, ja sigui en el grau superior com en el grau en enginyeria i d'haver après més del món de l'electrònica, portant un projecte molt ambiciós a terme i que la part del hardware és un bon disseny.

7. BIBLIOGRAFIA

[1] Texas Instruments. *INA217 Low-Noise, Low-Distortion Instrumentation Amplifier Replacement for SSM2017*, rev.2015

[2] Texas Instruments. *TL97x Output Rail-To-Rail Very-Low-Noise Operational Amplifiers*, rev.2015

[3] Paul Stoffregen. (s.d.). *Teensy 3.5 & 3.6*. Recuperat de: https://www.kickstarter.com/projects/paulstoffregen/teensy-35-and-36?lang=es

[4] NXP Semiconductors. *Kinetis K66 Sub-family. 180 MHz ARM Cortex-M4F Microcontroller*, rev.2017. 4

[5] Newhaven Display. *NHD-4.3-480272FT-CTXL-T*, rev.2017

[6] *Color 320x240 TFT Touchscreen, ILI9341 Controller Chip*. Recuperat de: https://www.pjrc.com/store/display_ili9341_touch.html

[7] Arduino. (Novembre 2017). *analogReadResolution().* Recuperat de: https://www.arduino.cc/reference/en/language/functions/zero-due-mkr-family/analogreadresolution/

[8] *Teensy audio library*. Recuperat de: https://www.pjrc.com/teensy/td_libs_Audio.html

[9] *Audio connections & memory*. Recuperat de: https://www.pjrc.com/teensy/td_libs_AudioConnection.html

[10] James Bowman. (Desembre 2013). *The Gameduino 2 Tutorial, Reference and Cookbook.* Recuperat de: http://excamera.com/files/gd2book_v0.pdf

[11] James Bowman. (Maig 2018). *Gd2-asset.* Recuperat de: https://github.com/jamesbowman/gd2-asset

[12] Lightcalamar. (Juliol 2017). *GD3_HOTMCU.* Recuperat de: https://github.com/lightcalamar/GD3_HOTMCU

[13] Lightcalamar. (s.d.). *Analizador espectro LCD 20x4 con MSGEQ7*. Recuperat de: https://forum.arduino.cc/index.php?topic=392378.0

8. ANNEX

8.1. ANNEX 1: CODI TEENSY

8. 1. 1. LECTURA I ESCRIPTURA DE DADES (ADC/DAC)

```
// Crear un objeto de IntervalTimer para la generacion del ruido rosa
// y la lectura del ADC.
IntervalTimer ADC_DAC_Timer;
// Variables relacionadas con la frecuencia y periodo de muestreo
const double samplingFrequency = 50000; //Hz
unsigned int sampling_period_us;
// Variable de la lectura del ADC
int read_ADC_L, read_ADC_R;
const int readPin = A1; // ADC1
const int readPin2 = A2; // ADC2
// Escritura del DAC
const int writePin_L = A21;
const int writePin_R = A22;
void setup() {
  // ADC0 y ADC1 como entrada
        pinMode(readPin, INPUT);
        pinMode(readPin2, INPUT);
        sampling_period_us = round(1000000 * (1.0 / samplingFrequency)); // calculo del
        periodo de muestreo
```

```
        ADC_DAC_Timer.begin(lectura_ADC_DAC, sampling_period_us);  //
        ADC_DAC_Timer to run every Ts seconds

        analogReadResolution(12);

        analogWriteResolution(12);

        // Configurar los ADC

        // init_ADC();

}

void loop() {

}

void lectura_ADC_DAC() {

        analogWrite(writePin_L, read_ADC_L);

        analogWrite(writePin_R, read_ADC_R);

        read_ADC_L = analogRead(readPin);

        read_ADC_R = analogRead(readPin2);

}
```

8. 1. 2. TRANSFORMADA RÀPIDA DE FOURIER (FFT)

```
#include <Audio.h>

#include <Wire.h>

#include <SPI.h>

#include <SD.h>

#include <SerialFlash.h>

// Create the Audio components.

AudioInputAnalog      adc0(A0);

AudioAnalyzeFFT1024   myFFT;

AudioConnection       patchCord1(adc0, myFFT);
```

```
void setup() {
  // Audio connections require memory to work.
  AudioMemory(12);
  // Configure the window algorithm to use
  myFFT.windowFunction(AudioWindowHanning1024);
}
void loop() {
  float n;
  int i;
  if (myFFT.available()) {
    // each time new FFT data is available
    // print it all to the Arduino Serial Monitor
    Serial.print("FFT: ");
    for (i=0; i<100; i++) {
      n = myFFT.read(i);
      if (n >= 0.01) {
        Serial.print(n);
        Serial.print(" ");
      } else {
        Serial.print("  -  "); // don't print "0.00"
      }
    }
    Serial.println();
  }
}
```

8. 1. 3. ANALITZADOR D'ESPECTRES DE 7 BANDES

```
#include <EEPROM.h>

#include <SPI.h>

#include <GD3.h>

char TX[50];

int Lectura;

int Valor;

const int maxlecturas = 35;

int lectura[maxlecturas];

int k, j;

//Posición base de la gráfica

int xinicial = 160;

int ybaseT = -2;

//Posición base de la gráfica

int DiamPT = 1;

int separacion = 3 + (DiamPT * 1.5);

//Posición del listado de la base de datos

int xTextoT = 375;

int yTextoT = 10;

//Millis en lugar de delay para gráfica lineal

long previousMillis = 0;

long interval = 0;

int left[7], R[7];

int band;

int sepY = 10, sepX = 20, largoX = 30;
```

```
int xi = 0, yi = 185;  //xi=75
int segmentos = 14;
void readMSGEQ7()
{
 for (band = 0; band < 7; band++)
 {
  left[band] = analogRead(A2);
 }
}
void barrasVSegmentadas()
{
 GD.Begin(LINES);
 GD.LineWidth(35);
 GD.ColorRGB(0xffffff);   //color de los segmentos
 for (int j = 0; j < band; j++)
 {
  int dato = map(left[j], 0, 1023, 0, segmentos);
  for (int i = 0; i < dato; i++)
  {
   GD.Vertex2ii(xi + (j * largoX + j * sepX), yi - (i * sepY)); // inicio
   GD.Vertex2ii(xi + ((j + 1)*largoX + j * sepX), yi - (i * sepY)); //  fin
  }
 }
}
void LecturaMSGEQ7()
{
```

```
  // readMSGEQ7();
  Lectura = left[1]; //canal selecciionado
  // Lectura=(left[0]+left[1]+left[2]+left[3]+left[4]+left[5]+left[6])/7;  //canal seleccionado
  Valor = Lectura / 15; //ajuste para que el gráfico quede dentro de la ventana
}
void lineabaseMSGEQ7()
{
  for (int k = 0; k < maxlecturas; k++)
  {
    LecturaMSGEQ7();
    lectura[k] = Valor;
  }
  k = maxlecturas;
}
void setup() {
  lineabaseMSGEQ7();  //Solo genera la linea base
  Serial.begin(115200);
  GD.begin();
  GD.cmd_setrotate(0);
  analogReadResolution(12);
  menuPrincipal();
}
void loop() {}
int DeltaXR = 250;
int XbaseD = 15, Xdelta = 30; //15, 30
void fondopantalla()
```

```
{
  //GD.Vertex2ii(0, 0, 15);  //fondo de pantalla
  GD.Vertex2ii(XbaseD + Xdelta * 0, 300, map(left[0], 0, 1024, 0, 15)); // de 0-1024 pasas a 0-15
  GD.Vertex2ii(XbaseD + Xdelta * 1, 300, map(left[1], 0, 1024, 0, 15));
  GD.Vertex2ii(XbaseD + Xdelta * 2, 300, map(left[2], 0, 1024, 0, 15));
  GD.Vertex2ii(XbaseD + Xdelta * 3, 300, map(left[3], 0, 1024, 0, 15));
  GD.Vertex2ii(XbaseD + Xdelta * 4, 300, map(left[4], 0, 1024, 0, 15));
  GD.Vertex2ii(XbaseD + Xdelta * 5, 300, map(left[5], 0, 1024, 0, 15));
  GD.Vertex2ii(XbaseD + Xdelta * 6, 300, map(left[6], 0, 1024, 0, 15));
  GD.SaveContext();
  GD.VertexTranslateX(16 * 260);
  GD.Vertex2ii(500 - 180, 300, map(R[0], 0, 1024, 0, 15));
  GD.Vertex2ii(500 - 150, 300, map(R[1], 0, 1024, 0, 15));
  GD.Vertex2ii(500 - 120, 300, map(R[2], 0, 1024, 0, 15));
  GD.Vertex2ii(500 - 90, 300, map(R[3], 0, 1024, 0, 15));
  GD.Vertex2ii(500 - 60, 300, map(R[4], 0, 1024, 0, 15));
  GD.Vertex2ii(500 - 30, 300, map(R[5], 0, 1024, 0, 15));
  GD.Vertex2ii(500, 300, map(R[6], 0, 1024, 0, 15));
  GD.RestoreContext();
  GD.VertexTranslateX(8 * 140); //centra elementos restantes 16*140
  // Diseño de marco/fondo para gráfica
  GD.ColorA(75); //transperencia
  GD.ColorRGB(255, 0, 0); //Color del texto
  GD.cmd_text(240, 12, 28, OPT_CENTER, "Spectrum analyzer");
  GD.ColorRGB(255, 255, 255); //Color del texto
```

```
  GD.cmd_text(xi, yi + 8, 16, 0, " 63         400   1000  2500  6250  16000 Hz");
  GD.cmd_text(xi + 55, yi + 8, 16, 0, "160");
  GD.ColorA(40); //transperencia
  GD.ColorRGB(0x00ff00);
  GD.LineWidth(1 * 16);
  GD.Begin(LINES);
  marcadoresverticales();
  marcadoreshorizontales();
  GD.ColorA(255); //color solido
}
int option = 0;
void seleccion()
{
  if (option == 0) {
    GD.cmd_button(5, 5, 50, 35, 26, 0,  "MP"); //boton 3D
  }
  if (option == 1) {
    GD.cmd_button(5, 5, 50, 35, 26, OPT_FLAT,  "PM"); //boton plano
  }
}
void menuPrincipal()
{
  GD.cmd_fgcolor(0x954500); //color del boton
  do {
    GD.Clear();
    GD.Begin(BITMAPS);
```

```
   readMSGEQ7();
   fondopantalla();
   Baseactual2();    //  Gráfica lineal
   barrasVSegmentadas();
   GD.get_inputs();
   GD.swap();
  }
  while (1);
}
// Diseño de marco para gráfica
int RectWidth = 178, RectHeight = 70;
int XGT = 157, YGT = 200;
void fondografica()
{
  GD.Vertex2f(XGT * 16, (YGT) * 16);                    //Esquina superior izquierda
  GD.Vertex2f((XGT + RectWidth) * 16, (YGT + RectHeight) * 16); //esquina inferior derecha
}
void marcadoresverticales()
{
  for (int k = 0; k < 15; k++)
  {
   GD.Vertex2f((XGT + (k * 10)) * 16, YGT * 16); //starting coordinates
   GD.Vertex2f((XGT + (k * 10)) * 16, (YGT + RectHeight) * 16); //ending coordinates
  }
}
void marcadoreshorizontales()
```

```
{
  for (int k = 0; k < 8; k++)
  {
    GD.Vertex2f((XGT) * 16, (YGT + (k * 10)) * 16); //starting coordinates
    GD.Vertex2f((XGT + 140) * 16, (YGT + (k * 10)) * 16); //ending coordinates
  }
}
// Diseño de marco para gráfica
void Arraydatos2()
{
  for (j = 0; j < 13; j++)
  {
    GD.ColorRGB(0, 255, 0);
    sprintf(TX, "%02d", j);         GD.cmd_text(xTextoT, (yTextoT + 15 * j), 16, 0, TX);
    sprintf(TX, "%02d", lectura[j]); GD.cmd_text(xTextoT + 40, (yTextoT + 15 * j), 16, 0, TX);
  }
}
void insertadatos1()
{
  // Esta función debe recorrer los datos hacia atrás y luego insertará el dato actual al final
de la lista
  //  La base nueva corresponde a la base previa -1 dato
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis > interval)
  {
    previousMillis = currentMillis;
```

```
   for (j = 0; j < maxlecturas - 1; j++)

   {

    lectura[j] = lectura[j + 1]; // genera la base de datos previa, recorriendo un dato hacia atrás

   }

   LecturaMSGEQ7();  //Toma la lectura actual del sensor

   lectura[maxlecturas - 1] = Valor;     //base de maxlecturas puntos

  }

}

void Baseactual2()  // lineas

{

  //lee la base de datos actual

  //Gráfica lineal

  GD.ColorRGB(0xFF, 0xFF, 0xFF);

  GD.Begin(LINE_STRIP);

  for (j = 0; j < maxlecturas; j++)

  {

   GD.Vertex2f(((j * separacion) + xinicial) * 16, (272 + (ybaseT) - (lectura[j])) * 16);

  }

  lectura[maxlecturas - 1] = Valor;   GD.ColorRGB(255, 255, 0); sprintf(TX, "%02d", lectura[maxlecturas - 1]); GD.cmd_text(318, 240, 26, 0, TX); //Presenta lectura actual

  GD.ColorRGB(0, 255, 0); sprintf(TX, "%04d", Lectura); GD.cmd_text(305, 225, 16, 0, TX);

  j = 0; //al salir deja j=0 para empezar la matriz en (0,0)

  insertadatos1();  //Inserta la lectura actual en la base previa

}
```

8.2. ANNEX 2: DATASHEET

8. 2. 1. INA217

TEXAS INSTRUMENTS
INA217
SBOS247C – JUNE 2002 – REVISED NOVEMBER 2015

INA217 Low-Noise, Low-Distortion Instrumentation Amplifier Replacement for SSM2017

1 Features

- Low Noise: 1.3 nV/M√Hz at 1 kHz
- Low THD+N: 0.004% at 1 kHz, G = 100
- Wide Bandwidth: 800 kHz at G = 100
- Wide Supply Range: ±4.5 V to ±18 V
- High CMR: > 100 dB
- Gain Set With External Resistor
- DIP-8 and SOL-16 Widebody Packages

2 Applications

- Professional Microphone Preamps
- Moving-coil Transducer Amplifiers
- Differential Receivers
- Bridge Transducer Amplifiers

3 Description

The INA217 device is a low-noise, low-distortion, monolithic instrumentation amplifier. Current-feedback circuitry allows the INA217 device to achieve wide bandwidth and excellent dynamic response over a wide range of gain. The INA217 device is ideal for low-level audio signals such as balanced low-impedance microphones. Many industrial, instrumentation, and medical applications also benefit from its low noise and wide bandwidth.

Unique distortion cancellation circuitry reduces distortion to extremely low levels, even in high gain. The INA217 device provides near-theoretical noise performance for 200-Ω source impedance. The INA217 device features differential input, low noise, and low distortion that provides superior performance in professional microphone amplifier applications.

The INA217device features wide supply voltage, excellent output voltage swing, and high output current drive, making it an optimal candidate for use in high-level audio stages.

The INA217 device is available in the same DIP-8 and SOL-16 wide body packages and pinouts as the SSM2017. For a smaller package, see the INA163 device in SO-14 narrow. The INA217 device is specified over the temperature range of –40°C to 85°C.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
INA217	SOIC (16)	10.30 mm × 7.50 mm
	PDIP (8)	9.81 mm × 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

8. 2. 2. TL972

TEXAS INSTRUMENTS
TL971, TL972, TL974
SLOS467H – OCTOBER 2006 – REVISED JANUARY 2015

TL97x Output Rail-To-Rail Very-Low-Noise Operational Amplifiers

1 Features

- Rail-to-Rail Output Voltage Swing: ±2.4 V at V_{CC} = ±2.5 V
- Very Low Noise Level: 4 nV/√Hz
- Ultra-Low Distortion: 0.003%
- High Dynamic Features: 12 MHz, 5 V/µs
- Operating Range: 2.7 V to 12 V
- Latch-Up Performance Exceeds 100 mA Per JESD 78, Class II
- ESD Performance Tested Per JESD 22
  - 2000-V Human-Body Model
  - 1500-V Charged-Device Model

2 Applications

- Portable Equipment
  - Music Players
  - Tablets
  - Cell Phones
- Instrumentation and Sensors
- Professional Audio Circuits

3 Description

The TL97x family of single, dual, and quad operational amplifiers operates at voltages as low as ±1.35 V and features output rail-to-rail signal swing. The TL97x boast characteristics that make them particularly well suited for portable and battery-supplied equipment. Very low noise and low distortion characteristics make them ideal for audio preamplification.

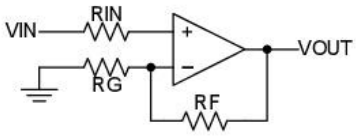
The TL971 is housed in the space-saving 5-pin SOT-23 package, which simplifies board design because of the ability to be placed anywhere (outside dimensions are 2.8 mm × 2.9 mm).

Device Information(1)

PART NUMBER	PACKAGE (PIN)	BODY SIZE (NOM)
TL971	SOIC (8)	4.90 mm × 3.90 mm
	SOT-23 (5)	2.80 mm × 2.90 mm
TL972	MSOP (8)	3.00 mm × 3.00 mm
	PDIP (8)	9.60 mm × 6.40 mm
	SOIC (8)	4.90 mm × 3.90 mm
	TSSOP (8)	3.00 mm × 4.40 mm
TL974	PDIP (14)	19.30 mm × 6.40 mm
	SOIC (14)	8.60 mm × 3.90 mm
	TSSOP (14)	5.00 mm × 4.40 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

4 Simplified Schematic



8. 2. 3. MK66FX1M0VMD18

NXP Semiconductors
Data Sheet: Technical Data

K66P144M180SF5V2
Rev. 4, 04/2017

Kinetis K66 Sub-Family

180 MHz ARM® Cortex®-M4F Microcontroller.

MK66FN2M0VMD18
MK66FX1M0VMD18
MK66FN2M0VLQ18
MK66FX1M0VLQ18

The K66 sub-family members provide greater performance, memory options up to 2 MB total flash and 256 KB of SRAM, as well as higher peripheral integration with features such as Dual USB and a 10/100 Mbit/s Ethernet MAC. These devices maintain hardware and software compatibility with the existing Kinetis family. This product also offers:

- Integration of a High Speed USB Physical Transceiver
- Greater performance flexibility with a High Speed Run mode
- Smarter peripherals with operation in Stop modes

144 MAPBGA (MD) 13 mm x 13 mm Pitch 1 mm

144 LQFP (LQ) 20 mm x 20 mm Pitch 0.5 mm

Performance
- Up to 180 MHz ARM Cortex-M4 based core with DSP instructions and Single Precision Floating Point unit

System and Clocks
- Multiple low-power modes to provide power optimization based on application requirements
- Memory protection unit with multi-master protection
- 3 to 32 MHz main crystal oscillator
- 32 kHz low power crystal oscillator
- 48 MHz internal reference

Security
- Hardware random-number generator
- Supports DES, AES, SHA accelerator (CAU)
- Multiple levels of embedded flash security

Timers
- Four Periodic Interrupt timers
- 16-bit low-power timer
- Two 16-bit low-power timer PWM modules
- Two 8-channel motor control/general purpose/PWM timers
- Two 2-ch quad decoder/general purpose timers
- Real-time clock

Human-machine Interface
- Low-power hardware touch sensor interface (TSI)
- General-purpose input/output

Memories and memory expansion
- Up to 2 MB program flash memory on non-FlexMemory devices with 256 KB RAM
- Up to 1 MB program flash memory and 256 KB of FlexNVM on FlexMemory devices
- 4 KB FlexRAM on FlexMemory devices
- FlexBus external bus interface and SDRAM controller

Analog modules
- Two 16-bit SAR ADCs and two 12-bit DAC
- Four analog comparators (CMP) containing a 6-bit DAC and programmable reference input
- Voltage reference 1.2V

Communication interfaces
- Ethernet controller with MII and RMII interface to external PHY and hardware IEEE 1588 capability
- USB high-/full-/low-speed On-the-Go with on-chip high speed transceiver
- USB full-/low-speed OTG with on-chip transceiver
- Two CAN, three SPI and four I2C modules
- Low Power Universal Asynchronous Receiver/Transmitter 0 (LPUART0) and five standard UARTs
- Secure Digital Host Controller (SDHC)
- I2S module

Operating Characteristics
- Voltage/Flash write voltage range:1.71 to 3.6 V
- Temperature range (ambient): -40 to 105°C

NXP reserves the right to change the production detail specifications as may be required to permit improvements in the design of its products.

8. 2. 4. NEWHAVEN DISPLAY

NEWHAVEN DISPLAY INTERNATIONAL

NHD-4.3-480272FT-CTXL-T

EVE2 TFT Module (SPI) Supports: Display | Touch | Audio

NHD-	Newhaven Display
4.3-	4.3" Diagonal
480272-	480xRGBx272 Pixels
FT-	Model
C-	On-board Controller
T-	White LED Backlight
X-	TFT
L-	6:00 Optimal View, Wide Temperature
T-	Resistive Touch Panel

Newhaven Display International, Inc.
2661 Galvin Ct.
Elgin IL, 60124
Ph: 847-844-8795 Fax: 847-844-8796

www.newhavendisplay.com
nhtech@newhavendisplay.com nhsales@newhavendisplay.com